An extended abstract of this paper appears in the Proceedings of the 35th Annual Cryptology Conference (CRYPTO 2015), Part I, Rosario Gennaro and Matthew Robshaw (Eds.), volume 9215 of Lecture Notes in Computer Science, pages 388–409, Springer, August 2015. This is the full version.

An Algebraic Framework for Pseudorandom Functions and Applications to Related-Key Security

Michel Abdalla Fabrice Benhamouda Alain Passelègue

ENS, CNRS, INRIA, and PSL
45 Rue d'Ulm, 75230 Paris Cedex 05, France
{michel.abdalla,fabrice.ben.hamouda,alain.passelegue}@ens.fr
http://www.di.ens.fr/∼{mabdalla,fbenhamo,passeleg}

Abstract

In this work, we provide a new algebraic framework for pseudorandom functions which encompasses many of the existing algebraic constructions, including the ones by Naor and Reingold (FOCS'97), by Lewko and Waters (CCS'09), and by Boneh, Montgomery, and Raghunathan (CCS'10), as well as the related-key-secure pseudorandom functions by Bellare and Cash (Crypto'10) and by Abdalla *et al.* (Crypto'14). To achieve this goal, we introduce two versions of our framework. The first, termed linearly independent polynomial security, states that the values $(g^{P_1(\vec{a})}, \ldots, g^{P_q(\vec{a})})$ are indistinguishable from a random tuple of the same size, when $P_1, \ldots, P_q$ are linearly independent multivariate polynomials of the secret key vector $\vec{a}$. The second, which is a natural generalization of the first framework, additionally deals with constructions based on the decision linear and matrix Diffie-Hellman assumptions. In addition to unifying and simplifying proofs for existing schemes, our framework also yields new results, such as related-key security with respect to arbitrary permutations of polynomials. Our constructions are in the standard model and do not require the existence of multilinear maps.

Keywords. Related-Key Security, Pseudorandom Functions.

Contents

1 Introduction

Pseudorandom functions (PRFs), originally defined by Goldreich, Goldwasser, and Micali [GGM86], are one of the most fundamental primitives in cryptography. Informally speaking, a function is said to be pseudorandom if its outputs are indistinguishable from that of a random function with respect to a computationally bounded adversary which only has black-box access to it. Hence, even if the adversary can control the inputs on which the function is computed and see the corresponding outputs, he or she should still not be able to distinguish this function from a perfectly random one.

Due to their simplicity and security properties, pseudorandom functions have been used in numerous applications, including symmetric encryption, authentication, and key exchange. In particular, since pseudorandom functions can be used to model real-world block-ciphers, such as AES [AES01], they are also extremely useful for the security analysis of protocols that rely on these primitives.

Number-Theoretic Constructions. Despite its elegance, the original construction of pseudorandom functions by Goldreich, Goldwasser, and Micali based on pseudorandom generators was not very efficient. In order to improve its efficiency while still being able to prove its security under reasonable complexity assumptions, Naor and Reingold [NR97] proposed a new construction based on the Decisional Diffie-Hellman assumption (DDH) [NR97]. Let $\vec{a} = (a_0, \ldots, a_n) \in \mathbb{Z}_p^{n+1}$ be the key and $x = x_1 \| \ldots \| x_n \in \{0,1\}^n$ be the input of the PRF. Let g be a fixed public generator of a group $\mathbb{G}$ of prime order p. The Naor-Reingold PRF is then defined as

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

where for any $a \in \mathbb{Z}_p$, $[a]$ stands for g^a, as defined in [EHK+13].

As mentioned in [BMR10], the algebraic nature of the Naor-Reingold PRF has led to many applications, such as verifiable random functions [ACF09, HW10], distributed PRFs [NR97], and related-key-secure PRFs [BC10], which are hard to obtain from generic PRFs. Hence, due to its importance, several other extensions of the Naor-Reingold PRF have been proposed [LW09, BMR10] based on different assumptions, such as the Decision Linear assumption (DLin) [BBS04] and the d-DDHI assumption [BMR10, GOR11].

In this work, our main contribution is to further extend the above line of work by providing a *generic algebraic framework* for building pseudorandom functions. In particular, all of the algebraic constructions mentioned above can be seen as a particular instantiations of our framework. In addition, our framework is general enough that it captures and extends other constructions such as the related-key-secure PRF constructions by Bellare and Cash [BC10] (BC) and by Abdalla *et al.* [ABPP14] (ABPP).

Linearly Independent Polynomial Security. To obtain our results, our first contribution is to introduce a new notion of linearly independent polynomial (LIP) security. Informally, it states that the values $([P_1(\vec{a})], \ldots, [P_q(\vec{a})])$ are indistinguishable from a random tuple of the same size, when $P_1, \ldots, P_q$ are linearly independent multivariate polynomials of degree at most d in any indeterminate and $\vec{a}$ is the PRF secret key vector. The new notion is based on a new MDDH assumption [EHK+13] over the underlying group $\mathbb{G}$, denoted $\mathcal{E}_{1,d}$-MDDH, which can be (tightly) reduced to either DDH or DDHI depending on value of d.

In order to illustrate the usefulness of the new notion, we show in Section 4 how to use it to provide alternative security proofs for the Naor-Reingold PRF [NR97] and the PRF by Boneh, Montgomery, and Raghunathan (BMR) in [BMR10] as well as generalizations of both these PRFs, that we call *weighted* NR and *weighted* BMR. Intuitively, all these PRFs are defined over a prime order group $\mathbb{G} = \langle g \rangle$ as a function F that takes a key $\vec{a}$ and an input x and outputs an element in $\mathbb{G}$ of the shape $F(\vec{a}, x) = [P_x(\vec{a})]$ where the polynomial P_x depends on x. Hence, to prove the security of such constructions, we just need to prove that all polynomials P_x, for any entries x, are linearly independent.

We would like to remark that the actual formulation of the LIP security in Section 3 includes a value $a' \in \mathbb{Z}_p$ multiplying each $P_i(\vec{a})$ term, which allows for the use of different generators in the PRF constructions. While we could dispense with a' in the case where a' and the a_i values in $\vec{a}$ are scalars, we opted to use it to be consistent with the case in which these values are matrices, as in Section 6.

Applications to Related-Key Security. Related-key attacks (RKAs) were first introduced by Biham and Knudsen [Bih94, Knu93] and consider the setting in which an adversary could force a given cryptographic primitive to execute under a different but related key. Over the years, such attacks became more predominant and several related-key attacks have been proposed against existing block-ciphers (e.g., [BDK05, BKN09, KHP07]). Since these attacks are quite powerful and hard to defend against,

Bellare and Kohno [BK03] introduced a formal treatment of these attacks in the context of PRFs and pseudorandom permutations (PRPs) to better understand if and how one could achieve security in the presence of related-key attacks. One of their main observations is that certain classes of related-key attacks are impossible to protect against and, hence, their goal was to identify the set of classes Φ for which one could design secure RKA-PRFs and RKA-PRPs.

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 related-key deriving (RKD) functions on the key space $\mathcal{K}$. Let $G\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ be a random function and let $K \in \mathcal{K}$ be a random target key. Informally, in the RKA security model of [BK03], F is said to be a Φ-RKA-PRF if no polynomial-time adversary can distinguish the output of $F(\phi(K), x)$ from the output of $G(\phi(K), x)$, for pairs (ϕ, x) of its choice, with non-negligible probability.

Our second contribution is to show that the new LIP security notion can be used to prove directly the related-key security of certain constructions. In particular, we show that a particular case of our weighted BMR PRF construction is secure against permutations of the secret key. In these attacks, the attacker can obtain the output of the PRF with respect to any key that is a permutation of the original one.

To understand why RKA security can follow from the LIP security notion, let F be a PRF defined over a prime-order group $\mathbb{G} = \langle g \rangle$ that takes a key $\vec{a}$ and an input x and outputs $F(\vec{a}, x) = [P_x(\vec{a})]$. Let Φ be a class of RKD functions, where functions $\vec{\phi} = (\phi_1, \ldots, \phi_n) \in \Phi$ are such that ϕ_i are multivariate polynomials in $\mathbb{Z}_p[T_1, \ldots, T_n]$. Then, for a RKD function $\vec{\phi}$ and an input x, the PRF outputs $F(\vec{\phi}(\vec{a}), x) = \left[P_{\vec{\phi},x}(\vec{a})\right]$, where the polynomial $P_{\vec{\phi},x}(\vec{T}) = P_x(\vec{\phi}(\vec{T})) = P_x(\phi_1(\vec{T}), \ldots, \phi_n(\vec{T}))$ depends on $\vec{\phi}$ and x, with $\vec{T} = (T_1, \ldots, T_n)$. Hence, when all polynomials $P_{\vec{\phi},x}$ are linearly independent, the LIP security notion directly shows that F is Φ-RKA-secure.

Related-Key Security With Respect to Unique-Input Adversaries. Unfortunately, the case in which the polynomials $P_{\vec{\phi},x}$ are all linearly independent is not so easy to instantiate as we would like, and we have only been able to directly obtain RKA security for very restricted classes. Hence, to overcome these restrictions, our third contribution is to further extend our results in Section 5.2 to deal with the case where polynomials are only linearly independent when all the inputs x are distinct. This scenario is similar to the one considered in [ABPP14]. In particular, our new algebraic framework extends the one from [ABPP14] and provides constructions for new and larger classes of RKD functions. More precisely, we build in Section 5.2 RKA-PRFs against classes of permutations of univariate polynomials. Furthermore, in Appendix F, we also consider classes of univariate polynomials and multivariate affine RKD functions.

For simplicity, the results in Section 5.2 only hold with respect to PRFs of the form $[P_x(\vec{a})]$ where P_x is a polynomial that depends on x. However, a more general framework which does not make this assumption is described in Appendix G.

An Algebraic Framework For Non-Commutative Structures. Finally, our last contribution is to extend the LIP security notion to work under weaker assumptions than DDH, such as DLin. As we point out in Section 6, the main difficulty in this case is that the key values a_i's may be matrices, which do not necessarily commute. To address this issue, we introduce natural conditions on the order of indeterminates which makes non-commutative and commutative polynomials behave in a similar manner. Through the new generalization, we not only deal with cases already covered by the LIP security notion, but we also capture PRFs based on the DLin and MDDH assumptions [EHK+13].

Further discussions. In addition to the seminal work of Goldreich, Goldwasser, and Micali [GGM86], several other frameworks for constructing PRFs have appeared in the literature, including [BCK96, NR99, BMR10] to name a few.

In [NR99], Naor and Reingold proposed the notion of pseudorandom synthesizers and provided several instantiations for it based on different complexity assumptions. Informally speaking, a pseudorandom synthesizer is a two-variable function, $S(\cdot, \cdot)$, so that, for polynomially many random and independent input assignments $(x_1, \ldots, x_m)$ and $(y_1, \ldots, y_m)$, the set of values $\{S(x_i, y_j)\}$ are computationally indistinguishable from uniform for i and j in $\{1, \ldots, m\}$.

In [BCK96], Bellare, Canetti, and Krawczyk provide a framework for building variable-length input PRFs from fixed-length input ones, known as the cascade construction. In their framework, one obtains a larger-domain PRF F' simply by partitioning the input x into a number n of small blocks $x_1, \ldots, x_n$ matching the domain of the underlying PRF F and using the output of F on key k_i and input x_i as the secret key k_{i+1} for the next stage. Since their framework requires the output of the underlying PRF to be at least as long as the secret key, it cannot be applied to PRFs with very small domains.

To circumvent the restrictions of the cascade construction, Boneh, Montgomery, and Raghunathan proposed an extension in [BMR10], known as the augmented cascade construction, in which supplemental

secret information is provided in every iteration. Unlike the cascade construction, its security does not follow from the standard security of the underlying PRF, requiring it to meet a new notion called parallel security.

While these frameworks are more general than ours and capable of handling different complexity assumptions (e.g., [BPR12]), they are more combinatorial in nature and do not fully exploit the algebraic nature of the underlying PRFs. In particular, it is not clear how to extend them to the RKA setting, which is one of the main applications of our new algebraic framework. Moreover, even in the standard PRF setting, our framework seems to possess complementary features compared to the existing ones. Notably, it only requires the verification of an algebraic condition (such as testing the linear independence of the polynomials) for each instantiation, which is generally easier to prove.

Other Related Work. It is worth mentioning that in the context of related-key security, Lewi, Montgomery and Raghunathan [LMR14] designed RKA-PRFs for similar classes of polynomial RKD functions. However, unlike their constructions, ours do not require multilinear maps. Also, our constructions are proven fully RKA-secure while theirs are only proven unique-input RKA-secure.

2 Definitions

Notations and Conventions. We denote by κ the security parameter. Let $F\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ be a function that takes a key $K \in \mathcal{K}$ and an input $x \in \mathcal{D}$ and returns an output $F(K, x) \in \mathcal{R}$. The set of all functions $F\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ is then denoted by $\mathsf{Fun}(\mathcal{K}, \mathcal{D}, \mathcal{R})$. Likewise, $\mathsf{Fun}(\mathcal{D}, \mathcal{R})$ denotes the set of all functions mapping $\mathcal{D}$ to $\mathcal{R}$. 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 $\vec{x}$ is a vector then we denote by $|\vec{x}|$ its length, so $\vec{x} = (x_1, \ldots, x_{|\vec{x}|})$. For a binary string x, we denote its length by $|x|$ so $x \in \{0,1\}^{|x|}$, x_i its i-th bit, so $x = x_1 \,\|\, \ldots \,\|\, x_n$. We extend these notations to any d-ary string x, for $d \geq 2$. For a matrix $\boldsymbol{A}$ of size $k \times m$, we denote by $a_{i,j}$ the coefficient of $\boldsymbol{A}$ in the i-th row and the j-th column. For a vector $\vec{\phi} = (\phi_1, \ldots, \phi_n)$ of n functions from S_1 to S_2 with $|\vec{\phi}| = n$ and $\vec{a} \in S_1$, we denote by $\vec{\phi}(\vec{a})$ the vector $(\phi_1(\vec{a}), \ldots, \phi_n(\vec{a})) \in S_2^n$. We denote by $\mathbb{Z}_p[T_1, \ldots, T_n]$ the ring of multivariate polynomials in indeterminates $T_1, \ldots, T_n$. For a polynomial $P \in \mathbb{Z}_p[T_1, \ldots, T_n]$, we denote $P(T_1, \ldots, T_n)$ by $P(\vec{T})$ and by $P(\vec{a})$ the evaluation of P by setting $\vec{T}$ to $\vec{a}$, meaning that we set $T_1 = a_1, \ldots, T_n = a_n$. For $F\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ and for a vector $\vec{x}$ over $\mathcal{D}$, we denote by $F(K, \vec{x})$ the vector $(F(K, x_1), \ldots, F(K, x_{|\vec{x}|}))$. We denote by $\mathfrak{S}_n$ the set of all permutations of $\{1, \ldots, n\}$.

Finally, we often implicitly consider a multiplicative group $\mathbb{G} = \langle g \rangle$ with public generator g of order p and we denote by $[a]_g$, or simply $[a]$ if there is no ambiguity about the generator, the element g^a, for any $a \in \mathbb{Z}_p$. Similarly, if $\boldsymbol{A}$ is a matrix in $\mathbb{Z}_p^{k \times m}$, $[\boldsymbol{A}]$ is a matrix $\boldsymbol{U} \in \mathbb{G}^{k \times m}$, such that $u_{i,j} = [a_{i,j}]$ for $i = 1, \ldots, k$ and $j = 1, \ldots, m$.

Games [BR06]. Most of our definitions and proofs use the code-based game-playing framework, in which a game has an **Initialize** procedure, procedures to respond to adversary oracle queries, and a **Finalize** procedure. To execute a game G with an adversary $\mathscr{A}$, we proceed as follows. First, **Initialize** is executed and its outputs become the input of $\mathscr{A}$. When $\mathscr{A}$ executes, its oracle queries are answered by the corresponding procedures of G. When $\mathscr{A}$ terminates, its outputs become the input of **Finalize**. The output of the latter, denoted $\mathrm{G}^{\mathscr{A}}$ is called the output of the game, and we let "$\mathrm{G}^{\mathscr{A}} \Rightarrow 1$" denote the event that this game output takes the value 1. 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 [GGM86, BC10]. The advantage of an adversary $\mathscr{A}$ in attacking the standard PRF security of a function $F\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ 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 $\mathrm{PRFReal}_F$ first picks $K \xleftarrow{\$} \mathcal{K}$ and responds to oracle query $\mathbf{Fn}(x)$ via $F(K, x)$. Game $\mathrm{PRFRand}_F$ first picks $f \xleftarrow{\$} \mathsf{Fun}(\mathcal{D}, \mathcal{R})$ and responds to oracle query $\mathbf{Fn}(x)$ via $f(x)$.

RKA-PRFs [BK03, BC10]. Let $F\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ be a function 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 RKA-PRF

Table 1: Security of $\mathcal{E}_{k,d}$-MDDH

	$k=1$	$k=2$	$k\geq 3$
$d=1$	$=\mathbf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}$	$\lesssim 2\cdot\mathbf{Adv}^{\mathcal{U}_2\text{-mddh}}_{\mathbb{G}}$	$\lesssim k\cdot\mathbf{Adv}^{\mathcal{U}_k\text{-mddh}}_{\mathbb{G}}$
$d\geq 2$	$\lesssim d\cdot\mathbf{Adv}^{d\text{-ddhi}}_{\mathbb{G}}$	generic bilinear group†	$?^\ddagger$

$\mathbf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}$, $\mathbf{Adv}^{d\text{-ddhi}}_{\mathbb{G}}$ and $\mathbf{Adv}^{\mathcal{U}_k\text{-mddh}}_{\mathbb{G}}$ are advantages for DDH, DDHI, and $\mathcal{U}_k$-MDDH. This latter assumption is weaker than k-Lin;

† proven in the generic (symmetric) bilinear group model [BB04] in Appendix H.3;

‡ (trivially) secure in the generic cyclic group model [Sho97], but nothing known about security in generic (symmetric) k-linear group model [Sha07, HK07].

security of F is defined via

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

Game $\mathrm{RKPRFReal}_F$ first picks $K \xleftarrow{\$} \mathcal{K}$ and responds to oracle query $\mathbf{RKFn}(\phi, x)$ via $F(\phi(K), x)$. Game $\mathrm{RKPRFRand}_F$ first picks $K \xleftarrow{\$} \mathcal{K}$ and $G \xleftarrow{\$} \mathsf{Fun}(\mathcal{K}, \mathcal{D}, \mathcal{R})$ 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 adversary, its advantage in attacking the RKA-PRF security is negligible.

Group Generators. All our PRFs and RKA-PRFs use a cyclic group of prime order p. The generator(s) used in their construction is supposed to be public. In particular, RKD functions *cannot* modify the generator(s). Our security proofs will then start by giving the generators to the adversary.

Hardness Assumptions. To get a simpler and unified framework, we introduce a particular MDDH assumption [EHK+13]: the $\mathcal{E}_{k,d}$-MDDH assumption, defined by the matrix distribution $\mathcal{E}_{k,d}$ which samples matrices $\mathbf{\Gamma}$ as follows

$$\mathbf{\Gamma} = \begin{pmatrix} \boldsymbol{A}_1^0 \cdot \boldsymbol{A}_0 \\ \boldsymbol{A}_1^1 \cdot \boldsymbol{A}_0 \\ \vdots \\ \boldsymbol{A}_1^d \cdot \boldsymbol{A}_0 \end{pmatrix} \in \mathbb{Z}_p^{k(d+1)\times k} \quad \text{with } \boldsymbol{A}_0, \boldsymbol{A}_1 \xleftarrow{\$} \mathbb{Z}_p^{k\times k}. \tag{1}$$

The advantage of an adversary $\mathscr{D}$ against the $\mathcal{E}_{k,d}$-MDDH assumption is

$$\mathbf{Adv}^{\mathcal{E}_{k,d}\text{-mddh}}_{\mathbb{G}}(\mathscr{D}) = \Pr\left[\mathscr{D}(g, [\mathbf{\Gamma}], [\mathbf{\Gamma}\cdot\boldsymbol{W}])\right] - \Pr\left[\mathscr{D}(g, [\mathbf{\Gamma}], [\boldsymbol{U}])\right],$$

where $\mathbf{\Gamma} \xleftarrow{\$} \mathcal{E}_{k,d}$, $\boldsymbol{W} \xleftarrow{\$} \mathbb{Z}_p^{k\times 1}$, $\boldsymbol{U} \xleftarrow{\$} \mathbb{Z}_p^{k(d+1)\times 1}$. As any MDDH assumption and as recalled in Appendix A.2, this assumption is random self-reducible, which enables us to make relatively tight proofs.

In Table 1, we summarize security results for $\mathcal{E}_{k,d}$-MDDH. For $k=1$ or $d=1$, the $\mathcal{E}_{k,d}$-MDDH assumption is implied by standard assumptions (DDH, DDHI, or k-Lin, recalled in Appendix A). $\mathcal{E}_{1,1}$-MDDH is actually exactly DDH.

For our RKA framework, we also make use of the d-Strong Discrete Logarithm (SDL) problem given in [GOR11] and recalled in Appendix A.

3 Linearly Independent Polynomial Security

In this section, we define a new security notion, termed linearly independent polynomial (LIP) security, which captures that, given a cyclic group $\mathbb{G} = \langle g \rangle$ of prime order p, the hardness of distinguishing a tuple $(y_1, \ldots, y_q) = ([P_1(\vec{a})\cdot a'], \ldots, [P_q(\vec{a})\cdot a']) \in \mathbb{G}^q$ from a random tuple in $(y_1, \ldots, y_q) \xleftarrow{\$} \mathbb{G}^q$, where $\vec{a}$ is a secret random vector in $\mathbb{Z}_p^n$, a' is a secret random scalar in $\mathbb{Z}_p$, and P_j are linearly independent multivariate polynomials. Our LIP theorem (Theorem 3.1) shows that distinguishing these two tuples is harder than the $\mathcal{E}_{1,d}$-MDDH problem in $\mathbb{G}$, where d is the maximum degree in one indeterminate in polynomials $P_1, \ldots, P_q$. We point out that, on the one hand, if there were a linear relation between the polynomials, i.e., if there exists $(\lambda_1, \ldots, \lambda_q) \in \mathbb{Z}_p^q \backslash \{(0, \ldots, 0)\}$, such that $\sum_{j=1}^q \lambda_j P_j = 0$, then it would be straightforward to break the LIP security by checking whether $\prod_{j=1}^q {y_j}^{\lambda_j} = 1$ (real case) or not (random case). So the linear independence of the P_j's is required.

On the other hand, if the polynomials P_j are linearly independent, then distinguishing the two tuples is hard in the generic group model, since in this model, the adversary can only compute linear combinations

of the group elements it is given (and check for equality). The LIP security is therefore not surprising. What is surprising, is that it is possible to prove it under classical assumptions such as $\mathcal{E}_{1,d}$-MDDH, without an exponential blow-up.

In the following, we first consider a particular case of the LIP theorem in which the polynomials are given in their expanded form. This section not only serves as a warm-up for the sequel, but it also helps better grasp the challenges of the proof of the full theorem and gives a nice overview. Next, we formally state the LIP theorem.

3.1 Warm-up: Expanded Multilinear Polynomials

As a warm-up, let us first suppose the polynomials P_j are multilinear and given in their expanded form: $P_j \in \mathbb{Z}_p[T_1, \ldots, T_n]$ and

$$P_j(\vec{T}) = \sum_{i \in \{0,1\}^n} \alpha_{j,i} T_1^{i_1} \cdots T_n^{i_n}.$$

There are 2^n monomials $T_1^{i_1} \cdots T_n^{i_n}$, even in that restricted case. So we need to suppose that either n is logarithmic in the security parameter, or, more generally, only a polynomial (in the security parameter) number of $\alpha_{j,i}$ are non-zero.

Let us now prove the LIP security of these polynomials. In the real case, we have:

$$y_j = [P_j(\vec{a})a'] = \left[\sum_{i \in \{0,1\}^n} \alpha_{j,i} a_1^{i_1} \cdots a_n^{i_n} a' \right] = \prod_{i \in \{0,1\}^n} \mathsf{NR}((a', \vec{a}), i)^{\alpha_{j,i}}, \tag{2}$$

where $\mathsf{NR}((a', \vec{a}), i) = \left[a' \prod_{k=1}^n a_k^{i_k}\right]$ (for $i \in \{0,1\}^n$). NR is a secure PRF under the DDH assumption, meaning that all the values $\mathsf{NR}((a', \vec{a}), i)$ for all $i \in \{0,1\}^n$ look independent and uniformly random. Let us write $\vec{U}$ the column vector, with rows indexed by $i \in \{0,1\}^n$, containing all the discrete logarithm of these values, i.e., $u_i = a' \prod_{k=1}^n a_k^{i_k}$. Let us also write $\boldsymbol{M}$ the $q \times 2^n$ matrix, with columns indexed by $i \in \{0,1\}^n$, defined by $m_{j,i} = \alpha_{j,i}$. Then we can rewrite Equation (2) as:

$$\begin{pmatrix} y_1 & \ldots & y_q \end{pmatrix}^\mathsf{T} = \left[\boldsymbol{M} \cdot \vec{U} \right].$$

Since the polynomials P_j are linearly independent, the rows of $\boldsymbol{M}$ are linearly independent. Therefore, as $\left[\vec{U}\right]$ looks uniformly random in $\mathbb{G}^{2^n}$, $(y_1, \ldots, y_q)$ looks like a uniformly random tuple in $\mathbb{G}^q$. This proves the result of the LIP theorem in this multilinear case with expanded polynomial. Extending this result to non multilinear polynomial would just require slightly changing the assumption, as long as polynomials are given in their expanded form.

This result is already very useful. We will see in Section 4 that it enables to prove the security of the Naor-Reingold PRF and variants thereof.

Challenges for its Extension. Unfortunately, for certain settings such as those considered in the context of related-key security, or even for the Boneh-Montgomery-Raghunathan PRF [BMR10], we cannot have polynomials in an expanded form, but only as a polynomial-size (in the number n of indeterminates and the maximum degree d in each indeterminate) formula (given by an abstract tree).[1] The problem is that the expanded version of these polynomials may be exponentially large. For example, $(T_1 + 1) \cdots (T_n + 1)$ has 2^n monomials.

Therefore, the main challenge is to prove the theorem without expanding the polynomials. This requires a much more subtle proof that we sketch here. This first idea is the following: instead of replacing all monomials by independent random values at once, we first fix all values $T_2, \ldots, T_n$ to randomly chosen $a_2, \ldots, a_n$, and get polynomials in T_1 only. These polynomials can be expanded without an exponential blow-up, and each monomial $T_1, T_1^2, \ldots$ can be replaced by an independent random value (instead of $a_1, a_1^2, \ldots$ for some value a_1). Then, we can fix only $T_3, \ldots, T_n$ to randomly chosen $a_3, \ldots, a_n$, get a polynomial in T_1 and T_2, and replace all distinct monomial $(T_1, T_1^2, T_1T_2, T_2^2, \ldots)$ by independent random values. And we can continue like that until all monomials are replaced.

Obviously, if we do that so naively, we get back to the original problem: we have an exponential number of monomials. The second idea is to remark that we actually do not need to expand polynomials to replace all distinct monomials by random values and get the result, at each step of the previous

[1] Details on the representation of polynomials are given in Appendix B.1.

proc Initialize	**proc Pl**(P)	**proc Finalize**(b')
$\vec{a} \xleftarrow{\$} \mathbb{Z}_p^n$; $a' \xleftarrow{\$} \mathbb{Z}_p$	If $b = 0$ then $y \leftarrow [P(\vec{a}) \cdot a']$	Return $b' = b$
$b \xleftarrow{\$} \{0,1\}$	Else $y \xleftarrow{\$} \mathbb{G}$	
	Return y	

Figure 1: Game defining the (n, d)-LIP security for a group $\mathbb{G}$

idea. We could just assign random values to all polynomials (after fixing $T_{i+1}, \ldots, T_n$ to $a_{i+1}, \ldots, a_n$), if they are all linearly independent: this is exactly what we showed in the previous proof for expanded polynomials. And if they are not all linearly independent, we just need to take care of linear combinations, and compute the resulting value accordingly.

More precisely, for any polynomial P, let us write $Q_P \in \mathbb{Z}_p[T_1, \ldots, T_i]$ the polynomial obtained after fixing $T_{i+1}, \ldots, T_n$ to $a_{i+1}, \ldots, a_n$. To answer the j-th query P_j, we check whether Q_{P_j} is linearly independent from $(Q_{P_l})_{l=1,\ldots,j-1}$. If that is the case, we answer with an independent random value y_j. Otherwise, we find some linear combination between Q_{P_j} and $(Q_{P_l})_{l=1,\ldots,j-1}$, and we write $Q_{P_j} = \sum_{l=1}^{j-1} \lambda_l Q_{P_l}$ and outputs $\prod_{l=1}^{j-1} y_l^{\lambda_l}$, with y_l the output given for P_l.

The last difficulty is that this proof requires a test of linear dependence of multivariate polynomials. One way to do that would be to expand them, which is exactly what we are trying to avoid. So, instead, we use a statistical test based on the Schwartz-Zippel lemma, which basically consists in evaluating the polynomials in enough random points and looking for linear combination among the vectors of these evaluations.

3.2 Main Theorem: LIP Security

LIP **Security.** Let $\mathbb{G} = \langle g \rangle$ be a group of prime order p. We define the advantage of an adversary $\mathscr{A}$ against the (n, d)-LIP security of $\mathbb{G}$, denoted $\mathbf{Adv}_{\mathbb{G}}^{(n,d)\text{-lip}}(\mathscr{A})$ as the probability of success in the game defined in Figure 1, with $\mathscr{A}$ being restricted to make queries $P \in \mathbb{Z}_p[T_1, \ldots, T_n]$ such that for any query P, the maximum degree in one indeterminate in P is at most d, and for any sequence $(P_1, \ldots, P_q)$ of queries, the polynomials $(P_1, \ldots, P_q)$ are always *linearly independent* over $\mathbb{Z}_p$. Another way to look at the security definition is to consider that when $b = 0$, $\mathbf{Pl}(P)$ outputs $[P(\vec{a})]_h = [P(\vec{a}) \cdot a']_g$, where the generator is $h = [a']_g$, which is not public (but can be obtained by querying the polynomial 1), and g is a public generator.

Theorem 3.1 (LIP)**.** *Let $\mathbb{G} = \langle g \rangle$ be a group of prime order p. Let $\mathscr{A}$ be an adversary against the (n, d)-LIP security of $\mathbb{G}$ that makes q oracle queries $P_1, \ldots, P_q$. Then we can design an adversary $\mathscr{B}$ against the $\mathcal{E}_{1,d}$-MDDH problem in $\mathbb{G}$, such that $\mathbf{Adv}_{\mathbb{G}}^{(n,d)\text{-lip}}(\mathscr{A}) \leq n \cdot d \cdot \mathbf{Adv}_{\mathbb{G}}^{\mathcal{E}_{1,d}\text{-mddh}}(\mathscr{B}) + O(ndq/p)$. The running time of $\mathscr{B}$ is that of $\mathscr{A}$ plus the time to perform a polynomial number (in q, n, and d) of operations in $\mathbb{Z}_p$ and $\mathbb{G}$.*

The proof is detailed in Appendix C.

4 Recovering and Extending Existing Number-Theoretic PRFs

In Table 2, we recall known number-theoretic PRFs, namely the Naor-Reingold (NR) PRF [NR97], its variant NR* defined in [BC10], and the algebraic PRF by Boneh, Montgomery, and Raghunathan (BMR) in [BMR10]. We also introduce weighted (extended) versions of these PRFs, namely weighted NR (WNR) and weighted BMR (WBMR), in order to construct RKA-secure PRFs for new classes of RKD functions (Section 5). These weighted PRFs are obtained by applying particular permutations to the key space. Then, as PRFs, it is straightforward that the security of NR and BMR implies the security of their weighted versions. However, as detailed in Section 5, in the RKA setting, we can prove that some of these weighted PRFs are secure against certain classes of RKD functions while both NR and BMR are not, even if we apply the BC/ABPP frameworks.

Using the LIP theorem and changing the generators used (to get PRFs of the form $F(\vec{a}, x) = [P_x(\vec{a}) \cdot a']$), the security proof of WNR and WBMR is straightforward, and so is the security proof of NR, NR*, and BMR, as particular cases of WNR and WBMR. Concretely, for WBMR$^{\vec{w}}$, we start by revealing

Table 2: Existing Number-Theoretic PRFs and their Weighted Extensions

PRF F	Key $\vec{a}$ Key domain $\mathcal{K}$	Domain $\mathcal{D}$	Output	$\mathbf{Adv}^{\mathsf{prf}}_F \lesssim$
NR	$(a_0,\ldots,a_n)$ $\mathcal{K}=\mathbb{Z}_p^{n+1}$	$\{0,1\}^n$	$\left[a_0\prod_{i=1}^n a_i^{x_i}\right]$	$n\cdot\mathbf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}$
NR*	$(a_1,\ldots,a_n)$ $\mathcal{K}=\mathbb{Z}_p^{n}$	$\{0,1\}^n\setminus\{0^n\}$	$\left[\prod_{i=1}^n a_i^{x_i}\right]$	$n\cdot\mathbf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}$
BMR	$(a_1,\ldots,a_n)$ $\mathcal{K}=\mathbb{Z}_p^{n}$	$\{0,\ldots,d\}^n$	$\left[\prod_{i=1}^n \frac{1}{a_i+x_i}\right]$	$nd\cdot\mathbf{Adv}^{d\text{-}\mathsf{ddhi}}_{\mathbb{G}}$
$\mathsf{WNR}^{\vec{w}}$ $(\vec{w}\in\mathbb{Z}_p^{n+1})^*$	$(a_0,\ldots,a_n)$ $\mathcal{K}=\mathbb{Z}_p^{n+1}$	if $w_0\neq 0$: $\{0,1\}^n$, else: $\{0,1\}^n\setminus\{0^n\}$	$\left[a_0^{w_0}\prod_{i=1}^n a_i^{w_i x_i}\right]$	$n\cdot\mathbf{Adv}^{\mathsf{ddh}\dagger}_{\mathbb{G}}$
$\mathsf{WBMR}^{\vec{w}}$ $(\vec{w}\in\mathbb{Z}_p^{n})^\ddagger$	$(a_1,\ldots,a_n)$ $\mathcal{K}=\mathbb{Z}_p^{n}$	$\{0,\ldots,d\}^n$	$\left[\prod_{i=1}^n \frac{1}{a_i+w_i+x_i}\right]$	$nd\cdot\mathbf{Adv}^{d\text{-}\mathsf{ddhi}}_{\mathbb{G}}$

$\mathbb{G}=\langle g\rangle$ is a prime order group, and g is the generator used for the PRF construction;
The last column show approximate simplified bounds on the advantage $\mathbf{Adv}^{\mathsf{prf}}_F$ of a polynomial-time adversary against the security of the PRF F; exact bounds can be found in Appendix D;
Remarks: $\mathsf{NR}=\mathsf{WNR}^{(1,\ldots,1)}$, $\mathsf{NR}^*=\mathsf{WNR}^{(0,1,\ldots,1)}$, and $\mathsf{BMR}=\mathsf{WBMR}^{(0,\ldots,0)}$;
* for WNR, weights are $\vec{w}=(w_0,\ldots,w_n)\in\mathbb{Z}_p^{n+1}$;
† when $w_1,\ldots,w_n$ are coprime to $p-1$, and w_0 is 0 or coprime to $p-1$;
‡ for WBMR, weights are $\vec{w}=(w_1,\ldots,w_n)\in\mathbb{Z}_p^{n}$.

the generator h to the adversary where

$$h=\left[\left(\prod_{i=1}^n\prod_{k\in\{0,\ldots,d\}}(a_i+w_i+k)\right)\cdot a'\right]_g=[P(\vec{a})\cdot a']_g$$

which is a generator with overwhelming probability. Then, when the adversary makes a query x, it is clear that

$$\left[\prod_{i=1}^n\frac{1}{a_i+w_i+x_i}\right]_h=\left[\left(\prod_{i=1}^n\prod_{k\in\{0,\ldots,d\}\setminus\{x_i\}}(a_i+w_i+k)\right)\cdot a'\right]_g=[P_x(\vec{a})\cdot a']_g$$

As each polynomial P_x is null on every input $-x'$ for $x'\in\{0,\ldots,d\}^n$, seen as a vector of $\mathbb{Z}_p^n$, except when $x'=x$, and as P is null on all $-x'$, P and $(P_x)_x$ are linearly independent. Then, we conclude the security proof of $\mathsf{WBMR}^{\vec{w}}$ by applying the LIP theorem. Formal proofs are provided in Appendix D (Lemmas D.1 and D.2).

5 Application to Related-Key Security

In this section, we show how our theorem can be used to build RKA-secure PRFs from a PRF F defined over a prime order group $\mathbb{G}=\langle g\rangle$ that takes a key $\vec{a}$ and an input x and outputs a group element $F(\vec{a},x)=[P_x(\vec{a})]$. Let Φ be a class of RKD functions, where functions $\vec{\phi}=(\phi_1,\ldots,\phi_n)\in\Phi$ are such that ϕ_i are multivariate polynomials in $\mathbb{Z}_p[T_1,\ldots,T_n]$. Then, for an RKD function $\vec{\phi}$ and an input x, the PRF outputs $F(\vec{\phi}(\vec{a}),x)=\left[P_{\vec{\phi},x}(\vec{a})\right]$, where the polynomial $P_{\vec{\phi},x}(\vec{T})=P_x(\vec{\phi}(\vec{T}))=P_x(\phi_1(\vec{T}),\ldots,\phi_n(\vec{T}))$ depends on $\vec{\phi}$ and x. In particular, $P_{\mathsf{id},x}=P_x$ for all x, where id is the identity function.

When all polynomials $P_{\vec{\phi},x}$ and the constant polynomial 1 are linearly independent, the LIP theorem directly shows that F is Φ-RKA-secure. To illustrate this, we construct in Section 5.1 a PRF that is secure against permutations of the secret key using this method.

However, to assume that all polynomials $P_{\vec{\phi},x}$ are linearly independent is a very strong property and, in general, this is not the case for all x and $\vec{\phi}$. Hence, in Section 5.2, we consider the less restrictive case where the polynomials $P_{\vec{\phi}_1,x_1},\ldots,P_{\vec{\phi}_q,x_q}$ are linearly independent as long as the inputs $x_1,\ldots,x_q$ are distinct (in which case the adversary is said to be unique-input). More precisely, we first design a new

algebraic framework that extends the one from [ABPP14], when the PRF F is of the form $[P_x(\vec{a})]$ and the RKD functions are multivariate polynomials, and then use it to construct RKA-secure PRFs from F for new and larger classes of RKD functions.

5.1 Direct Constructions of RKA-Secure PRFs

In this section, we show how the LIP theorem can be used to prove the Φ-RKA-PRF security in the particular case where all polynomials $P_{\vec{\phi},x}$ are linearly independent, for any $\vec{\phi} \in \Phi$ and any input x.

Specifically, we consider the class $\Phi_{\mathfrak{S}_n}$ of functions defined as $\{\sigma \mid \sigma \in \mathfrak{S}_n\}$ such that, applying a function $\sigma \in \Phi_{\mathfrak{S}_n}$ to a key $\vec{a} = (a_1, \ldots, a_n) \in \mathbb{Z}_p^n$ leads to the key $\sigma(\vec{a}) = (a_{\sigma^{-1}(1)}, \ldots, a_{\sigma^{-1}(n)})$, so the i-th component of $\vec{a}$ becomes the $\sigma(i)$-th component of the key $\sigma(\vec{a})$.

It is clear that BMR is not $\Phi_{\mathfrak{S}_n}$-RKA-secure, since we can distinguish BMR from a random function with only 2 queries. Indeed, let id be the identity function and (12) be the permutation which switches the first two components of the key. Then, one can just first query $(\mathsf{id}, 100\ldots0)$ and $((12), 010\ldots0)$ and check whether the output of these queries are the same, which is the case in the real case while they are independent in the random case. However, we show in what follows that a particular case of WBMR, defined below, is a $\Phi_{\mathfrak{S}_n}$-RKA-secure PRF.

Linear WBMR PRF. We define $\mathsf{WBMR}^{\mathsf{lin}}$ as the particular case of WBMR, where $w_i = (i-1)(d+1)$, for $i = 1, \ldots, n$. Please refer to Table 2 for details.

Theorem 5.1. *Let $\mathbb{G} = \langle g \rangle$ be a group of prime order p and let $\mathsf{WBMR}^{\mathsf{lin}}$ be the function defined above. Then we can reduce the $\Phi_{\mathfrak{S}_n}$-RKA-PRF security of $\mathsf{WBMR}^{\mathsf{lin}}$ to the hardness of the $(n(d+1)-1)$-DDHI problem in $\mathbb{G}$, with a loss of a factor $n(n(d+1)-1)$. Moreover, the time overhead of this reduction is polynomial in n, d and in the number of queries made by the adversary.*

The proof is given in Appendix E.1 and is very similar to the proof of security of WBMR sketched in Section 4. The construction can actually be extended to also tolerates small additive factors in addition to permutations (see Remark E.3 on page 27).

5.2 Constructions via Unique-Input RKA-Secure PRFs

In this section, we address the less restrictive case where the polynomials $P_{\vec{\phi}_1,x_1}, \ldots, P_{\vec{\phi}_q,x_q}$ are linearly independent for any $\vec{\phi}_1, \ldots, \vec{\phi}_q$ only when the inputs $x_1, \ldots, x_q$ are all distinct. Please notice that this is the case for all the classes considered in [BC10] and [ABPP14]. We now denote by M the "original" PRF: $M(\vec{a}, x) = [P_x(\vec{a})]$.

In order to build RKA-secure PRFs from such PRFs, we would like to apply the ABPP generic framework [ABPP14] that allows to transform a PRF M which is RKA-secure with respect to unique-input adversaries (UI-RKA-secure) into an RKA-secure PRF F, when M is key-collision and statistical-key-collision secure. The latter means that it is hard to find two functions $\phi_1, \phi_2 \in \Phi$ such that $\phi_1(K) = \phi_2(K)$, even with access to an oracle $(\phi, x) \mapsto f(\phi(K), x)$, when $f = M$ (key-collision security), and when f is a random function (statistical key-collision security). The framework consists in transforming this UI-RKA-secure PRF M into an RKA-secure PRF F, as follows:

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

where H is a compatible collision-resistant hash function, and the vector $\vec{\omega}$ is a strong key fingerprint, meaning that it is a vector of inputs such that the vector of outputs $M(K, \vec{\omega})$ completely defines K (recall that $M(K, \vec{\omega}) = (M(K, \omega_1), \ldots, M(K, \omega_{|\vec{\omega}|}))$. As defined in [BC10], a hash function is said to be compatible if it guarantees that the inner calls to M in the construction above will never collide with the outer calls to M even under related keys.

Unfortunately, if we consider the PRF $\mathsf{WNR}^{\vec{w}}$ with some $w_i > 1$, then it is not clear how to find a strong key fingerprint, which can be used to apply the ABPP framework. Furthermore, this ABPP framework requires to prove several non-algebraic properties (statistical or computational), namely key-collision, statistical-key-collision, and UI-RKA securities.

For this reason, we design a new algebraic framework, that generalizes the ABPP framework in the particular case of PRFs of the shape $M(\vec{a}, x) = [P_x(\vec{a})]$ and of RKD functions which are multivariate polynomials. For completeness, a more general framework, which does not make any assumptions about the shape of a PRF, is also given in Appendix G. Afterwards, we use our algebraic framework to design

new RKA-secure PRFs based on WNR for larger classes for which previous constructions from [BC10] and [ABPP14] are not secure.

An Algebraic Framework for Related-Key Security. Here, we describe a new framework that transforms any PRF that satisfies that $P_{\vec{\phi}_1,x_1},\ldots,P_{\vec{\phi}_q,x_q}$ are linearly independent, for any $\vec{\phi}_1,\ldots,\vec{\phi}_q$ as long as $x_1,\ldots,x_q$ are all distinct inputs, into a RKA-secure PRF. To do so, we first introduce three new notions, termed *algebraic fingerprint*, *helper information*, and *expansion function*, and defined as follows.

Group Generator. In this framework and its applications, we assume for simplicity that the generator used in the PRF construction, that is revealed to the adversary, is $[a']$.

Algebraic Fingerprint. In order to overcome the eventual lack of a strong key fingerprint, we introduce algebraic fingerprint, which will be used to replace $M(K,\vec{\omega})$ in the construction in [ABPP14], where $\vec{\omega}$ is a strong fingerprint. An algebraic fingerprint is simply an injective function $\vec{\Omega}\colon \mathbb{Z}_p^n \to \mathbb{G}^m$ such that the image $\vec{\Omega}(\vec{a})$ is a vector of group elements $([\Omega_1(\vec{a})a'],\ldots,[\Omega_m(\vec{a})a'])$ with $\Omega_1,\ldots,\Omega_m$ being polynomials in $\mathbb{Z}_p[T_1,\ldots,T_n]$ and $a'\in\mathbb{Z}_p$. In our applications, we will simply have $\vec{\Omega}(\vec{a}) = ([a_1a'],\ldots,[a_na'])$, so $m=n$ and $\Omega_i(\vec{T})=T_i$ for $i=1,\ldots,n$.

Helper Information. In order to prove the security of our framework, we need to be able to compute the image of the algebraic fingerprint, $\vec{\Omega}(\vec{\phi}(\vec{a})) = ((\Omega_1\circ\vec{\phi})(\vec{a}),\ldots,(\Omega_m\circ\vec{\phi})(\vec{a}))$, for any related key $\vec{\phi}(\vec{a})\in\mathbb{Z}_p^n$, with $\vec{\phi}\in\Phi$, from some information which can somehow be made public without hurting security. We call this information a helper information, write it $\mathsf{Help}_\Phi(\vec{a})$, and call Help_Φ the helper function. We suppose that $\mathsf{Help}_\Phi(\vec{a}) = ([\mathsf{help}_1(\vec{a})a'],\ldots,[\mathsf{help}_l(\vec{a})a'])$, with $\mathsf{help}_1,\ldots,\mathsf{help}_l$ linearly independent polynomials which generate a vector subspace of $\mathbb{Z}_p[T_1,\ldots,T_n]$ containing the polynomials $\Omega_i\circ\vec{\phi}$ for $i=1,\ldots,m$, and $\vec{\phi}\in\Phi$.

Hash Function and Expansion Function. Let $\overline{\mathcal{D}} = \mathcal{D}\times\mathbb{G}^m$ where $\mathcal{D}$ is the domain of the PRF M, and let h be a collision-resistant hash function $h\colon \overline{\mathcal{D}}\to\mathsf{hSp}$ (definition recalled in Appendix A.1), where hSp is a large enough space. The last thing we need to define is an expansion function, which is simply an injective function $\mathsf{E}\colon \mathsf{hSp}\to\mathcal{S}\subseteq\mathcal{D}$ such that for any sequence $(\vec{\phi}_1,x_1),\ldots,(\vec{\phi}_q,x_q)$ where $x_1,\ldots,x_q$ are distinct inputs in $\mathcal{S}$ and $\vec{\phi}_1,\ldots,\vec{\phi}_q$ are RKD functions, polynomials $\mathsf{help}_1,\ldots,\mathsf{help}_l$ and polynomials $P_{\vec{\phi}_1,x_1},\ldots,P_{\vec{\phi}_q,x_q}$ and 1 (which needs to be queried to define the generator $[a']$) are linearly independent over $\mathbb{Z}_p$ (in particular, E has to be injective).

Using these new tools, we obtain the following framework.

Theorem 5.2. *Let $\mathbb{G}$ be a group of prime order p. We use the above definitions, with $M\colon \mathbb{Z}_p^n\times\mathcal{D}\to\mathbb{G}$ defined by $M(\vec{a},x)=[P_x(\vec{a})]$. Let d be a upper bound for the maximum degree in any indeterminate of polynomials in $\{\mathsf{help}_1,\ldots,\mathsf{help}_l\}\cup\{P_{x,\vec{\phi}}\mid x\in\mathcal{S},\vec{\phi}\in\Phi\}$. Define $F\colon \mathbb{Z}_p^n\times\mathcal{D}\to\mathbb{G}$ by*

$$F(\vec{a},x) = M(\vec{a},\mathsf{E}(h(x,\vec{\Omega}(\vec{a}))))$$

for all $\vec{a}\in\mathbb{Z}_p^n$ and $x\in\mathcal{D}$. Then, we can reduce the Φ-RKA-PRF security of F to the (n,d)-LIP security, the collision-resistance security of h without any loss, and to the d-SDL assumption with a loss of a factor $2n$. The running time overhead of this reduction is polynomial in n,d and q.

Proof Overview. The proof of the above theorem is detailed in Appendix E.2 and relies on the sequence of 10 games (games $\mathrm{G}_0-\mathrm{G}_9$) described in Figure 7 and on Lemma E.4. We first prove an intermediate statement whose proof is very similar to the proof of Theorem 3.1 from [ABPP14], under a notion termed extended key-collision security (that states the hardness of finding key collisions given access to PRF values and helper information) which is defined in the appendix. Afterwards, we reduce this notion to the hardness of the SDL in $\mathbb{G}$. Here we provide a brief overview of the proof of the intermediate statement.

We start by giving the generator used for the PRF by querying polynomial 1. Hence, the generator is simply $[a']$. Since we may have key collisions (i.e., two RKD functions $\phi_1\neq\phi_2$, such that $\phi_1(\vec{a})=\phi_2(\vec{a})$), we start by dealing with possible collisions on the related keys in the RKAPRFReal case, using the extended key-collision notion (games $\mathrm{G}_0-\mathrm{G}_2$). These claws can be detected by looking for collisions on images of $\vec{\Omega}$ for different RKD functions.

Then, in games $\mathrm{G}_3-\mathrm{G}_4$, we deal with possible collisions on hash values in order to ensure that the inputs $t=\mathsf{E}(h(x,\vec{\Omega}(\vec{a})))$ used to compute the output y are distinct (recall that E is injective).

Then, we use the (n,d)-LIP security notion to show that it is hard to distinguish the output of F and the helper information from uniformly random values (games $\mathrm{G}_5-\mathrm{G}_6$).

Finally, we use once again the extended-key-collision security notion to deal with possible key collisions in the RKAPRFRand case (games $G_7 - G_9$) so that G_9 matches the description of the RKAPRFRand game. These key collisions can still be detected in these games by making crucial use of the helper information.

RKA-PRFs for Permutations of Univariate Polynomial Functions. We now apply our framework to a particular case of WNR and build the first RKA-secure PRF secure against permutations of univariate polynomials. We chose to set w_0 to 0 in our construction in order to ease the readability so that the key space of the PRF stays $\mathbb{Z}_p^n$, but similar results can be proven with $w_0 = 1$ or set to a prime number $p_0 > d$ (and distinct to $p_1, \ldots, p_n$ defined below).

For $d \geq 1$, let Φ_d be the class of degree at most d non-constant univariate polynomials defined as $\Phi_d = \{\vec{\phi}\colon\ \mathbb{Z}_p^n \to \mathbb{Z}_p^n \mid \phi_i : \vec{T} \mapsto \sum_{j=0}^{d} \alpha_{i,j} T_i^j, (\alpha_{i,1}, \ldots, \alpha_{i,d}) \neq 0^d, \forall i = 1, \ldots, n\}$. Then we consider the class $\Phi_{\mathfrak{S}_n,d}$ of permutations of degree at most d non-constant univariate polynomials, defined as follows:

$$\Phi_{\mathfrak{S}_n,d} = \{\sigma \circ \vec{\phi} \mid (\sigma, \vec{\phi}) \in \mathfrak{S}_n \times \Phi_d\} .$$

For a key $\vec{a} = (a_1, \ldots, a_n) \in \mathbb{Z}_p^n$, applying an RKD function $\sigma \circ \vec{\phi} \in \Phi_{\mathfrak{S}_n,d}$, where $\vec{\phi} = (\phi_1, \ldots, \phi_n)$ leads to the key $(\phi_{\sigma^{-1}(1)}(\vec{a}), \ldots, \phi_{\sigma^{-1}(n)}(\vec{a})) \in \mathbb{Z}_p^n$, so the i-th component a_i of the key is changed into $\phi_i(\vec{a})$ and becomes the $\sigma(i)$-th component of the related key.

Before explaining our construction, we would like to point out that, even if we just consider the simple class of permutations $\Phi_{\mathfrak{S}_n} \subset \Phi_{\mathfrak{S}_n,1}$ introduced in Section 5.1, we can already show that NR and NR* are not $\Phi_{\mathfrak{S}_n}$-RKA secure, even with respect to unique-input adversaries.

Indeed, let us consider NR*: let id be the identity function and (12) be the permutation which switches the first two components of the key. Then, the output of the queries $(\mathsf{id}, 100\ldots0)$ and $((12), 010\ldots0)$ will be the same in the real case and independent in the random case.

In fact, we can generalize the attack above to show that there even exists a compatible collision-resistant hash function h such that the PRF that one obtains when applying the Bellare-Cash (or ABPP) transform to NR* would not be RKA-secure with respect to the class of permutations. Indeed, let h' be a collision-resistant hash function. The counter-example for h could be as follows (where x_1 and x_2 are two arbitrary distinct inputs):

$$h(x, [a_1], \ldots, [a_n]) = \begin{cases} 1110 \,\|\, h'(x_1, [a_1], \ldots, [a_n]) & \text{if } x = x_1 \\ 1101 \,\|\, h'(x_1, [a_2], [a_1], [a_3], \ldots, [a_n]) & \text{if } x = x_2 \\ 1111 \,\|\, h'(x, [a_1], \ldots, [a_n]) & \text{otherwise.} \end{cases}$$

Note that h is a compatible collision-resistant hash function. It is easy to see that the output of the queries (id, x_1) and $((12), x_2)$ will be the same in the real case and independent in the random case. The same kind of attack can be mounted against NR.

However, while NR and NR* are not RKA-secure against permutations attacks, we show in what follows that a particular case of WNR, defined below, yields a $\Phi_{\mathfrak{S}_n,d}$-RKA-secure PRF.

d-Linear Weighted NR PRF. Let $d \geq 1$. Let $p_1 < p_2 < \cdots < p_n$ be distinct prime numbers such that $p_1 > d$. We define WNR$^{d\text{-lin}}$ as the particular case of WNR, where $w_0 = 0$ and $w_i = p_i$. Please refer to Table 2 for details. Using standard inequalities over prime numbers, it is easy to see that we can find $p_1, \ldots, p_n$ such that $p_n = \tilde{O}(d + n)$.

In order to apply the framework from Theorem 5.2 to WNR$^{d\text{-lin}}$ and $\Phi_{\mathfrak{S}_n,d}$, we define:

- $[a'] \in \mathbb{G}$ is the generator used for the PRF construction
- $\vec{\Omega}\colon\ \vec{a} \in \mathbb{Z}_p^n \mapsto ([a_1 a'], \ldots, [a_n a']) \in \mathbb{G}^n$
- $\mathsf{Help}_{\Phi_{\mathfrak{S}_n,d}}\colon\ \vec{a} \in \mathbb{Z}_p^n \mapsto ([a'], [a_1 a'], \ldots, [a_1^d a'], \ldots, [a_n a'], \ldots, [a_n^d a']) \in \mathbb{G}^{nd+1}$
- h can be any collision-resistant hash function $h\colon\ \{0,1\}^n \times \mathbb{G}^n \to \{0,1\}^{n-2}$
- E$\colon\ z \in \{0,1\}^{n-2} \mapsto 11 \,\|\, z \in \{0,1\}^n$

We just need to prove that E satisfies the linear independence property required to apply the framework, which is done in Appendix E.3, and sketched here. We order monomials of multivariate polynomials, with any order respecting the total degree of polynomials (e.g., the graded lexicographic order). The leading monomial (i.e., the first monomial for that order) of the polynomial $P_{\vec{\phi},x}$ is $T_1^{x_{\sigma(1)} p_{\sigma(1)} d_1} \cdots T_n^{x_{\sigma(n)} p_{\sigma(n)} d_n}$, with $d_i > 0$ the degree of ϕ_i. The polynomials for the helper information (help_k) are T_i^j. Therefore, the

leading monomials of $\mathsf{help}_1, \ldots, \mathsf{help}_l, P_{\vec{\phi}_1,x_1}, \ldots, P_{\vec{\phi}_q,x_q}, 1$ are all distinct, when $x_1, \ldots, x_q$ are distinct inputs. This means that the matrix whose columns correspond to monomials (ordered as specified above) and whose rows correspond to the polynomials $\mathsf{help}_1, \ldots, \mathsf{help}_l, P_{\vec{\phi}_1,x_1}, \ldots, P_{\vec{\phi}_q,x_q}, 1$ (ordered according to their leading monomial) is in echelon form. Hence, the latter polynomials are linearly independent. Finally, by combining Theorem 5.2 and the LIP theorem, we obtain the following theorem.

Theorem 5.3. *Let $\vec{\Omega}$, h and E be defined as above. Define $F\colon \mathbb{Z}_p^n \times \{0,1\}^n \to \mathbb{G}$ by $F(\vec{a}, x) = \mathsf{WNR}^{d\text{-}\mathsf{lin}}(\vec{a}, \mathsf{E}(h(x, \vec{\Omega}(\vec{a}))))$, for all $\mathbf{a} \in \mathbb{Z}_p^n$ and $x \in \{0,1\}^n$. Then we can reduce the $\Phi_{\mathfrak{S}_n,d}$-RKA-PRF security of F to the hardness of the $p_n d$-DDHI problem in $\mathbb{G}$ and the $p_n d$-SDL problem in $\mathbb{G}$, respectively with a loss of a factor np_nd and of a factor n, and to the* CR *security of h. Moreover, the time overhead of this reduction is polynomial in n, d, p_n and in the number of queries made by the adversary attacking the $\Phi_{\mathfrak{S}_n,d}$-RKA-PRF security of F.*

6 Extension to PRFs in Symmetric Bilinear Groups

6.1 High-Level Overview of Existing Constructions and Challenges

All the previous constructions (of classical PRF and RKA-secure PRF) require at least DDH to hold. In particular, they are insecure if there exists a symmetric pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. In this section, we investigate how to adapt our linearly independent polynomials framework and the corresponding LIP theorem to handle constructions of PRFs under weaker assumptions, which may hold in symmetric bilinear groups.

The first algebraic PRF based on DLin is the Lewko-Waters PRF [LW09], which is defined as follows:

$$\mathsf{LW}(\vec{\boldsymbol{A}}, x) = \left[\prod_{i=1}^{n} \boldsymbol{A}_i^{x_i} \cdot \boldsymbol{A}'\right],$$

with $\vec{\boldsymbol{A}} = (\boldsymbol{A}_1, \ldots, \boldsymbol{A}_n)$ being a vector of n uniformly random matrices in $\mathbb{Z}_p^{2\times 2}$ and $\boldsymbol{A}'$ a uniformly random matrix in $\mathbb{Z}_p^{2\times m}$, for some $m \geq 1$. $\boldsymbol{A}'$ was actually in $\mathbb{Z}_p^{2\times 1}$ (i.e., $m = 1$) in [LW09] (with only the first group element being returned). This PRF is secure under DLin, and even under a weaker assumption, namely the $\mathcal{U}_2$-MDDH-assumption of Escala et al. [EHK+13]. In the latter paper, this PRF is extended to any MDDH-assumption, which particularly encompasses DDH and DLin. These instantiations differ by the size of the matrices and their distribution. Except for constructions using multilinear maps and lattices [BLMR13, BP14] or trivial variants, we are not aware of any other construction.

Commutation Challenge From a high level point of view, these PRFs are very similar to the one considered in our algebraic framework in Section 3, except elements of keys are now matrices. Unfortunately, matrices do not commute in general, and this lack of commutativity makes everything more complex.

One naive solution would be to extend the LIP theorem by considering non-commutative polynomials, or in other words elements of the free algebra $\mathbb{Z}_p\langle T_1, \ldots, T_n\rangle$. In this algebra, for example, T_1T_2 and T_2T_1 are distinct and linearly independent elements. The problem is that, as proven by Amitsur and Levitzki [AL50], for any matrices $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_4 \in \mathbb{Z}_p^{2\times 2}$, $\sum_{\sigma\in\mathfrak{S}_4} \mathrm{sgn}(\sigma) \cdot \boldsymbol{A}_{\sigma(1)} \cdot \boldsymbol{A}_{\sigma(2)} \cdot \boldsymbol{A}_{\sigma(3)} \cdot \boldsymbol{A}_{\sigma(4)} = 0$, with $\mathrm{sgn}(\sigma)$ being the parity of the permutation σ. Thus, while the family of non-commutative polynomials $(P_\sigma = T_{\sigma(1)}T_{\sigma(2)}T_{\sigma(3)}T_{\sigma(4)})_{\sigma\in\mathcal{S}_4}$ is linearly independent in the free algebra, the PRF of domain $\mathcal{D} = \mathcal{S}_4$, the PRF defined by $F(\vec{\boldsymbol{A}}, \sigma) = \left[\boldsymbol{A}_{\sigma(1)}\boldsymbol{A}_{\sigma(2)}\boldsymbol{A}_{\sigma(3)}\boldsymbol{A}_{\sigma(4)}\boldsymbol{A}'\right]$ would clearly be insecure.

Assumption Challenge and Generic Symmetric Bilinear Group The second challenge is to prove the hardness of the $\mathcal{E}_{2,d}$-MDDH assumption in the generic bilinear group, which is done in Appendix H.3, using a non-trivial technical lemma: Lemma H.3. Notably, contrary to the cyclic group case, it is not straightforward to check whether a PRF defined by $F(\vec{\boldsymbol{A}}, x) = \left[P_x(\vec{\boldsymbol{A}}) \cdot \boldsymbol{A}'\right]$ is secure in the generic bilinear group model, where $(P_x)_{x\in\mathcal{D}}$ is a family of non-commutative polynomials, $\vec{\boldsymbol{A}}$ is a vector of matrices from $\mathbb{Z}_p^{2\times 2}$, and $\boldsymbol{A}'$ is a matrix from $\mathbb{Z}_p^{2\times m}$, for some $m \geq 1$.

6.2 Generalized Polynomial Framework

Let us show how we address these challenges.

Generalized Polynomial (GP) Security. Let us introduce the (k, n, d)-GP security of a cyclic group $\mathbb{G} = \langle g\rangle$ as a generalization of the (n, d)-LIP security in Section 3.2, where the secret scalar $a' \xleftarrow{\$} \mathbb{Z}_p$

and the secret vector of scalars $\vec{a} \xleftarrow{\$} \mathbb{Z}_p^n$ are replaced by a secret matrix $\boldsymbol{A}' \xleftarrow{\$} \mathbb{Z}_p^{k\times m}$ (for some $m \geq 1$; for the sake of simplicity, in the sequel, we choose $k = m$) and a secret vector of matrices $\vec{\boldsymbol{A}} \xleftarrow{\$} (\mathbb{Z}_p^{k\times k})^n$, respectively.

Result under $\mathcal{E}_{2,d}$-MDDH. To extend Theorem 3.1 to symmetric bilinear groups and avoid the commutativity problem, we suppose that all indeterminates appear "in the same order when multiplied together" in each subexpression of the representation of the non-commutative polynomials P_j (e.g., $P_1 = T_1T_3 + T_3T_2$ and $P_2 = T_3 + T_1T_2$, where T_1 appears before T_3 which appears before T_2). The condition is quite natural and is formally defined in Appendix B.2. That makes these non-commutative polynomials behave very similarly to commutative polynomial, and we get the following theorem.

Theorem 6.1. *Let $\mathbb{G} = \langle g\rangle$ be a group of prime order p. Let $\mathscr{A}$ be an adversary against the $(2, n, d)$-GP security of $\mathbb{G}$ that makes q oracle queries $P_1, \ldots, P_q$. We suppose that all indeterminates appear in the same order in each monomial of each non-commutative polynomials P_j. Then we can build an adversary $\mathscr{B}$ against the $\mathcal{E}_{2,d}$-MDDH problem in $\mathbb{G}$, such that $\mathbf{Adv}^{(2,n,d)\text{-gp}}_{\mathbb{G}}(\mathscr{A}) \leq n\cdot d\cdot \mathbf{Adv}^{\mathcal{E}_{2,d}\text{-mddh}}_{\mathbb{G}}(\mathscr{B}) + O(ndq/p)$. The running time of $\mathscr{B}$ is that of $\mathscr{A}$ plus the time to perform a polynomial number (in q, n, and d) of operations in $\mathbb{Z}_p$ and $\mathbb{G}$.*

The proof is similar to the proof of the LIP theorem (with some additional care when partially evaluating polynomials to avoid having polynomials with matrix coefficients) and is given in Appendix H.4. Actually, this theorem can trivially be extended to the (k, n, d)-GP security and the $\mathcal{E}_{k,d}$-MDDH assumption. But for $k \geq 3$ and $n \geq 2$, it is not known if the latter assumption is secure in the generic k-linear group model.

Results in the Generic Bilinear Group Model. We may wonder whether the $(2, k, d)$-GP security still holds in the generic bilinear group model, when indeterminates do not necessarily appear in the same order in each polynomial P_j. As seen before, it is not sufficient to suppose that $(P_j)_{j=1,\ldots,q}$ is a linearly independent family. But we show here that under a relatively natural condition, the *DLM* (distinct leading monomial) condition, the $(2, k, d)$-GP security still holds.

To formally state our result, we need to introduce some notions, which are formally defined in Appendix H and which are informally described here. We consider a monomial order for $\mathbb{Z}_p[T_1, \ldots, T_n]$, which is a total order on monomials $T_1^{i_1}\cdots T_n^{i_n}$ compatible with multiplications and where 1 is the smallest monomial. We then define the commutative leading monomials of a non-commutative polynomial as the monomials which are the highest for our monomial order, when considered as commutative monomials. There may be many commutative leading monomials for a given polynomial (for example $T_1T_2^2 + 5T_2T_1T_2$ has two commutative leading monomials: $T_1T_2^2$ and $T_2T_1T_2$). We say a polynomial has a unique commutative leading monomial if there is only one such monomial.

Finally, we say that a family of polynomials $(P_j)_j$ satisfies the DLM condition, if there exists a monomial order and an invertible matrix $\boldsymbol{M} \in \mathbb{Z}_p^{q\times q}$ such that $\boldsymbol{M}\cdot(P_j)_j$ is a vector of non-commutative polynomials with unique and distinct commutative leading monomials, where $(P_j)_j$ is the column vector of polynomials P_j.

Theorem 6.2. *Let $\mathbb{G} = \langle g\rangle$ be a group of prime order p. Let $\mathscr{A}$ be an adversary against the $(2, n, d)$-GP security of $\mathbb{G}$ that makes q oracle queries $P_1, \ldots, P_q$. We suppose that $(P_j)_j$ satisfies the DLM condition. Then, the advantage $\mathbf{Adv}^{(2,n,d)\text{-gp}}_{\mathbb{G}}(\mathscr{A})$ is negligible in the generic bilinear group model.*

The proof of Theorem 6.2 is given in Appendix H.5 and follows from Lemma H.3 in Appendix H.2. We remark that, in the case of commutative polynomials (i.e., LIP theorem), the DLM condition is exactly the same as saying that the polynomials P_j are linearly independent (using the Gauss reduction). However, this is not the case with non-commutative polynomials (e.g., consider $P_1 = T_1T_2$ and $P_2 = T_2T_1$ which are linearly independent but which have the same leading monomial).

Summary. Table 3 provides a summary of all our results about GP security.

6.3 Applications

RKA-PRFs in Generic Bilinear Groups. The RKA-PRF for permutation of univariate polynomial functions based on WNR (Section 5.2) can easily be transformed into an RKA-secure PRF for symmetric bilinear groups for the same set of RKD functions. It is sufficient to change keys from $\vec{a} \xleftarrow{\$} \mathbb{Z}_p^n$ to $\vec{\boldsymbol{A}} \xleftarrow{\$} (\mathbb{Z}_p^{2\times 2})^n$. Indeed, the RKA framework extends to this case easily, and the polynomials family we considered verifies the DLM condition as non-commutative polynomials. Actually, our proof of their

Table 3: Summary of our Results Related to Generalized Polynomial Security

Cyclic Group $\mathbb{G}$	**Symmetric Bilinear Group** (pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$)
$P_j \in \mathbb{Z}_p[T_1, \ldots, T_n]$ commutative polynomial $(a', a_1, \ldots, a_n) \xleftarrow{\$} \mathcal{K} = \mathbb{Z}_p^{n+1}$	$P_j \in \mathbb{Z}_p\langle T_1, \ldots, T_n\rangle$ non-commutative polynomial $(a', a_1, \ldots, a_n) \xleftarrow{\$} \mathcal{K} = (\mathbb{Z}_p^{2\times 2})^{n+1}$
In generic cyclic group: $(1, n, d)$-GP security $\Leftrightarrow (P_j)_j$ satisfies the DLM condition $\Leftrightarrow (P_j)_j$ is linearly independent (easy)	*In generic bilinear group:* $(2, n, d)$-GP security $\Leftarrow (P_j)_j$ satisfies the DLM condition (Theorem 6.2)
Under $\mathcal{E}_{1,d}$-MDDH*:* $(1, n, d)$-GP security $\Leftrightarrow$ same condition as above (Theorem 3.1, the LIP theorem)	*Under* $\mathcal{E}_{2,d}$-MDDH*:* $(2, n, d)$-GP security $\Leftarrow$ same condition as above + same order for indeterminates or equivalently, $(P_j)_j$ is linearly independent + same order for indeterminates (Theorem 6.1)

linear independence can be seen as exhibiting a monomial order (namely the graded lexicographic order) for which these polynomials have distinct leading monomials. In addition, their leading monomials are always unique even as non-commutative polynomials.

RKA-PRFs under $\mathcal{E}_{2,d}$-MDDH. Unfortunately, Theorem 6.1 does not apply to RKA-PRF for permutation, as permutation change the order of the indeterminates. However, it still easily enables to construct the first RKA-PRF for univariate polynomial functions, secure in symmetric bilinear groups, using the construction of Section 5.2 (or a slightly more efficient variant thereof in Appendix F.1). Again, the construction is straightforward and so is the proof.

Acknowledgements

This work was supported by the French ANR-10-SEGI-015 PRINCE Project, the *Direction Générale de l'Armement* (DGA), the CFM Foundation, and the European Research Council under the European Union's Seventh Framework Program (FP7/2007-2013 Grant Agreement 339563 – CryptoCloud).

References

[ABPP14] Michel Abdalla, Fabrice Benhamouda, Alain Passelègue, and Kenneth G. Paterson. Related-key security for pseudorandom functions beyond the linear barrier. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 77–94. Springer, Heidelberg, August 2014. (Cited on pages 3, 4, 10, 11, 21, 27, 31, 34, 35, and 36.)

[ACF09] Michel Abdalla, Dario Catalano, and Dario Fiore. Verifiable random functions from identity-based key encapsulation. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 554–571. Springer, Heidelberg, April 2009. (Cited on page 3.)

[AES01] Advanced Encryption Standard (AES). National Institute of Standards and Technology (NIST), FIPS PUB 197, U.S. Department of Commerce, November 2001. (Cited on page 3.)

[AL50] Avraham Shimshon Amitsur and Jacob Levitzki. Minimal identities for algebras. *Proceedings of the American Mathematical Society*, 1(4):449–463, 1950. (Cited on page 13.)

[BB04] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 56–73. Springer, Heidelberg, May 2004. (Cited on page 6.)

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 440–456. Springer, Heidelberg, May 2005. (Cited on pages 41 and 43.)

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, Heidelberg, August 2004. (Cited on page 3.)

[BC10] 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, Heidelberg, August 2010. (Cited on pages 3, 5, 8, 10, 11, 17, and 34.)

[BCK96] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Pseudorandom functions revisited: The cascade construction and its concrete security. In *37th FOCS*, pages 514–523. IEEE Computer Society Press, October 1996. (Cited on page 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, Heidelberg, May 2005. (Cited on page 3.)

[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, Heidelberg, May 1994. (Cited on page 3.)

[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, Heidelberg, May 2003. (Cited on pages 4 and 5.)

[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, Heidelberg, August 2009. (Cited on page 3.)

[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, Heidelberg, August 2013. (Cited on page 13.)

[BMR10] Dan Boneh, Hart William Montgomery, and Ananth Raghunathan. Algebraic pseudorandom functions with improved efficiency from the augmented cascade. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM CCS 10*, pages 131–140. ACM Press, October 2010. (Cited on pages 3, 4, 7, and 8.)

[Boy08] Xavier Boyen. The uber-assumption family (invited talk). In Steven D. Galbraith and Kenneth G. Paterson, editors, *PAIRING 2008*, volume 5209 of *LNCS*, pages 39–56. Springer, Heidelberg, September 2008. (Cited on pages 41 and 43.)

[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, Heidelberg, August 2014. (Cited on page 13.)

[BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 719–737. Springer, Heidelberg, April 2012. (Cited on page 5.)

[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, Heidelberg, May / June 2006. (Cited on page 5.)

[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, Heidelberg, August 2013. (Cited on pages 3, 4, 6, 13, 18, 41, and 43.)

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, October 1986. (Cited on pages 3, 4, and 5.)

[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, Heidelberg, March 2011. (Cited on pages 3, 6, and 18.)

[HK07] Dennis Hofheinz and Eike Kiltz. Secure hybrid encryption from weakened key encapsulation. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 553–571. Springer, Heidelberg, August 2007. (Cited on page 6.)

[HW10] Susan Hohenberger and Brent Waters. Constructing verifiable random functions with large input spaces. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 656–672. Springer, Heidelberg, May 2010. (Cited on page 3.)

[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, Heidelberg, March 2007. (Cited on page 3.)

[Knu93] Lars R. Knudsen. Cryptanalysis of LOKI91. In Jennifer Seberry and Yuliang Zheng, editors, *AUSCRYPT'92*, volume 718 of *LNCS*, pages 196–208. Springer, Heidelberg, December 1993. (Cited on page 3.)

[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, Heidelberg, June 2014. (Cited on page 5.)

[LW09] Allison B. Lewko and Brent Waters. Efficient pseudorandom functions from the decisional linear assumption and weaker variants. In Ehab Al-Shaer, Somesh Jha, and Angelos D. Keromytis, editors, *ACM CCS 09*, pages 112–120. ACM Press, November 2009. (Cited on pages 3 and 13.)

[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 pages 3 and 8.)

[NR99] Moni Naor and Omer Reingold. Synthesizers and their application to the parallel construction of pseudo-random functions. *Journal of Computer and System Sciences*, 58(2):336–375, 1999. (Cited on page 4.)

[Sha07] Hovav Shacham. A cramer-shoup encryption scheme from the linear assumption and from progressively weaker linear variants. Cryptology ePrint Archive, Report 2007/074, 2007. http://eprint.iacr.org/2007/074. (Cited on page 6.)

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Heidelberg, May 1997. (Cited on page 6.)

A Usual Definitions and Assumptions

A.1 Standard Definitions

Collision-Resistant (CR) Hash Function. As in [BC10], we view hash functions as unkeyed for simplicity. More precisely, the advantage of $\mathscr{C}$ in attacking the collision-resistance security of hash function $h\colon \mathcal{D} \to \mathsf{hSp}$ 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 $(x, x') \xleftarrow{\$} \mathscr{C}$. Please note that we assume the probability to be only over the randomness of the adversary in order to ease notation, but the traditional definition could be used as well.

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

Figure 2: Games defining the advantage of an adversary $\mathscr{D}$ against the SDL and DDHI problems in $\mathbb{G}$.

Hardness Assumptions. Our proofs make use of the d-Strong Discrete Logarithm (SDL) and Decisional d-Diffie-Hellman Inversion (DDHI) problems given in [GOR11] and described in Figure 2. For the SDL problem, we define the advantage of an adversary $\mathscr{D}$ against the SDL problem in $\mathbb{G}$ as

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

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. For the DDHI problem, the advantage of an adversary $\mathscr{D}$ against the DDHI problem in $\mathbb{G}$ is

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

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

We also recall the definition of the k-Lin problem in $\mathbb{G}$, which states the hardness of distinguishing whether $z = [w_1 + \cdots + w_k]$ or a random group element, when given a tuple $([1], [a_1], \ldots, [a_k], [a_1 w_1], \ldots, [a_k w_k], z)$, where $a_i, w_i \xleftarrow{\$} \mathbb{Z}_p$ for $i = 1, \ldots, k$.

We have three assumptions corresponding to the hardness of these problems, the d-SDL assumption, the d-DDHI assumption, and the k-Lin assumption.

Setting $d = 1$ in the d-SDL problem, we recover the usual definition of the DL problem in $\mathbb{G}$. Setting $k = 1$ in the k-Lin problem, we recover the usual definition of the DDH problem in $\mathbb{G}$.

A.2 Random Self-Reducibility of $\mathcal{E}_{k,d}$-MDDH and $(\mathcal{E}_{k,d}, N)$-MDDH

$\mathcal{E}_{k,d}$-MDDH assumption is random self-reducible. Namely, let $(\mathcal{E}_{k,d}, N)$-MDDH denote the N-fold $\mathcal{E}_{k,d}$-MDDH assumption, which is similar to the $\mathcal{E}_{k,d}$-MDDH assumption, except that $\boldsymbol{W} \xleftarrow{\$} \mathbb{Z}_p^{k \times N}$, $\boldsymbol{U} \xleftarrow{\$} \mathbb{Z}_p^{k(d+1) \times N}$. Then, we have the following lemma.

Lemma A.1 (Lemma 1 from [EHK+13])**.** *Let $\mathscr{A}$ be an adversary against the $(\mathcal{E}_{k,d}, N)$-MDDH assumption in $\mathbb{G}$. Then we can construct an adversary against the $\mathcal{E}_{k,d}$-MDDH such that:*

$$\mathbf{Adv}_{\mathbb{G}}^{(\mathcal{E}_{k,d},N)\text{-}\mathsf{mddh}}(\mathscr{A}) \leq \begin{cases} N \cdot \mathbf{Adv}_{\mathbb{G}}^{\mathcal{E}_{k,d}\text{-}\mathsf{mddh}}(\mathscr{B}) & \text{if } 1 \leq N \leq kd \\ kd \cdot \mathbf{Adv}_{\mathbb{G}}^{\mathcal{E}_{k,d}\text{-}\mathsf{mddh}}(\mathscr{B}) + \frac{1}{p-1} & \text{if } N > kd \end{cases}$$

A.3 From $\mathcal{E}_{1,d}$-MDDH to DDHI

In this section, we simply show that the $\mathcal{E}_{1,d}$-MDDH assumption in $\mathbb{G}$ is implied by the hardness of the DDHI assumption in $\mathbb{G}$. In order to do that, we introduce new intermediate assumptions.

$\mathcal{E}_{1,(d,l)}$-MDDH **Assumption** Let $\mathbb{G}$ be a group of prime order p. Let $1 \leq l \leq d$. The $\mathcal{E}_{1,(d,l)}$-MDDH simply states that, given

$$g\,, \quad [\boldsymbol{\Gamma}] = \begin{pmatrix} [a'] \\ [a_1 \cdot a'] \\ \vdots \\ [a_1^d \cdot a'] \end{pmatrix} \in \mathbb{G}^{(d+1)\times 1} \quad \text{and} \quad [\boldsymbol{Z}] = \begin{pmatrix} [a' \cdot w] \\ [a_1 \cdot a' \cdot w] \\ \vdots \\ [a_1^{l-1} \cdot a' \cdot w] \\ z \end{pmatrix} \in \mathbb{G}^{(l+1)\times 1}$$

with g a generator of $\mathbb{G}$ and $a', a_1, w \xleftarrow{\$} \mathbb{Z}_p$, then it is hard to distinguish whether $z = [a_1^l \cdot a' \cdot w]$ or whether z is a random group element. Hence, it simply describes, given $[\boldsymbol{\Gamma}]$ and the l first rows of $[\boldsymbol{\Gamma} \cdot \boldsymbol{W}]$, where $\boldsymbol{W} = (w) \in \mathbb{Z}_p^{1\times 1}$, the hardness of distinguishing the $(l+1)$-th row of $[\boldsymbol{\Gamma} \cdot \boldsymbol{W}]$ from a random value.

We immediately have the following lemma.

Lemma A.2. *Let $\mathbb{G}$ be a group of prime order p and $d \geq 1$. Let $\mathscr{A}$ be an adversary against the $\mathcal{E}_{1,d}$-MDDH problem in $\mathbb{G}$. Then we can construct adversaries $\mathscr{B}_l$ against the $\mathcal{E}_{1,(d,l)}$-MDDH problem in $\mathbb{G}$, for $l = 1, \ldots, d$ such that*

$$\begin{aligned}\mathbf{Adv}_{\mathbb{G}}^{\mathcal{E}_{1,d}\text{-mddh}}(\mathscr{A}) &\leq \sum_{l=1}^{d} \frac{p}{p-1} \cdot \mathbf{Adv}_{\mathbb{G}}^{\mathcal{E}_{1,(d,l)}\text{-mddh}}(\mathscr{B}_l) \\ &\leq \frac{p}{p-1} \cdot d \cdot \mathbf{Adv}_{\mathbb{G}}^{\mathcal{E}_{1,(d,d)}\text{-mddh}}(\mathscr{B}_d)\ .\end{aligned}$$

The running time of $\mathscr{B}_j$ is that of $\mathscr{A}$ plus the time required to compute at most d exponentiations in $\mathbb{G}$.

Lemma A.2. The proof follows a standard hybrid argument. We define games H_l for $l = 0, \ldots, d$ as in Figure 3. Clearly, we have $\mathrm{H}_0 \equiv \mathcal{E}_{1,d}$-MDDH-Rand and $\mathrm{H}_d \equiv \mathcal{E}_{1,d}$-MDDH-Real. Moreover, it is straightforward to construct an adversary $\mathscr{B}_l$ such that $\Pr[\mathrm{H}_l \Rightarrow 1] - \Pr[\mathrm{H}_{l-1} \Rightarrow 1] \leq \mathbf{Adv}_{\mathbb{G}}^{\mathcal{E}_{1,(d,l)}\text{-mddh}}(\mathscr{B}_l)$, for $l = 1, \ldots, d$. Adversary $\mathscr{B}_l$ simply samples $d - l$ random group elements to complete its matrix and the simulation is perfect.

Finally, it is clear that $\mathbf{Adv}_{\mathbb{G}}^{\mathcal{E}_{1,(d,l)}\text{-mddh}}(\mathscr{B}_l) \leq \mathbf{Adv}_{\mathbb{G}}^{\mathcal{E}_{1,(d,d)}\text{-mddh}}(\mathscr{B}_d)$ for $l = 1, \ldots, d$, since one can obtain the $\mathcal{E}_{1,(d,l)}$-MDDH matrices from a $\mathcal{E}_{1,(d,d)}$-MDDH by simply taking the last $l+1$ rows of the second matrix (which is a perfect tuple fixing $w = a_1^{d-l} w$).

Lemma A.2 easily follows. □

proc Initialize // H_l, $l = 0, \ldots, d$	**proc Finalize**(b)
$a', a_1, w \xleftarrow{\$} \mathbb{Z}_p$, $u_k \xleftarrow{\$} \mathbb{Z}_p$, for $k = l+1, \ldots, d$	Return b
$\Gamma_k \leftarrow [a_1^k \cdot a']$, for $k = 0, \ldots, d$	
$Z_k \leftarrow [a_1^k \cdot a' \cdot w]$, for $k = 0, \ldots, l$	
$Z_k \leftarrow [u_k]$, for $k = l+1, \ldots, d$	
Return $(\Gamma, \vec{Z})$	

Figure 3: Game for the proof of Lemma A.2.

Lemma A.3. *Let $\mathbb{G}$ be a group of prime order p. Let $\mathscr{A}$ be an adversary against the $\mathcal{E}_{1,(d,d)}$-MDDH problem in $\mathbb{G}$. Then, we can construct an adversary $\mathscr{B}$ against the* DDHI *problem in $\mathbb{G}$ such that*

$$\mathbf{Adv}_{\mathbb{G}}^{\mathcal{E}_{1,(d,d)}\text{-mddh}}(\mathscr{A}) \leq \frac{p}{p-1} \cdot \mathbf{Adv}_{\mathbb{G}}^{d\text{-ddhi}}(\mathscr{B})$$

Moreover, the running time of $\mathscr{B}$ is that of $\mathscr{A}$ plus the time required to compute $2d$ operations in $\mathbb{Z}_p$ and $\mathbb{G}$.

Lemma A.3. Let $\mathbb{G}$ be a group of prime order p and $d \geq 1$. Let us assume adversary $\mathscr{B}$ has a tuple $(\vec{\Gamma}, \gamma) = (([a'], [aa'], \ldots, [a^d a']), \gamma)$ where $\gamma = [a^{d+1}a']$ or γ is a random group element $[c]$. Then it chooses $\alpha, \beta \xleftarrow{\$} \mathbb{Z}_p$ at random and computes the tuple $\vec{Z}$ by letting $Z_l = \Gamma_{l+1}^{\alpha} \cdot \Gamma_l^{\beta}$ for $l = 0, \ldots, d-1$ and $Z_d = \gamma^{\alpha} \cdot \Gamma_d^{\beta}$. Hence, for $l = 0, \ldots, d-1$, it is clear that $Z_l = [(\alpha a + \beta) a^l a']$. If $\gamma = [a^{d+1}]$, we have $Z_d = [(\alpha a + \beta) a^d a']$. If $\gamma = [c]$ with $c \xleftarrow{\$} \mathbb{Z}_p$, then $Z_d = [(\alpha c + \beta a^d) a']$. Let us fix $a, c \in \mathbb{Z}_p$ and let $b' = \alpha a + \beta$ and $c' = \alpha c + \beta a^d$. Then, the matrices $[\boldsymbol{\Gamma}]$ and $[\boldsymbol{Z}]$ of dimension $(d+1) \times 1$ obtained by writing vectors $(\vec{\Gamma}, \vec{Z})$ as column vectors forms exactly a $\mathcal{E}_{1,(d,d)}$-MDDH tuple 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 $\frac{p-1}{p}$.

To finish the proof, we simply need to explain how adversary $\mathscr{B}$ can get such a tuple $(\vec{\Gamma}, \gamma)$ from its DDHI tuple. Let $([1], [a], \ldots, [a^d], x) \in \mathbb{G}^{d+2}$ be a DDHI tuple. Then the tuple $(([a^d], [a^{d-1}], \ldots, [a], [1]), x)$

is such a tuple since $h = \left[a^d\right]$ is a random generator of $\mathbb{G}$ (since [1] was a random generator) and $\frac{1}{a}$ is random in $\mathbb{Z}_p^*$, so we have $\left[a^{d-j}\right] = h^{(\frac{1}{a})^j}$, for $j = 0, \ldots, d$ and either x is random, either $x = \left[\frac{1}{a}\right]$ and then $x = h^{(\frac{1}{a})^{d+1}}$. Hence, this is exactly a tuple of the wanted form $(\vec{\Gamma}, \gamma)$.

The claim now easily follows. □

B Multivariate Polynomial Representation

B.1 Multivariate Polynomial Representation for the LIP Theorem (Theorem 3.1)

As explained in Section 3.1, the theorem would be quite straightforward assuming the polynomials are given in expanded form. However, it would not encompass a lot of interesting cases. Then, there are some subtle points to consider in order to avoid to make such an assumption.

Indeed, while we do want to be as general as possible, it is pretty clear that one needs to make some restrictions. For example, the minimal requirement seems at least to be able to evaluate the polynomial in arbitrary points, so it would be unconceivable to define a polynomial P by an RSA modulus N, as the polynomial $P = (X - p_1)(X - p_2)$, with p_1 and p_2 the two prime factors of N.

From now on, we will denote by $\tilde{P}$ the representation of the polynomial, while P is the mathematical polynomial object. We assume in the body of the paper that there is at least one representation $\tilde{P}$ of P which has polynomial size in n and d. This assumption simplifies the bounds and seems reasonable, but we could avoid it. However, bounds in the theorems would need to be changed.

We also assume that one can, in polynomial time, evaluate P on any points of $\mathbb{Z}_p^n$ as well as evaluate it partially on any points of $\mathbb{Z}_p^k$ with $k < n$, obtaining a polynomial Q with $n - k$ indeterminates, and such that Q has the same properties. These two properties are summarized in Condition 1 below.

In fact, in the whole paper, $\tilde{P}$ is just an expression or an abstract syntax tree (AST) where internal nodes are either $+$ or $\cdot$, while leaves are either an indeterminate T_i or a scalar in $\mathbb{Z}_p$. A partial evaluation can be performed by replacing T_i by a_i (when $i > j$) in the (AST), while a full evaluation can be performed by evaluating the AST (after the previous replacement, with $j = 0$). Both operations are polynomial-time in the size of the AST.

Condition 1. *It is possible to get from $\tilde{P}$ (in polynomial time):*

full evaluation *the value $P(a_1, \ldots, a_n) \in \mathbb{Z}_p$, given $a_1, \ldots, a_n \in \mathbb{Z}_p$;*

partial evaluation *for any $j = 0, \ldots, n$, a representation $\tilde{Q}$ of $Q = P(T_1, \ldots, T_j, a_{j+1}, \ldots, a_n)$, given $a_{j+1}, \ldots, a_n \in \mathbb{Z}_p$. This representation $\tilde{Q}$ has again to verify (recursively) Condition 1.*

B.2 Extension to the GP Security Notion

The lack of commutativity in the GP case (6) makes everything more complex. Hence, we have to make additional assumptions. Intuitively, we want that all the indeterminates always appear in the same order. That is, without loss of generality, T_n appears before T_{n-1}, T_{n-1} before T_{n-2}, and so on. Moreover, the latter has to be true for both the mathematical object P as well as for any of its representation $\tilde{P}$ in order to be able to run the proof.

Specifically, we want to ensure that when evaluating the polynomial, we never have to compute $\boldsymbol{A}_j \boldsymbol{A}_{j'}$ with $j' > j$, even if this expression does not appear in the resulting polynomial. For instance, $T_2T_3 - T_2(T_3 + T_1)$ is not an acceptable representation of the polynomial T_2T_1, while the representation T_2T_1 is acceptable.

Precisely, we assume that the representation of the polynomials satisfies the following condition.

Condition 2. *The representation of a polynomial is an expression or AST (where internal nodes are either $+$ or $\cdot$, while leaves are either an indeterminate T_i or a scalar in $\mathbb{Z}_p$) with the following additional (natural) property (to deal with non-commutativity): if $\tilde{P}_1 \cdot \tilde{P}_2$ is a sub-expression of $\tilde{P}$, and if T_j is a leaf of $\tilde{P}_1$ for some j, then for any $j' > j$, $T_{j'}$ is not a leaf of $\tilde{P}_2$.*

We remark that, since we always perform multiplications of the indeterminates in the same order because of this condition, polynomials can actually be viewed as commutative polynomials, even if the polynomials we consider would normally be non-commutative

Please also note that, when $k = 1$, Condition 2 is stronger than Condition 1.

C Proof of the LIP Theorem (Theorem 3.1)

Let $\mathscr{A}$ be an adversary against the (n,d)-LIP security of $\mathbb{G}$ that makes q oracle queries. Let us construct an adversary $\mathscr{B}$ against the $(\mathcal{E}_{k,d}, N)$-MDDH assumption, defined in Appendix A.2, such that:

$$\mathbf{Adv}_{\mathbb{G}}^{(n,d)\text{-lip}}(\mathscr{A}) \leq n \cdot \mathbf{Adv}_{\mathbb{G}}^{(\mathcal{E}_{1,d}, d\cdot q)\text{-mddh}}(\mathscr{B}) + \frac{2n(d+1)q}{p} + \frac{n}{p} \,. \tag{3}$$

the LIP theorem follows from the above equation and Lemma A.1.

The proof of the above equation is based on the sequence of games in Figure 4. The games are used in the following order: $\mathrm{G}_{0,1}, \mathrm{G}_{1,1}, \mathrm{G}_{0,2}, \ldots, \mathrm{G}_{1,n}$. We denote by $\textsc{Succ}_i$ the event that game G_i output takes the value 1.

Preliminaries. Let $\vec{a} \in \mathbb{Z}_p^n$. For any polynomial $P \in \mathbb{Z}_p[T_1, \ldots, T_n]$ with degree in one indeterminate bounded by d and for $j = 1, \ldots, n$, we define the polynomial $Q_{P,\vec{a},j} \in \mathbb{Z}_p[T_1, \ldots, T_j]$ as the evaluation of P with $T_i = a_i$, for $i = j+1, \ldots, n$, so:

$$Q_{P,\vec{a},j}(T_1, \ldots, T_j) = P(T_1, \ldots, T_j, a_{j+1}, \ldots, a_n) \,.$$

Since the degree of P in any indeterminate is at most d, $Q_{P,\vec{a},j}$ may have up to $(d+1)^j$ (distinct) monomials and so cannot be expanded efficiently. But we can formally consider it as a row vector $(Q^{(z)}_{P,\vec{a},j})_{z\in\{0,\ldots,d\}^j}$ in $\mathbb{Z}_p^{(d+1)^j}$, where $Q^{(z)}_{P,\vec{a},j}$ 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 denote by $Q_{P,\vec{a},j} \odot \vec{U}$ the multiplication of such a vector with a column vector $\vec{U} = (U_z)_z$ with entries from $\mathbb{G}$, defined as:

$$Q_{P,\vec{a},j} \odot \vec{U} = \prod_{z\in\{0,\ldots,d\}^j} U_z^{Q^{(z)}_{P,\vec{a},j}} \,.$$

In addition, since $Q_{P,\vec{a},j}$ is a polynomial in $\mathbb{Z}_p[T_1, \ldots, T_j]$, with degree in any indeterminate bounded by d, there exist unique polynomials $Q_{P,\vec{a},j,0}, \ldots, Q_{P,\vec{a},j,d}$ in $\mathbb{Z}_p[T_1, \ldots, T_{j-1}]$, such that

$$Q_{P,\vec{a},j} = Q_{P,\vec{a},j,0} + Q_{P,\vec{a},j,1} \cdot T_j + \ldots + Q_{P,\vec{a},j,d} \cdot T_j^d \,.$$

In particular, we have $Q_{P,\vec{a},j-1} = Q_{P,\vec{a},j,0} + Q_{P,\vec{a},j,1} \cdot a_j + \ldots + Q_{P,\vec{a},j,d} \cdot a_j^d$. It is important to notice that while the polynomials $Q_{P,\vec{a},j,i}$ may not be expanded efficiently (because they may have up to $(d+1)^{j-1}$ monomials), it is easy (i.e., possible in polynomial time in n and d) to compute them formally (i.e., without actually doing and expanding multiplications and additions). This "formal" form is sufficient for our purpose.

Finally, we denote by **TestLin** the statistical polynomial-time procedure, described in [ABPP14], which takes as a list $\mathcal{L}$ of polynomials $(R_1, \ldots, R_L)$ (such that $R_1, \ldots, R_L$ are linearly independent as polynomials) and a polynomial R and which outputs:

$$\begin{cases} \perp & \text{if } R \text{ is linearly independent of the set } \{R_1, \ldots, R_L\} \\ \vec{\lambda} = (\lambda_1, \ldots, \lambda_L) & \text{otherwise, so that } R = \lambda_1 R_1 + \ldots + \lambda_L R_L \end{cases}$$

There is at most one such $\vec{\lambda}$, since polynomials in inputs are assumed to be linearly independent. For polynomials in inputs with n indeterminates and degree at most d in any indeterminate, assuming $n(d+1) \leq \sqrt{p}$, there is a statistical procedure which is incorrect with probability at most $\frac{1}{p}$ and which runs in $O(L^2 dn + L^3)$. Please refer to [ABPP14] for the details of this procedure. **TestLin** only needs to be able to evaluate the polynomials $R_1, \ldots, R_L$ and R and do not need an expanded form of them.

While **TestLin** is a statistical procedure since no polynomial time deterministic procedure is known, we will first suppose that **TestLin** is perfect and will deal with statistical errors at the end of the proof.

Let us start with the proof. We first show that game $\mathrm{G}_{0,1}$ instantiates exactly the game defining the (n,d)-LIP security of $\mathbb{G}$ when $b = 0$. For any query P, we have $Q_{P,\vec{a},1} \in \mathbb{Z}_p[T_1]$ and $Q_{P,\vec{a},1} = \sum_{i=0}^{d} Q_{P,\vec{a},1,i} \cdot T_1^i$, with $Q_{P,\vec{a},1,i} \in \mathbb{Z}_p$. The first time we see a non-zero coefficient $\alpha = Q_{P,\vec{a},1,i}$: $\mathcal{L}[1] \leftarrow \alpha$, $\mathsf{T}[1,0] \xleftarrow{\$} \mathbb{G}$ (and let us write this element $[a']$), and $\mathsf{T}[1,k] \leftarrow a_j^k a'$ for $k = 1, \ldots, d$. Afterwards,

proc Initialize // $\mathrm{G}_{0,j}$; $j = 1, \ldots, n$
$\vec{a} \xleftarrow{\$} \mathbb{Z}_p^n$
$\mathcal{L} \leftarrow$ empty list
$\mathsf{T} \leftarrow$ empty 2-dimensional table
$L \leftarrow 0$ (length of $\mathcal{L}$)
proc RKFn(P) // $\mathrm{G}_{0,j}$; $j = 1, \ldots, n$
$y \leftarrow 1$
For $i = 0, \ldots, d$
  $\vec{\lambda}^{(i)} \leftarrow$ **TestLin**$(\mathcal{L}, Q_{P,\vec{a},j,i})$
  If $\vec{\lambda}^{(i)} = \perp$ then
    $L \leftarrow L + 1$
    $\mathcal{L}[L] \leftarrow Q_{P,\vec{a},j,i}$
    $\mathsf{T}[L, 0] \xleftarrow{\$} \mathbb{G}$
    For $k = 1, \ldots, d$
      $\mathsf{T}[L, k] \xleftarrow{\$} \mathsf{T}[L, 0]^{a_j^k}$
    $\vec{\lambda}^{(i)} \leftarrow (0, \ldots, 0, 1) \in \mathbb{Z}_p^{L+1}$
  $y \leftarrow y \cdot \prod_{l=0}^{L} \mathsf{T}[l, i]^{\lambda_l^{(i)}}$
Return y

proc Initialize // $\mathrm{G}_{1,j}$; $j = 1, \ldots, n$
$\vec{a} \xleftarrow{\$} \mathbb{Z}_p^n$
$\mathcal{L} \leftarrow$ empty list
$\mathsf{T} \leftarrow$ 2-dimensional table
$L \leftarrow 0$ (length of $\mathcal{L}$)
proc RKFn(P) // $\mathrm{G}_{1,j}$; $j = 1, \ldots, n$
$y \leftarrow 1$
For $i = 0, \ldots, d$
  $\vec{\lambda}^{(i)} \leftarrow$ **TestLin**$(\mathcal{L}, Q_{P,\vec{a},j,i})$
  If $\vec{\lambda}^{(i)} = \perp$ then
    $L \leftarrow L + 1$
    $\mathcal{L}[L] \leftarrow Q_{P,\vec{a},j,i}$
    $\mathsf{T}[L, 0] \xleftarrow{\$} \mathbb{G}$
    For $k = 1, \ldots, d$
      $\mathsf{T}[L, k] \xleftarrow{\$} \mathbb{G}$
    $\vec{\lambda}^{(i)} \leftarrow (0, \ldots, 0, 1) \in \mathbb{Z}_p^{L+1}$
  $y \leftarrow y \cdot \prod_{l=0}^{L} \mathsf{T}[l, i]^{\lambda_l^{(i)}}$
Return y

Figure 4: Games $\mathrm{G}_{0,j}$ and $\mathrm{G}_{1,j}$ for the proof of the LIP theorem. Differences between the two games are in blue.

TestLin$(\mathcal{L}, Q_{P,\vec{a},1,i})$ always outputs $\vec{\lambda}^{(i)} = Q_{P,\vec{a},1,i}/\alpha$, for $i = 0, \ldots, d$. Then, the output y is computed as

$$y = \prod_{i=0}^{d} T[0,i]^{\vec{\lambda}^{(i)}} = \prod_{i=0}^{d} \left[a_1^i \cdot a'\right]^{Q_{P,\vec{a},1,i}} = \left[\sum_{i=0}^{d} Q_{P,\vec{a},1,i} \cdot a_1^i \cdot a'\right]$$
$$= \left[Q_{P,\vec{a},1}(a_1) \cdot a'\right] = \left[P(\vec{a}) \cdot a'\right] .$$

Hence, this is exactly the game defining the (n, d)-LIP security of $\mathbb{G}$ when $b = 0$.

Now, let us show Game $\mathrm{G}_{0,j}$ and Game $\mathrm{G}_{1,j}$ are indistinguishable under the $(\mathcal{E}_{1,d}, d \cdot q)$-MDDH assumption. Afterwards, we will show that Game $\mathrm{G}_{1,j}$ and Game $\mathrm{G}_{0,j+1}$ are perfectly indistinguishable.

Indistinguishability of Game $\mathrm{G}_{0,j}$ and Game $\mathrm{G}_{1,j}$ under the $(\mathcal{E}_{1,d}, d \cdot q)$-MDDH assumption. We design adversaries $\mathscr{B}_j$ attacking the $(\mathcal{E}_{1,d}, d \cdot q)$-MDDH problem in $\mathbb{G}$ such that

$$\Pr\left[\mathrm{SUCC}_{0,j}\right] - \Pr\left[\mathrm{SUCC}_{1,j}\right] \leq \mathbf{Adv}_{\mathbb{G}}^{(\mathcal{E}_{1,d}, d\cdot q)\text{-mddh}}(\mathscr{B}_j) \; ; \; \forall j = 1, \ldots, n.$$

The adversary $\mathscr{B}_j$ takes as input a tuple $([\boldsymbol{\Gamma}], [\boldsymbol{Z}]) \in \mathbb{G}^{(d+1)\times 1} \times \mathbb{G}^{(d+1)\times(d\cdot q)}$, where either $\boldsymbol{Z} = \boldsymbol{\Gamma} \cdot \boldsymbol{W}$, and $\boldsymbol{W} \xleftarrow{\$} \mathbb{Z}_p^{1\times(d\cdot q)}$, or $\boldsymbol{Z} = \boldsymbol{U} \xleftarrow{\$} \mathbb{Z}_p^{(d+1)\times(d\cdot q)}$, with $\boldsymbol{\Gamma}$ defined as in Equation (1) in Section 2, and has to distinguish these two cases. For that purpose, the adversary $\mathscr{B}_j$ simulates everything as in Game $\mathrm{G}_{0,j}$ or $\mathrm{G}_{1,j}$ for $\mathscr{A}$, except it sets $\mathsf{T}[l, k] = z_{k,l}$. Assuming the matrix $A_0 \in \mathbb{Z}_p^{1\times 1}$ (in the definition of $\boldsymbol{\Gamma}$ in Equation (1)) is invertible (which happens with probability $1 - 1/p$), in the first case, everything is simulated as in Game $\mathrm{G}_{0,j}$, while in the second case, everything is simulated as in Game $\mathrm{G}_{1,j}$.

Perfect Indistinguishability of Game $\mathrm{G}_{1,j}$ and Game $\mathrm{G}_{0,j+1}$. We introduce an intermediate Game $\mathrm{G}_{2,j}$, described in Figure 5. We will use it to prove that Game $\mathrm{G}_{1,j}$ is perfectly indistinguishable from Game $\mathrm{G}_{0,j+1}$ by showing that both these games are perfectly indistinguishable from game $\mathrm{G}_{2,j}$. This intermediate game is not polynomial-time, since $\vec{U}$ contains $(d+1)^j$ entries, but this does not affect our proof since we show that it is perfectly indistinguishable from Game $\mathrm{G}_{1,j}$ and Game $\mathrm{G}_{0,j+1}$ which are both polynomial-time.

First, we prove that game $\mathrm{G}_{1,j}$ is perfectly indistinguishable from Game $\mathrm{G}_{2,j}$. Indeed, we can compute T in $\mathrm{G}_{1,j}$ as $\mathsf{T}[l, k] = \left(Q_l \cdot T_j^k\right) \odot \vec{U}$, for $k = 0, \ldots, d$, with $\vec{U}$ computed as in the Initialize procedure in $\mathrm{G}_{2,j}$ and $\mathcal{L}[l] = Q_l \in \mathbb{Z}_p[T_1, \ldots, T_{j-1}]$. Considering $(\mathsf{T}[l, k])_{l,k}$ as a vector over $\mathbb{Z}_p^{L(d+1)}$ and $\boldsymbol{M} = (Q_l \cdot T_j^k)_{l,k}$ as a matrix of $L(d+1)$ rows and $(d+1)^j$ columns (such that each row corresponds to a polynomial

proc Initialize // $\mathrm{G}_{2,j}$; $j = 1, \ldots, n$	**proc RKFn**(P) // $\mathrm{G}_{2,j}$; $j = 1 \ldots, n$
$\mathbf{a} \xleftarrow{\$} \mathbb{Z}_p^n$	$y \leftarrow Q_{P,\vec{a},j} \odot \vec{U}$
$U_x \xleftarrow{\$} \mathbb{G}$; $\forall x \in \{0, \ldots, d\}^j$	Return y

Figure 5: Games $\mathrm{G}_{2,j}$ for the proof of the LIP theorem.

$Q_l \cdot T_j^k$), then we have $\mathsf{T} = \boldsymbol{M} \odot \vec{U}$. Hence, since the polynomials Q_l are linearly independent (and do not contain the indeterminate T_j), the rows of $\boldsymbol{M}$ are also linearly independent, and $\boldsymbol{M}$ is full rank. Hence, if $\vec{U}$ is random, so is T, exactly as in Game $\mathrm{G}_{1,j}$. Now, if T is computed as $\mathsf{T} = \boldsymbol{M} \odot \vec{U}$ as above. Since $\vec{\lambda}^{(i)}$ is a vector in $\mathbb{Z}_p^{L+1}$ and since L may grow during the execution, we let $\lambda_l^{(i)} = 0$ if $l > |\vec{\lambda}^{(i)}|$. Then for a query P, the output y in Game $\mathrm{G}_{1,j}$ is computed as:

$$y = \prod_{i=0}^{d} \prod_{l=0}^{L} \mathsf{T}[l,i]^{\lambda_l^{(i)}} = \prod_{i=0}^{d} \prod_{l=0}^{L} \left(\left(\lambda_l^{(i)} \cdot Q_l \cdot T_j^i \right) \odot \vec{U} \right)$$
$$= \left(\sum_{i=0}^{d} \sum_{l=0}^{L} \lambda_l^{(i)} \cdot Q_l \cdot T_j^i \right) \odot \vec{U} = \left(\sum_{i=0}^{d} Q_{P,\vec{a},j,i} \cdot T_j^i \right) \odot \vec{U} = Q_{P,\vec{a},j} \odot \vec{U}.$$

Then, it is exactly computed as in Game $\mathrm{G}_{2,j}$. Therefore, games $\mathrm{G}_{1,j}$ and $\mathrm{G}_{2,j}$ are perfectly indistinguishable, for $j = 1, \ldots, n$.

Second, we prove that game $\mathrm{G}_{2,j}$ is perfectly indistinguishable from Game $\mathrm{G}_{0,j+1}$. The proof is similar to the previous one. Indeed, we can compute T in $\mathrm{G}_{0,j+1}$ as $\mathsf{T}[l,0] = Q_l \odot \vec{U}$ with $\vec{U}$ computed as in $\mathrm{G}_{2,j}$ and $\mathcal{L}[l] = Q_l \in \mathbb{Z}_p[T_1, \ldots, T_j]$, and letting $\mathsf{T}[l,k] = \mathsf{T}[l,0]^{a_{j+1}^k}$, for $k = 1, \ldots, d$. Considering $(\mathsf{T}[l,0])_l$ as a vector over $\mathbb{Z}_p^L$ and $\boldsymbol{M} = (Q_l)_l$ as a matrix of L rows and $(d+1)^j$ columns (each row corresponding to a polynomial Q_l for $l = 1, \ldots, L$), then we have $(\mathsf{T}[l,0])_l = \boldsymbol{M} \odot \vec{U}$. Hence, since the polynomials Q_l are linearly independent, the rows of $\boldsymbol{M}$ are also linearly independent, and $\boldsymbol{M}$ is full rank. Thus, if $\vec{U}$ is random, then so is $(\mathsf{T}[l,0])_l$, exactly as in Game $\mathrm{G}_{0,j+1}$. Now, if T is computed as above, then, for a query P, the output y in Game $\mathrm{G}_{0,j+1}$ is computed as:

$$y = \prod_{i=0}^{d} \prod_{l=0}^{L} \mathsf{T}[l,i]^{\lambda_l^{(i)}} = \prod_{i=0}^{d} \prod_{l=0}^{L} \mathsf{T}[l,0]^{\lambda_l^{(i)} \cdot a_{j+1}^i} = \prod_{i=0}^{d} \prod_{l=0}^{L} \left(\left(\lambda_l^{(i)} \cdot Q_l \cdot a_{j+1}^i \right) \odot \vec{U} \right)$$
$$= \left(\sum_{i=0}^{d} \sum_{l=0}^{L} \lambda_l^{(i)} \cdot Q_l \cdot a_{j+1}^i \right) \odot \vec{U} = \left(\sum_{i=0}^{d} Q_{P,\vec{a},j+1,i} \cdot a_{j+1}^i \right) \odot \vec{U} = Q_{P,\vec{a},j} \odot \vec{U}.$$

Then, it is exactly computed as in Game $\mathrm{G}_{2,j}$. Therefore, games $\mathrm{G}_{2,j}$ and $\mathrm{G}_{0,j+1}$ are perfectly indistinguishable, for $j = 1, \ldots, n-1$.

We finally prove that game $\mathrm{G}_{1,n}$ is perfectly indistinguishable from the game defining the (n,d)-LIP security of $\mathbb{G}$ when $b = 1$. Indeed, since $Q_{P,\vec{a},n} = P$ for any polynomial $P \in \mathbb{Z}_p[T_1, \ldots, T_n]$, we just need to prove that for any sequence $(P_1, \ldots, P_q)$ of linearly independent polynomials, the values $P_1 \odot \vec{U}, \ldots, P_q \odot \vec{U}$, where $\vec{U}$ is computed as in game $\mathrm{G}_{2,n}$, are uniformly random and independent, which is straightforward since $(P_1, \ldots, P_q)$ are linearly independent. But since games $\mathrm{G}_{1,n}$ and $\mathrm{G}_{2,n}$ are equivalent, $\mathrm{G}_{1,n}$ is already perfectly indistinguishable from the game defining the (n,d)-LIP security of $\mathbb{G}$ when $b = 1$.

Dealing with an Imperfect TestLin. To deal with an imperfect **TestLin**, we just remark that the only part where we supposed **TestLin** to be perfect in the proof was to prove the perfect indistinguishability of Game $\mathrm{G}_{1,j}$ and Game $\mathrm{G}_{0,j+1}$, and the perfect indistinguishability between Game $\mathrm{G}_{0,1}$ (respectively Game $\mathrm{G}_{1,n}$) with the game defining the (n,d)-LIP security of $\mathbb{G}$ when $b = 0$ (respectively $b = 1$). All this properties are statistical, so that it is possible to replace the real **TestLin** (with error $1/p$) by a perfect (with error 0, as used in the proof). This just loses an additive factor at most $(d+1)Q_A/p$ each time, and so at most $2n(d+1)q/p$ in total.

Equation (3) easily follows from the bounds arising in the different game hops. □

D Proofs for Section 4

D.1 Weighted NR

Lemma D.1. *Let $\mathbb{G} = \langle g \rangle$ be a group of prime order p and $\vec{w} = (w_0, \ldots, w_n) \in \mathbb{Z}_p \times (\mathbb{Z}_p^*)^n$, such that w_i and $p-1$ are coprime for $i = 1, \ldots, n$ and w_0 and $p-1$ are coprime if $w_0 \neq 0$. Let $\mathsf{WNR}^{\vec{w}}$ be the function defined via Table 2. Let $\mathscr{A}$ be an adversary against the PRF security of $\mathsf{WNR}^{\vec{w}}$ that makes q oracle queries. Then we can construct an adversary $\mathscr{B}$ against the DDH problem in $\mathbb{G}$ such that*

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{WNR}^{\vec{w}}}(\mathscr{A}) \leq n \cdot \mathbf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(\mathscr{B}) + O\big(\frac{nq}{p}\big) \ .$$

Moreover, $\mathscr{B}$'s running time is that of $\mathscr{A}$ plus the time to compute a polynomial number (in q and n) of operations in $\mathbb{G}$ and $\mathbb{Z}_p$.

Proof. Since we just apply a permutation to the key space, it is straightforward that the PRF security of NR implies the security of WNR. However, we just want to emphasize here that we can easily prove the security of all these PRFs (including NR) by simply applying the LIP theorem.

The proof is based on the LIP theorem. Let us assume that $w_0 \neq 0$. Since the application $a \in \mathbb{Z}_p \mapsto a^w \in \mathbb{Z}_p$ is a bijection as long as w and $p-1$ are coprime, it is clear that if NR is a secure PRF, then so is $\mathsf{WNR}^{\vec{w}}$, as long as w_i and $p-1$ are coprime, for $i = 0, \ldots, n$, since we just apply a permutation to the key space.

Let $\mathscr{A}$ be an adversary against the PRF security of NR that makes q oracle queries. We can assume without loss of generality that $\mathscr{A}$ never repeats a query. Then we can construct an adversary $\mathscr{B}$ against the $(n,1)$-LIP security of $\mathbb{G}$ that makes q queries $P_0, P_1, \ldots, P_q$, defined as follows.

First, $\mathscr{B}$ sends the public generator g to $\mathscr{A}$ as the generator used in the PRF construction. By doing so, it implicitly sets $a_0 = a'$.

Next, $\mathscr{B}$ runs adversary $\mathscr{A}$. When the latter makes a query $x \in \{0,1\}^n$, adversary $\mathscr{B}$ makes the query $P_x(T_1, \ldots, T_n) = \prod_{i=1}^n T_i^{x_i}$ to its oracle and returns the value it gets to $\mathscr{A}$. When $\mathscr{A}$ halts, $\mathscr{B}$ halts with the same output. If $\mathscr{B}$'s oracle responds to a query P by the value $[P(\vec{a}) \cdot a']$, then $\mathscr{B}$ sends $[P_x(\vec{a}) \cdot a'] = [a' \cdot \prod_{i=1}^n a_i^{x_i} a'] = \mathsf{NR}((a', \vec{a}), x)$ to $\mathscr{A}$ when the latter makes a query x, while if $\mathscr{B}$'s oracle responds to a query P by a uniformly random value, then $\mathscr{B}$ sends uniformly random values to $\mathscr{A}$. Hence, $\mathscr{B}$ simulates exactly the game defining the prf security of NR for the key $\vec{a}$ which is taken at random over $\mathbb{Z}_p^{n+1}$.

The only thing we need to prove is that all these queries are allowed to $\mathscr{B}$, meaning that for any distinct queries $x_1, \ldots, x_q$ of $\mathscr{A}$, polynomials $P_{x_1}, \ldots, P_{x_q}$ are linearly independent over $\mathbb{Z}_p$, which is straightforward. Hence, we have

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{NR}}(\mathscr{A}) \leq \mathbf{Adv}^{(n,1)\text{-}\mathsf{lip}}_{\mathbb{G}}(\mathscr{B}) \ .$$

Applying the LIP theorem, we then obtain

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{NR}}(\mathscr{A}) \leq n \cdot \mathbf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(\mathscr{C}) + O\big(\frac{nq}{p}\big) \ .$$

The running time of $\mathscr{C}$ is that of $\mathscr{A}$ plus the time to compute a polynomial number (in q and n) operations in $\mathbb{Z}_p$ and $\mathbb{G}$. The statistical loss could be removed since the polynomials queried are either equal or linearly independent, and we can therefore use a simpler perfect **TestLin** procedure which just checks equality in the proof of the LIP theorem is perfect.

This also proves the security of $\mathsf{WNR}^{\vec{w}}$ for any $\vec{w}$ such that w_i and $p-1$ are coprime, for $i = 0, \ldots, n$. Please also note that if $w_0 = 0$ (for instance for NR^*), we can do a similar proof using the $(n,1)$-LIP security of $\mathbb{G}$, by querying 1 at first and revealing $[a']$ as the generator instead of g. □

D.2 Weighted BMR

Lemma D.2. *Let $\mathbb{G} = \langle g \rangle$ be a group of prime order p and $\vec{w} = (w_1, \ldots, w_n) \in \mathbb{Z}_p^n$. Let $\mathsf{WBMR}^{\vec{w}}$ be the function defined via Table 2. Let $\mathscr{A}$ be an adversary against the PRF security of $\mathsf{WBMR}^{\vec{w}}$ that makes q oracle queries. Then we can construct an adversary $\mathscr{B}$ against the d-DDHI problem in $\mathbb{G}$ such that*

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{WBMR}^{\vec{w}}}(\mathscr{A}) \quad \leq \quad \left(\frac{p}{p-d-1}\right)^n \cdot n \cdot \left(\left(\frac{p}{p-1}\right)^2 \cdot d^2 \cdot \mathbf{Adv}^{d\text{-}\mathsf{ddhi}}_{\mathbb{G}}(\mathscr{C}) \ + \ O\big(\frac{ndq}{p}\big)\right) \quad .$$

Moreover, $\mathscr{B}$'s running time is that of $\mathscr{A}$ plus a time polynomial in n, d, q.

Proof. Once again, since we simply apply a permutation to the key space, the PRF security of WBMR is clearly implied by the PRF security of BMR. We just want to emphasize here that we can easily prove the security of all these PRFs (including BMR) by simply applying the LIP theorem.

Let $\mathscr{A}$ be an adversary against the PRF security of $\mathsf{WBMR}^{\vec{w}}$ that makes q oracle queries. We can assume without loss of generality that $\mathscr{A}$ never repeats a query. Then we can construct an adversary $\mathscr{B}$ against the (n,d)-LIP security of $\mathbb{G}$ that makes $q+1$ queries $P_0, P_1, \ldots, P_q$, defined as follows.

First, $\mathscr{B}$ queries polynomial $P_0(\vec{T}) = \prod_{i=1}^{n} \prod_{k\in\{0,\ldots,d\}} (T_i + w_i + k)$ and gets

$$h = \left[a' \prod_{i=1}^{n} \prod_{k\in\{0,\ldots,d\}} (a_i + w_i + k) \right]_g ,$$

that it sends to $\mathscr{A}$ as the generator used in the PRF construction (this is a generator with probability $(\frac{p-d-1}{p})^n$).

Next, $\mathscr{B}$ runs adversary $\mathscr{A}$. When the latter makes a query $x \in \{0,\ldots,d\}^n$, adversary $\mathscr{B}$ makes the query $P_x(T_1,\ldots,T_n) = \prod_{i=1}^{n} \prod_{k\in\{0,\ldots,d\}\setminus\{x\}} (T_i + w_i + k)$ to its oracle and returns the value he gets to $\mathscr{A}$. When $\mathscr{A}$ halts, $\mathscr{B}$ halts with the same output. If $\mathscr{B}$'s oracle responds to a query P by the value $[P(\vec{a})a']_g$, then $\mathscr{B}$ sends

$$[P_x(\vec{a})a']_g = \left[\prod_{i=1}^{n} \prod_{k\in\{0,\ldots,d\}\setminus\{x\}} (a_i + w_i + k)a' \right]_g = \left[\frac{P_0(\vec{a})}{\prod_{i=1}^{n}(a_i + w_i + x_i)} \right]_g = \left[\prod_{i=1}^{n} \frac{1}{a_i + w_i + x_i} \right]_h ,$$

to $\mathscr{A}$ when the latter makes a query x, while if $\mathscr{B}$'s oracle responds to a query P by a uniformly random value, then $\mathscr{B}$ sends uniformly random values to $\mathscr{A}$. Hence, $\mathscr{B}$ simulates exactly the game defining the PRF security of $\mathsf{WNR}^{\vec{w}}$ for the key $\vec{a}$ which is taken at random over $\mathbb{Z}_p^n$ and the generator h.

Then, since $P_x(T_1,\ldots,T_n) = \prod_{i=1}^{n} \prod_{k\in\{0,\ldots,d\}\setminus\{x\}} (T_i + w_i + k)$ is a degree at most d polynomial in each indeterminate, the only thing that remains to prove is that all these queries are allowed to $\mathscr{B}$, meaning that for any distinct queries $x^{(1)},\ldots,x^{(q)}$ of $\mathscr{A}$, polynomials $P_0, P_{x^{(1)}},\ldots,P_{x^{(q)}}$ are linearly independent over $\mathbb{Z}_p$. By contradiction, let us assume there is a sequence of distinct queries $x^{(1)},\ldots,x^{(q)}$ such that there is a linear combination:

$$\lambda_0 P_0 + \sum_{m=1}^{q} \lambda_k P_{x^{(k)}}$$

with $\lambda_k \neq 0$ for all $k = 1,\ldots,q$.

By evaluating this sum in $-x^{(k)} - \vec{w} = (-x_1^{(k)} - w_1, \ldots, -x_n^{(k)} - w_n) \in \mathbb{Z}_p^n$, we get $P_0(-x^{(k)} - \vec{w}) = 0$ for all k and $P_{x^{(i)}}(-x^{(k)} - \vec{w}) = 0$ for all $i \neq k$ and then $\lambda_k P_{x^{(k)}}(-x^{(k)} - \vec{w}) = 0$, which directly implies $\lambda_k = 0$, since $P_{x^{(k)}}(-x^{(k)} - \vec{w}) \neq 0$ for all $k = 1,\ldots,q$. Then, we have proven that $\lambda_k = 0$, for all $k = 1,\ldots,q$. Finally, since P_0 is not the zero polynomial and $\lambda_0 P_0 = 0$, we have also $\lambda_0 = 0$.

Then, for any distinct queries $x^{(1)},\ldots,x^{(q)}$ of $\mathscr{A}$, polynomials $P_0, P_{x^{(1)}},\ldots,P_{x^{(q)}}$ are linearly independent over $\mathbb{Z}_p$. Hence, we have

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{WBMR}^{\vec{w}}}(\mathscr{A}) \leq \left(\frac{p}{p-d-1}\right)^n \cdot \mathbf{Adv}^{(n,d)\text{-}\mathsf{lip}}_{\mathbb{G}}(\mathscr{B}) .$$

So applying the LIP theorem and Lemmas A.1 and A.3, we finally obtain

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{WBMR}^{\vec{w}}}(\mathscr{A}) \quad \leq \quad \left(\frac{p}{p-d-1}\right)^n \cdot n \cdot \left(\left(\frac{p}{p-1}\right)^2 \cdot d^2 \cdot \mathbf{Adv}^{d\text{-}\mathsf{ddhi}}_{\mathbb{G}}(\mathscr{C}) + O(\frac{ndq}{p}) \right) .$$

The running time of $\mathscr{C}$ is that of $\mathscr{A}$ plus the time to compute a polynomial number (in q, d and n) of operations in $\mathbb{Z}_p$ and $\mathbb{G}$.

Hence, in particular, setting the weights w_i to 0, we obtain a security proof for BMR under the d-DDHI assumption. □

E Proof of Theorems in Section 5

E.1 Proof of Theorem 5.1

In order to prove Theorem 5.1, we will prove the following lemma which states the exact security bound of the $\Phi_{\mathfrak{S}_n}$-prk-rka security of $\mathsf{WBMR}^{\mathsf{lin}}$.

Lemma E.1. *Let $\mathbb{G} = \langle g \rangle$ be a group of prime order p. Let $\mathscr{A}$ be a $\Phi_{\mathfrak{S}_n}$-restricted adversary against the PRF-RKA security of $\mathsf{WBMR}^{\mathsf{lin}}$ that makes q oracle queries. Then we can construct an adversary $\mathscr{B}$ against the $(n(d+1)-1)$-DDHI problem in $\mathbb{G}$ such that*

$$\mathbf{Adv}^{\mathsf{prf\text{-}rka}}_{\Phi_{\mathfrak{S}_n},\mathsf{WBMR}^{\mathsf{lin}}}(\mathscr{A}) \leq \left(\frac{p}{p-n(d+1)}\right)^n \cdot n \cdot \left(\left(\frac{p}{p-1}\right)^2 \cdot (n(d+1)-1) \cdot \mathbf{Adv}^{(n(d+1)-1)\text{-}\mathsf{ddhi}}_{\mathbb{G}}(\mathscr{B}) + O(\frac{ndq}{p})\right).$$

The running time of $\mathscr{B}$ is that of $\mathscr{A}$, plus the time to compute a polynomial number (in q, n and d) of operations in $\mathbb{Z}_p$ and $\mathbb{G}$.

To prove the above lemma, we prove a first statement using the $(n, n(d+1)-1)$-LIP assumption and then we apply the LIP theorem.

Lemma E.2. *Let $\mathbb{G} = \langle g \rangle$ be a group of prime order p. Let $\mathscr{A}$ be a $\Phi_{\mathfrak{S}_n}$-restricted adversary against the PRF-RKA security of $\mathsf{WBMR}^{\mathsf{lin}}$ that makes q oracle queries. Then we can design an adversary $\mathscr{B}$ against the $(n, n(d+1)-1)$-LIP security problem in $\mathbb{G}$ that makes q oracle queries such that*

$$\mathbf{Adv}^{\mathsf{prf\text{-}rka}}_{\Phi_{\mathfrak{S}_n},\mathsf{WBMR}^{\mathsf{lin}}}(\mathscr{A}) \leq \left(\frac{p}{p-n(d+1)}\right)^n \cdot \mathbf{Adv}^{(n,n(d+1)-1)\text{-}\mathsf{lip}}_{\mathbb{G}}(\mathscr{B}).$$

Moreover, the running time of $\mathscr{B}$ is the time to compute a polynomial number (in q, n and d) of operations in $\mathbb{Z}_p$ and $\mathbb{G}$.

Lemma E.2. Let $\mathscr{A}$ be a $\Phi_{\mathfrak{S}_n}$-restricted adversary against the PRF-RKA security of $\mathsf{WBMR}^{\mathsf{lin}}$ that makes q queries. We design an adversary $\mathscr{B}$ against the $(n, n(d+1)-1)$-LIP security of $\mathbb{G}$ that makes $q+1$ queries as follows.

First, $\mathscr{B}$ queries polynomial $P_0(\vec{T}) = \prod_{i=1}^n \prod_{k\in\{0,\ldots,n(d+1)-1\}}(T_i + k)$ and gets

$$h = \left[a' \prod_{i=1}^n \prod_{k\in\{0,\ldots,n(d+1)-1\}} (a_i + k)\right]_g,$$

that it sends to $\mathscr{A}$ as the generator used in the PRF construction (this is a generator with probability $(\frac{p-n(d+1)-1}{p})^n$).

Next, $\mathscr{B}$ runs adversary $\mathscr{A}$. When the latter makes a query (σ, x), $\mathscr{B}$ computes the polynomial

$$P_{\sigma,x}(\vec{T}) = \frac{\prod_{i=1}^n \prod_{k=0}^{n(d+1)-1}(T_i + k)}{\prod_{i=1}^n (T_{\sigma^{-1}(i)} + w_i + x_i)} = \frac{P_0(\vec{T})}{\prod_{i=1}^n (T_i + w_{\sigma(i)} + x_{\sigma(i)})}$$

and queries $P_{\sigma,x}$. This is possible since $\prod_{i=1}^n (T_i + w_{\sigma(i)} + x_{\sigma(i)})$ divides P_0 and since $P_{\sigma,x}$ is a degree $n(d+1)-1$ polynomial in each indeterminate. It then returns the value it gets to $\mathscr{A}$.

It is clear that if $\mathscr{B}$'s oracle returns uniformly random values, $\mathscr{B}$ simulates exactly the game $\mathrm{RKPRFRand}_{\mathsf{WBMR}^{\mathsf{lin}}}$, whereas if it returns $[P(\vec{a})a']$ for a query P, then $\mathscr{B}$ simulates exactly game $\mathrm{RKPRFReal}_{\mathsf{WBMR}^{\mathsf{lin}}}$ with the generator defined above. Indeed, in that case, let $h = [P_0(\vec{a})a']_g$ be the generator given to $\mathscr{A}$. Then for a query (σ, x), $[P_{\sigma,x}(\vec{a})a']_g = \left[\prod_{i=1}^n \frac{1}{\vec{a}[\sigma^{-1}(i)]+w_i+x_i}\right]_h = \mathsf{WBMR}^{\mathsf{lin}}(\sigma(\vec{a}), x)$, where $\mathsf{WBMR}^{\mathsf{lin}}$ is defined using the generator h. Then, the only thing that remains to prove is that all the polynomials queried by $\mathscr{B}$ to its oracle are linearly independent over $\mathbb{Z}_p$.

Hence, let us now prove that all polynomials $P_{\sigma,x}(\vec{T})$ corresponding to queries (σ, x) are linearly independent. By contradiction, let us assume that there exists a sequence of distinct queries $(\sigma^{(1)}, x^{(1)}), \ldots, (\sigma^{(q)}, x^{(q)})$ such that there exists a linear combination:

$$\lambda_0 P_0 + \sum_{k=1}^q \lambda_k \cdot P_{\sigma^{(k)},x^{(k)}} = 0$$

with $\lambda_k \neq 0$ for all $k = 1, \ldots, q$.

Then, by evaluating the above sum of polynomials on points

$$(-w_{\sigma^{(k)}(1)} - x^{(k)}_{\sigma^{(k)}(1)}, \ldots, -w_{\sigma^{(k)}(n)} - x^{(k)}_{\sigma^{(k)}(n)}),$$

since all the above polynomials except $P_{\sigma^{(k)},x^{(k)}}$ take the value 0 on these points, we obtain, for $k = 1, \ldots, q$, $\lambda_k = 0$. Then, we have $\lambda_0 P_0 = 0$ and since $P_0 \neq 0$, we have $\lambda_0 = 0$. Hence, all these polynomials are linearly independent. Lemma E.2 follows. □

Finally, Lemma E.1 easily follows by combining Lemma E.2, the LIP theorem and Lemmas A.1 and A.3.

Remark E.3. It is interesting to notice that, by slightly changing the construction, we can also achieve $\Phi_{\mathfrak{S}_n,+_k}$-RKA-security, where $\Phi_{\mathfrak{S}_n,+_k}$ is the class of functions $\sigma_{\vec{b}}$ that, when applied to a key $\vec{a} \in \mathbb{Z}_p^n$, leads to the key $(a_{\sigma^{-1}(1)} + b_1, \ldots, a_{\sigma^{-1}(n)} + b_n)$, for any $\sigma \in \mathfrak{S}_n$ and any $\vec{b} \in \{0, \ldots, k\}^n$. In other words, $\Phi_{\mathfrak{S}_n,+_k}$ also tolerates small additive factors in addition to permutations. Indeed, considering the function $\mathsf{WBMR}^{\mathsf{lin}}_k$ as the particular case of WBMR where $w_i = (i-1)(d+1)(k+1)$ for $i = 1, \ldots, n$ one can prove that this construction is $\Phi_{\mathfrak{S}_n,+_k}$-RKA-secure under the $(d(k+1)+k+(n-1)(d+1)(k+1))$-DDHI assumption in $\mathbb{G}$ and the proof follows closely the proof of Theorem 5.1. In particular, with $k = 0$, this is exactly the result from Theorem 5.1.

E.2 Proof of Theorem 5.2

We first prove a similar statement under a security notion termed *extended* key-collision security, whose security game is depicted in Figure 6, which extends the key-collision security notion defined in [ABPP14] but where **Initialize** also leaks Help_Φ to the adversary. Afterwards, we will generically reduce this notion to the SDL problem in $\mathbb{G}$.

proc Initialize	**proc RKFn**$(\vec{\phi}, x)$	**proc Finalize**$(\vec{\phi}_1, \vec{\phi}_2)$
$\vec{a} \xleftarrow{\$} \mathbb{Z}_p^n$	$y \leftarrow M(\vec{\phi}(\vec{a}), x)$	Return $(\vec{\phi}_1 \neq \vec{\phi}_2$ and $\vec{\phi}_1(\vec{a}) = \vec{\phi}_2(\vec{a}))$
Return $\mathsf{Help}_\Phi(\vec{a})$	Return y	

Figure 6: Game defining the extended Φ-key-collision security of a PRF M for a class Φ.

Specifically, we will show at first that:

$$\mathbf{Adv}^{\mathsf{prf\text{-}rka}}_{\Phi,F}(\mathscr{A}) \;\leq\; \mathbf{Adv}^{(n,d)\text{-}\mathsf{lip}}_{\mathbb{G}}(\mathscr{B}) + \mathbf{Adv}^{\mathsf{cr}}_{H}(\mathscr{C}) + 2 \cdot \mathbf{Adv}^{\mathsf{ext\text{-}kc}}_{\Phi,M}(\mathscr{D})\,. \tag{4}$$

Proof of this Intermediate Statement.

The proof is based on the sequence of games in Figure 7. We denote by $\textsc{Succ}_i$ the event that game G_i output takes the value 1. Boolean flags are assumed initialized to false. Games G_i, G_j are said to be identical until flag if their code differs only in statements that follow the setting of flag to true. We assume that adversary $\mathscr{A}$ never repeats an oracle query.

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

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

Game G_2 adds the boxed code which changes how the repetition of a value $\vec{C}$ is handled, by picking instead a random value from $\mathbb{G}^m \backslash E$ 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 extended Φ-key-collision security of M such that

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

Adversary $\mathscr{D}$ gets helper information $\mathsf{help}_\Phi = \mathsf{Help}_\Phi(\vec{a})$, then runs $\mathscr{A}$. When the latter makes a **RKFn**-query $(\vec{\phi}, x)$, adversary $\mathscr{D}$ computes $\vec{C} = \vec{\Omega}(\vec{\phi}(\vec{a}))$ using its helper information, $z = h(x, \vec{C})$ and $t = \mathsf{E}(z)$

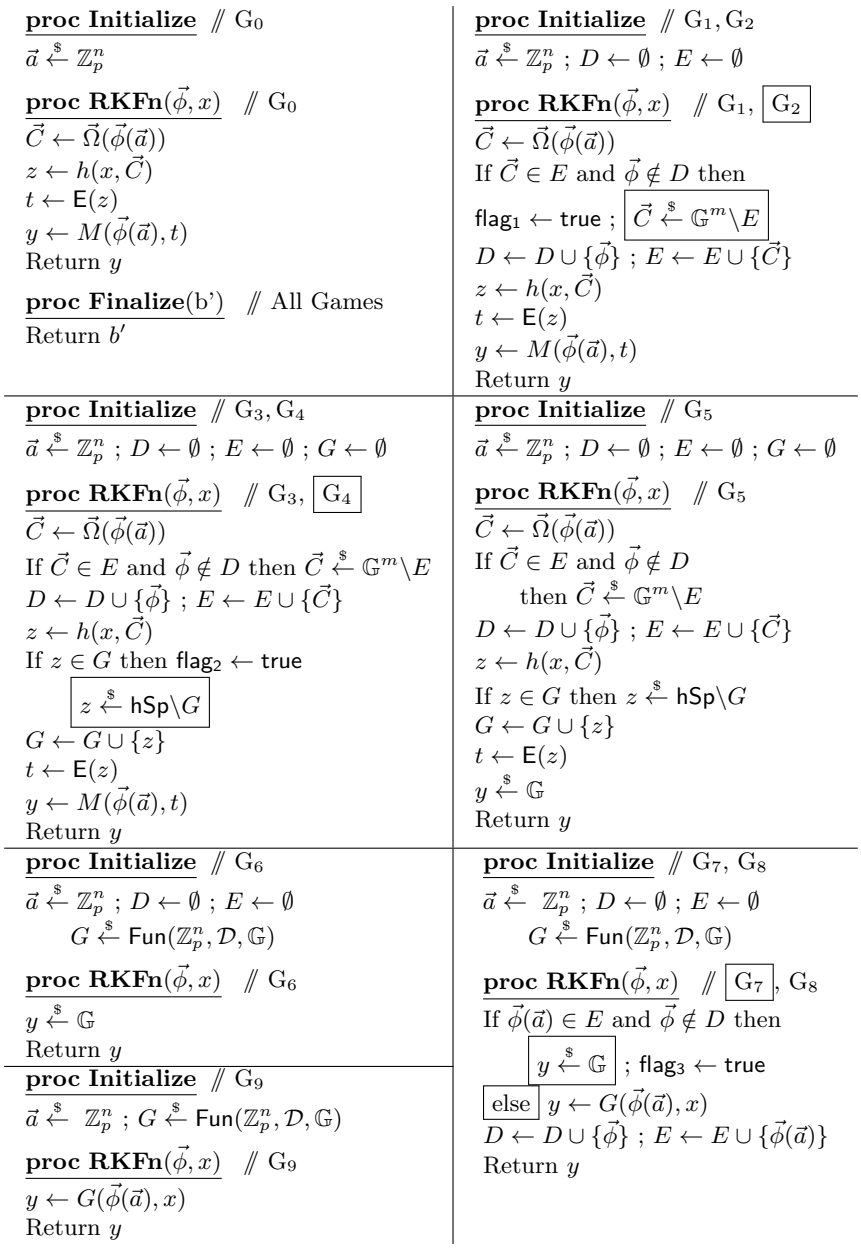

Figure 7: Games for the proof of Theorem 5.2.

and finally queries $(\vec{\phi}, t)$ to its oracle and sends the value it gets to $\mathscr{A}$. When $\mathscr{A}$ halts, $\mathscr{D}$ searches for two different RKD functions $\vec{\phi}_1, \vec{\phi}_2$ queried by $\mathscr{A}$ that lead to the same value $\vec{C}$ and returns these two functions if found. Since $\vec{\Omega}$ is a Algebraic Fingerprint, such two functions lead to the same key, so $\mathscr{D}$ wins if he finds such two functions.

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

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

Game G_4 adds the boxed code which changes how repetition of hash values is handled, by picking instead a random value z from $\mathsf{hSp}\backslash G$ 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

$$\Pr[\,\mathrm{SUCC}_3\,] \leq \Pr[\,\mathrm{SUCC}_4\,] + \Pr[\,E_2\,]$$

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 collision-resistance security of h such that

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

Adversary $\mathscr{C}$ starts by picking $\vec{a} \xleftarrow{\$} \mathbb{Z}_p^n$ and initializes $j \leftarrow 0$. It runs $\mathscr{A}$. When the latter makes a **RKFn**-query $(\vec{\phi}, x)$, adversary $\mathscr{C}$ responds via

$\vec{C} \leftarrow \vec{\Omega}(\vec{\phi}(\vec{a}))$
$j \leftarrow j+1$; $\vec{\phi}_j \leftarrow \vec{\phi}$; $x_j \leftarrow x$
If $\vec{C} \in E$ and $\vec{\phi} \notin D$ then $\vec{C} \xleftarrow{\$} \mathbb{G}^m \backslash E$ $\qquad (*)$
$D \leftarrow D \cup \{\vec{\phi}\}$; $E \leftarrow E \cup \{\vec{C}\}$
$\vec{C}_j \leftarrow \vec{C}$
$z \leftarrow h(x, \vec{C})$
$z_j \leftarrow z$
$t \leftarrow \mathsf{E}(z)$ $y \leftarrow M(\vec{\phi}(\vec{a}), t)$
Return y.

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

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

In game G_5, instead of returning the value $M(\vec{\phi}(\vec{a}), t)$, 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 (n,d)-LIP security of $\mathbb{G}$ such that

$$\Pr[\,\textsc{Succ}_4\,] \le \Pr[\,\textsc{Succ}_5\,] + \mathbf{Adv}^{(n,d)\text{-}\mathsf{lip}}_{\mathbb{G}}(\mathscr{B})\ .$$

Adversary $\mathscr{B}$ starts by querying polynomial 1 to $[a']$ and returns it as the generator used for the PRF to $\mathscr{A}$. Next, it initializes sets $D \leftarrow \emptyset$, $E \leftarrow \emptyset$, $G \leftarrow \emptyset$. Then $\mathscr{B}$ queries $\mathsf{help}_1, \ldots, \mathsf{help}_l$ such that $\mathsf{Help}_\Phi(\vec{a}) = ([a'\mathsf{help}_1(\vec{a})], \ldots, [a'\mathsf{help}_l(\vec{a})])$. Afterwards, it runs $\mathscr{A}$. When the latter makes an **RKFn**-query $(\vec{\phi}, x)$, $\mathscr{B}$ responds as follows. First, it computes $\vec{C} = \vec{\Omega}(\vec{\phi}(\vec{a}))$ using its helper information. Then, $\mathscr{B}$ checks if $\vec{C} \in E$ and $\vec{\phi} \in D$. If they do, $\mathscr{B}$ picks $\vec{C} \xleftarrow{\$} \mathbb{G}^m \backslash E$ at random. $\mathscr{B}$ then sets $D \leftarrow D \cup \{\vec{\phi}\}$ and $E \leftarrow E \cup \{\vec{C}\}$. Next, $\mathscr{B}$ computes $z \leftarrow h(x, \vec{C})$ and checks if $z \in G$. If it does, $\mathscr{B}$ picks $z \xleftarrow{\$} \mathsf{hSp} \backslash G$ at random. Notice that this step guarantees that all values z are distinct as long as $\mathscr{A}$ makes at most $|\mathsf{hSp}|$ queries. Finally, $\mathscr{B}$ sets $G \leftarrow G \cup \{z\}$, computes $t = \mathsf{E}(z)$ and makes the query $P_{\vec{\phi},t}$ to its oracle, where $P_{\vec{\phi},t}$ is the polynomial such that $M(\vec{\phi}(\vec{a}), t) = \left[P_{\vec{\phi},t}(\vec{a})\right]$, and returns the value it gets, which is either $\left[P_{\vec{\phi},t}(\vec{a})a'\right]$ or a uniformly random value, to $\mathscr{A}$. When $\mathscr{A}$ halts, $\mathscr{B}$ halts with the same output. It is clear that all the polynomials queried by $\mathscr{B}$ are linearly independent via the definition of E. Finally, it is clear that if $\mathscr{B}$'s oracle answers to a query P by the value $[P(\vec{a})a']$, then it simulates exactly game G_4 (where the generator used for the PRF construction is $[a']$, and if $\mathscr{B}$'s oracle gives uniformly random values, then the outputs are correctly simulated, since they are uniformly random. This concludes the proof of the above statement.

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 $\mathbb{G}$ even when two different queries result in the same key, games G_6 and G_7 are identical. Hence,

$$\Pr[\,\textsc{Succ}_6\,] = \Pr[\,\textsc{Succ}_7\,]\ .$$

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

$$\Pr[\,\textsc{Succ}_7\,] \le \Pr[\,\textsc{Succ}_8\,] + \Pr[\,E_3\,]$$

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 $\mathscr{D}$ attacking the extended Φ-key-collision security of M such that

$$\Pr[\,E_3\,] \le \mathbf{Adv}^{\mathsf{ext\text{-}kc}}_{\Phi,M}(\mathscr{D})\ .$$

Adversary $\mathscr{D}$ starts by initializing a list $\mathcal{L} \leftarrow$ empty list and by choosing an element $\vec{\psi}$ in Φ and by setting $\vec{\psi}_1 \leftarrow \vec{\psi}$ and $\vec{\psi}_2 \leftarrow \vec{\psi}$. Then, it runs $\mathscr{A}$. When the latter makes an **RKFn**-query $(\vec{\phi}, x)$, if $\vec{\psi}_1 = \vec{\psi}_2$, adversary $\mathscr{D}$ does the following: it first computes $\vec{C} \leftarrow \vec{\Omega}(\vec{\phi}(\vec{a}))$ using its helper information and searches for all tuples $(\vec{\phi}_i, \vec{C}_i) \in \mathcal{L}$ such that $\vec{C}_i = \vec{C}$. If it does find such tuples, it checks for all of them if $\vec{\phi} \neq \vec{\phi}_i$. If it does for a certain i, it sets $\vec{\psi}_1 \leftarrow \vec{\phi}_i$ and $\vec{\psi}_2 \leftarrow \vec{\phi}$. Then, it adds $(\vec{\phi}, \vec{C})$ to $\mathcal{L}$. Finally, $\mathscr{D}$ picks $y \xleftarrow{\$} \mathbb{G}$ at random, returns y to $\mathscr{A}$. When $\mathscr{A}$ halts, $\mathscr{D}$ halts and outputs $(\vec{\psi}_1, \vec{\psi}_2)$. If the execution of $\mathscr{A}$ sets flag_3 to true, then $\mathscr{A}$ has queried $\vec{\phi}_1 \neq \vec{\phi}_2$ such that $\vec{\phi}_1(\vec{a}) = \vec{\phi}_2(\vec{a})$ and assuming it has first queried $\vec{\phi}_1$, when $\mathscr{D}$ computes $\vec{\Omega}(\vec{\phi}_2(\vec{a}))$ and checks if this value is already in $\mathcal{L}$, it finds that this value matches $\vec{\Omega}(\vec{\phi}_1(\vec{a}))$ and since $\vec{\phi}_1 \neq \vec{\phi}_2$, it sets $\vec{\psi}_1 = \vec{\phi}_1$ and $\vec{\psi}_2 = \vec{\phi}_2$, so $\mathscr{D}$ wins. Then, we have

$$\Pr[E_3] \leq \mathbf{Adv}^{\mathsf{ext\text{-}kc}}_{\Phi,M}(\mathscr{D}) .$$

Since 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,

$$\Pr[\textsc{Succ}_8] = \Pr[\textsc{Succ}_9] .$$

Equation (4) now follows by combining the bounds arising in the different game hops. □

From Extended-Key-Collision Security to SDL.

In the following lemma, we reduce the extended-kc security to the hardness of the SDL problem in $\mathbb{G}$.

Lemma E.4. *Let $\mathbb{G}$ be a group of prime order p. Let M: $\mathbb{Z}_p^n \times \mathcal{D} \to \mathbb{G}$ be a function and Φ be a class of RKD functions. Let $\mathscr{A}$ be an adversary against the Φ-key-collision security of M that is given $\mathsf{Help}_\Phi(\vec{a})$ and that makes q oracle queries in $\mathcal{S}$. Let d be an integer such that, for any polynomial in $\{\mathsf{help}_1, \ldots, \mathsf{help}_l\} \cup \{P_{x,\vec{\phi}} \mid x \in \mathcal{S}, \vec{\phi} \in \Phi\}$, its maximum degree in any indeterminate is at most d. Then, we can construct an adversary $\mathscr{B}$ against the* SDL *problem in $\mathbb{G}$ such that*

$$\mathbf{Adv}^{\mathsf{ext\text{-}kc}}_{\Phi,M}(\mathscr{A}) \;\leq\; \frac{p}{p-(n-1)d} \cdot n \cdot \mathbf{Adv}^{\mathbb{G}\text{-}\mathsf{sdl}}_{\mathscr{B}}(\,). \tag{5}$$

Moreover, the running time of $\mathscr{B}$ is that of $\mathscr{A}$ plus the time to compute a polynomial number (in n, d and q) of operations in $\mathbb{Z}_p$ and in $\mathbb{G}$.

Lemma E.4. Adversary $\mathscr{B}$ receives a d-SDL tuple $([1], [a], \ldots, [a^d])$ where $a \xleftarrow{\$} \mathbb{Z}_p$ and where $[1]$. Adversary $\mathscr{B}$ then picks $j \xleftarrow{\$} \{1, \ldots, n\}$ at random.

$\mathscr{B}$ start by picking a' at random in $\mathbb{Z}_p$ and by sending $[a']$ to $\mathscr{A}$ as the generator used for the PRF construction. Then $\mathscr{B}$ picks $a_i \xleftarrow{\$} \mathbb{Z}_p$ for $i = 1, \ldots, n, i \neq j$ at random. Implicitly, $\mathscr{B}$ sets $a_j = a$.

Then, $\mathscr{B}$ computes the helper information $\mathsf{Help}_\Phi(\vec{a}) = ([\mathsf{help}_1(\vec{a}) \cdot a'], \ldots, [\mathsf{help}_l(\vec{a}) \cdot a'])$ using its d-SDL tuple and the known values a', a_i for $i \neq j$ and sends this helper information to $\mathscr{A}$. When $\mathscr{A}$ makes a query $(\vec{\phi}, x)$, $\mathscr{B}$ wants to return to $\mathscr{A}$ the value $M(\vec{\phi}(\vec{a}), x) = \left[P_x(\vec{\phi}(\vec{a}))a'\right] = \left[P_{\vec{\phi},x}(\vec{a})a'\right]$.

To do so, $\mathscr{B}$ starts by computing $P_{\vec{\phi},x}(a_1, \ldots, a_{j-1}, T_j, a_{j+1}, \ldots, a_n)$ using chosen values $a_i, i \neq j$. It then gets a degree at most d polynomial P_j in T_j. Hence, using its d-SDL tuple and the known value a', it can now easily compute $[P_j(a_j)a'] = \left[P_{\vec{\phi},x}(\vec{a})a'\right]$ and returns this value to $\mathscr{A}$.

At the end, $\mathscr{A}$ sends $(\vec{\phi}_1, \vec{\phi}_2)$ to $\mathscr{B}$ and wins if $\vec{\phi}_1 \neq \vec{\phi}_2$ and $\vec{\phi}_1(\vec{a}) = \vec{\phi}_2(\vec{a})$. Hence, $\mathscr{B}$ computes $\vec{\psi} = \vec{\phi}_1 - \vec{\phi}_2$. Since $\vec{\phi}_1 \neq \vec{\phi}_2$, there is at least a component of $\vec{\psi}$ which is a non-zero multivariate polynomial. Let us call this component $R \in \mathbb{Z}_p[T_1, \ldots, T_n]$. Since we chose j at random in $\{1, \ldots, n\}$, the indeterminate T_j appears in R with probability at least $\frac{1}{n}$. $\mathscr{B}$ now evaluates R in $(a_1, \ldots, a_{j-1}, T_j, a_{j+1}, \ldots, a_n)$ to obtain a univariate polynomial $S \in \mathbb{Z}_p[T_j]$. This polynomial is a non-zero polynomial with probability at least $\frac{p-(n-1)d}{p}$, via the Schwartz-Zippel lemma.

Finally, $\mathscr{B}$ simply factorizes S (for instance using the Kedlaya-Umans algorithm), and outputs the unique root r of S such that $[r] = [a]$. Lemma E.4 follows. □

The proof of Theorem 5.2 follows by combining Equation (4) with Equation (5). □

E.3 Proof of Linearly Independence Property for Section 5.2

Let $P_{\sigma\circ\vec{\phi},x}(\vec{T}) = \prod_{i=1}^{n} \phi_i(\vec{T})^{p_{\sigma(i)}\cdot x_{\sigma(i)}}$ for $\sigma \circ \vec{\phi} \in \Phi_{\mathfrak{S}_n,d}$ and $x \in \{0,1\}^n$. Let $\mathcal{S} = \{11 \,\|\, z \mid z \in \{0,1\}^{n-2}\}$. The only thing we need to prove is that, for any sequence $(x_1, \sigma_1\circ\vec{\phi}_1), \ldots, (x_q, \sigma_q\circ\vec{\phi}_q)$, polynomials 1, T_i^j for $i = 1,\ldots,n$ and $j = 1,\ldots,d$ (revealed in the helper information) and polynomials $P_{\sigma_1\circ\vec{\phi}_1,x_1}, \ldots, P_{\sigma_q\circ\vec{\phi}_q,x_q}$ are linearly independent, as long as $x_1,\ldots,x_q$ are all distinct in $\mathcal{S}$.

By contradiction, let us assume that there exists a sequence of distinct polynomials $P_1,\ldots,P_q$, where for $k = 1,\ldots,q$, $P_k = P_{\sigma\circ\vec{\phi},x}$ for distinct $x \in \mathcal{S}$ or $P_k = T_i^j$ for some $(i,j) \in \{1,\ldots,n\} \times \{1,\ldots,d\}$ or $P_k = 1$, such that:

$$\sum_{k=1}^{q} \lambda_k \cdot P_k = 0$$

with $\lambda_k \neq 0$ for all k. Since polynomials T_i^j, for $i = 1,\ldots,n$ and $j = 1,\ldots,d$ and polynomial 1 are clearly linearly independent over $\mathbb{Z}_p$, then there is at least one polynomial P_k in the above sum such that P_k corresponds to $P_{\sigma\circ\vec{\phi},x}$ for a query $(\sigma\circ\vec{\phi},x)$ with $x \in \mathcal{S}$.

Let $P_{\sigma\circ\vec{\phi},x}(\vec{T})$ denote a polynomial in the above sum such that $x \in \mathcal{S}$ has the maximum Hamming weight amongst all the bitstrings $z \in \mathcal{S}$ such that there exists $k \in \{1,\ldots,q\}$ and $(\sigma\circ\vec{\phi}) \in \Phi_{\mathfrak{S}_n,d}$ such that $P_k = P_{\sigma\circ\vec{\phi},z}$. Since $x \in \mathcal{S}$, $\mathsf{hw}(x) \geq 2$. Since ϕ_i is a non-constant polynomial for all $i = 1,\ldots,n$, letting d_i denote the degree of ϕ_i, the monomial $\prod_{i=1}^{n} T_i^{d_i p_{\sigma(i)}\cdot x_{\sigma(i)}}$ appears in $P_{\sigma\circ\vec{\phi},x}(\vec{T})$. Hence, since $\sum_{k=1}^{q} \lambda_k \cdot P_k$ is the zero polynomial and $\lambda_k \neq 0$ for all k, there exists another query $P_{\sigma'\circ\vec{\phi}',x'}$ such that the monomial $\prod_{i=1}^{n} T_i^{d_i p_{\sigma(i)}\cdot x_{\sigma(i)}}$ also appears in $P_{\sigma'\circ\vec{\phi}',x'}$. Hence $\mathsf{hw}(x') \geq \mathsf{hw}(x)$.

But x has maximum Hamming weight by assumption, so $\mathsf{hw}(x') \leq \mathsf{hw}(x)$ and then $\mathsf{hw}(x') = \mathsf{hw}(x)$. We note that the degree in T_i in this monomial is either 0, if $x_{\sigma(i)} = 0$, or $d_i p_{\sigma(i)}$ otherwise, where $1 \leq d_i \leq d$. But by definition of $P_{\sigma'\circ\vec{\phi}',x}$, the possible degrees in T_i in a monomial that appears in $P_{\sigma'\circ\vec{\phi}',x}$ is either 0 if $x'_{\sigma'(i)} = 0$ or a non-zero multiple $l \cdot p_{\sigma'(i)}$ of $p_{\sigma'(i)}$ with $1 \leq l \leq d$ otherwise.

However, $p_1,\ldots,p_n$ are clearly coprime and $d < p_1 < \ldots < p_n$ by assumption. Hence, for all $i = 1,\ldots,n$ such that $x_{\sigma(i)} = 1$, we have, by Gauss's Lemma, $p_{\sigma'(i)} = p_{\sigma(i)}$ and $x_{\sigma(i)} = 1$ implies $x'_{\sigma'(i)} = 1$. Finally, since $p_{\sigma'(i)} = p_{\sigma(i)}$ implies $\sigma'(i) = \sigma'(i)$ for all i such that $x_{\sigma(i)} = 1$, and since they have both same Hamming weight, it implies that $x = x'$ and then to have such a linear combination, we need to use twice the same entry $x \in \mathcal{S}$, which is not possible.

The linear independence property follows. □

F Other Applications to Related-Key Security

In this appendix, we describe how our new framework can be used to build RKA-secure PRFs for two other classes.

The first class we address, in Appendix F.1, is the class Φ_d of degree at most d univariate polynomials. We use our framework to show that for any choice of weights, WNR is Φ_d-RKA-secure. This extends the result from [ABPP14] where NR* was proven Φ_d-RKA-secure and proves that, in particular, NR also is Φ_d-RKA-secure.

The second class we address, in Appendix F.2, is the class $\Phi_{n+1,\mathsf{multi\text{-}aff}}$ of non-constant affine multivariate functions. However, this construction is of limited interest and should only be seen as a first step towards building PRFs for large classes of RKD functions.

All the proofs of statements in these Appendices F.1 and F.2 are detailed in Appendix F.3.

F.1 RKA-PRFs for Univariate Polynomial Functions

Here, we apply our framework to WNR, defined via Table 2, for the class of RKD functions $\Phi_d = \{\vec{\phi}\colon \mathcal{K} \to \mathcal{K} \mid \vec{\phi}_i : \vec{T} \mapsto \sum_{j=0}^{d} \alpha_{i,j} \cdot T_i^j, (\alpha_{i,1},\ldots,\alpha_{i,d}) \neq 0^d, \forall i = 0,\ldots,n\}$, for any choice of weights. In what follows, we assume that $w_0 \neq 0$, but similar results can easily be proven if $w_0 = 0$.

Hence, we recover in particular all the results from [ABPP14] in this subsection, under the same assumption, but also prove that, even if the previous framework could not deal with NR but only with NR*, both constructions can lead to a Φ_d-RKA-secure PRF.

In order to apply our framework to WNR and Φ_d, we need to define associated algebraic fingerprint, helper function, collision-resistant hash function and expansion function. We define these as follows:

- $[a'] \in \mathbb{G}$ is the generator used for the PRF construction
- $\vec{\Omega}$: $\vec{a} \in \mathbb{Z}_p^n \mapsto ([a_1 \cdot a'], \ldots, [a_n \cdot a']) \in \mathbb{G}^n$
- Help_{Φ_d}: $\vec{a} \in \mathbb{Z}_p^n \mapsto ([a'], [a_1 \cdot a'], \ldots, [a_1^d \cdot a'], \ldots, [a_n \cdot a'], \ldots, [a_n^d \cdot a']) \in \mathbb{G}^{nd+1}$
- h can be any collision-resistant hash function h: $\{0,1\}^n \times \mathbb{G}^n \to \{0,1\}^{n-2}$
- E: $z \in \{0,1\}^{n-2} \mapsto 11 \,\|\, z \in \{0,1\}^n$

We just need to prove that E satisfies the linear independence property required to apply the framework, which is done in Appendix F.3.

Finally, by combining the above statements, Theorem 5.2 and the LIP theorem, we obtain the following theorem.

Theorem F.1. *Let $\mathbb{G} = \langle g \rangle$ be a group of prime order p. Using above definitions, let us define F: $\mathbb{Z}_p^{n+1} \times \{0,1\}^n \to \mathbb{G}$ by*

$$F(\vec{a}, x) = \mathsf{WNR}^{\vec{w}}(\vec{a}, \mathsf{E}(h(x, \vec{\Omega}(\vec{a}))))$$

for all $\vec{a} \in \mathbb{Z}_p^{n+1}$ and $x \in \{0,1\}^n$. Let $m = \max(w_0, \ldots, w_n)$ be the maximum component of $\vec{w}$. Let $\mathscr{A}$ be a Φ_d-restricted adversary against the PRF-RKA security of F that makes $q \leq |\{0,1\}^{n-2}|$ oracle queries. Then we can construct an adversary $\mathscr{B}$ against the md-DDHI problem in $\mathbb{G}$, an adversary $\mathscr{C}$ against the collision-resistance security of h, an adversary $\mathscr{D}$ against the md-SDL problem in $\mathbb{G}$ such that

$$\mathbf{Adv}^{\mathsf{prf\text{-}rka}}_{\Phi_d,F}(\mathscr{A}) \;\leq\; (n+1) \cdot md \cdot \frac{p}{p-1} \cdot \mathbf{Adv}^{md\text{-}\mathsf{ddhi}}_{\mathbb{G}}(\mathscr{B}) + \mathbf{Adv}^{\mathsf{cr}}_{h}(\mathscr{C}) + (n+1) \cdot \mathbf{Adv}^{md\text{-}\mathsf{sdl}}_{\mathbb{G}}(\mathscr{D}) + \frac{O(qnd)}{p} \,.$$

$\mathscr{C}$ has the same running time as $\mathscr{A}$. The running time of $\mathscr{B}$ and $\mathscr{D}$ is that of $\mathscr{A}$, plus the time required to compute a polynomial number (in n, d, m and q) of operations in $\mathbb{Z}_p$ and in $\mathbb{G}$.

F.2 RKA-PRF for Affine Multivariate Functions

In order to deal with larger classes of related-key attacks, it would be important to consider multivariate RKD functions in which the attacker is allowed to mix different components of the secret key. In fact, the d-Linear Weighted NR PRF construction given in the previous section seems like a plausible candidate if we assume that the RKD functions that are being applied to each sub key are linearly independent. However we have not been able to prove this to be the case or to disprove it. We construct an alternative PRF that can be shown to be RKA-secure against an adversary that can modify the key by applying functions from $\Phi_{n+1,\mathsf{multi\text{-}aff}}$, defined as:

$$\{(\boldsymbol{M}, \vec{b}) \mid (\boldsymbol{M}, \vec{b}) \in \mathbb{Z}_p^{(n+1)\times(n+1)} \times \mathbb{Z}_p^{n+1} \text{ s.t. } \vec{M}_i \neq 0^{n+1}, \forall i = 0, \ldots, n\}$$

where $\vec{M}_i$ denote the i-th row of $\boldsymbol{M}$, for $i = 0, \ldots, n$. Hence, for a key $\vec{a} = (a_0, \ldots, a_n) \in \mathbb{Z}_p^{n+1}$, applying an RKD function $(\boldsymbol{M}, \vec{b}) \in \Phi_{n+1,\mathsf{multi\text{-}aff}}$, leads to the key $\boldsymbol{M} \cdot \vec{a} + \vec{b} = (\vec{M}_0 \cdot \vec{a} + b_0, \ldots, \vec{M}_n \cdot \vec{a} + b_n) \in \mathbb{Z}_p^{n+1}$.

Since the new construction is based on the Weighted NR PRF with exponential weights $w_i = 2^i$, it is of limited interest given that its security relies on an assumption whose input is of exponential size. Moreover, in settings in which it is acceptable to have security assumptions with exponential-size inputs, much simpler constructions are possible. Nevertheless, we still believe that the new construction provides a first small step towards building PRFs for large classes of RKD functions such as multivariate affine functions or even multivariate polynomial functions.

To construct a $\Phi_{n+1,\mathsf{multi\text{-}aff}}$-RKA-secure PRF, we apply our framework to a particular case of the Weighted NR PRF, defined as follows:

Exponential Weighted NR PRF. Let $\mathbb{G} = \langle g \rangle$ be a group of prime order p. Let $d \geq 1$. We define $\mathsf{WNR}^{\mathsf{exp}}$ as the particular case of WNR, where $w_i = 2^i$, for $i = 0, \ldots, n$. Please refer to Table 2 for details.

In order to apply our framework to $\mathsf{WNR}^{\mathsf{exp}}$ and $\Phi_{n+1,\mathsf{multi\text{-}aff}}$, we need to define associated algebraic fingerprint, helper function, collision-resistant hash function and expansion function. We define these as follows:

- $\vec{\Omega}$: $\vec{a} \in \mathbb{Z}_p^n \mapsto ([a_1 \cdot a'], \ldots, [a_n \cdot a']) \in \mathbb{G}^n$

- $\mathsf{Help}_{\Phi_{n+1,\text{multi-aff}}}$: $\vec{a} \in \mathbb{Z}_p^n \mapsto ([a'], [a_1 \cdot a'], \ldots, [a_n \cdot a']) \in \mathbb{G}^{n+1}$
- h can be any collision-resistant hash function h: $\{0,1\}^n \times \mathbb{G}^n \to \{0,1\}^{n-2}$
- E: $z \in \{0,1\}^{n-2} \mapsto 11 \,\|\, z \in \{0,1\}^n$

We just need to prove that E satisfies the linear independence property required to apply the framework, which is done in Appendix F.3.

Finally, combining the above statements, Theorem 5.2 and the LIP theorem, we obtain the following theorem.

Theorem F.2. *Let $\mathbb{G} = \langle g \rangle$ be a group of prime order p. Using above definitions, let us define F: $\mathbb{Z}_p^{n+1} \times \{0,1\}^n \to \mathbb{G}$ by*

$$F(\vec{a}, x) = \mathsf{WNR}^{\mathsf{exp}}(\vec{a}, \mathsf{E}(h(x, \vec{\Omega}(\vec{a}))))$$

for all $\vec{a} \in \mathbb{Z}_p^{n+1}$ and $x \in \{0,1\}^n$. Let $\mathscr{A}$ be a $\Phi_{n+1,\textit{multi-aff}}$-restricted adversary against the PRF-RKA security of F that makes $q \leq |\{0,1\}^{n-2}|$ oracle queries. Then we can construct an adversary $\mathscr{B}$ against the 2^{n+1}-DDHI problem in $\mathbb{G}$, an adversary $\mathscr{C}$ against the collision-resistance security of h, an adversary $\mathscr{D}$ against the 2^{n+1}-SDL problem in $\mathbb{G}$ such that

$$\mathbf{Adv}^{\mathsf{prf\text{-}rka}}_{\Phi_{n+1,\textit{multi-aff}},F}(\mathscr{A}) \;\leq\; (n+1) \cdot 2^{n+1} \cdot \left(\frac{p}{p-1}\right)^2 \cdot \mathbf{Adv}^{2^{n+1}\text{-}\mathsf{ddhi}}_{\mathbb{G}}(\mathscr{B}) + \mathbf{Adv}^{\mathsf{cr}}_{h}(\mathscr{C}) + (n+1) \cdot \mathbf{Adv}^{2^{n+1}\text{-}\mathsf{sdl}}_{\mathbb{G}}(\mathscr{D}) + \frac{O(2^n nq)}{p} \,.$$

The running time of $\mathscr{B}$ is that of $\mathscr{A}$, plus the time to compute $2^{O(n)}$ operations in $\mathbb{Z}_p$ and $\mathbb{G}$. $\mathscr{C}$ has the same running time as $\mathscr{A}$. The running time of $\mathscr{D}$ is that of $\mathscr{A}$ plus the time to make $O(q \cdot \mathsf{M}(2^{n+1}))$ operations in $\mathbb{Z}_p$ where $\mathsf{M}(d)$ is the complexity of the multiplication of two degree at most d polynomials.

Remark F.3. The above construction could be easily generalized to multivariate polynomial RKD functions of degree at most d, by properly changing the exponential sequence used in the construction to $w_i = (d+1)^i$ in order to guarantee that, for any two different values of x and x' and any $d_i, d'_i \in \{1, \ldots, d\}$ for $i = 0, \ldots, d$, the sums $w_0 d_0 + \sum_{i=1}^n w_i d_i x_i$ and $w_0 d'_0 + \sum_{i=1}^n w_i d'_i x'_i$ are distinct. We can then use exactly the same helper function than in Section 5.2.

F.3 Proof of Linearly Independence Properties for Section F.1 and Section F.2

For Univariate Polynomials. Let $P_{\vec{\phi},x}(\vec{T}) = \phi_0(\vec{T})^{w_0} \cdot \prod_{i=1}^n \phi_i(\vec{T})^{w_i x_i}$ for $\vec{\phi} \in \Phi_d$ and $x \in \{0,1\}^n$. Let $\mathcal{S} = \{11 \,\|\, h \mid z \in \{0,1\}^{n-2}\}$. The only thing we need to prove is that, for any sequence $(x_1, \vec{\phi}_1), \ldots, (x_q, \vec{\phi}_q)$, polynomials $1, T_i^j$ for $i = 0, \ldots, n$ and $j = 1, \ldots, d$ and polynomials $P_{\vec{\phi}_1,x_1}, \ldots, P_{\vec{\phi}_q,x_q}$ are linearly independent, as long as $x_1, \ldots, x_q$ are distinct in $\mathcal{S}$.

By contradiction, let us assume that there exists a sequence of distinct polynomials $P_1, \ldots, P_q$, where for $k = 1, \ldots, q$, $P_k = P_{\vec{\phi},x}$ for distinct $x \in \mathcal{S}$ or $P_k = T_i^j$ for some $(i,j) \in \{0, \ldots, n\} \times \{1, \ldots, d\}$ or $P_k = 1$, such that:

$$\sum_{k=1}^{q} \lambda_k \cdot P_k = 0$$

with $\lambda_k \neq 0$ for all k.

Since polynomials T_i^j, for $i = 0, \ldots, n$ and $j = 1, \ldots, d$ and polynomial 1 are clearly linearly independent over $\mathbb{Z}_p$, there is at least one polynomial P_k in the above sum such that P_k corresponds to $P_{\vec{\phi},x}$ for a query $(\vec{\phi}, x)$ with $x \in \mathcal{S}$.

We now consider an arbitrary monomial $T_0^{z_0} \cdots T_n^{z_n}$ appearing in at least one of the polynomials in this combination and such that z has highest Hamming weight. It is clear that $\mathsf{hw}(z_1 \,\|\, \ldots \,\|\, z_n) \geq 2$, since the Hamming weight of $x \in \mathcal{S}$ is at least 2. But, since the sum is the zero polynomial, there must exist at least two distinct polynomials $P_{\vec{\phi}_1,x_1}$ and $P_{\vec{\phi}_2,x_2}$ containing this monomial $T_0^{z_0} \cdots T_n^{z_n}$ in the above sum.

Let $\hat{z}$ denote the n-bit string such that $\hat{z}_i = 0$ if $z_i = 0$, while $\hat{z}_i = 1$ otherwise, for $i = 1, \ldots, n$. It is clear that $\mathsf{hw}(\hat{z}) \geq 2$ and then $\hat{z} \in \mathcal{S}$. Also, since $\hat{z}$ has the highest possible Hamming weight, $x_1 = x_2 = \hat{z}$ (from the definitions of $P_{\vec{\phi}_1,x_1}$ and $P_{\vec{\phi}_2,x_2}$ and since the first components of $\vec{\phi}_1$ and $\vec{\phi}_2$, which are the

polynomials that apply to A_0, have degree at least 1). This means $\hat{z} \in \mathcal{S}$ has been used twice, which is forbidden. Hence, all the polynomials are linearly independent.

The linear independence property follows.

For Multivariate Affine Functions. Let $P_{(\boldsymbol{M},\vec{b}),x}(\vec{T}) = (\vec{M}_0 \cdot \vec{T} + b_0) \prod_{i=1}^{n} (\vec{M}_i \cdot \vec{T} + b_i)^{2^i \cdot x_i}$ for $(\boldsymbol{M}, \vec{b}) \in \Phi_{n+1,\mathsf{multi\text{-}aff}}$ and $x \in \{0,1\}^n$, where $\vec{M}_i$ denote the i-th row of matrix $\boldsymbol{M}$. Let $\mathcal{S} = \{11 \,\|\, h \mid z \in \{0,1\}^{n-2}\}$. The only thing we need to prove is that, for any sequence $(x_1, (\boldsymbol{M}_1, \vec{b}_1)), \ldots, (x_q, (\boldsymbol{M}_q, \vec{b}_q))$ with, for $k = 1, \ldots, q$, x_k all distinct in $\mathcal{S}$ and $(\boldsymbol{M}_k, \vec{b}_k) \in \Phi_{n+1,\mathsf{multi\text{-}aff}}$, polynomials 1, T_i for $i = 0, \ldots, n$ and polynomials $P_{(\boldsymbol{M}_1,\vec{b}_1),x_1}, \ldots, P_{(\boldsymbol{M}_q,\vec{b}_q),x_q}$ are linearly independent.

Polynomial 1 is a degree 0 polynomial and polynomials T_i, for $i = 0, \ldots, n$ are degree 1 polynomials, and they are all linearly independent. Then, we just prove that for any $((\boldsymbol{M}, \vec{b}), x)$ and $((\boldsymbol{M}', \vec{b}'), x')$ with $x, x' \in \mathcal{S}$ and $x \neq x'$, the degrees of polynomials $P_{(\boldsymbol{M},\vec{b}),x}$ and $P_{(\boldsymbol{M}',\vec{b}'),x'}$ are always distinct and greater than 2.

Let $((\boldsymbol{M}, \vec{b}), x)$ with $x \in \mathcal{S}$, then $P_{(\boldsymbol{M},\vec{b}),x}(\vec{T}) = (\vec{M}_0 \cdot \vec{T} + b_0) \prod_{i=1}^{n} (\vec{M}_i \cdot \vec{T} + b_i)^{2^i \cdot x_i}$. Since for any $i = 0, \ldots, n$, $\vec{M}_i \neq 0^{n+1}$, $\vec{M}_i \cdot \vec{T} + b_i$ is always a multivariate polynomial of degree 1, $(\vec{M}_i \cdot \vec{T} + b_i)^{2^i}$ is always a polynomial of degree 2^i. Hence, $P_{(\boldsymbol{M},\vec{b}),x}(\vec{T})$ is a multivariate polynomial of degree $\sum_{i=0}^{n} 2^i x_i$, which is clearly greater than 2 since $\mathsf{hw}(x) \geq 2$ for any $x \in \mathcal{S}$.

Finally, since the entries x used in distinct queries has to be always distinct, and since for any $x \neq x'$, $\sum_{i=0}^{n} 2^i x_i \neq \sum_{i=0}^{n} 2^i x'_i$ by the uniqueness of the binary decomposition, the degrees of polynomials $P_{(\boldsymbol{M},\vec{b}),x}$ and $P_{(\boldsymbol{M}',\vec{b}'),x'}$ are always distinct and at least 2 for any queries $((\boldsymbol{M}, \vec{b}), x)$ and $((\boldsymbol{M}', \vec{b}'), x')$ with $x, x' \in \mathcal{S}$ and $x \neq x'$.

The linear independence property follows.

G A Further Generalization of the Framework

G.1 Previous Frameworks for Building RKA-Secure PRFs

In [BC10], Bellare and Cash introduced a framework that allowed to transform a key-malleable PRF into an RKA-secure PRF. Their framework however had two shortcomings. The first one was the fact that they only consider claw-free function classes, i.e., such that there does not exist $\phi_1, \phi_2 \in \Phi$ so that $\phi_1 \neq \phi_2$ but $\phi_1(K) = \phi_2(K)$ for some key $K \in \mathcal{K}$. The second one was that they required a form of key malleability from the underlying PRF, which restricted the set of classes of RKD functions to which their framework could be applied. To address these shortcomings, in [ABPP14], Abdalla, Benhamouda, Passelègue, and Paterson generalized this framework. In particular, by applying their framework to NR^*, the authors showed how to obtain a Φ_d-RKA-PRF under the d-DDHI assumption.

In a nutshell, their framework first consists in building a PRF M, which verifies a weaker notion of security, called unique-input-PRF-RKA security or UI-PRF-RKA security. This notion is similar to PRF-RKA security, recalled in Section 2, except the adversary is restricted to use different inputs x for each query. This PRF M is also supposed to be key-collision and statistical-key-collision secure, meaning that it is hard to find two functions $\phi_1, \phi_2 \in \Phi$ such that $\phi_1(K) = \phi_2(K)$, even with access to an oracle $(\phi, x) \mapsto f(\phi(K), x)$, when $f = M$ (key-collision security), and when f is a random function (statistical key-collision security).

Then, the framework consists in transforming this UI-PRF-RKA-secure PRF M into an rka-secure PRF F, as follows:

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

where H is a collision-resistant hash function, and the vector $\vec{\omega}$ is a strong key fingerprint, i.e., it is a vector of inputs such that $M(K, \vec{\omega})$ completely defines K. This transform is exactly the one given by Bellare and Cash in [BC10]. Under some compatibility conditions on the hash function (to avoid mixing the inputs ω_i with the outputs of the hash function), Theorem 3.1 in [ABPP14] shows that the resulting PRF is rka-secure.

G.2 Our New Framework

We would like to prove a similar statement for the *weighted* NR and BMR PRFs. However, given $\vec{w} \in \mathbb{Z}_p \times (\mathbb{Z}_p^*)^n$, if $w_0 \neq 0$ or if there exists $i \in \{1, \ldots, n\}$ such that $w_i > 1$, there is no practical strong key

fingerprint for $\mathsf{WNR}^{\vec{w}}$. Also, for any weight, it is not very clear that there exists a strong key-fingerprint for WBMR. Hence, in general, one cannot apply the above framework to WNR or WBMR.

For the above reason, in this section, we design a new framework, that generalizes the framework given in [ABPP14] and that can be applied, in particular, to both WNR and WBMR. This generic framework encompasses in particular the framework we propose in Section 5.2. However, our framework in the main body of this paper is significantly easier to use, which explains why we chose to give it first and propose this generic framework only as an appendix, for completeness.

We introduce new notions, defined as follows, that extends the notions introduced in Section 5.2.

Perfectly Binding Key-Commitment. In order to overcome the lack of a strong key fingerprint, we introduce perfectly binding key-commitment. A perfectly binding key-commitment is a (deterministic) algorithm $\mathsf{Com}\colon \mathcal{K} \to \mathsf{ComSp}$ that takes a key $K \in \mathcal{K}$ as input and outputs a value $\mathsf{Com}(K)$ such that for any K, K' in $\mathcal{K}$, we have $\mathsf{Com}(K) = \mathsf{Com}(K')$ if and only if $K = K'$ (perfectly binding). As we will see later, we also want that for $K \in \mathcal{K}$, $\mathsf{Com}(K)$ hides the value of K. However, we do not need special requirement for this in the present definition, since this requirement will be implied by the extended definitions of the key-collision and UI-PRF-RKA security problems defined below.

Helper Information. In order to prove the security of our framework, we need to be able to compute commitments of any related key from some (public) information. Then, we enable the adversary to have access to some helper information $\mathsf{help}_\Phi = \mathsf{Help}_\Phi(K) \in \mathsf{HelpSp}$ about the secret K, where Help_Φ is a function from $\mathcal{K}$ to HelpSp. The helper function Help_Φ depends on the class Φ of RKD functions we are interested in. We suppose that it is possible to compute $\mathsf{Com}(\phi(K))$ just by knowing ϕ and $\mathsf{Help}_\Phi(K)$ but not K.

Then, we use the extended version of the key-collision and unique-input-rka-prf security games depicted in Figure 8 and Figure 9, where **Initialize** also leaks help_Φ to the adversary. We remark that UI-PRF-RKA security implies that help_Φ hides K, otherwise the extended UI-PRF-RKA security would be trivial to break. This directly implies that the commitment of K is hiding, since it can be computed from help_Φ.

Remark G.1. We do not need a statistical-key-collision security property, because it is implied by the extended key-collision security property. The compatibility of the hash function is also simplified. We just require that it is a collision-resistant hash function and that its range $\mathcal{S}$ is such that the extended $(\mathcal{S}, \Phi)$-unique-input-prf-rka security is hard.

proc Initialize	**proc RKFn**(ϕ, x)
$K \xleftarrow{\$} \mathcal{K}$	$y \leftarrow M(\phi(K), x)$
$\mathsf{help}_\Phi \leftarrow \mathsf{Help}_\Phi(K)$	Return y
Return help_Φ	**proc Finalize**(ϕ_1, ϕ_2)
	Return $(\phi_1 \neq \phi_2$ and $\phi_1(K) = \phi_2(K))$

Figure 8: Game defining the extended Φ-key-collision security of a PRF M and helper function Help.

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

Figure 9: Game defining the extended $(\mathcal{S}, \Phi)$-unique-input-prf-rka security of a PRF M and helper function Help.

Using these new tools, we obtain the following framework, which generalizes [ABPP14, Theorem 3.1], as well as Theorem 5.2 from Section 5.2.

Theorem G.2. *Let $M\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ be a function and Φ be a class of RKD functions. Let* $\mathsf{Com}\colon \mathcal{K} \to \mathsf{ComSp}$ *be a perfectly binding key-commitment. Let* $\mathsf{Help}_\Phi\colon \mathcal{K} \to \mathsf{HelpSp}$ *be the helper function associated*

to Com *and* Φ*. Let* $\overline{\mathcal{D}} = \mathcal{D} \times \mathsf{ComSp}$ *and let* $H\colon\ \overline{\mathcal{D}} \to \mathcal{S}$ *be a compatible collision-resistant hash function, with* $\mathcal{S} \subseteq \mathcal{D}$*. Define* $F\colon\ \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ *by*

$$F(K, x) = M(K, H(x, \mathsf{Com}(K)))$$

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* $q \leq |\mathcal{S}|$ *oracle queries. Then we can construct an adversary* $\mathscr{B}$ *against the extended* $(\mathcal{S}, \Phi)$*-unique-input-prf-rka security of* M*, an adversary* $\mathscr{C}$ *against the collision-resistance (cr) security of* H*, and an adversary* $\mathscr{D}$ *against the extended* Φ*-kc security of* M *such that*

$$\mathbf{Adv}^{\mathsf{prf\text{-}rka}}_{\Phi,F}(\mathscr{A}) \ \leq\ \mathbf{Adv}^{\mathsf{ext\text{-}ui\text{-}prf\text{-}rka}}_{\Phi,\mathcal{S},M}(\mathscr{B}) + \mathbf{Adv}^{\mathsf{cr}}_{H}(\mathscr{C}) + 2 \cdot \mathbf{Adv}^{\mathsf{ext\text{-}kc}}_{\Phi,M}(\mathscr{D})\ .$$

Adversaries $\mathscr{C}$ *has the same running time as* $\mathscr{A}$*. Adversaries* $\mathscr{B}$ *and* $\mathscr{D}$ *have the same running time as* $\mathscr{A}$ *plus the time to compute* q *key-commitments using their helper information.*

Proof Overview. The proof of the above theorem is detailed in Appendix G.3 and relies on the sequence of 10 games (games $\mathrm{G}_0 - \mathrm{G}_9$) described in Figure 10. It is very similar to the proof of Theorem 3.1 from [ABPP14]. 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 RKAPRFReal case, using the extended key-collision notion (games $\mathrm{G}_0 - \mathrm{G}_2$). These claws can be detected by looking for collisions of perfectly binding key-commitments for different RKD functions. 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 $h = H(x, \mathsf{Com}(K))$ used to compute the output y are distinct. Then, we use the new extended $(\mathcal{S}, \Phi)$-unique-input-prf-rka security notion to show that it is hard to distinguish the output of F from a uniformly random output (games $\mathrm{G}_5 - \mathrm{G}_6$). Finally, we use once again the extended key-collision security notion to deal with possible key collisions in the RKAPRFRand case (games $\mathrm{G}_7 - \mathrm{G}_9$) so that G_9 matches the description of the RKAPRFRand Game. These key collisions can still be detected in these games by making crucial use of the helper function.

G.3 Proof of Theorem G.2

The proof is based on the sequence of games in Figure 10. Much of the proof is similar to the proof of the original framework that was given in [ABPP14]. We denote by $\textsc{Succ}_i$ the event that game G_i output takes the value 1. Boolean flags are assumed initialized to false. Games G_i, G_j are said to be identical until flag if their code differs only in statements that follow the setting of flag to true. We assume that adversary $\mathscr{A}$ never repeats an oracle query.

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

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

Game G_2 adds the boxed code which changes how the repetition of a commitment value com is handled, by picking instead a random value from $\mathsf{ComSp}\backslash E$ 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

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

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 extended Φ-key-collision security of M such that

$$\Pr\left[\,E_1\,\right] \leq \mathbf{Adv}^{\mathsf{ext\text{-}kc}}_{\Phi,M}(\mathscr{D})\ .$$

Adversary $\mathscr{D}$ gets helper information $\mathsf{help}_\Phi = \mathsf{Help}_\Phi(K)$, then runs $\mathscr{A}$. When the latter makes a **RKFn**-query (ϕ, x), adversary $\mathscr{D}$ computes $\mathsf{com} = \mathsf{Com}(\phi(K))$ using its helper information and then $h = H(x, \mathsf{com})$ and finally queries (ϕ, h) to its oracle and sends the value it gets to $\mathscr{A}$. When $\mathscr{A}$ halts, $\mathscr{D}$ searches for two different RKD functions ϕ queried by $\mathscr{A}$ that lead to the same commitment value com and returns these two functions if found. Since Com is a perfectly binding key-commitment, two such functions lead to the same key, so $\mathscr{D}$ wins if he finds such two functions.

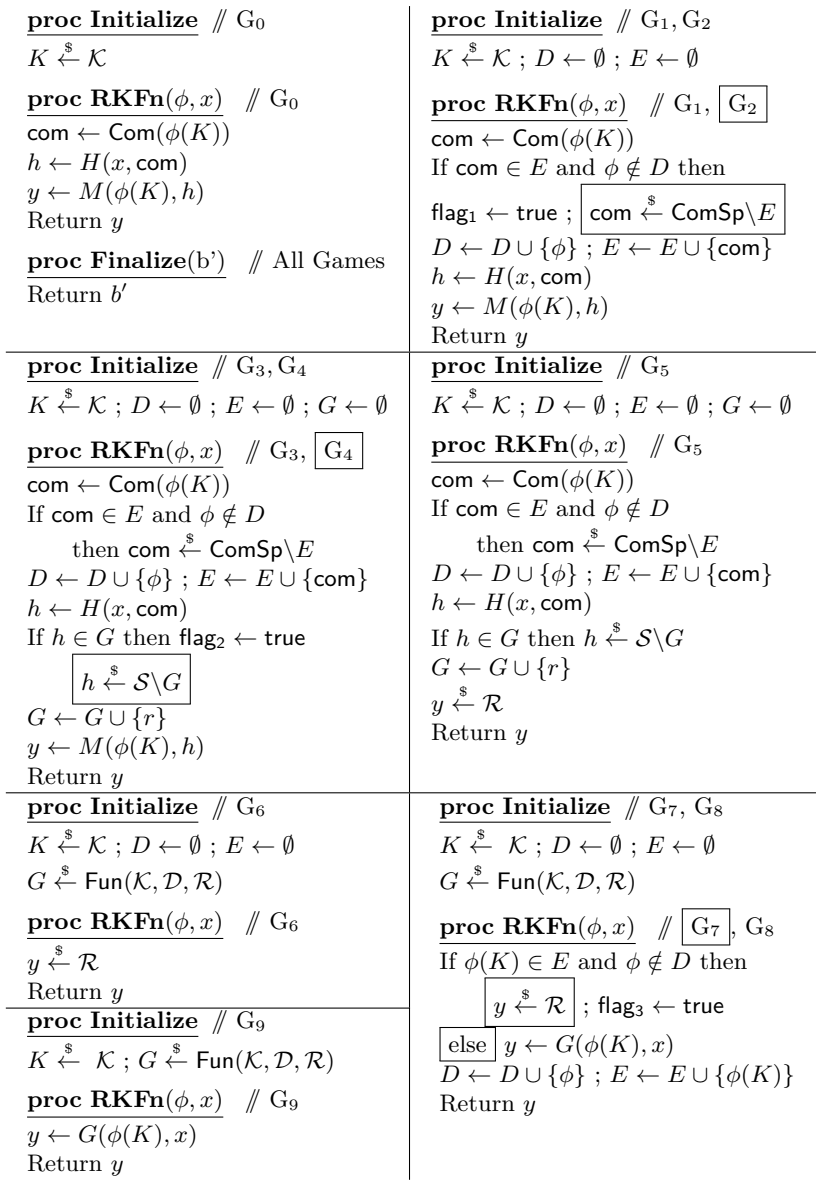

Figure 10: Games for the proof of Theorem G.2.

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

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

Game G_4 adds the boxed code which changes how repetition of hash values is handled, by picking instead a random value h from $\mathcal{S}\backslash G$ 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

$$\Pr[\,\mathrm{SUCC}_3\,] \le \Pr[\,\mathrm{SUCC}_4\,] + \Pr[\,E_2\,]$$

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 a **RKFn**-query (ϕ, x), adversary $\mathscr{C}$ responds via

$\mathsf{com} \leftarrow \mathsf{Com}(\phi(K))$

$j \leftarrow j+1$; $\phi_j \leftarrow \phi$; $x_j \leftarrow x$
If $\mathsf{com} \in E$ and $\phi \notin D$ then $\mathsf{com} \xleftarrow{\$} \mathsf{ComSp} \backslash E$ $\quad (*)$
$D \leftarrow D \cup \{\phi\}$; $E \leftarrow E \cup \{\mathsf{com}\}$
$\mathsf{com}_j \leftarrow \mathsf{com}$
$h \leftarrow H(x, \mathsf{com})$
$h_j \leftarrow h$
$y \leftarrow M(\phi(K), h)$
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 $(x_a, \mathsf{com}_a), (x_b, \mathsf{com}_b)$ and halts. The pairs (x_a, com_a) and (x_b, com_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 $(x_a, \mathsf{com}_a) \neq (x_b, \mathsf{com}_b)$. Second, if $\phi_a \neq \phi_b$, then condition $(*)$ ensures that $\mathsf{com}_a \neq \mathsf{com}_b$. Hence once again, $(x_a, \mathsf{com}_a) \neq (x_b, \mathsf{com}_b)$, and then

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

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 extended $(\mathcal{S}, \Phi)$-unique-input-prf-rka security of M such that

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

Adversary $\mathscr{B}$ starts by initializing sets $D \leftarrow \emptyset$, $E \leftarrow \emptyset$, $G \leftarrow \emptyset$. Then $\mathscr{B}$ gets helper information $\mathsf{help}_\Phi = \mathsf{Help}_\Phi(K)$, then runs $\mathscr{A}$. When the latter makes an **RKFn**-query (ϕ, x), $\mathscr{B}$ responds as follows. First, it computes $\mathsf{com} = \mathsf{Com}(\phi(K))$ using its helper information. Then, $\mathscr{B}$ checks if $\mathsf{com} \in E$ and $\phi \in D$. If they do, $\mathscr{B}$ picks $\mathsf{com} \xleftarrow{\$} \mathsf{ComSp} \backslash E$ at random. $\mathscr{B}$ then sets $D \leftarrow D \cup \{\phi\}$ and $E \leftarrow E \cup \{\mathsf{com}\}$. Next, $\mathscr{B}$ computes $h \leftarrow H(x, \mathsf{com})$ and checks if $h \in G$. If it does, $\mathscr{B}$ picks $h \xleftarrow{\$} \mathcal{S} \backslash G$ 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. 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,

$$\Pr[\,\textsc{Succ}_6\,] = \Pr[\,\textsc{Succ}_7\,]\,.$$

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

$$\Pr[\,\textsc{Succ}_7\,] \le \Pr[\,\textsc{Succ}_8\,] + \Pr[\,E_3\,]$$

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 $\mathscr{D}$ attacking extended Φ-key-collision security of M such that

$$\Pr[\,E_3\,] \le \mathbf{Adv}^{\mathsf{ext\text{-}kc}}_{\Phi,M}(\mathscr{D})\,.$$

Adversary $\mathscr{D}$ starts by initializing a list $\mathcal{L} \leftarrow$ empty list and by choosing an element ψ in Φ and by setting $\psi_1 \leftarrow \psi$ and $\psi_2 \leftarrow \psi$. Then, it runs $\mathscr{A}$. When the latter makes an **RKFn**-query (ϕ, x), adversary $\mathscr{D}$ does the following: it first computes $\mathsf{com} \leftarrow \mathsf{Com}(\phi(K))$ using its helper information and searches for all tuples $(\phi_i, \mathsf{com}_i) \in \mathcal{L}$ such that $\mathsf{com}_i = \mathsf{com}$. If it does find such tuples, it checks for all of them if $\phi \neq \phi_l$

and in that case sets $\psi_1 \leftarrow \phi_l$ and $\psi_2 \leftarrow \phi$. Finally, $\mathscr{D}$ picks $y \xleftarrow{\$} \mathcal{R}$ at random, returns y to $\mathscr{A}$ and adds (ϕ, com) to $\mathcal{L}$. When $\mathscr{A}$ halts, $\mathscr{B}$ halts and outputs (ψ_1, ψ_2). If the execution of $\mathscr{A}$ sets $\mathsf{flag_3}$ to true, then $\mathscr{A}$ has queried $\phi_1 \neq \phi_2$ such that $\phi_1(K) = \phi_2(K)$ and assuming it has first queried ϕ_1, when $\mathscr{D}$ computes $\mathsf{Com}(\phi_2(K))$ and checks if this value is already in $\mathcal{L}$, it finds that this value matches $\mathsf{Com}(\phi_1(K))$ and since $\phi_1 \neq \phi_2$, it sets $\psi_1 = \phi_1$ and $\psi_2 = \phi_2$, so $\mathscr{D}$ wins. Then, we have

$$\Pr[E_3] \leq \mathbf{Adv}^{\mathsf{ext\text{-}kc}}_{\Phi,M}(\mathscr{D}) .$$

Since 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,

$$\Pr[\textsc{Succ}_8] = \Pr[\textsc{Succ}_9] .$$

Theorem G.2 now follows by combining the bounds arising in the different game hops.

H Definitions and Proofs for Section 6

H.1 Definitions: Monomial Order and Leading Commutative Monomials

Definition H.1. [Monomial order] Let n be a positive integer. A monomial order for $\mathbb{Z}_p[T_1, \ldots, T_n]$ is a total order such that, for any monomials u, v, w:

- if $u < v$, then $uw < vw$,
- $1 \leq u$.

We write $\vec{T}^{\vec{i}} = T_1^{i_1} \cdots T_n^{i_n}$ for $\vec{i} = (i_1, \ldots, i_n)$. The *leading monomial* of a polynomial $P(\vec{T}) = \sum_{\vec{i}} \alpha_{\vec{i}} \vec{T}^{\vec{i}}$ is the maximum of the set $\{\vec{T}^{\vec{i}} \mid \alpha_{\vec{i}} \neq 0\}$ for the monomial order $<$, and is denoted $\mathrm{LM}(P)$. The *leading term* of this polynomial P is $\alpha_{\vec{i}^*} \vec{T}^{\vec{i}^*}$, when $\mathrm{LM}(P) = \vec{T}^{\vec{i}^*}$.

We extend this definition to non-commutative polynomials as follows: let

$$\pi\colon\ \mathbb{Z}_p\langle T_1, \ldots, T_n\rangle \to \mathbb{Z}_p[T_1, \ldots, T_n]$$

be the (canonical) linear map defined by $\pi(T_{j_1} \cdots T_{j_k}) = T_{j_1} \cdots T_{j_k}$. The *leading monomials set* of a non-commutative polynomial

$$P(\vec{T}) = \sum_{\substack{k \geq 1 \\ j_1, \ldots, j_k \in \{1, \ldots, n\}}} \alpha_{j_1, \ldots, j_k} T_{j_1} \cdots T_{j_k}$$

as the set of monomials $T_{j_1} \cdots T_{j_k}$ such that $\pi(T_{j_1} \cdots T_{j_k})$ is the maximum of

$$\{\pi(T_{j_1} \cdots T_{j_k}) \mid \alpha_{j_1, \ldots, j_k} \neq 0\} .$$

It is denoted $\mathrm{CLM}(P)$. We say a polynomial has *unique* commutative leading monomial if $\mathrm{CLM}(P)$ is a singleton $\{T_{j_1} \cdots T_{j_k}\}$, in which case, we also often write $\mathrm{CLM}(P) = T_{j_1} \cdots T_{j_k}$, to simplify notations.

We remark that if we identify (commutative) polynomials with non-commutative polynomials (by writing them as $P = \sum_{\vec{i}} \alpha_{\vec{i}} \vec{T}^{\vec{i}} = \sum_{\vec{i}} \alpha_{\vec{i}} T_1^{i_1} \cdots T_n^{i_n}$), then polynomials have unique commutative leading monomial.

Example H.2. For $n = 2$ and $<$ the lexicographic order with $T_1 > T_2$, we have:

$$\mathrm{LM}(5T_1^2T_2 + T_1T_2^3 + T_2) = T_1^2T_2 \qquad \mathrm{LM}(T_1^3 + 3T_1T_2^7) = T_1^3$$

for commutative polynomials, and

$$\mathrm{LM}(5T_1^2T_2 + T_1T_2^3 + T_2) = \{T_1^2T_2\}$$
$$\mathrm{LM}(5T_1^2T_2 + T_1T_2T_1 + T_2T_1^2 + T_2 + T_1) = \{T_1^2T_2, T_1T_2T_1, T_2T_1^2\}$$

for non-commutative polynomials.

H.2 Main Lemma

We will make use of the following lemma in the security proof of $\mathcal{E}_{2,d}$-MDDH in generic bilinear groups in Section H.3 and in the proof of Theorem 6.2 in Section H.5.

Lemma H.3. *Let n and m be two positive integers. We suppose fixed a monomial order $<$ for $\mathbb{Z}_p[T_1,\ldots,T_n]$. Let $(P_s)_{s=1,\ldots,q}$ be a family of polynomials with distinct and unique commutative leading monomial. Let*

$$\mathscr{R} = \mathbb{Z}_p[(X_{k,i,j})_{\substack{k=1,\ldots,n\\ i=1,2\\ j=1,2}}, (Y_{i,j})_{\substack{i=1,2\\ j=1,\ldots,m}}].$$

Let us define $\vec{\boldsymbol{A}} \in \left(\mathscr{R}^{2\times 2}\right)^n$ a vector of 2×2 matrices of (commutative) polynomials with indeterminates $X_{k,i,j}$, such that $a_{k,i,j} = X_{k,i,j}$. Let us also define $\boldsymbol{A}' \in \mathscr{R}^{2\times m}$, such that $a_{0,i,j} = Y_{i,j}$. In other words:

$$\boldsymbol{A}' = \begin{pmatrix} Y_{1,1} & \ldots & Y_{1,m} \\ Y_{2,1} & \ldots & Y_{2,m} \end{pmatrix} \qquad \boldsymbol{A}_k = \begin{pmatrix} X_{k,1,1} & X_{k,1,2} \\ X_{k,2,1} & X_{k,2,2} \end{pmatrix}.$$

Let $Q_{s,i,j} \in \mathscr{R}$ be the polynomial corresponding to the coordinate $(i,j) \in \{1,2\}\times\{1,\ldots,m\}$ of the matrix $P_s(\vec{\boldsymbol{A}})\cdot\boldsymbol{A}'$.

Finally let assume there exists coefficients $\lambda_{s_1,s_2,i_1,i_2,j_1,j_2}$ such that:

$$\sum_{\substack{s_1=0,\ldots,q\\ i_1=1,2\\ j_1=1,\ldots,m}} \sum_{\substack{s_2=0,\ldots,q\\ i_2=1,2\\ j_2=1,\ldots,m\\ (s_2,i_2,j_2)\succeq(s_1,i_1,j_1)}} \lambda_{s_1,s_2,i_1,i_2,j_1,j_2} Q_{s_1,i_1,j_1} Q_{s_2,i_2,j_2} = 0, \tag{6}$$

with $\succeq$ the lexicographic order (just to ensure that each term $Q_{s_1,i_1,j_1}Q_{s_2,i_2,j_2}$ to appear only once). Then, all these coefficients $\lambda_{s_1,s_2,i_1,i_2,j_1,j_2}$ are zero.

Proof. Let us assume, without loss of generality that:

$$\mathrm{CLM}(P_1) < \ldots < \mathrm{CLM}(P_q).$$

First, we order monomials of $\mathscr{R}$ using the product order on $\{Y_{i,j}\}_k \times \{X_{k,2,2}\}_k \times \{X_{k,1,2}\}_k \times \{X_{k,2,1}\}_k \times \{X_{k,1,1}\}_k$, with the lexicographic order on $\{Y_{i,j}\}$ (with $Y_{1,1} > Y_{1,2} > \ldots > Y_{1,m} > Y_{2,1} > \ldots > Y_{2,m}$), and, for any i,j, the same order on $\{X_{k,i,j}\}_{k,i,j}$ than on $\{T_1,\ldots,T_n\}$.

Second, let us prove that $\lambda_{s_1,s_2,i_1,i_2,j_1,j_2} = 0$ when $i_1 \neq i_2$. For that purpose, let us set $X_{s,1,2}$ and $X_{s,2,1}$ to 0 in Equation (6) (i.e., we choose diagonal matrices $\boldsymbol{A}_i$). Then, we get:

$$\begin{aligned} P_s(\vec{\boldsymbol{A}})\cdot\boldsymbol{A}' &= \begin{pmatrix} P_s(X_{1,1,1},\ldots,X_{n,1,1}) & 0 \\ 0 & P_s(X_{1,2,2},\ldots,X_{n,2,2}) \end{pmatrix} \cdot \boldsymbol{A}' \\ &= \begin{pmatrix} P_s(X_{1,1,1},\ldots,X_{n,1,1})\cdot Y_{1,1} & \ldots & P_s(X_{1,1,1},\ldots,X_{n,1,1})\cdot Y_{1,m} \\ P_s(X_{1,2,2},\ldots,X_{n,2,2})\cdot Y_{2,1} & \ldots & P_s(X_{1,2,2},\ldots,X_{n,2,2})\cdot Y_{2,m} \end{pmatrix}. \end{aligned}$$

where $P_s(X_{1,1,1},\ldots,X_{n,1,1}) = \pi(P_s)(X_{1,1,1},\ldots,X_{n,1,1})$, with π the canonical surjection from the vector space $\mathbb{Z}_p\langle T_1,\ldots,T_n\rangle$ to the vector space $\mathbb{Z}_p[T_1,\ldots,T_n]$ defined above. Thus,

$$Q_{s,i,j} = P_s(X_{1,i,i},\ldots,X_{n,i,i})\cdot Y_{i,j}.$$

As all the commutative leading monomials of P_s are unique and distinct, so are the leading monomials of

$$P_{s_1}(X_{1,i_1,i_1},\ldots,X_{n,i_1,i_1})P_{s_2}(X_{1,i_2,i_2},\ldots,X_{n,i_2,i_2})Y_{i_1,j_1}Y_{i_2,j_2},$$

for any s_1,s_2,i_1,i_2,j_1,j_2 such that $i_1 \neq i_2$ (the proof is straightforward from the definition of the monomial order $<$ on $\mathscr{R}$: we first compare the $Y_{i,j}$ part, then the $P_{s_1}(X_{1,i_1,i_1},\ldots,X_{n,i_1,i_1})$ part and finally the $P_{s_2}(X_{1,i_2,i_2},\ldots,X_{n,i_2,i_2})$ part; each part corresponds to a different set of monomials of the product order $<$). Therefore $\lambda_{s_1,s_2,i_1,i_2,j_1,j_2} = 0$, when $i_1 \neq i_2$.

Third, let us prove that $\lambda_{s_1,s_2,1,1,j_1,j_2} = 0$ for all s_1,s_2,j_1,j_2. For that purpose, let us set $X_{s,2,1} = 0$ and $X_{s,1,2} = X_{s,2,2}$. In other words:

$$\boldsymbol{A}_k = \begin{pmatrix} X_{k,1,1} & X_{k,2,2} \\ 0 & X_{k,2,2} \end{pmatrix}.$$

Then, we get (by recursion, linearity, and the product definition of $<$):

$$P_s(\vec{\boldsymbol{A}}) \cdot \boldsymbol{A}' = \begin{pmatrix} P_s(X_{1,1,1}, \ldots, X_{n,1,1}) & \mathrm{LT}(P_s(X_{1,2,2}, \ldots, X_{n,2,2})) + \ldots \\ 0 & P_s(X_{1,2,2}, \ldots, X_{n,2,2}) \end{pmatrix} \cdot \boldsymbol{A}',$$

where $\mathrm{LT}(P_s(X_{1,2,2}, \ldots, X_{n,2,2})) + \ldots$ is a polynomial for which the leading term $\mathrm{LT}(P_s(X_{1,2,2}, \ldots, X_{n,2,2}))$. By contradiction, let us suppose that there exists some $\lambda_{s_1,s_2,1,1,j_1,j_2} \neq 0$, and let us consider one with the highest (s_2, s_1) for the lexicographic order (as $(s_2, 1, j_2) \succeq (s_1, 1, j_1)$, $s_2 \geq s_1$ and this means that s_2 and s_1 are both the highest possible). We have:

$$\begin{aligned} &Q_{s_1,1,j_1} Q_{s_2,1,j_2} \\ &= (P_{s_1}(X_{1,1,1}, \ldots, X_{n,1,1}) Y_{1,j_1} + (\mathrm{LT}(P_{s_1}(X_{1,2,2}, \ldots, X_{n,2,2})) + \ldots) Y_{2,j_1}) \\ &\quad \cdot (P_{s_2}(X_{1,1,1}, \ldots, X_{n,1,1}) Y_{1,j_2} + (\mathrm{LT}(P_{s_2}(X_{1,2,2}, \ldots, X_{n,2,2})) + \ldots) Y_{2,j_2}), \end{aligned}$$

which contains the following monomial:

$$u = \mathrm{LM}(P_{s_1}(X_{1,1,1}, \ldots, X_{n,1,1})) Y_{1,j_1} \mathrm{LM}(P_{s_2}(X_{1,2,2}, \ldots, X_{n,2,2})) Y_{2,j_2}.$$

Let us prove this monomial is in no other term of Equation (6). By contradiction, let us suppose there is some other term $\lambda_{s'_1,s'_2,1,1,j'_1,j'_2} Q_{s'_1,1,j'_1} Q_{s'_2,1,j'_2}$ containing this monomial. Then the monomial u can be written:

$$u = v_1 \cdot Y_{1,j_1} \cdot v_2 \cdot Y_{2,j_2}$$

with (v_1, v_2) monomials of $P_{s'_1}(X_{1,1,1}, \ldots, X_{n,1,1})$ and $\mathrm{LT}(P_{s'_2}(X_{1,2,2}, \ldots, X_{n,2,2})) + \ldots$ (respectively; in which case $j_1 = j'_1$ and $j_2 = j'_2$), or of $P_{s'_2}(X_{1,1,1}, \ldots, X_{n,1,1})$ and $\mathrm{LT}(P_{s'_1}(X_{1,2,2}, \ldots, X_{n,2,2})) + \ldots$ (respectively; in which case $j_1 = j'_2$ and $j_2 = j'_1$). But, from the choice of s_1, s_2 and the fact that we suppose $(s'_2, 1, j_2) \succeq (s'_1, 1, j_1)$, we have that $s_2 \geq s'_2 \geq s'_1$, and so $\mathrm{LM}(P_{s_2}) \geq \mathrm{LM}(P_{s'_2}) \geq \mathrm{LM}(P_{s'_1})$. Therefore $s_2 = s'_2$. We have two cases:

- if $s'_2 > s'_1$, then necessarily (v_1, v_2) comes from the polynomials $P_{s'_1}(X_{1,1,1}, \ldots, X_{n,1,1})$ and $\mathrm{LT}(P_{s'_2}(X_{1,2,2}, \ldots, X_{n,2,2})) + \ldots$ (respectively), and $j_1 = j'_1$, $j_2 = j'_2$. In addition, as $s_1 \geq s'_1$ and $\mathrm{LM}(P_{s_1}) \geq \mathrm{LM}(P_{s'_1})$, $s_1 = s'_1$. So $(s_1, s_2, j_1, j_2) = (s'_1, s'_2, j'_1, j'_2)$, which is a contradiction;
- otherwise $s'_2 = s'_1$, then as $s_1 \geq s'_1$ and $\mathrm{LM}(P_{s_1}) \geq \mathrm{LM}(P_{s'_1})$, $s_1 = s'_1 = s_2 = s'_2$. In that case, as $(s_2, 1, j_2) \succeq (s_1, 1, j_1)$ and $(s'_2, 1, j'_2) \succeq (s'_1, 1, j'_1)$, $j_2 \geq j_1$ and $j'_2 \geq j'_1$. In addition, it is clear that $\{j_1, j_2\} = \{j'_1, j'_2\}$, hence $j_1 = j'_1$ and $j_2 = j'_2$. So $(s_1, s_2, j_1, j_2) = (s'_1, s'_2, j'_1, j'_2)$, which is a contradiction.

That concludes the third point.

Fourth, using a similar proof with

$$\boldsymbol{A}_k = \begin{pmatrix} X_{k,1,1} & 0 \\ X_{k,1,1} & X_{k,2,2} \end{pmatrix},$$

we get that $\lambda_{s_1,s_2,2,2,j_1,j_2} = 0$.

This concludes the proof. □

H.3 Proof of Security of $\mathcal{E}_{2,d}$-MDDH in Generic Bilinear Groups

Similarly to the proof of Theorem 3 of [EHK+13] and the proof for Über assumptions [BBG05, Boy08], to prove the security of $\mathcal{E}_{2,d}$-MDDH in generic symmetric bilinear groups, we just need to show that there is no (non-trivial) polynomial relation of degree 2 between entries of $\boldsymbol{\Gamma}$ and $\boldsymbol{Z}$, both when $\boldsymbol{Z} = \boldsymbol{\Gamma} \cdot \boldsymbol{W}$ and when $\boldsymbol{Z} = \boldsymbol{U}$, with

$$\boldsymbol{\Gamma} = \begin{pmatrix} \boldsymbol{A}' \\ \boldsymbol{A}_1 \cdot \boldsymbol{A}' \\ \vdots \\ \boldsymbol{A}_d \cdot \boldsymbol{A}' \end{pmatrix}.$$

Indeterminates are entries of $\boldsymbol{A}_1$ and $\boldsymbol{A}'$ ($a_{1,i,j}$, $a'_{i,j}$, for $i = 1, 2$, $j = 1, 2$), entries of $\boldsymbol{W}$ (w_i, for $i = 1, 2$), and entries of $\boldsymbol{U}$ ($u_{i,j}$, for $i = 1, \ldots, 2(d+1)$, $j = 1, 2$). The polynomial independence follows from Lemma H.3, with $n = 1$, $q = d + 1$, and $P_s = T_1^{s-1}$, for $s = 1, \ldots, d + 1$. □

H.4 Proof of Theorem 6.1

Proof Intuition. By renaming the indeterminates, we can assume that T_n appears before T_{n-1}, T_{n-1} before T_{n-2}, ... Then, the proof is similar to the one for the LIP theorem, except the assumption $\mathcal{E}_{1,d}$-MDDH is replaced by $\mathcal{E}_{2,d}$-MDDH (and if we write $[\boldsymbol{A}]^{\boldsymbol{B}} = [\boldsymbol{BA}]$ and $[\boldsymbol{A}] \cdot [\boldsymbol{B}] = [\boldsymbol{A} + \boldsymbol{B}]$ for any matrices $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{Z}_p^{k \times k}$). In particular, the statistical test for linear dependence of multivariate polynomials remains the same.

An important technical detail remain: we need to partially evaluate polynomials in some matrices. If we do it naively, we will end up with polynomials with matrix coefficients and the proof will fail. Instead, we remark that we can decompose these polynomials with matrix coefficients as sum of products of one matrix and one classical polynomial, when polynomial representations satisfy Condition 2 (in Section B.2). This intuition is captured by the following decomposition lemmas (which are stated even for matrices of size $k \times k$ and not just for 2×2 matrices).

The full proof of Theorem 6.1 is then straightforward.

Decomposition Lemmas.

Lemma H.4. *Let $k \geq 2$ be an integer. There exists a polynomial-time algorithm which takes as input:*

- *an integer $j \in \{0, \ldots, n\}$,*
- *$n - j$ matrices $\boldsymbol{A}_{j+1}, \ldots, \boldsymbol{A}_n$ in $\mathbb{Z}_p^{k \times k}$,*
- *an expression $\tilde{P}$ of a multivariate polynomial $P \in \mathbb{Z}_p[T_1, \ldots, T_n]$ satisfying Condition 2,*

and which outputs a decomposition of $\tilde{P}$ as N polynomials $Q_1, \ldots, Q_N \in \mathbb{Z}_p[T_1, \ldots, T_j]$ and N matrices $\boldsymbol{C}_1, \ldots, \boldsymbol{C}_N$ such that:

$$P(T_1, \ldots, T_j, \boldsymbol{A}_{j+1}, \ldots, \boldsymbol{A}_n) = \sum_{\nu=1}^{N} \boldsymbol{C}_\nu \cdot Q_\nu(T_1, \ldots, T_j).$$

In addition, N is less than the number of internal nodes in the expression or AST $\tilde{P}$; and the representations of the polynomials $Q_1, \ldots, Q_N$ satisfy Condition 2.

Proof. We do the proof by recursion:

- *Base case (a leaf): an indeterminate T_i or a scalar in $\mathbb{Z}_p$.* Straightforward.
- *Recursive case 1: additive node $\tilde{P}_1 + \tilde{P}_2$.* We decompose recursively $\tilde{P}_1$ and $\tilde{P}_2$.
- *Recursive case 2: multiplicative node $\tilde{P}_1 \cdot \tilde{P}_2$.* This is the most important case. We consider two sub-cases:
  - $\tilde{P}_1$ only contain leaves with scalars or indeterminates $T_{j+1}, \ldots, T_n$. In that case, its decomposition is just a matrix in $\mathbb{Z}_p^{k \times k}$. The decomposition of $\tilde{P}_1 \cdot \tilde{P}_2$ then contains as many terms as in the decomposition of $\tilde{P}_2$.
  - Otherwise, $\tilde{P}_2$ does not contain indeterminates $T_{j+1}, \ldots, T_n$ (otherwise that would break Condition 2, and so the decomposition of $\tilde{P}_2$ is just a polynomial (matrices are identity matrices). The decomposition of $\tilde{P}_1 \cdot \tilde{P}_2$ then contains as many terms as in the decomposition of $\tilde{P}_1$.

□

Lemma H.5. *Let $k \geq 1$ and $j \geq 1$ be two integers. There exists a polynomial-time algorithm which takes as input an expression $\tilde{P}$ of a multivariate polynomial $P \in \mathbb{Z}_p[T_1, \ldots, T_j]$ of degree at most $d < p$ in T_j and satisfying Condition 1, and which outputs $d + 1$ polynomials $Q_0, \ldots, Q_d \in \mathbb{Z}_p[T_1, \ldots, T_{j-1}]$ such that*

$$P = Q_0 + Q_1 \cdot T_j + \cdots + Q_d \cdot T_j^d.$$

In addition, the representations of $Q_0, \ldots, Q_d$ satisfy Condition 1.

Proof. We can use the Lagrange interpolation

$$P = \sum_{i=0}^{d} P(T_1, \ldots, T_{j-1}, i) \prod_{\substack{i'=0,\ldots,d \\ i' \neq i}} (T_j - i'),$$

and regroup terms correctly. □

H.5 Proof of Theorem 6.2

Similarly to the proof of Theorem 3 of [EHK+13] and the proof for Über assumptions [BBG05, Boy08], to prove Theorem 6.2 in generic symmetric bilinear groups, we just need to show that there is no (non-trivial) polynomial relation of degree 2, between entries of matrices $P_j(\vec{\boldsymbol{A}}) \cdot \boldsymbol{A'}$, where indeterminates are entries of $\vec{\boldsymbol{A}}$ and of $\boldsymbol{A'}$ ($a_{s,i,j}$, $a'_{i,j}$, for $s = 1, \ldots, n$, $i = 1, 2$, and $j = 1, 2$).

When polynomials P_j have distinct and unique leading monomials (i.e., when $\boldsymbol{M}$ is the identity matrix in the DLM condition), this is exactly what shows Lemma H.3.

In the general case, where $\boldsymbol{M}$ is any invertible matrix, let us write $(\tilde{P}_s)_s = \boldsymbol{M} \cdot (P_s)_s$ the family obtained after applying $\boldsymbol{M}$, where here, we view the families $(\tilde{P}_s)_s$ and $(P_s)_s$ as column vectors. Let us write $Q_{s,i,j}$ the polynomials corresponding to the entries of the matrices $P_s(\vec{\boldsymbol{A}}) \cdot \boldsymbol{A'}$ (as in Lemma H.3), and $\tilde{Q}_{s,i,j}$ the polynomials corresponding to the entries of the matrices $\tilde{P}_s(\vec{\boldsymbol{A}}) \cdot \boldsymbol{A'}$. Polynomials $\tilde{P}_s$ have distinct and unique leading monomials, so that we can apply Lemma H.3 on them, and show that there is no non-trivial relation of the form:

$$\sum_{s_1,s_2,i_1,i_2,j_1,j_2} \lambda_{s_1,s_2,i_1,i_2,j_1,j_2} \tilde{Q}_{s_1,i_1,j_1} \tilde{Q}_{s_2,i_2,j_2} = 0.$$

Suppose now by contradiction that there exists a non-trivial polynomial relation of degree 2 between $(Q_{s,i,j})$. This means that there exists a non-zero polynomial R of degree 2 in indeterminates $U_{s,i,j}$, such that $R((Q_{s,i,j})) = 0$ (where $R((Q_{s,i,j}))$ is the polynomial R, where $U_{s,i,j}$ is replaced by $U_{s,i,j}$). But then, let $\tilde{R}$ be the polynomial R after the linear coordinate transform $(U_{s,i,j}) \mapsto \boldsymbol{M} \cdot (U_{s,i,j})$. This polynomial $\tilde{R}$ has degree 2, is non-zero (as $\boldsymbol{M}$ is invertible), and verifies $\tilde{R}((\tilde{Q}_{s,i,j})) = 0$. That contradicts the absence of non-trivial polynomial relation between the polynomials $\tilde{Q}_{s,i,j}$, and conclude the proof. □