An extended abstract of this paper appears in the Proceedings of the 21th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT 2015), Part I, Tetsu Iwata and Jung Hee Cheon (Eds.), volume 9452 of Lecture Notes in Computer Science, pages 103–120, Springer, November 29 – December 3, 2015. This is the full version.

Multilinear and Aggregate Pseudorandom Functions: New Constructions and Improved 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

Since its introduction, pseudorandom functions (PRFs) have become one of the main building blocks of cryptographic protocols. In this work, we revisit two recent extensions of standard PRFs, namely multilinear and aggregate PRFs, and provide several new results for these primitives. In the case of aggregate PRFs, one of our main results is a proof of security for the Naor-Reingold PRF with respect to read-once boolean aggregate queries under the standard Decision Diffie-Hellman problem, which was an open problem. In the case of multilinear PRFs, one of our main contributions is the construction of new multilinear PRFs achieving indistinguishability from random symmetric and skew-symmetric multilinear functions, which was also left as an open problem. In order to achieve these results, our main technical tool is a simple and natural generalization of the recent linear independent polynomial framework for PRFs proposed by Abdalla, Benhamouda, and Passelègue in Crypto 2015, that can handle larger classes of PRF constructions. In addition to simplifying and unifying proofs for multilinear and aggregate PRFs, our new framework also yields new constructions which are secure under weaker assumptions, such as the decisional k-linear assumption.

Keywords. Pseudorandom functions, multilinear PRFs, aggregate PRFs.

Contents

1 Introduction

Pseudorandom functions (PRFs) are one of the most fundamental primitives in cryptography. One of the features that makes PRFs so useful is the fact that they behave as truly random functions with respect to computationally bounded adversaries. Since being introduced by Goldreich, Goldwasser, and Micali [GGM86], PRFs have been used in many cryptographic applications, varying from symmetric encryption and authentication schemes to key exchange. In particular, they are very useful for modeling the security of concrete block ciphers, such as AES [AES01].

Given the large applicability of pseudorandom functions, several extensions have been proposed in the literature over the years, with the goal of providing additional functionalities to these functions. One concrete example of such an extension are constrained PRFs [KPTZ13, BGI14, BW13], which provides the owner of the secret key with the capability of delegating the computation of the pseudorandom function for different subsets of the input domain, without compromising the pseudorandomness property for the other points of the input domain. In this paper, we focus on two recent extensions of pseudorandom functions, namely multilinear PRFs [CH15], and aggregate PRFs [CGV15], and solve several open problems related to the construction of these primitives.

Aggregate Pseudorandom Functions. Aggregate pseudorandom functions were introduced by Cohen, Goldwasser, and Vaikuntanathan in [CGV15]. The main interest of an aggregate PRF is to provide the user with the possibility of aggregating the values of the function over *super-polynomially* many PRF values with only a *polynomial-time* computation, without enabling a polynomial-time adversary to distinguish the function from a truly random function. For instance, one such example of an aggregate query could be to compute the product of all the output values of the PRF corresponding to a given exponentially-sized interval of the input domain.

In addition to proposing the notion of aggregate PRFs, Cohen, Goldwasser, and Vaikuntanathan [CGV15] also proposed new constructions for several different classes of aggregate queries, such as decision trees, hypercubes, and read-once boolean formulas, achieving different levels of expressiveness. Unfortunately, for most of the constructions proposed in [CGV15], the proofs of security suffer from an exponential (in the input length) overhead in their running time and have to rely on the sub-exponential hardness of the Decisional Diffie-Hellman (DDH) problem.

Indeed, to prove the security of their constructions, the authors use a generic result which is simply saying the following: given an adversary $\mathscr{A}$ against the AGG-PRF security of a PRF F, one can build an adversary $\mathscr{B}$ against the standard PRF security of F. $\mathscr{B}$ simply queries all the values required to compute the aggregate values (or the PRF values), and computes the aggregate values itself before sending them to $\mathscr{A}$.

Clearly, this reduction proves that any secure PRF is actually also a secure aggregate PRF. However, this reduction is *not efficient*, since to answer to just one aggregate query, the adversary $\mathscr{B}$ may have to query an exponential number of values to its oracle. Hence, as soon as we can aggregate in one query a superpolynomial number of PRF values, this generic reduction does not run in polynomial time.

Multilinear Pseudorandom Functions. In order to overcome the shortcomings of the work of Cohen, Goldwasser, and Vaikuntanathan [CGV15], Cohen and Holmgren introduced the concept of multilinear pseudorandom functions in [CH15]. Informally speaking, a multilinear pseudorandom function is a variant of the standard notion of pseudorandom functions, which works with vector spaces and which guarantees indistinguishability from random multilinear functions with the same domain and range. As shown in [CH15], multilinear pseudorandom functions can be used to prove the AGG-PRF security of the Naor-Reingold (NR) PRF [NR97] with a polynomial time reduction for the case of hypercubes and decision trees aggregations. Unfortunately, their technique does not extend to the more general case of read-once formulas aggregation, which is the most expressive form of aggregation in [CGV15].

Our Techniques. In this work, we provide an alternative way of overcoming the limitations of the work of Cohen, Goldwasser, and Vaikuntanathan [CGV15], based on a natural extension of the recent algebraic framework for pseudorandom functions proposed by Abdalla, Benhamouda, and Passelègue in [ABP15], known as the linear independent polynomial (LIP) framework.

In a nutshell, the LIP framework essentially says that for any linearly independent polynomials $P_1, \ldots, P_q \in \mathbb{Z}_p[T_1, \ldots, T_n]$, the group elements

$$[P_1(\vec{a}) \cdot b], \ldots, [P_q(\vec{a}) \cdot b],$$

with $\vec{a} \xleftarrow{\$} \mathbb{Z}_p^n$ and $b \xleftarrow{\$} \mathbb{Z}_p$, are computationally indistinguishable from independent random group elements in $\mathbb{G}$, under the DDH assumption (when polynomials are multilinear) or the d-DDHI assumption

(where d is the maximum degree of $P_1, \ldots, P_q$ in any indeterminate T_i). As a toy example, the LIP framework directly proves the security of the NR PRF defined as:

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

where $(b, \vec{a} = (a_1, \ldots, a_n)) \in \mathcal{K} = \mathbb{Z}_p \times \mathbb{Z}_p^n$ and $x \in \mathcal{D} = \{0,1\}^n$. Indeed, all the polynomials $P_x = b \prod_{i=1}^{n} a_i^{x_i}$ are linearly independent.

Unfortunately, the LIP framework is not enough to prove the security of multilinear PRFs or aggregate PRFs, as the outputs of the function (and the corresponding polynomials) may not be independent. To overcome these limitations, we provide a natural extension of the LIP framework, which we call polynomial linear pseudorandomness security (PLP), that can handle such dependences. Despite being a simple extension, the new PLP framework yields significant improvements over previous works on multilinear and aggregate PRFs. In particular, the multilinear constructions in [CH15] can be seen as a special case of our new PLP framework.

Main Results. Using our new PLP framework for pseudorandom functions, we obtain the following results.

First, we prove the security of the aggregate PRF for read-once formulas proposed in [CGV15], under the DDH assumption and with a polynomial-time reduction. This in turn implies the security of all the other aggregate PRFs in [CGV15], as the latter are particular cases of the aggregate PRFs for read-once formulas. The proof is very simple and based on linear algebra. Up to now, the only known reduction incurred an exponential blow-up in the length n of the input.

Second, we show that our PLP framework enables to very easily prove the security of the multilinear pseudorandom function construction in [CH15]. More importantly, it enables us to directly show the security of the symmetric variant of this construction, under the d-DDHI assumption, which was left as an open problem in [CH15].

Third, we extend all the above constructions to weaker assumptions, as the k-Lin assumption, which can hold in symmetric k-linear groups, contrary to DDH or d-DDHI. Again, these extensions are straightforward to prove thanks to our PLP framework.

Additionally, we solve two other open problems respectively in [CGV15, end of Section 1 and Section 2.2] and in [CH15]: We show that unless NP=BPP, there cannot exist aggregate PRFs for DNF formulas, although satisfiability of DNF formulas can be tested in polynomial time; and we propose the first skew-symmetric multilinear PRF.

Additional Contributions. As a side contribution, we prove the hardness of $\mathcal{E}_{k,d}$-MDDH (defined in [ABP15] and recalled in Section 2) in the generic (symmetric) k-linear group model, which was left as an open problem in [ABP15] for $k > 2$ and $d > 1$. This result directly implies that all the results stated in [ABP15] under the $\mathcal{E}_{2,d}$-MDDH now holds also for $\mathcal{E}_{k,d}$-MDDH, for any $k \geq 2$, which is also an interesting side contribution. To prove this result, we essentially need to prove there are no non-trivial polynomial relations of degree k between the elements of the assumptions (these elements being themselves polynomials), as in [BBG05, Boy08, EHK+13]. The latter polynomial independence is a consequence of Lemma E.5 in Appendix E. The technically difficult part is the proof of this lemma. The proof is by induction over k: for the base case $k = 1$, the proof is straightforward as all the elements we consider are linearly independent; for the inductive case $k = 2$, we basically set some indeterminates to some carefully chosen values (for the polynomials defining the elements we consider) to come down to previous cases.

Paper Organization. The rest of the paper is composed of the following sections. In Section 2 and Appendix A, we give necessary background and notations. We introduce our general PLP security notion and explain our main result, termed PLP theorem (Theorem 3.1), in Section 3. Its proof and some intermediate lemmas are detailed in Appendices B and C. We then present our new constructions and improved security bounds for aggregate and multilinear pseudorandom functions in Section 4 as well as some side results. The proofs of these results are detailed in Appendix D. Finally, in Appendix E, we prove the hardness of our main assumption (the $\mathcal{E}_{k,d}$-MDDH assumption) in the generic k-linear group model.

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

procTestLin$(\mathcal{L}, R)$
// $\mathcal{L}[\ell] = R_\ell$ for $\ell = 1, \ldots, L$ and $L = |\mathcal{L}|$
$R_{L+1} \leftarrow R$
$N \leftarrow 2L + 4$
For $k = 1, \ldots, N$
    $\overrightarrow{\gamma_k} \xleftarrow{\$} \mathbb{Z}_p^n$
$\boldsymbol{M}$ matrix over $\mathbb{Z}_p$ of $L + 1$ rows and N columns
For $\ell = 1, \ldots, L + 1$
    For $k = 1, \ldots, N$
        $m_{\ell,k} \leftarrow R_\ell(\overrightarrow{\gamma_k})$
Apply Gaussian elimination on $\boldsymbol{M}$
If $\boldsymbol{M}$ is full-rank then
    Return $\perp$
Else
    Let $\overrightarrow{\lambda'}$ be the row vector such that $\overrightarrow{\lambda'} \cdot \boldsymbol{M} = \overrightarrow{0}$
    $\overrightarrow{\lambda} \leftarrow (\lambda'_1/\lambda'_{L+1}, \ldots, \lambda'_L/\lambda'_{L+1})$
    Return $\overrightarrow{\lambda}$

Figure 1: **TestLin** procedure

functions $F\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ is 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 $\mathcal{R}$. Also, if $\mathcal{D}$ and $\mathcal{R}$ are vector spaces, we denote by $\mathsf{L}(\mathcal{D}, \mathcal{R})$ the vector space of linear functions from $\mathcal{D}$ to $\mathcal{R}$. In addition, if $\mathcal{D}_1, \ldots, \mathcal{D}_n$ are n vector spaces, then $\mathsf{L}(\mathcal{D}_1 \otimes \cdots \otimes \mathcal{D}_n, \mathcal{R})$ is the vector space of n-linear functions from $\mathcal{D}_1 \times \cdots \times \mathcal{D}_n$ 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 $x = x_1 \,\|\, \ldots \,\|\, x_{|x|}$. 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. We denote by $\mathbb{Z}_p[T_1, \ldots, T_n]$ the subspace of multivariate polynomials in indeterminates $T_1, \ldots, T_n$, and by $\mathbb{Z}_p[T_1, \ldots, T_n]_{\leq d}$ the subring of polynomials of degree at most d in each indeterminate. For a polynomial $P \in \mathbb{Z}_p[T_1, \ldots, T_n]$, we denote by $P(\vec{T})$ the polynomial $P(T_1, \ldots, T_n)$ and by $P(\vec{a})$ its evaluation by setting $\vec{T}$ to $\vec{a}$, meaning that we set $T_1 = a_1, \ldots, T_n = a_n$.

We often implicitly consider a multiplicative group $\mathbb{G} = \langle g \rangle$ with public generator g of order p and we denote by $[a]$ 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 $j = 1, \ldots, m$. All vector spaces are implicitly supposed to be $\mathbb{Z}_p$-vector spaces.

We denote by **TestLin** a procedure which takes as inputs 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:

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

$\overrightarrow{\lambda}$ is uniquely defined since we assume that polynomials from the input list are linearly independent. No such procedure is known for multivariate polynomials, if we require the procedure to be deterministic and polynomial-time. However, it is easy to construct such a randomized procedure which is correct with overwhelming probability. Such a statistical procedure is sufficient for our purpose and was given in [ABPP14]. We recall this procedure in Figure 1. This procedure is correct with probability at least $\frac{p-1}{p}$ as soon as $nd \leq \sqrt{p}$, where d is the maximum degree in one indeterminate and n is the number of indeterminates.

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. In the case where the **Finalize** procedure is not explicitly defined, it is implicitly defined as the procedure that simply outputs its input. 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.

Pseudorandom Functions. A PRF is an efficiently computable ensemble of functions $F\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$, implicitly indexed by the security parameter κ, such that, when $K \xleftarrow{\$} \mathcal{K}$, the function $x \in \mathcal{D} \mapsto F(K, x) \in \mathcal{R}$ is indistinguishable from a random function. Formally, we say that F is a pseudorandom function if the advantage of any adversary $\mathscr{A}$ in attacking the standard PRF security of F is negligible, where this advantage is defined via

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

where games $\mathrm{PRFReal}_F$ and $\mathrm{PRFRand}_F$ are depicted in Figure 2.

Aggregation Function. Let $f\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ be a function. We define an aggregation function by describing two objects:

- a collection $\mathscr{S}$ of subsets S of the domain $\mathcal{D}$;
- an aggregation function $\Gamma\colon \mathcal{R}^* \to \mathcal{V}$ that takes as input a tuple of values from the range $\mathcal{R}$ of F and aggregates them to produce a value in an output set $\mathcal{V}$.

In addition, we require the set ensemble $\mathscr{S}$ to be *efficiently recognizable*, meaning that for any $S \in \mathscr{S}$, there exists a polynomial time procedure to check if $x \in S$, for any $x \in \mathcal{D}$. Also, we require the aggregation function Γ to be polynomial time and the output of the function not to depend on the order of the elements provided as inputs. Finally, we require all sets S to have a representation of size polynomial in the security parameter κ.

Given an aggregation function $(\mathscr{S}, \Gamma)$, we define the aggregate function $\mathrm{AGG} = \mathrm{AGG}_{f,\mathscr{S},\Gamma}$ as the function that takes as input a set $S \in \mathscr{S}$ and outputs the aggregation of all values $f(x)$ for all $x \in S$. That is, $\mathrm{AGG}(S)$ outputs $\Gamma(f(x_1), \ldots, f(x_{|S|}))$, where $S = \{x_1, \ldots, x_{|S|}\}$. We will require the computation of AGG to be polynomial time (even if the input set S is exponentially large) if the function f provided is the pseudorandom function $F(K, \cdot)$ we consider, where K is some key.

Aggregate Pseudorandom Functions. Let $F\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ be a pseudorandom function and let $(\mathscr{S}, \Gamma)$ be an associated aggregation function. We say that F is an $(\mathscr{S}, \Gamma)$-aggregate pseudorandom function ($(\mathscr{S}, \Gamma)$-AGG-PRF) if the advantage of any adversary in attacking the AGG-PRF security of F is negligible, where this advantage is defined via

$$\mathbf{Adv}^{\mathsf{agg\text{-}prf}}_{F,\mathscr{S},\Gamma}(\mathscr{A}) = \Pr\left[\mathrm{AGGPRFReal}^{\mathscr{A}}_{F} \Rightarrow 1\right] - \Pr\left[\mathrm{AGGPRFRand}^{\mathscr{A}}_{F} \Rightarrow 1\right],$$

where games $\mathrm{AGGPRFReal}_F$ and $\mathrm{AGGPRFRand}_F$ are depicted in Figure 2. Game $\mathrm{AGGPRFRand}_F$ may not be polynomial-time, as $\mathrm{AGG}_{f,\mathscr{S},\Gamma}$ may not require to compute an exponential number of values $f(x)$. However, for all the aggregate PRFs that we consider, this game is statistically indistinguishable from a polynomial-time game, using the **TestLin** procedure, similarly to what is done in our new PLP security notion (see Section 3 and Figure 3).

Multilinear Pseudorandom Functions. Multilinear pseudorandom functions are a variant of the standard notion of pseudorandom functions, which works with vector spaces. More precisely, a multilinear pseudorandom function $F\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$, is an efficiently computable function with key space $\mathcal{K}$, domain $\mathcal{D} = \mathcal{D}_1 \times \cdots \times \mathcal{D}_n$ (a cartesian product of n vector spaces $\mathcal{D}_1, \ldots, \mathcal{D}_n$, for some integer n), range $\mathcal{R}$ which is a vector space, and which is indistinguishable from a random n-linear function with same domain and range. We say that F is a multilinear pseudorandom function (MPRF) if the advantage of any adversary in attacking the MPRF security of F is negligible, where this advantage is defined via

$$\mathbf{Adv}^{\mathsf{mprf}}_{F}(\mathscr{A}) = \Pr\left[\mathrm{MPRFReal}^{\mathscr{A}}_{F} \Rightarrow 1\right] - \Pr\left[\mathrm{MPRFRand}^{\mathscr{A}}_{F} \Rightarrow 1\right],$$

where games $\mathrm{MPRFReal}_F$ and $\mathrm{MPRFRand}_F$ are depicted in Figure 2. As explained in [CH15], Game $\mathrm{MPRFRand}_F$ can be implemented in polynomial time using a deterministic algorithm checking linearity of simple tensors [BW04]. Also, similarly to Game $\mathrm{AGGPRFRand}_F$, it is also possible to implement a polynomial-time game that is statistically indistinguishable from $\mathrm{MPRFRand}_F$ using **TestLin**.

$\mathrm{PRFReal}_F$	$\mathrm{PRFRand}_F$
proc Initialize $K \xleftarrow{\$} \mathcal{K}$ **proc Fn**(x) Return $F(K, x)$	**proc Initialize** $f \xleftarrow{\$} \mathsf{Fun}(\mathcal{D}, \mathcal{R})$ **proc Fn**(x) Return $f(x)$

$\mathrm{AGGPRFReal}_F$	$\mathrm{AGGPRFRand}_F$
proc Initialize $K \xleftarrow{\$} \mathcal{K}$ **proc Fn**(x) Return $F(K, x)$ **proc AGG**(S) Return $\mathrm{AGG}_{F(K,\cdot),\mathscr{S},\Gamma}(S)$	**proc Initialize** $f \xleftarrow{\$} \mathsf{Fun}(\mathcal{D}, \mathcal{R})$ **proc Fn**(x) Return $f(x)$ **proc AGG**(S) Return $\mathrm{AGG}_{f,\mathscr{S},\Gamma}(S)$

$\mathrm{MPRFReal}_F$	$\mathrm{MPRFRand}_F$
proc Initialize $K \xleftarrow{\$} \mathcal{K}$ **proc Fn**$(\vec{x})$ Return $F(K, \vec{x})$	**proc Initialize** $f \xleftarrow{\$} \mathsf{L}(\mathcal{D}_1 \otimes \cdots \otimes \mathcal{D}_n, \mathcal{R})$ **proc Fn**$(\vec{x})$ Return $f(\vec{x})$

Figure 2: Security games for (classical, aggregate, multilinear — from top to bottom) pseudorandom functions

Assumptions. Our main theorem is proven under the same MDDH assumption [EHK+13] introduced in [ABP15] and termed $\mathcal{E}_{k,d}$-MDDH assumption. This MDDH assumption is defined by the matrix distribution $\mathcal{E}_{k,d}$ which samples matrices $\boldsymbol{\Gamma}$ as follows

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

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

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

where $\boldsymbol{\Gamma} \xleftarrow{\$} \mathcal{E}_{k,d}$, $\boldsymbol{W} \xleftarrow{\$} \mathbb{Z}_p^{k\times 1}$, $\boldsymbol{U} \xleftarrow{\$} \mathbb{Z}_p^{k(d+1)\times 1}$. This assumption is random self-reducible, as any other MDDH assumption (we will make use of this property in the proof of our main theorem, and recall this property in Appendix A).

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, as recalled in Appendix A). $\mathcal{E}_{1,1}$-MDDH is actually exactly DDH. In [ABP15], the question of the hardness of the $\mathcal{E}_{k,d}$-MDDH problem in the generic k-linear group model was left as an open problem when $d > 1$ and $k > 2$. One of our contributions is to give a proof of hardness of these assumptions, which is detailed in Appendix E.

3 Polynomial Linear Pseudorandomness Security

As we already mentioned in the introduction, while the LIP theorem from [ABP15] is quite powerful to prove the security of numerous constructions of pseudorandom functions (and related-key secure pseudorandom functions), it falls short when we need to prove the security of multilinear pseudorandom functions or aggregate pseudorandom functions. Indeed, the LIP theorem requires that there is no linear dependence between the outputs of the function. Thus, for the latter primitives, it is clear that one

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†	generic k-linear group‡

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

¶ proven in [ABP15];

† proven in the generic (symmetric) bilinear group model [BB04] in [ABP15] (and also as a particular case of Appendix E);

‡ proven in the generic (symmetric) k-linear group model [Sha07, HK07] in Appendix E.

cannot use the LIP theorem, since the main point of these primitives is precisely that outputs can be related.

In order to deal with these primitives, we introduce a new security notion, termed polynomial linear pseudorandomness security (PLP), which encompasses the LIP security notion, but allows to handle multilinear pseudorandom functions and aggregate pseudorandom functions.

3.1 Intuition

Intuitively, the polynomial linear pseudorandomness security notion says that for any polynomials $P_1,\ldots,P_q\in\mathbb{Z}_p[T_1,\ldots,T_n]$, the group elements

$$[P_1(\vec{a})\cdot b],\ldots,[P_q(\vec{a})\cdot b],$$

with $\vec{a}\xleftarrow{\$}\mathbb{Z}_p^n$ and $b\xleftarrow{\$}\mathbb{Z}_p$, are computationally indistinguishable from the group elements:

$$[U(P_1)],\ldots,[U(P_q)],$$

with $U\xleftarrow{\$}\mathsf{L}(\mathbb{Z}_p[T_1,\ldots,T_n]_{\leq d},\mathbb{Z}_p)$ being a random linear function from the polynomial vector space $\mathbb{Z}_p[T_1,\ldots,T_n]_{\leq d}$ (with d the maximum degree of $P_1,\ldots,P_q$ in any indeterminate T_i) to the base field $\mathbb{Z}_p$. Our main theorem (Theorem 3.1) shows that this security notion holds under the $\mathcal{E}_{1,d}$-MDDH assumption (and thus also under DDH for $d=1$ and d-DDHI for $d\geq 2$).

When $P_1,\ldots,P_q$ are linearly independent, $[U(P_1)],\ldots,[U(P_q)]$ are independent random group elements in $\mathbb{G}$. In that sense, the polynomial linear pseudorandomness security notion is a generalization of the LIP security notion.

We remark that, in the generic group model, the polynomial linear pseudorandomness security notion holds trivially, by definition. The difficulty of the work is to prove it under classical assumptions such as the $\mathcal{E}_{1,d}$-MDDH assumption.

Polynomial-Time Games. When we want to formally define the polynomial linear pseudorandomness security notion, we quickly face a problem: how to compute $[U(P_i)]$ for a random linear map $U\xleftarrow{\$}\mathsf{L}(\mathbb{Z}_p[T_1,\ldots,T_n]_{\leq d},\mathbb{Z}_p)$? Such a map can be represented by a (random) vector with $(d+1)^n$ entries. But doing so would make the game in the security notion exponential time. The idea is to define or draw U lazily: each time we need to evaluate it on a polynomial P_i linearly independent of all the previous polynomials P_j (with $j<i$), we define $U(P_i)\xleftarrow{\$}\mathbb{Z}_p$; otherwise, we compute $U(P_i)$ as a linear combination of $U(P_j)$. More precisely, if $P_i=\sum_{j=1}^{i-1}\lambda_j\cdot P_j$, $U(P_i)=\sum_{j=1}^{i-1}\lambda_j\cdot U(P_j)$. As explained in Section 2, no deterministic polynomial-time algorithm for checking linear dependency between polynomials in $\mathbb{Z}_p[T_1,\ldots,T_n]$ is known. But we can use one which is correct which overwhelming probability. We recall that we denote by **TestLin** such an algorithm.

On the Representation of the Polynomials. A second challenge is to define how the polynomials are represented. We cannot say they have to be given in their expanded form, because it would restrict us to polynomials with a polynomial number of monomials and forbid polynomials such as $\prod_{i=1}^{n}(a_i+1)$.

Instead, we only suppose that polynomials can be (partially) evaluated, in polynomial time (in n and d, the maximum degree in each indeterminate). This encompasses polynomials defined by an expression (with $+$ and $\cdot$ operations, indeterminates, and scalars) of polynomial size (in n and d). Details are given in Appendix B.1.

Extension to Weaker Assumptions. Before, showing the formal definition and theorem, let us show an extension of our polynomial linear pseudorandomness security notion to handle weaker assumptions,

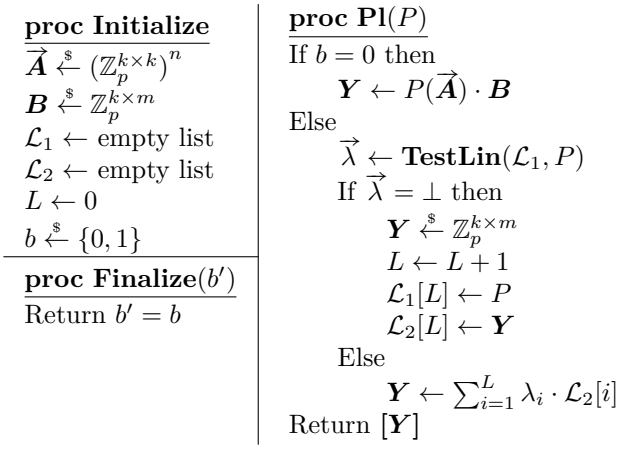

Figure 3: Game defining the (n, d, k, m)-PLP security for a group $\mathbb{G}$

namely $\mathcal{E}_{k,d}$-MDDH, with $k \geq 2$. In that case, we need to evaluate polynomials on matrices: $[P_i(\boldsymbol{A}) \cdot \boldsymbol{B}]$, with $\boldsymbol{A} \xleftarrow{\$} \mathbb{Z}_p^{k\times k}$ and $\boldsymbol{B} \xleftarrow{\$} \mathbb{Z}_p^{k\times m}$ (with $m \geq 1$ being a positive integer). As multiplication of matrices is not commutative, we need to be very careful. We therefore consider that T_n appears before T_{n-1} (in products), T_{n-1} before T_{n-2}, ... (or any other fixed ordering).

More formally, we suppose that polynomials are represented by an expression (similar to the case $k = 1$), such that in any subexpression $Q \cdot R$, if Q contains T_i (formally as an expression and not just when the expression is expanded), then R contains no monomial T_j with $j > i$. Details are given in Appendix B.1.

3.2 Formal Security Notion and Theorem

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, k, m)-PLP security of $\mathbb{G}$, denoted $\mathbf{Adv}_{\mathbb{G}}^{(n,d,k,m)\text{-plp}}(\mathscr{A})$ as the probability of success in the game defined in Figure 3, with $\mathscr{A}$ being restricted to make queries $P \in \mathbb{Z}_p[T_1, \ldots, T_n]_{\leq d}$. When not specified, $m = 1$. When $k = m = 1$, we get exactly the intuitive security notion defined previously, as in that case $\vec{\boldsymbol{A}} = \vec{a} \in \mathbb{Z}_p^n$ and $\boldsymbol{B} = b \in \mathbb{Z}_p$.

Theorem 3.1 (PLP)**.** *Let $\mathbb{G} = \langle g \rangle$ be a group of prime order p. Let $\mathscr{A}$ be an adversary against the (n, d, k, m)-PLP security of $\mathbb{G}$ that makes q oracle queries $P_1, \ldots, P_q$. Then we can design an adversary $\mathscr{B}$ against the $\mathcal{E}_{k,d}$-MDDH problem in $\mathbb{G}$, such that $\mathbf{Adv}_{\mathbb{G}}^{(n,d,k,m)\text{-plp}}(\mathscr{A}) \leq n \cdot d \cdot \mathbf{Adv}_{\mathbb{G}}^{\mathcal{E}_{k,d}\text{-mddh}}(\mathscr{B}) + O(ndqN/p)$, where N is an integer polynomial in the size of the representations of the polynomials and $N = 1$ when $k = 1$ (see Appendix C for details). 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 of Theorem 3.1 is detailed in Appendix C. It is similar to the proof of the LIP theorem (in the matrix case) in [ABP15]. More precisely, we show a series of indistinguishable games where the first game corresponds to the (n, d, k, m)-PLP security game when $b = 0$, and the last game corresponds to this security game when $b = 1$. Basically, all the games except for the last two games are the same as in the proof of the LIP theorem. The two last games differ, as follows: for the LIP theorem, all polynomials are supposed to be linearly independent, and so in the last two games, all the returned values are drawn uniformly and independently, while for the PLP theorem, the returned values still have linear dependencies.

4 Applications

In this section, we describe how PLP theorem (Theorem 3.1) can be used to prove the security of aggregate pseudorandom functions as well as multilinear pseudorandom functions. In particular, we obtain polynomial-time reduction for all previous constructions of aggregate-pseudorandom, even for aggregate where only exponential-time reduction were known (read-once formulas). We also obtain a very simple proof of the multilinear pseudorandom function designed in [CH15]. Finally, we briefly explain how these results can be extended to build constructions based on weaker assumptions in an almost

straightforward manner, by simply changing the key space. The proofs of security remain almost the same and consist in reducing the security to the adequate PLP security game.

4.1 Aggregate Pseudorandom Functions

In this subsection, we show that for all constructions proposed in [CGV15], one can prove the AGG-PRF security with a polynomial time reduction, while proofs proposed in this seminal paper suffered from an exponential (in the input size) overhead in the running time of the reduction. Moreover, our reductions are almost straightforward via the PLP theorem.

A first attempt to solve the issue of the exponential time of the original reductions was done in [CH15]. By introducing multilinear pseudorandom functions and giving a particular instantiation, Cohen and Holmgren showed that one can prove the AGG-PRF security of NR with a polynomial time reduction for hypercubes and decision trees aggregation. However, their technique does not extend to the more general case of read-once formulas aggregation. Also, as we will show it the next subsection, their construction can be seen as a particular case of our main theorem, and then can be proven secure very easily using our result.

Here, we provide a polynomial time reduction for the general case of read-once formulas. This implies in particular the previous results on hypercubes and decision trees which are particular cases of read-once formulas.

Intuitively, if we consider the PLP security for $k = 1$ and aggregation with the Naor-Reingold PRF, our PLP theorem (Theorem 3.1) implicitly says that as long as the aggregate values can be computed as a group element whose discrete logarithm is the evaluation of a multivariate polynomial on the key, then, if the corresponding polynomials have a small representation, the PLP theorem guarantees the security (with a polynomial time reduction), even if the number of points aggregated is superpolynomial. Please notice that if these polynomials do not have any small representation (e.g. the smallest representation is exponential in the input size), then there is no point of considering such aggregation, since the whole point of aggregate pseudorandom function lies in the possibility of aggregating superpolynomially many PRF values with a very efficient computation.

Read-Once Formulas. A read-once formula is a circuit on $x = (x_1, \ldots, x_n) \in \{0,1\}^n$ composed of only AND, OR and NOT gates with fan-out 1, so that each input literal is fed into at most one gate and each gate output is fed into at most one other gate. We denote by ROF_n the family of all read-once boolean formulas over $x_1, \ldots, x_n$ variables. In order to ease the reading, we restrict these circuits to be in a standard form, so that they are composed of fan-in 2 and fan-out 1 AND and OR gates, and NOT gates occurring only at the inputs. This common restriction can be done without loss of generality. Hence, one can see such a circuit as a binary tree where each leaf is labeled by a variable x_i or its negation $\bar{x}_i$ and where each internal node has a label C and has two children with labels C_L and C_R and represents either an AND or an OR gate (with fan-in 2). We identify a formula (and the set it represents) with the label of its root C_ϕ.

Aggregation for Read-Once Formulas. We recall the definition of read-once formula aggregation used in [CGV15]. For the sake of simplicity, we only consider the case of the Naor-Reingold PRF, defined as $\mathsf{NR}(\vec{a}, x) = [a_0 \prod_{i=1}^n a_i^{x_i}]$, where $a_0, \ldots, a_n \xleftarrow{\$} \mathbb{Z}_p$ and $x \in \{0,1\}^n$. We define the aggregation function for read-once formulas of length n as follows.

The collection $\mathscr{S}_{\mathsf{rof}} \subseteq \{0,1\}^n$ corresponds to all the subsets of $S \subseteq \{0,1\}^n$ such that there exists a read-once formula $C_\phi \in \mathrm{ROF}_n$ such that $S = \{x \in \{0,1\}^n \mid C_\phi(x) = 1\}$.

The aggregation function Γ_{rof} is defined as the product (assuming the group is a multiplicative group) of the values on such a subset. Hence, we have:

$$\begin{aligned}\mathrm{AGG}_{\mathsf{NR},\mathscr{S}_{\mathsf{rof}},\Gamma_{\mathsf{rof}}}(C_\phi) &= \prod_{x|C_\phi(x)=1} \left[a_0 \prod_{i=1}^n a_i^{x_i}\right] = \left[a_0 \sum_{x|C_\phi(x)=1} \prod_{i=1}^n a_i^{x_i}\right] \\ &= \left[a_0 \cdot A_{C_\phi,1}(\vec{a})\right] ,\end{aligned}$$

where $A_{C,b}$ is the polynomial $\sum_{x\in\{0,1\}^n|C(x)=b} \prod_{i=1}^n T_i^{x_i}$ for any $C \in \mathrm{ROF}_n$ and $b \in \{0,1\}$.

Efficient Evaluation of $A_{C,b}$. One can efficiently compute $A_{C,b}$ recursively as follows:

- If C is a literal for variable x_i, then $A_{C,1} = T_i$ and $A_{C,0} = 1$ if $C = x_i$; and $A_{C,1} = 1$ and $A_{C,0} = T_i$ if $C = \bar{x}_i$;

- If C is an AND gate with C_L and C_R its two children, then we have:
  $A_{C,1} = A_{C_L,1} \cdot A_{C_R,1}$
  $A_{C,0} = A_{C_L,0} \cdot A_{C_R,0} + A_{C_L,1} \cdot A_{C_R,0} + A_{C_L,0} \cdot A_{C_R,1}$;
- If C is an OR gate with C_L and C_R its two children, then we have:
  $A_{C,1} = A_{C_L,1} \cdot A_{C_R,1} + A_{C_L,1} \cdot A_{C_R,0} + A_{C_L,0} \cdot A_{C_R,1}$
  $A_{C,0} = A_{C_L,0} \cdot A_{C_R,0}$.

Now we have introduced everything, we can prove that NR (or more general constructions) is an $(\mathscr{S}_{\mathsf{rof}}, \Gamma_{\mathsf{rof}})$-AGG-PRF under the standard DDH assumption, as stated in the lemma below.

Lemma 4.1. *Let $\mathbb{G} = \langle g \rangle$ be a group of prime order p and* NR *be the Naor-Reingold PRF defined as* $\mathsf{NR}(\vec{a}, x) = [a_0 \prod_{i=1}^{n} a_i^{x_i}]$, *where the key is* $(a_0, \ldots, a_n) \xleftarrow{\$} \mathbb{Z}_p^{n+1}$ *and the input is* $x \in \{0,1\}^n$. *Then one can reduce the* $(\mathscr{S}_{\mathsf{rof}}, \Gamma_{\mathsf{rof}})$-AGG-PRF *security of* NR *to the hardness of the* DDH *problem in* $\mathbb{G}$, *with a loss of a factor* n. *Moreover, the time overhead is* polynomial *in* n *and in the number of queries made by the adversary.*

The proof is straightforward using the PLP theorem: all queries in the security game for the aggregate PRF can be seen as a queries of the form $\mathbf{Pl}(P)$ for some polynomial P with a small representation: $\mathbf{Fn}(x)$ returns $\mathbf{Pl}(T_0 \prod_{i=1}^{n} T_i^{x_i})$ and $\mathrm{AGG}(C_\phi)$ returns $\mathbf{Pl}(T_0 \cdot A_{C_\phi,1}(\vec{T}))$. Details can be found in Appendix D.1.

Extensions. One can easily extend this result for k-Lin-based PRFs similar to NR using our main theorem. Also, one can easily use our PLP theorem (Theorem 3.1) to prove the security for any aggregate (for instance with NR) as soon as the aggregate values can be represented as group elements whose discrete logarithms are the evaluation of a (multivariate) polynomial on the key (and that this polynomial is efficiently computable).

Impossibility Result for CNF (Conjunctive Normal Form) and DNF (Disjunctive Normal Form) Formulas. In [CGV15], the authors show that, unless NP=BPP, there does not exist an $(\mathscr{S}, \Gamma)$-aggregate pseudorandom function[1], with $\mathcal{D} = \{0,1\}^n$, $\mathscr{S}$ containing the following sets:

$$S_\phi = \{x \in \{0,1\}^n \mid \phi(x) = 1\}$$

with ϕ a CNF formula with n-bit input, and Γ a "reasonable" aggregate function, e.g., Γ_{rof} (assuming $\mathcal{R}$ is a cyclic group $\mathbb{G}$ of prime order p). The proof consists in showing that if such aggregate pseudorandom function exists, then we can solve SAT in polynomial time. More precisely, given a SAT instance, i.e., a CNF formula ϕ, we can compute $\mathrm{AGG}(\phi)$. If ϕ is not satisfiable, $\mathrm{AGG}(\phi) = 1 \in \mathbb{G}$, while otherwise $\mathrm{AGG}(\phi) = \prod_{x \in \{0,1\}^n,\ \phi(x)=1} F(K, x)$. This latter value is not 1 with high probability, otherwise we would get a non-uniform distinguisher against aggregate pseudorandomness.

The case of DNF formulas (or more generally of any class for which satisfiability is tractable) was left as an important open problem in [CGV15]. Here, we show that unless NP=BPP, there also does not exist an $(\mathscr{S}, \Gamma)$-aggregate pseudorandom function as above, when $\mathscr{S}$ contains S_ϕ for any DNF (instead of CNF) formula ϕ with n-bit input. For that, we first remark that the formula $\top$, always true, is a DNF formula (it is the disjunction of all the possible literals), and that the negation $\bar{\phi}$ of a CNF formula ϕ is a DNF formula. Then, given a SAT instance, a CNF formula ϕ, we compute $\mathrm{AGG}(\bar{\phi})$ and $\mathrm{AGG}(\top)$. If ϕ is not satisfiable, $\bar{\phi}$ is always true and $\mathrm{AGG}(\bar{\phi}) = \mathrm{AGG}(\top)$, while otherwise, $\mathrm{AGG}(\bar{\phi}) = \mathrm{AGG}(\top) / \prod_{x \in \{0,1\}^n,\ \phi(x)=1} F(K, x)$. This latter value is not $\mathrm{AGG}(\top)$ with high probability, otherwise we would get a non-uniform distinguisher against aggregate pseudorandomness.

4.2 Multilinear Pseudorandom Functions

Here, we explain how our main theorem can be used to prove directly the security of the multilinear pseudorandom function built in [CH15]. We first recall their construction before explaining how to prove its security.

Cohen-Holmgren multilinear pseudorandom function (CH). Let $\mathbb{G} = \langle g \rangle$ be a group of prime order p. The key space of the multilinear pseudorandom function is $\mathbb{Z}_p^{l_1} \times \cdots \times \mathbb{Z}_p^{l_n}$. The input space is the same as the key space. Given a key $(\overrightarrow{a_1}, \ldots, \overrightarrow{a_n})$ taken uniformly at random in the key space, the

[1] We suppose that the aggregate pseudorandomness security property holds non-uniformly. When $\mathscr{S}$ is expressive enough, we can also do the proof when this security property holds uniformly, see [CGV15, Section 2.2] for details.

evaluation of the multilinear pseudorandom function on the input $(\overrightarrow{x_1}, \ldots, \overrightarrow{x_n})$) outputs:

$$\mathsf{CH}((\overrightarrow{a_1}, \ldots, \overrightarrow{a_n}), (\overrightarrow{x_1}, \ldots, \overrightarrow{x_n})) = \left[\prod_{i=1}^{n} \langle \overrightarrow{a_i}, \overrightarrow{x_i} \rangle\right]$$

where $\langle \overrightarrow{a}, \overrightarrow{x} \rangle$ denotes the canonical inner product $\langle \overrightarrow{a}, \overrightarrow{x} \rangle = \sum_{i=1}^{l} a_i \cdot x_i$, with l being the length of vectors $\overrightarrow{a}$ and $\overrightarrow{x}$.

In [CH15], Cohen and Holmgren prove that this construction is a secure multilinear pseudorandom function under the standard DDH assumption. One of their main contributions is to achieve a polynomial time reduction. Their technique can be seen as a special case of ours. In particular, using our main theorem, one can easily obtain the following lemma.

Lemma 4.2. *Let $\mathbb{G} = \langle g \rangle$ be a group of prime order p and CH: $(\mathbb{Z}_p^{l_1} \times \cdots \times \mathbb{Z}_p^{l_n}) \times (\mathbb{Z}_p^{l_1} \times \cdots \times \mathbb{Z}_p^{l_n}) \to \mathbb{G}$ denote the above multilinear pseudorandom function. Then we can reduce the multilinear PRF security of CH to the hardness of the* DDH *problem in $\mathbb{G}$, with a loss of a factor $l = \sum_{i=1}^{n} l_i$. Moreover, the time overhead is* polynomial *in l and in the number of queries made by the adversary.*

A detailed proof can be found in Appendix D.2, but we give an intuition of the proof in what follows.

Proof. Let $\overrightarrow{T} = (T_{1,1}, \ldots, T_{1,l_1}, \ldots, T_{n,1}, \ldots, T_{n,l_n})$ be a vector of indeterminates, and let $\overrightarrow{T_i} = (T_{i,1}, \ldots, T_{i,l_i})$. The PLP theorem shows that $\mathsf{CH}(\overrightarrow{a_1}, \ldots, \overrightarrow{a_n}, \overrightarrow{x_1}, \ldots, \overrightarrow{x_n})$ (using a random key $\overrightarrow{a}$) is computationally indistinguishable from

$$\left[U\left(\prod_{i=1}^{n} \langle \overrightarrow{T_i}, \overrightarrow{x_i} \rangle\right)\right] = [f(\overrightarrow{x_1}, \ldots, \overrightarrow{x_n})]$$

with $U \xleftarrow{\$} \mathsf{L}(\mathbb{Z}_p[\overrightarrow{T}]_{\leq 1}, \mathbb{Z}_p)$ and

$$f : \begin{pmatrix} \mathbb{Z}_p^{l_1} \times \cdots \times \mathbb{Z}_p^{l_n} & \to & \mathbb{Z}_p \\ (\overrightarrow{x_1}, \ldots, \overrightarrow{x_n}) & \mapsto & U(\prod_{i=1}^{n} \langle \overrightarrow{T_i}, \overrightarrow{x_i} \rangle) \end{pmatrix}.$$

To conclude, we just need to prove that f is a random n-linear function in $\mathsf{L}(\mathbb{Z}_p^{l_1} \otimes \cdots \otimes \mathbb{Z}_p^{l_n}, \mathbb{Z}_p)$.

For that purpose, let us introduce the following n-linear application:

$$\psi : \begin{pmatrix} \mathbb{Z}_p^{l_1} \times \cdots \times \mathbb{Z}_p^{l_n} & \to & \mathbb{Z}_p[\overrightarrow{T}]_{\leq 1} \\ (\overrightarrow{x_1}, \ldots, \overrightarrow{x_n}) & \mapsto & \prod_{i=1}^{n} \langle \overrightarrow{T_i}, \overrightarrow{x_i} \rangle \end{pmatrix}.$$

We remark that f is the composition of U and ψ: $f = U \circ \psi$.

Furthermore, if we write $\overrightarrow{e_{i,l}} = (0, \ldots, 0, 1, 0, \ldots, 0)$ the i-th vector of the canonical base of $\mathbb{Z}_p^l$, then:

$$\psi(\overrightarrow{e_{i_1,l_1}}, \ldots, \overrightarrow{e_{i_n,l_n}}) = T_{1,i_1} \cdots T_{n,i_n};$$

and as the monomials $T_{1,i_1} \cdots T_{n,i_n}$ are linearly independent, ψ is injective. Since $f = U \circ \psi$ and $U \xleftarrow{\$} \mathsf{L}(\mathbb{Z}_p[\overrightarrow{T}]_{\leq 1}, \mathbb{Z}_p)$, the function f is a uniform random linear function from $\mathsf{L}(\mathbb{Z}_p^{l_1} \otimes \cdots \otimes \mathbb{Z}_p^{l_n}, \mathbb{Z}_p)$. This is exactly what we wanted to show. □

Symmetric Multilinear Pseudorandom Function. In [CGV15], constructing symmetric multilinear pseudorandom functions was left as an open problem. The definition of this notion is the same as the notion of multilinear pseudorandom function, except that we only require the function to be indistinguishable from a random *symmetric* multilinear function. In that case, we suppose that $l_1 = \cdots = l_n = l$, i.e., all the vectors $\overrightarrow{x_1}, \ldots, \overrightarrow{x_n}$ have the same size l. The authors wrote in [CGV15] that the natural modification of the CH construction to obtain a symmetric construction consisting in setting $\overrightarrow{a_1} = \overrightarrow{a_2} = \cdots = \overrightarrow{a_n}$ (simply denoted $\overrightarrow{a}$ in what follows) leads to a symmetric multilinear pseudorandom function whose security is less clear, but claimed that it holds under the $\mathcal{E}_{1,n}$-MDDH assumption (which is exactly the n-Strong DDH assumption), when $l = |\overrightarrow{a}| = 2$. We show that this construction is actually secure under the same assumption for any $l = |\overrightarrow{a}| \geq 2$ as stated in the following lemma, whose proof is detailed in Appendix D.3 and is almost the same as the proof of Lemma 4.2.

Lemma 4.3. *Let $\mathbb{G} = \langle g \rangle$ be a group of prime order p and $\mathsf{CH}_{\mathsf{sym}}$: $\mathbb{Z}_p^l \times (\mathbb{Z}_p^l)^n \to \mathbb{G}$ that takes as input a key $\overrightarrow{a} \in \mathbb{Z}_p^l$ and an input $\overrightarrow{x} = (\overrightarrow{x_1}, \ldots, \overrightarrow{x_n}) \in (\mathbb{Z}_p^l)^n$ and outputs $[\prod_{i=1}^{n} \langle \overrightarrow{a}, \overrightarrow{x_i} \rangle]$. Then we can reduce the symmetric multilinear PRF security of $\mathsf{CH}_{\mathsf{sym}}$ to the hardness of the n-*DDHI *problem in $\mathbb{G}$, with a loss of a factor l. Moreover, the time overhead is* polynomial *in l and in the number of queries made by the adversary.*

Skew-Symmetric Multilinear Pseudorandom Function. In [CGV15], the author left as an open problem the construction of a skew-symmetric multilinear pseudorandom function. The definition of this notion is the same as the notion of multilinear pseudorandom function, except that we only require the function to be indistinguishable from a random *skew-symmetric* multilinear function. We assume that $l_1 = \cdots = l_n = l = n$, i.e., all the vectors $\overrightarrow{x_1}, \ldots, \overrightarrow{x_n}$ have the same size $l = n$. We need $l = n$ because there is no skew-symmetric n-multilinear map from $\left(\mathbb{Z}_p^l\right)^n$ to $\mathbb{Z}_p$, when $l < n$.

We know that any skew-symmetric n-multilinear map f is of the form:

$$f(\overrightarrow{x_1}, \ldots, \overrightarrow{x_n}) = c \cdot \det(\overrightarrow{x_1}, \ldots, \overrightarrow{x_n}),$$

with c being a scalar in $\mathbb{Z}_p$ and det being the determinant function. Therefore, the function

$$F(a, (\overrightarrow{x_1}, \ldots, \overrightarrow{x_n})) = [a \cdot \det(\overrightarrow{x_1}, \ldots, \overrightarrow{x_n})]$$

is a skew-symmetric multilinear PRF with key $a \in \mathbb{Z}_p$. The proof is trivial since, $(\overrightarrow{x_1}, \ldots, \overrightarrow{x_n}) \mapsto F(a, (\overrightarrow{x_1}, \ldots, \overrightarrow{x_n}))$ is actually a random skew-symmetric n-multilinear map when a is a random scalar in $\mathbb{Z}_p$. No assumption is required. Our analysis shows that skew-symmetric multilinear PRFs are of limited interest, but our construction still solves an interesting open problem in [CGV15].

Extensions. As for aggregate pseudorandom functions, it is very easy to build multilinear pseudorandom functions under k-Lin and to prove their security applying our PLP theorem (Theorem 3.1), for instance using the same construction but changing the key components from elements in $\mathbb{Z}_p$ to elements in $\mathbb{Z}_p^{k\times k}$ while keeping the same inputs space, and by defining $\langle \vec{\boldsymbol{A}}, \vec{x} \rangle = \sum_{i=1}^{l} x_i \cdot \boldsymbol{A_i}$, with $\vec{\boldsymbol{A}} = (\boldsymbol{A}_1, \ldots, \boldsymbol{A}_l) \in \left(\mathbb{Z}_p^{k\times k}\right)^l$ and $x = (x_1, \ldots, x_l) \in \mathbb{Z}_p^l$. This leads to the following construction:

$$F : \left(\begin{array}{rcl} \mathbb{Z}_p^{l_1} \times \cdots \times \mathbb{Z}_p^{l_n} & \to & \mathbb{G}^{k\times m} \\ (\overrightarrow{x_1}, \ldots, \overrightarrow{x_n}) & \mapsto & \left[\left(\prod_{i=1}^{n} \langle \overrightarrow{\boldsymbol{A}_i}, \vec{x_i} \rangle \right) \cdot \boldsymbol{B} \right] \end{array} \right)$$

with $(\overrightarrow{\boldsymbol{A}_1}, \ldots, \overrightarrow{\boldsymbol{A}_n}) \in \left(\mathbb{Z}_p^{k\times k}\right)^{l_1} \times \cdots \times \left(\mathbb{Z}_p^{k\times k}\right)^{l_n}$ and $\boldsymbol{B} \in \mathbb{Z}_p^{k\times m}$.

Acknowledgements

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

References

[ABP15] Michel Abdalla, Fabrice Benhamouda, and Alain Passelègue. An algebraic framework for pseudorandom functions and applications to related-key security. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 388–409. Springer, Heidelberg, August 2015. (Cited on pages 3, 4, 7, 8, and 9.)

[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 page 5.)

[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.)

[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 8.)

[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 4 and 29.)

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 501–519. Springer, Heidelberg, March 2014. (Cited on page 3.)

[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 4 and 29.)

[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.)

[BW04] Andrej Bogdanov and Hoeteck Wee. A stateful implementation of a random function supporting parity queries over hypercubes. In Klaus Jansen, Sanjeev Khanna, José D. P. Rolim, and Dana Ron, editors, *APPROX-RANDOM 2004*, volume 3122 of *LNCS*, pages 298–309. Springer, Heidelberg, August 2004. (Cited on page 6.)

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 280–300. Springer, Heidelberg, December 2013. (Cited on page 3.)

[CGV15] Aloni Cohen, Shafi Goldwasser, and Vinod Vaikuntanathan. Aggregate pseudorandom functions and connections to learning. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 61–89. Springer, Heidelberg, March 2015. (Cited on pages 3, 4, 10, 11, 12, and 13.)

[CH15] Aloni Cohen and Justin Holmgren. Multilinear pseudorandom functions. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *ICALP 2015, Part I*, volume 9134 of *LNCS*, pages 331–342. Springer, Heidelberg, July 2015. (Cited on pages 3, 4, 6, 9, 10, 11, and 12.)

[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 4, 7, 16, and 29.)

[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 page 3.)

[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 page 15.)

[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 8.)

[KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 13*, pages 669–684. ACM Press, November 2013. (Cited on page 3.)

[NR97] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *38th FOCS*, pages 458–467. IEEE Computer Society Press, October 1997. (Cited on page 3.)

[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 8.)

A Supplementary Definitions

A.1 Standard Assumptions.

Hardness Assumptions. We recall the definition of the Decisional d-Diffie-Hellman Inversion (DDHI) problems given in [GOR11] and described in Figure 4. The advantage of an adversary $\mathscr{D}$ against the d-DDHI problem in $\mathbb{G}$ is

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

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

We also recall the definition of the k-Lin problem in $\mathbb{G}$, also described in Figure 4. The advantage of an adversary $\mathscr{D}$ against the DDHI problem in $\mathbb{G}$ is

$$\mathbf{Adv}^{k\text{-lin}}_{\mathbb{G}}(\mathscr{D}) = \Pr\left[\, k\text{-Lin-Real}^{\mathscr{D}}_{\mathbb{G}} \Rightarrow 1 \,\right] - \Pr\left[\, k\text{-Lin-Rand}^{\mathscr{D}}_{\mathbb{G}} \Rightarrow 1 \,\right]$$

The two assumptions corresponding to the hardness of these problems are the d-DDHI assumption and the k-Lin assumption.

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

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

k-Lin-Real	k-Lin-Rand
proc Initialize $g \xleftarrow{\$} \mathbb{G}$ $a_1, \ldots, a_k \xleftarrow{\$} \mathbb{Z}_p^*$ $w_1, \ldots, w_k \xleftarrow{\$} \mathbb{Z}_p^*$ $z \leftarrow w_1 + \cdots + w_k$ Return $([1], [a_1], \ldots, [a_k],$ $[a_1 w_1], \ldots, [a_k w_k], [z])$ **proc Finalize**(b) Return b	**proc Initialize** $g \xleftarrow{\$} \mathbb{G}$ $a_1, \ldots, a_k \xleftarrow{\$} \mathbb{Z}_p^*$ $w_1, \ldots, w_k \xleftarrow{\$} \mathbb{Z}_p^*$ $z \xleftarrow{\$} \mathbb{Z}_p^*$ Return $([1], [a_1], \ldots, [a_k],$ $[a_1 w_1], \ldots, [a_k w_k], [z])$ **proc Finalize**(b) Return b

Figure 4: Games defining the advantage of an adversary $\mathscr{D}$ against DDHI and the k-Lin problems 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 $(\mathcal{E}_{k,d}, M)$-MDDH denote the M-fold $\mathcal{E}_{k,d}$-MDDH assumption, which is similar to the $\mathcal{E}_{k,d}$-MDDH assumption, except that $\boldsymbol{W} \xleftarrow{\$} \mathbb{Z}_p^{k \times M}$, $\boldsymbol{U} \xleftarrow{\$} \mathbb{Z}_p^{k(d+1) \times M}$.

Specifically, the $(\mathcal{E}_{k,d}, M)$-MDDH assumption is defined by the matrix distribution $\mathcal{E}_{k,d}$ which samples matrices $\boldsymbol{\Gamma}$ as follows

$$\boldsymbol{\Gamma} = \begin{pmatrix} \boldsymbol{A}^0 \cdot \boldsymbol{B} \\ \boldsymbol{A}^1 \cdot \boldsymbol{B} \\ \vdots \\ \boldsymbol{A}^d \cdot \boldsymbol{B} \end{pmatrix} \in \mathbb{Z}_p^{k(d+1) \times k} \quad \text{with } \boldsymbol{A}, \boldsymbol{B} \xleftarrow{\$} \mathbb{Z}_p^{k \times k}. \tag{2}$$

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

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

where $\boldsymbol{\Gamma} \xleftarrow{\$} \mathcal{E}_{k,d}$, $\boldsymbol{W} \xleftarrow{\$} \mathbb{Z}_p^{k\times M}$, $\boldsymbol{U} \xleftarrow{\$} \mathbb{Z}_p^{k(d+1)\times M}$. By random self-reducibility of the $\mathcal{E}_{k,d}$-MDDH assumption, this assumption is actually implied by the latter one, as stated in the above lemma.

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

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

B Multivariate Polynomial Representation

B.1 Multivariate Polynomial Representation in Polynomial Linear Pseudorandomness Security

An important but subtle point of our work is that we do not need polynomials to be given in expanded form in the polynomial linear pseudorandomness security notion. Indeed, otherwise, the theorem would be quite easy to prove but would not encompass a lot of interesting cases.

On the other hand, some representations of polynomials will even not enable us to define properly the PLP game (Figure 3). It is indeed at the very least necessary to be able to evaluate the polynomial at arbitrary points (which may be scalars in $\mathbb{Z}_p$ when $k = 1$, or matrices in $\mathbb{Z}_p^{k\times k}$ when $k \ge 2$). For example, it would be inconceivable 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.

For this appendix only, let us write $\tilde{P}$ the representation of the polynomial, while P is the mathematical polynomial object. The same polynomial may have many representations. In the core of the paper, we always suppose that $\tilde{P}$ has polynomial size in n and d. This assumption is reasonable and simplifies the bounds, but is not required (bounds in theorems would then need to be changed).

The Scalar Case ($k = 1$). In this case, we could actually just require the following condition

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.*

In all cases in our paper, actually, $\tilde{P}$ 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.

The Matrix Case ($k > 1$). In this case, everything is more contrived because of the absence of commutativity. Intuitively, we want that all the indeterminates always appears in the same order, without loss of generality, T_n appears before T_{n-1}, T_{n-1} before T_{n-2}, ... This has to hold not only in the polynomial as a mathematical object, but somehow also "in the representation" if we want to be able to prove something. More precisely, we do not want that, at some point, when evaluating the polynomial, we have to compute $\boldsymbol{A}_j\boldsymbol{A}_{j'}$ with $j' > j$, even if this expression does not appear in the resulting polynomial. For example, the representation $T_2T_3 - T_2(T_3 + T_1)$ is not acceptable (because of the presence of T_2T_3), while the representation T_2T_1 (which corresponds to the same polynomial) is acceptable.

More formally, we assume 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, because of this condition, even if the polynomials we consider would normally be non-commutative, we can as well view them as commutative polynomials (when we evaluate a polynomial from a mathematical point of view, we perform multiplications of the indeterminates in the right order).

We also remark that, when $k = 1$, Condition 2 is stronger than Condition 1.

B.2 Decomposition Lemmas

In order to prove the PLP theorem, we will need to able to run some decomposition algorithms on the polynomials queried. The following lemmas detail the types of decomposition that we require in our proof.

Lemma B.1. *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 $\boldsymbol{C}_1, \ldots, \boldsymbol{C}_N \in \mathbb{Z}_p^{k \times k}$ 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 B.2. *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

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

and regroup terms correctly. □

C Proof of the PLP Theorem (Theorem 3.1)

Preliminaries. We recall that the representations of the polynomials we consider satisfy Condition 1 (when $k = 1$, see Section B.1) or Condition 2 (when $k \geq 2$). Let $\vec{\boldsymbol{A}} \in (\mathbb{Z}_p^{k\times k})^n$. When $k \geq 2$, for any polynomial $P \in \mathbb{Z}_p[T_1, \ldots, T_n]$ whose degree in one indeterminate is at most d and for $j = 1, \ldots, n$, using Lemma B.1, we can decompose $\boldsymbol{Q}_{P,\vec{\boldsymbol{A}},j} = P(T_1, \ldots, T_j, \boldsymbol{A}_{j+1}, \ldots, \boldsymbol{A}_n)$ as follows (in polynomial time):

$$\boldsymbol{Q}_{P,\vec{\boldsymbol{A}},j} = \sum_{\nu=1}^{N_{P,\vec{\boldsymbol{A}},j}} \boldsymbol{C}_{P,\vec{\boldsymbol{A}},j,\nu} \cdot Q_{P,\vec{\boldsymbol{A}},j,\nu}(T_1, \ldots, T_j),$$

with $N_{P,\vec{A},j}$ a positive integer, $\boldsymbol{C}_{P,\vec{\boldsymbol{A}},j,\nu}$ a matrix in $\mathbb{Z}_p^{k\times k}$, and $Q_{P,\vec{\boldsymbol{A}},j,\nu}$ a polynomial in $\mathbb{Z}_p[T_1, \ldots, T_j]$ (given by a representation still satisfying Condition 2), for $\nu = 1, \ldots, N_{P,\vec{A},j}$. We remark that this decomposition exists and can trivially be obtained when $k = 1$ (in this case $N_{P,\vec{\boldsymbol{A}},j} = 1$ and $\boldsymbol{C}_{P,\vec{\boldsymbol{A}},j,1} = 1$. When the index $\vec{\boldsymbol{A}}$ is clear from context, it is omitted. We write N the maximum possible value of $N_{P,j}$ (when $k = 1$, $N = 1$).

Since $Q_{P,j,\nu}$ is a polynomial in $\mathbb{Z}_p[T_1, \ldots, T_j]$, with degree in any indeterminate bounded by d, according to Lemma B.2, we can decompose it in polynomial time as:

$$Q_{P,j,\nu} = Q_{P,j,\nu,0} + T_j \cdot Q_{P,j,\nu,1} + \cdots + T_j^d \cdot Q_{P,j,\nu,d},$$

with $Q_{P,k,\nu,0}, \ldots, Q_{P,k,\nu,d}$ polynomials in $\mathbb{Z}_p[T_1, \ldots, T_j]$ (given by a representation satisfying Condition 1).

In particular, we have:

$$\boldsymbol{Q}_{P,j} = \sum_{\nu=1}^{N_{P,j}} \sum_{i=0}^{d} \boldsymbol{C}_{P,j,\nu} \cdot T_j^i \cdot Q_{P,j,\nu,i} \tag{3}$$

$$\boldsymbol{Q}_{P,j-1} = \sum_{\nu=1}^{N_{P,j}} \sum_{i=0}^{d} \boldsymbol{C}_{P,j,\nu} \cdot A_i^j \cdot Q_{P,j,\nu,i} \tag{4}$$

Finally, we write $\boldsymbol{C}_{P,j,z_1,\ldots,z_j} \in \mathbb{Z}_p^{k\times k}$ the (matrix) coefficient of $T_j^{z_j} \cdots T_1^{z_1}$ in $\boldsymbol{Q}_{P,j}$, and we write $c_{P,j,\nu,z_1,\ldots,z_j} \in \mathbb{Z}_p$ the coefficient of the previous monomial in $Q_{P,j,\nu}$ (or equivalently in $Q_{P,j,\nu,z_j} \cdot T_j^i$). As we have:

$$\boldsymbol{Q}_{P,j} = \sum_{z_1,\ldots,z_j} \boldsymbol{C}_{P,j,z_1,\ldots,z_j} \cdot T_j^{z_j} \cdots T_1^{z_1}$$

$$\boldsymbol{Q}_{P,j} = \sum_{\nu=1}^{N_{P,j}} \sum_{z_1,\ldots,z_j} \boldsymbol{C}_{P,j,\nu} \cdot c_{P,j,\nu,z_1,\ldots,z_j} \cdot T_j^{z_j} \cdots T_1^{z_1}$$

$$\boldsymbol{Q}_{P,j} = \sum_{\nu=1}^{N_{P,j+1}} \sum_{z_1,\ldots,z_j} \sum_{i=0}^{d} \boldsymbol{C}_{P,j+1,\nu} \cdot A_{j+1}^i \cdot c_{P,j+1,\nu,z_1,\ldots,z_j,i} \cdot T_j^{z_j} \cdots T_1^{z_1}$$

we have:

$$\boldsymbol{C}_{P,j,z_1,\ldots,z_j} = \sum_{\nu=1}^{N_{P,j}} \boldsymbol{C}_{P,j,\nu} \cdot c_{P,j,\nu,z_1,\ldots,z_j} \tag{5}$$

$$\boldsymbol{C}_{P,j,z_1,\ldots,z_j} = \sum_{\nu=1}^{N_{P,j+1}} \sum_{i=0}^{d} \boldsymbol{C}_{P,j+1,\nu} \cdot A_{j+1}^i \cdot c_{P,j+1,\nu,z_1,\ldots,z_j,i}. \tag{6}$$

PLP Theorem (Theorem 3.1) . We write N the maximum of the $N_{P,j}$'s and $M = (d+1) \cdot q \cdot N \cdot m$. Let $\mathscr{A}$ be an adversary against the (n, d, k, m)-PLP security of $\mathbb{G}$ that makes q oracle queries. We prove a first statement under the $(\mathcal{E}_{k,d}, M)$-MDDH assumption, which denotes the M-fold $\mathcal{E}_{k,d}$-MDDH assumption: namely, this is the $\mathcal{E}_{k,d}$-MDDH assumption, except that $\boldsymbol{W} \xleftarrow{\$} \mathbb{Z}_p^{k\times M}$. We can then use random self-reducibility to obtain our statement under the $\mathcal{E}_{k,d}$-MDDH assumption. Please refer to Appendix A.2 for formal definitions of this intermediate assumption and of random self-reducibility.

More precisely, we first design an adversary $\mathscr{A}$ against the $(\mathcal{E}_{k,d}, N)$-MDDH problem such that:

$$\mathbf{Adv}^{(n,d,k)\text{-plp}}_{\mathbb{G}}(\mathscr{A}) \leq n \cdot \mathbf{Adv}^{(\mathcal{E}_{k,d},M)\text{-mddh}}_{\mathbb{G}}(\mathscr{B}) + \frac{2n(d+1)qN}{p} + \frac{n}{p} + \frac{n}{p^2} \,. \tag{7}$$

The proof of the above equation is based on the sequence of games in Figure 5. 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.

$\mathrm{G}_{0,j}$	$\mathrm{G}_{1,j}$
proc Initialize // $\mathrm{G}_{0,j}$; $j = 1, \ldots, n$	**proc Initialize** // $\mathrm{G}_{1,j}$; $j = 1, \ldots, n$
$\overrightarrow{\boldsymbol{A}} \xleftarrow{\$} \mathbb{Z}_p^n$	$\overrightarrow{\boldsymbol{A}} \xleftarrow{\$} \mathbb{Z}_p^n$
$\mathcal{L} \leftarrow$ empty list	$\mathcal{L} \leftarrow$ empty list
$\mathsf{T} \leftarrow$ empty 2-dimensional table	$\mathsf{T} \leftarrow$ empty 2-dimensional table
$L \leftarrow 0$ (length of $\mathcal{L}$)	$L \leftarrow 0$ (length of $\mathcal{L}$)
proc RKFn(P) // $\mathrm{G}_{0,j}$; $j = 1, \ldots, n$	**proc RKFn**(P) // $\mathrm{G}_{1,j}$; $j = 1, \ldots, n$
$\boldsymbol{Y} \leftarrow \mathbf{0} \in \mathbb{Z}_p^{k \times m}$	$\boldsymbol{Y} \leftarrow \mathbf{0} \in \mathbb{Z}_p^{k \times m}$
For $\nu = 1, \ldots, N_{P,j}$	For $\nu = 1, \ldots, N_{P,j}$
For $i = 0, \ldots, d$	For $i = 0, \ldots, d$
$\overrightarrow{\lambda}^{(\nu,i)} \leftarrow \mathbf{TestLin}(\mathcal{L}, Q_{P,j,\nu,i})$	$\overrightarrow{\lambda}^{(\nu,i)} \leftarrow \mathbf{TestLin}(\mathcal{L}, Q_{P,j,\nu,i})$
If $\overrightarrow{\lambda}^{(\nu,i)} = \perp$ then	If $\overrightarrow{\lambda}^{(\nu,i)} = \perp$ then
$L \leftarrow L + 1$	$L \leftarrow L + 1$
$\mathcal{L}[L] \leftarrow Q_{P,j,\nu,i}$	$\mathcal{L}[L] \leftarrow Q_{P,j,\nu,i}$
$\mathsf{T}[L, 0] \xleftarrow{\$} \mathbb{Z}_p^{k \times m}$	$\mathsf{T}[L, 0] \xleftarrow{\$} \mathbb{Z}_p^{k \times m}$
For $\ell = 1, \ldots, d$	For $\ell = 1, \ldots, d$
$\mathsf{T}[L, \ell] \xleftarrow{\$} \boldsymbol{A}_j^{\ell} \cdot \mathsf{T}[L, 0]$	$\mathsf{T}[L, \ell] \xleftarrow{\$} \mathbb{Z}_p$
$\overrightarrow{\lambda}^{(\nu,i)} \leftarrow (0, \ldots, 0, 1) \in \mathbb{Z}_p^{L+1}$	$\overrightarrow{\lambda}^{(\nu,i)} \leftarrow (0, \ldots, 0, 1) \in \mathbb{Z}_p^{L+1}$
$\boldsymbol{Y} \leftarrow \boldsymbol{Y} + \boldsymbol{C}_{P,j,\nu} \cdot \sum_{l=1}^{L} \lambda_l^{(\nu,i)} \cdot \mathsf{T}[l, i]$	$\boldsymbol{Y} \leftarrow \boldsymbol{Y} + \boldsymbol{C}_{P,j,\nu} \cdot \sum_{l=1}^{L} \lambda_l^{(\nu,i)} \cdot \mathsf{T}[l, i]$
Return $[\boldsymbol{Y}]$	Return $[\boldsymbol{Y}]$

Figure 5: Games $\mathrm{G}_{0,j}$ and $\mathrm{G}_{1,j}$ for the proof of the PLP theorem. Differences between the two games are in blue

Let us start with the proof. For the sake of simplicity, let us first suppose the procedure **TestLin** is perfect. We will deal with its imperfection at the end of the proof.

We first show that game $\mathrm{G}_{0,1}$ instantiates exactly the game defining the (n, d, k, m)-PLP security of $\mathbb{G}$ when $b = 0$. For any query P, we have $Q_{P,\vec{\boldsymbol{A}},1,\nu} \in \mathbb{Z}_p[T_1]$ (for any $\nu = 1, \ldots, N_{P,1}$) and according to Equation (3):

$$\boldsymbol{Q}_{P,1} = \sum_{\nu=1}^{N_{P,1}} \sum_{i=0}^{d} \boldsymbol{C}_{P,1,\nu,i} \cdot T_1^i \cdot Q_{P,1,\nu,i} \,,$$

with $Q_{P,1,\nu,i} \in \mathbb{Z}_p$ and $\boldsymbol{C}_{P,1,\nu,i} \in \mathbb{Z}_p^{k \times k}$. The first time we see a non-zero coefficient $\alpha = Q_{P,1,\nu,i} \in \mathbb{Z}_p$: $\mathcal{L}[1] \leftarrow \alpha \in \mathbb{Z}_p$, $\mathsf{T}[1, 0] \xleftarrow{\$} \mathbb{Z}_p^{k \times k}$ (let us write this element $\alpha \boldsymbol{A'}$), and $\mathsf{T}[1, \ell] \leftarrow \alpha \cdot \boldsymbol{A}_j^{\ell} \cdot \boldsymbol{A'}$ for $\ell = 1, \ldots, d$. Afterwards, $\mathbf{TestLin}(\mathcal{L}, Q_{P,1,\nu,i})$ always outputs $\overrightarrow{\lambda}^{(\nu,i)} = Q_{P,1,\nu,i}/\alpha$, for $i = 0, \ldots, d$. Then, the matrix $\boldsymbol{Y}$ is computed as

$$\begin{aligned}
\boldsymbol{Y} &= \sum_{\nu=1}^{N_{P,1}} \sum_{i=0}^{d} \boldsymbol{C}_{P,1,\nu} \cdot \sum_{l=1}^{1} \lambda_l^{(\nu,i)} \cdot \mathsf{T}[l, i] \\
&= \sum_{\nu=1}^{N_{P,1}} \sum_{i=0}^{d} \boldsymbol{C}_{P,1,\nu} \cdot Q_{P,1,\nu,i} \cdot \alpha^{-1} \cdot \alpha \cdot \boldsymbol{A}_1^i \cdot \boldsymbol{A'} \\
&= \sum_{\nu=1}^{N_{P,1}} \sum_{i=0}^{d} \boldsymbol{C}_{P,1,\nu} \cdot Q_{P,1,\nu,i} \cdot \boldsymbol{A}_1^i \cdot \boldsymbol{A'} \\
&= \boldsymbol{Q}_{P,0} \cdot \boldsymbol{A'} = P(\boldsymbol{A}_1, \ldots, \boldsymbol{A}_n) \cdot \boldsymbol{A'},
\end{aligned}$$

where the last-but-one equality comes from Equation (4). Hence, this is exactly the game defining the (n, d, k, m)-PLP 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 $(\mathcal{E}_{k,d}, M)$-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 recall that $M = (d+1) \cdot q \cdot N \cdot m$. We design adversaries $\mathscr{B}_j$ attacking the $(\mathcal{E}_{1,d}, M)$-MDDH problem in $\mathbb{G}$ such that

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

The adversary $\mathscr{B}_j$ takes as input a tuple $([\boldsymbol{\Gamma}], [\boldsymbol{Z}]) \in \mathbb{G}^{k(d+1)\times k} \times \mathbb{G}^{(d+1)\times M}$, where either $\boldsymbol{Z} = \boldsymbol{\Gamma} \cdot \boldsymbol{W}$, and $\boldsymbol{W} \xleftarrow{\$} \mathbb{Z}_p^{k\times M}$, or $\boldsymbol{Z} = \boldsymbol{U} \xleftarrow{\$} \mathbb{Z}_p^{k(d+1)\times M}$, 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 $\mathsf{T}[l, i]$ to be the $k \times m$ matrix with $z_{\alpha+ki,\beta+lm}$ as the entry of index $(\alpha, \beta) \in \{1, \ldots, k\} \times \{1, \ldots, m\}$. Assuming the matrix $B \in \mathbb{Z}_p^{k\times k}$ (in the definition of $\boldsymbol{\Gamma}$ in Equation (1)) is invertible (which happens with probability $(1 - 1/p) \cdots (1 - 1/p^k) \geq 1 - 1/p - 1/p^2$, thanks to Euler's Pentagonal Number Theorem), 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 $\mathrm{G}_{1,j}$. 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 6. 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 U is a linear map from $\mathbb{Z}_p[T_1, \ldots, T_j]_{\leq d}$ to $\mathbb{Z}_p^{k\times m}$ and so would be represented by a matrix with $km(d+1)^j$ 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.

proc Initialize // $\mathrm{G}_{2,j}$; $j = 1, \ldots, n$	**proc RKFn**(P) // $\mathrm{G}_{2,j}$; $j = 1 \ldots, n$
$U \xleftarrow{\$} \mathsf{L}(\mathbb{Z}_p[T_1, \ldots, T_j]_{\leq d}, \mathbb{Z}_p^{k\times m})$	$\boldsymbol{Y} \leftarrow \sum_{z_1,\ldots,z_j} \boldsymbol{C}_{P,j,z_1,\ldots,z_j} \cdot U(T_j^{z_j} \cdots T_1^{z_1})$
	Return $[\boldsymbol{Y}]$

Figure 6: Games $\mathrm{G}_{2,j}$ for the proof of the PLP theorem

First, we prove that game $\mathrm{G}_{1,j}$ is perfectly indistinguishable from Game $\mathrm{G}_{2,j}$. For that, we remark that T in $\mathrm{G}_{1,j}$ can be seen as computed as $\mathsf{T}[l, i] = U(T_j^i \cdot Q_l)$, for $k = 0, \ldots, d$, with $U \xleftarrow{\$} \mathsf{L}(\mathbb{Z}_p[T_1, \ldots, T_j]_{\leq d}, \mathbb{Z}_p^{k\times m})$ and $\mathcal{L}[l] = Q_l \in \mathbb{Z}_p[T_1, \ldots, T_{j-1}]$. Indeed, the polynomials $T_j^i \cdot Q_l$ are linearly independent, and so $U(Tj^i \cdot Q_l)$ are independent uniform matrices in $\mathbb{Z}_p^{k\times m}$. We have:

$$Q_{P,j,\nu,i} = \sum_{l=1}^{L} \lambda_l^{(\nu,i)} \cdot Q_l.$$

Thus, for a query P, we remark that the matrix $\boldsymbol{Y}$ is computed as:

$$\begin{aligned}
\boldsymbol{Y} &= \sum_{\nu=1}^{N_{P,j}} \sum_{i=0}^{d} \boldsymbol{C}_{P,j,\nu} \cdot \sum_{l=1}^{L} \lambda_l^{(\nu,i)} \cdot \mathsf{T}[l,i] = \sum_{\nu=1}^{N_{P,j}} \sum_{i=0}^{d} \boldsymbol{C}_{P,j,\nu} \cdot \sum_{l=1}^{L} \lambda_l^{(\nu,i)} \cdot U(T_j^i \cdot Q_l) \\
&= \sum_{\nu=1}^{N_{P,j}} \boldsymbol{C}_{P,j,\nu} \cdot U\left(\sum_{i=0}^{d} \sum_{l=1}^{L} \lambda_l^{(\nu,i)} \cdot T_j^i \cdot Q_l \right) \\
&= \sum_{\nu=1}^{N_{P,j}} \sum_{i=0}^{d} \boldsymbol{C}_{P,j,\nu} \cdot U\left(\sum_{i=0}^{d} Q_{P,j,\nu,i} \cdot T_j^i \right) \\
&= \sum_{\nu=1}^{N_{P,j}} \boldsymbol{C}_{P,j,\nu} \cdot U(\sum_{z_1,\ldots,z_j} c_{P,j,\nu,z_1,\ldots,z_j} \cdot T_j^{z_j} \cdots T_1^{z_1}) \\
&= \sum_{z_1,\ldots,z_j} \left(\sum_{\nu=1}^{N_{P,j}} \boldsymbol{C}_{P,j,\nu} \cdot c_{P,j,\nu,z_1,\ldots,z_j} \right) \cdot U(T_j^{z_j} \cdots T_1^{z_1}) \\
&= \sum_{z_1,\ldots,z_j} \boldsymbol{C}_{P,j,z_1,\ldots,z_j} \cdot U(T_j^{z_j} \cdots T_1^{z_1}),
\end{aligned}$$

where the last equality comes from Equation (5) and most other equalities come from the linearity of U. The matrix $\boldsymbol{Y}$ is computed exactly 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. For that, we remark that T in $\mathrm{G}_{1,j}$ can be seen as computed as $\mathsf{T}[l,i] = \boldsymbol{A}_{j+1}^i \cdot U(Q_l)$, for $i = 0, \ldots, d$, with $U \xleftarrow{\$} \mathsf{L}(\mathbb{Z}_p[T_1, \ldots, T_j]_{\leq d}, \mathbb{Z}_p^{k \times m})$ with $\mathcal{L}[l] = Q_l \in \mathbb{Z}_p[T_1, \ldots, T_j]$. Indeed, the polynomials Q_l are linearly independent, and so $U(Q_l)$ are independent uniform matrices in $\mathbb{Z}_p^{k \times m}$. We also have:

$$Q_{P,j+1,\nu,i} = \sum_{l=1}^{L} \lambda_l^{(\nu,i)} Q_l.$$

Thus, for a query P, we remark that the matrix $\boldsymbol{Y}$ is computed as:

$$\begin{aligned}
\boldsymbol{Y} &= \sum_{\nu=1}^{N_{P,j+1}} \sum_{i=0}^{d} \boldsymbol{C}_{P,j+1,\nu} \cdot \sum_{l=1}^{L} \lambda_l^{(\nu,i)} \cdot \mathsf{T}[l,i] \\
&= \sum_{\nu=1}^{N_{P,j+1}} \sum_{i=0}^{d} \boldsymbol{C}_{P,j+1,\nu} \cdot \sum_{l=1}^{L} \lambda_l^{(\nu,i)} \cdot A_{j+1}^i \cdot U(Q_l) \\
&= \sum_{\nu=1}^{N_{P,j+1}} \sum_{i=0}^{d} \boldsymbol{C}_{P,j+1,\nu} \cdot A_{j+1}^i \cdot U\left(\sum_{l=1}^{L} \lambda_l^{(\nu,i)} \cdot Q_l \right) \\
&= \sum_{\nu=1}^{N_{P,j+1}} \sum_{i=0}^{d} \boldsymbol{C}_{P,j,\nu} \cdot A_{j+1}^i \cdot U(Q_{P,j+1,\nu,i}) \\
&= \sum_{\nu=1}^{N_{P,j+1}} \sum_{i=0}^{d} \boldsymbol{C}_{P,j,\nu} \cdot U(\sum_{z_1,\ldots,z_j} c_{P,j+1,\nu,z_1,\ldots,z_j,i} \cdot T_j^{z_j} \cdots T_1^{z_1}) \\
&= \sum_{z_1,\ldots,z_j} \left(\sum_{\nu=1}^{N_{P,j+1}} \sum_{i=0}^{d} \boldsymbol{C}_{P,j,\nu} \cdot c_{P,j+1,\nu,z_1,\ldots,z_j,i} \right) \cdot U(T_j^{z_j} \cdots T_1^{z_1}) \\
&= \sum_{z_1,\ldots,z_j} \boldsymbol{C}_{P,j,z_1,\ldots,z_j} \cdot U(T_j^{z_j} \cdots T_1^{z_1}),
\end{aligned}$$

where the last equality comes from Equation (6) and most other equalities come from the linearity of U. This is computed exactly as in Game $\mathrm{G}_{2,j}$. Therefore, games $\mathrm{G}_{1,j}$ and $\mathrm{G}_{2,j}$ are perfectly indistinguishable, for $j = 1, \ldots, n$.

We finally prove that Game $\mathrm{G}_{1,n}$ is perfectly indistinguishable from the game defining the (n,d,k,m)-PLP security of $\mathbb{G}$ when $b=1$. Since Game $\mathrm{G}_{1,n}$ is perfectly indistinguishable from Game $\mathrm{G}_{2,n}$, we just need to prove the $\mathrm{G}_{2,0}$ is perfectly indistinguishable from the game defining the (n,d,k,m)-PLP security of $\mathbb{G}$ when $b=1$. This is the case, since $\boldsymbol{Q}_{P,n}=P$ is a polynomial with scalar coefficients, and in the expression of the matrix $\boldsymbol{Y}=\sum_{z_1,\ldots,z_j}\boldsymbol{C}_{P,n,z_1,\ldots,z_n}\cdot U(T_n^{z_n}\cdots T_1^{z_1})$ (in Game $\mathrm{G}_{2,0}$), $\boldsymbol{C}_{P,n,z_1,\ldots,z_n}$ can be seen as a scalar, and $\boldsymbol{Y}=U(Q_{P,n})=U(P)$ by linearity.

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,k,m)-PLP 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 $(d+1)qN/p$ each time, and so at most $2n(d+1)qN/p$ in total.

Equation (7) easily follows from the bounds arising in the different game hops. □

D Proofs of Lemmas in Section 4

In order to prove the statements from Section 4, we will simply reduce the security of our constructions to the PLP security and apply PLP theorem (Theorem 3.1).

D.1 Proof of Lemma 4.1

Let $\mathscr{A}$ be an adversary against the $(\mathscr{S}_{\mathsf{rof}},\Gamma_{\mathsf{rof}})$-AGG-PRF security of NR that makes q oracle queries. We design an adversary $\mathscr{B}$ against the $(n,1,1)$-PLP security in $\mathbb{G}$ as follows, where we denote by $a_1,\ldots,a_n$ and $a_0=b$ the secret values chosen at random in the **Initialize** procedure of the game defining the $(n,1,1)$-PLP security (it corresponds to $\boldsymbol{A}_1,\ldots,\boldsymbol{A}_n$ and $\boldsymbol{B}$ but since $k=1$, these values are simply scalars in $\mathbb{Z}_p$).

$\mathscr{B}$ runs $\mathscr{A}$. The latter can make two types of queries, so let us first describe how $\mathscr{B}$ responds to both these types of queries. The first type consists in standard PRF queries. When adversary $\mathscr{A}$ makes a query x, then $\mathscr{B}$ makes the query $P_x=\prod_{i=1}^n T_i^{x_i}$ and returns the value it gets to $\mathscr{A}$. The second type of queries consists in aggregate PRF values. When $\mathscr{A}$ makes an aggregate query $C_\phi\in\mathrm{ROF}_l$, for some $l\leq n$, the adversary $\mathscr{B}$ computes $A_{C_\phi,1}(\vec{T})$ using the efficient recursive evaluation, as described in Section 4.1, then queries this polynomial $P_{C_\phi}=A_{C_\phi,1}(\vec{T})$, and returns the value it gets to $\mathscr{A}$. As $A_{C_\phi,1}(\vec{T})$ is a multivariate polynomial in $T_1,\ldots,T_l$ with degree at most 1 in any indeterminate (since it is a sum of such multivariate polynomials), clearly, P_{C_ϕ} is a multivariate polynomial with degree at most 1 in any indeterminate.

The only thing we need to prove is that $\mathscr{B}$ simulates correctly either the game AGGPRFReal or the game AGGPRFRand, which is almost straightforward by definition of the PLP security. On the one hand, it is clear that if $\mathscr{B}$'s oracle responds to a query P by $[P(\vec{a})\cdot b]$, then $\mathscr{A}$ gets exactly the (standard or aggregate) values of the Naor-Reingold PRF defined with the generator $g=[1]$ and with the key $(b,a_1,\ldots,a_n)\in\mathbb{Z}_p^{n+1}$. On the other hand, if $\mathscr{B}$'s oracle responds to a query P by random values computed taking into account related between P and previously queried polynomials, then the values $\mathscr{A}$ gets are statistically indistinguishable from the values it would get from the AGGPRFRand oracles. Indeed, the only difference lies in the fact that any value sent to $\mathscr{A}$ is random if the corresponding polynomial is linearly independent from previous queries, or is computed from previous values if the corresponding polynomial is linearly dependent, but these dependence are tested using a statistical procedure (which is correct with probability at least $\frac{p-1}{p}$) while in AGGPRFRand, no error can occur.

Hence, in the first case, $\mathscr{B}$ simulates perfectly AGGPRFReal, while in the second game, the simulation is statistically indistinguishable from AGGPRFRand. Thus, we have shown that

$$\mathbf{Adv}^{\mathsf{agg\text{-}prf}}_{\mathsf{NR},\mathscr{S}_{\mathsf{rof}},\Gamma_{\mathsf{rof}}}(\mathscr{A})\leq\frac{p}{p-1}\cdot\mathbf{Adv}^{(n,1,1)\text{-}\mathsf{plp}}_{\mathbb{G}}(\mathscr{B})$$

and Lemma 4.1 now follows from the PLP theorem (Theorem 3.1) and from the fact that the $\mathcal{E}_{1,1}$-MDDH and the DDH assumptions are equivalent. □

D.2 Proof of Lemma 4.2

Let l denotes $\sum_{i=1}^{n} l_i$. Let $\mathscr{A}$ be an adversary against the MPRF security of CH that makes q oracle queries. We design an adversary $\mathscr{B}$ against the $(l,1,1)$-PLP security in $\mathbb{G}$ as follows, where we denote by $a_{1,1},\ldots,a_{1,l_1},a_{2,1},\ldots,a_{n,l_n}$ the secret values chosen at random in the **Initialize** procedure of the game defining the $(l,1,1)$-PLP security (these secret values correspond to $\boldsymbol{A}_1\cdot\boldsymbol{B},\boldsymbol{A}_2\cdot\boldsymbol{B},\ldots$ but since $k=1$, these values are simply scalars in $\mathbb{Z}_p$).

$\mathscr{B}$ runs $\mathscr{A}$. When the latter makes a query $\overrightarrow{x}=(\overrightarrow{x_1},\ldots,\overrightarrow{x_n})$ where $\overrightarrow{x_i}\in\mathbb{Z}_p^{l_i}$, $\mathscr{B}$ does the following. First, it starts by computing $P_{\overrightarrow{x}}=\prod_{i=1}^{n}\langle\overrightarrow{T_i},\overrightarrow{x_i}\rangle$, where $\overrightarrow{T_i}=(T_{i,1},\ldots,T_{i,l_i})$ is a vector of indeterminates. The latter polynomial is a multivariate polynomial in $\mathbb{Z}_p[T_{1,1},\ldots,T_{1,l_1},T_{2,1},\ldots,T_{n,l_n}]$ with degree at most 1 in any indeterminate.

Afterwards, $\mathscr{B}$ queries $P_{\overrightarrow{x}}$ to its oracle and sends the value it gets to $\mathscr{A}$. Hence, on the one hand, if $\mathscr{B}$'s oracle outputs $[P_{\overrightarrow{x}}(a_{1,1},\ldots,a_{1,l_1},a_{2,1},\ldots,a_{n,l_n})\cdot b]$, then the value $\mathscr{A}$ gets is exactly the evaluation of CH in $\overrightarrow{x}$ with the key $\overrightarrow{a}=(\overrightarrow{a_1},\ldots,\overrightarrow{a_n})$ where $\overrightarrow{a_i}=(a_{i,1},\ldots,a_{i,l_i})$, for $i=1,\ldots,n$, and with the generator $[b]$ (which is a generator as soon as $b\neq 0$). On the other hand, if $\mathscr{B}$'s oracle responds to a query P by random values computed taking into account related between P and previously queried polynomials, then the values $\mathscr{A}$ gets are statistically indistinguishable from the values it would get from the MPRFRand oracle, as this is exactly how the values output by the MPRFRand oracle are computed in order to obtain a polynomial-time simulation (as mentioned in Section 2).

Hence, we have shown that

$$\mathbf{Adv}^{\mathsf{mprf}}_{\mathsf{CH}}(\mathscr{A})\leq\frac{p}{p-1}\cdot\mathbf{Adv}^{(l,1,1)\text{-}\mathsf{plp}}_{\mathbb{G}}(\mathscr{B})$$

and Lemma 4.2 now follows from the PLP theorem (Theorem 3.1) and from the fact that the $\mathcal{E}_{1,1}$-MDDH and the DDH assumptions are equivalent. □

D.3 Proof of Lemma 4.3

Let $\mathscr{A}$ be an adversary against the symmetric MPRF security of $\mathsf{CH}_{\mathsf{sym}}$ that makes q oracle queries. We design an adversary $\mathscr{B}$ against the $(l,1,n)$-PLP security in $\mathbb{G}$ as follows, where we denote by $a_1,\ldots,a_l$ and b the secret values chosen at random in the **Initialize** procedure of the game defining the $(l,1,n)$-PLP security (it corresponds to $\boldsymbol{A}_1,\ldots,\boldsymbol{A}_l$ and $\boldsymbol{B}$ but since $k=1$, these values are simply scalars in $\mathbb{Z}_p$).

$\mathscr{B}$ runs $\mathscr{A}$. When the latter makes a query $\overrightarrow{x}=(\overrightarrow{x_1},\ldots,\overrightarrow{x_n})$ where $\overrightarrow{x_i}\in\mathbb{Z}_p^{l_i}$, $\mathscr{B}$ does the following. First, it starts by computing $P_{\overrightarrow{x}}=\prod_{i=1}^{n}\langle\overrightarrow{T},\overrightarrow{x_i}\rangle$, where $\overrightarrow{T}=(T_1,\ldots,T_l)$ is a vector of indeterminates. Hence, it computes a multivariate polynomial in $\mathbb{Z}_p[T_1,\ldots,T_l]$ with degree at most n in any indeterminate.

Afterwards, $\mathscr{B}$ queries $P_{\overrightarrow{x}}$ to its oracle and sends the value it gets to $\mathscr{A}$. Hence, on the one hand, if $\mathscr{B}$'s oracle outputs $[P_{\overrightarrow{x}}(a_1,\ldots,a_l)\cdot b]$, then the value $\mathscr{A}$ gets is exactly the evaluation of $\mathsf{CH}_{\mathsf{sym}}$ in $\overrightarrow{x}$ with the key $\overrightarrow{a}\in\mathbb{Z}_p^l$ and with the generator $[b]$ (which is a generator as soon as $b\neq 0$). On the other hand, if $\mathscr{B}$'s oracle responds to a query P by random values computed taking into account the linear relations between P and the previously queried polynomials, then the values $\mathscr{A}$ gets are statistically indistinguishable from the values it would get from the symmetric MPRFRand oracle, as this is exactly how the values output by the symmetric MPRFRand oracle are computed in order to obtain a polynomial-time simulation (as mentioned in Section 2).

Hence, we have shown that

$$\mathbf{Adv}^{\mathsf{mprf}}_{\mathsf{CH}_{\mathsf{sym}}}(\mathscr{A})\leq\frac{p}{p-1}\cdot\mathbf{Adv}^{(l,1,n)\text{-}\mathsf{plp}}_{\mathbb{G}}(\mathscr{B})$$

and Lemma 4.2 now follows from the PLP theorem (Theorem 3.1) and from the fact that the $\mathcal{E}_{1,n}$-MDDH is implied by the n-DDHI assumption. □

E Proof of Security of $\mathcal{E}_{k,d}$-MDDH

E.1 Definitions: Monomial Order and Leading Commutative Monomials

Definition E.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 from the set of non-commutative polynomials $\mathbb{Z}_p\langle T_1, \ldots, T_n\rangle$ to the set of (commutative) polynomials $\mathbb{Z}_p[T_1, \ldots, T_n]$, 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 E.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

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

for non-commutative polynomials.

Finally, the *partial degree* of a polynomial P in a set $S \subseteq \{T_1, \ldots, T_n\}$ of indeterminates is the degree of the polynomial P seen as a polynomial with indeterminates $(T_i)_{i \in S}$ and coefficients in $\mathbb{Z}_p[(T_i)_{i \notin S}]$.

E.2 Main Lemma

We will make use of the following main lemma in the security proof of $\mathcal{E}_{k,d}$-MDDH. But before that, we need two technical lemmas.

Lemma E.3. *Let k_1, n_1, n_2, q_1 be positive integers.*

Let $Q_{1,1}, \ldots, Q_{1,q_1} \in \mathbb{Z}_p[X_{1,1}, \ldots, X_{1,n_1}]$ be q_1 polynomials. We suppose that, if there exists a polynomial $R \in \mathbb{Z}_p[U_{1,1}, \ldots, U_{1,q_1}]$ of total degree at most k_1 such that:

$$R(Q_{1,1}, \ldots, Q_{1,q_1}) = 0$$

then $R = 0$.

Then, the same is true when R is in $\mathbb{Z}_p[X_{2,1}, \ldots, X_{2,n_2}][U_{1,1}, \ldots, U_{1,q_1}]$ instead of being in $\mathbb{Z}_p[U_{1,1}, \ldots, U_{1,q_1}]$, i.e., when coefficients of R are polynomials in $\mathbb{Z}_p[X_{2,1}, \ldots, X_{2,n_2}]$ instead of being scalars.

Proof. Let us suppose that R is a polynomial in $\mathbb{Z}_p[X_{2,1}, \ldots, X_{2,n_2}][U_{1,1}, \ldots, U_{1,q_1}]$ of partial degree at most k_1 in $\{U_{1,1}, \ldots, U_{1,q_1}\}$, such that

$$R(Q_{1,1}, \ldots, Q_{1,q_1}) = 0.$$

We want to show that $R = 0$.

Let us look at R as a polynomial in $\mathbb{Z}_p[U_{1,1},\dots,U_{1,q_1}][X_{2,1},\dots,X_{2,n_2}]$:

$$R = \sum_{i_1,\dots,i_{n_2}} \lambda_{i_1,\dots,i_{n_2}} \cdot R_{i_1,\dots,i_{n_2}}(U_{1,1},\dots,U_{1,q_1}) \cdot X_{2,1}^{i_{2,1}} \cdots X_{2,n_2}^{i_{2,n_2}}, \tag{8}$$

where $\lambda_{i_1,\dots,i_{n_2}} \in \mathbb{Z}_p$ and $R_{1,i_1,\dots,i_{n_2}} \in \mathbb{Z}_p[U_{1,1},\dots,U_{1,q_1}]$. Let $R' \in \mathbb{Z}_p[X_{1,1},\dots,X_{1,n_1}][X_{2,1},\dots,X_{2,n_2}]$ be defined as:

$$\begin{aligned} R' &= R(Q_{1,1},\dots,Q_{1,q_1}) \\ &= \sum_{i_1,\dots,i_{n_2}} \lambda_{i_1,\dots,i_{n_2}} \cdot R_{i_1,\dots,i_{n_2}}(Q_{1,1},\dots,Q_{1,q_1}) \cdot X_{2,1}^{i_{2,1}} \cdots X_{2,n_2}^{i_{2,n_2}}. \end{aligned}$$

We know that $R' = 0$. Since $R_{i_1,\dots,i_{n_2}}(Q_{1,1},\dots,Q_{1,q_1}) \in \mathbb{Z}_p[X_{1,1},\dots,X_{1,n_1}]$, the polynomials

$$\lambda_{i_1,\dots,i_{n_2}} \cdot R_{i_1,\dots,i_{n_2}}(Q_{1,1},\dots,Q_{1,q_1})$$

can be seen as the coefficients of the polynomial R' (seen as a polynomial over $\mathbb{Z}_p[X_{1,1},\dots,X_{1,n_1}]$), we have:

$$R_{i_1,\dots,i_{n_2}}(Q_{1,1},\dots,Q_{1,q_1}) = 0\ .$$

As $R_{i_1,\dots,i_{n_2}}$ has degree at most k_1, from the assumption of the lemma, we have $R_{i_1,\dots,i_{n_2}} = 0$. It implies that $R = 0$ (from Equation (8)). □

Lemma E.4. *Let $k_1, k_2, n_1, n_2, q_1, q_2$ be positive integers. Let $Q_{1,1},\dots,Q_{1,q_1} \in \mathbb{Z}_p[X_{1,1},\dots,X_{1,n_1}]$ and $Q_{2,1},\dots,Q_{2,q_2} \in \mathbb{Z}_p[X_{2,1},\dots,X_{2,n_2}]$ be q_1+q_2 polynomials. We suppose that, if there exist polynomials $R_1 \in \mathbb{Z}_p[U_{1,1},\dots,U_{1,q_1}]$ of total degree at most k_1 and $R_2 \in \mathbb{Z}_p[U_{2,1},\dots,U_{2,q_2}]$ of total degree at most k_2 such that:*

$$R_1(Q_{1,1},\dots,Q_{1,q_1}) = 0 \qquad\qquad R_2(Q_{2,1},\dots,Q_{2,q_2}) = 0 \tag{9}$$

then $R_1 = 0$ and $R_2 = 0$. This condition is called Condition $\star$.

Let us suppose there exists a polynomial $R \in \mathbb{Z}_p[U_{1,1},\dots,U_{1,q_1},U_{2,1},\dots,U_{2,q_2}]$ such that any monomial of R is of the form

$$U_{1,1}^{i_{1,1}} \cdots U_{1,q_1}^{i_{1,q_1}} \cdot U_{2,1}^{i_{2,1}} \cdots U_{2,q_2}^{i_{2,q_2}} \qquad \text{with} \begin{cases} i_{1,1}+\cdots+i_{1,q_1} \le k_1 \\ i_{2,1}+\cdots+i_{2,q_2} \le k_2, \end{cases}$$

and such that

$$R(Q_{1,1},\dots,Q_{1,q_1},Q_{2,1},\dots,Q_{2,q_2}) = 0.$$

Then, $R = 0$.

Proof. Let us write R as follows:

$$R = \sum_{i_1+\cdots+i_{q_1} \le k_1} \lambda_{i_1,\dots,i_{q_1}} \cdot R_{2,i_1,\dots,i_{q_1}}(U_{2,1},\dots,U_{2,q_2}) \cdot U_{1,1}^{i_1} \cdots U_{1,q_1}^{i_{q_1}} \tag{10}$$

where $\lambda_{i_1,\dots,i_{q_1}} \in \mathbb{Z}_p$ and $R_{2,i_1,\dots,i_{q_1}} \in \mathbb{Z}_p[U_{2,1},\dots,U_{2,q_2}]$ of degree at most k_2. Let R' be the polynomial in $\mathbb{Z}_p[X_{2,1},\dots,X_{2,n_2}][U_{1,1},\dots,U_{1,q_1}]$, defined as:

$$\begin{aligned} R' &= R(U_{1,1},\dots,U_{1,q_1},Q_{2,1},\dots,Q_{2,q_2}) \\ &= \sum_{i_1+\cdots+i_{q_1} \le k_1} \lambda_{i_1,\dots,i_{q_1}} \cdot R_{2,i_1,\dots,i_{q_1}}(Q_{2,1},\dots,Q_{2,q_2}) \cdot U_{1,1}^{i_1} \cdots U_{1,q_1}^{i_{q_1}}. \end{aligned}$$

As a polynomial over $\mathbb{Z}_p[X_{2,1},\dots,X_{2,n_2}]$ in $U_{1,1},\dots,U_{1,q_1}$, R' has degree at most k_1 and we have that $R'(Q_{1,1},\dots,Q_{1,q_1}) = 0$. Therefore, $R' = 0$ thanks to Lemma E.3 and the assumption that there is no non-zero polynomial R_1 of degree at most k_1 such that $R_1(Q_{1,1},\dots,Q_{1,q_1}) = 0$.

This means that $R_{2,i_1,\dots,i_{q_1}}(Q_{2,1},\dots,Q_{2,q_2}) = 0$ (which is a polynomial in $Z_p[X_{2,1},\dots,X_{2,n_2}]$), for all $i_1,\dots,i_{q_1}$. Since $R_{2,i_1,\dots,i_{q_1}}$ has degree at most k_2, $R_{2,i_1,\dots,i_{q_1}} = 0$. From Equation (10), we get that $R = 0$. □

Lemma E.5 (Main Lemma)**.** *Let k, n, m, q be 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*

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

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

$$\boldsymbol{A}_\ell = \begin{pmatrix} X_{\ell,1,1} & \ldots & X_{\ell,1,k} \\ \vdots & & \vdots \\ X_{\ell,k,1} & \ldots & X_{\ell,k,k} \end{pmatrix} \qquad \boldsymbol{B} = \begin{pmatrix} Y_{1,1} & \ldots & Y_{1,m} \\ \vdots & & \vdots \\ Y_{k,1} & \ldots & Y_{k,m} \end{pmatrix}.$$

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

Finally, let us suppose there exists a polynomial $R \in \mathbb{Z}_p[(U_{s,i,j})_{\substack{s=1,\ldots,q \\ i=1,\ldots,k \\ j=1,\ldots,m}}]$ of total degree at most k, such that

$$R((Q_{s,i,j})_{s,i,j}) = 0. \tag{11}$$

Then, necessarily, $R = 0$ (R is the zero polynomial).

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

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

We do the proof by induction over k.

Base case ($k = 1$). When $k = 1$, $Q_{s,1,1} = P_s((X_{\ell,1,1})_\ell)$ and Equation (11) shows there is a linear combination between the P_s's, which is impossible as their leading monomials are distinct (here everything is commutative, as matrices have size 1×1).

Inductive step. We suppose the lemma holds for some value all values lower than k, and prove it for k.

First, let us show that R contains no monomial of the form:

$$U_{s_1,i_1,j_1} \cdots U_{s_{k_1},i_{k_1},j_{k_1}} \cdot U_{s_{k_1+1},i_{k_1+1},j_{k_1+1}} \cdots U_{s_{k_1+k_2},i_{k_1+k_2},j_{k_1+k_2}},$$

with k_1 and k_2 two positive integers such that $k_1 + k_2 \leq k$ and

$$i_1, \ldots, i_{k_1} \in \{1, \ldots, k_1\} \qquad\qquad i_{k_1+1}, \ldots, i_{k_1+k_2} \in \{k_1+1, \ldots, k\}.$$

More precisely, let k_1, k_2 be two positive integers such that $k_1 + k_2 \leq k$. Let us write R as a sum $R = \tilde{R} + \hat{R}$, with $\tilde{R}$ containing the monomials of the above form, and $\hat{R}$ the other monomials. We want to show that $\tilde{R} = 0$.

Now, in Equation (11), we set $X_{\ell,i,j}$ to 0 for all $\ell = 1, \ldots, n$ and:

$$(i,j) \notin (\{1,\ldots,k_1\} \times \{1,\ldots,k_1\}) \cup (\{k_1+1,\ldots,k\} \times \{k_1+1,\ldots,k\})$$

Concretely, this means that:

$$\boldsymbol{A}_\ell = \begin{pmatrix} \boldsymbol{A}'_\ell & 0 \\ 0 & \boldsymbol{A}''_\ell \end{pmatrix},$$

with

$$\boldsymbol{A}'_\ell = \begin{pmatrix} X_{\ell,1,1} & \ldots & X_{\ell,1,k_1} \\ \vdots & & \vdots \\ X_{\ell,k_1,1} & \ldots & X_{\ell,k_1,k_1} \end{pmatrix} \qquad \boldsymbol{A}''_\ell = \begin{pmatrix} X_{\ell,k_1+1,k_1+1} & \ldots & X_{\ell,k_1+1,k} \\ \vdots & & \vdots \\ X_{\ell,k,k_1+1} & \ldots & X_{\ell,k,k} \end{pmatrix}.$$

Let us also write

$$\boldsymbol{B} = \begin{pmatrix} \boldsymbol{B}' \\ \boldsymbol{B}'' \end{pmatrix},$$

with

$$\boldsymbol{B}' = \begin{pmatrix} Y_{1,1} & \dots & Y_{1,m} \\ \vdots & & \vdots \\ Y_{k_1,1} & \dots & Y_{k_1,m} \end{pmatrix} \qquad \boldsymbol{B}' = \begin{pmatrix} Y_{k_1+1,1} & \dots & Y_{k_1+1,m} \\ \vdots & & \vdots \\ Y_{k,1} & \dots & Y_{k,m} \end{pmatrix}.$$

Therefore, we have:

$$Q_{s,i,j} = \begin{cases} \text{coefficient } (i,j) \text{ of the matrix } P_s(\boldsymbol{A}') \cdot \boldsymbol{B}' & \text{if } 1 \le i \le k_1 \\ \text{coefficient } (i-k_1,j) \text{ of the matrix } P_s(\boldsymbol{A}'') \cdot \boldsymbol{B}'' & \text{if } k_1+1 \le i \le k \end{cases}$$

Thus, all monomials in $\tilde{R}((Q_{s,i,j})_{s,i,j})$ have partial degree k_1 in $\{Y_{i,j}\}_{\substack{i=1,\dots,k_1 \\ j=1,\dots,m}}$ (coming from the polynomials $Q_{s_1,i_1,j_1},\dots,Q_{s_{k_1},i_{k_1},j_{k_1}}$) and partial degree k_2 in $\{Y_{i,j}\}_{\substack{i=k_1,\dots,k \\ j=1,\dots,m}}$ (coming from the polynomials $Q_{s_{k_1+1},i_{k_1+1},j_{k_1+1}},\dots,Q_{s_{k_1+k_2},i_{k_1+k_2},j_{k_1+k_2}}$), while no monomial in $\hat{R}((Q_{s,i,j})_{s,i,j})$ has such partial degrees. Since $R((Q_{s,i,j})_{s,i,j}) = 0$, we have $\tilde{R}((Q_{s,i,j})_{s,i,j}) = 0$.

Now we can apply Lemma E.4, where

$$\begin{aligned} (Q_{1,i})_i \quad &\text{corresponds to} \quad (Q_{s,i,j})_{\substack{s=1,\dots,q \\ i=1,\dots,k_1 \\ j=1,\dots,m}} \\ (X_{1,i})_i \quad &\text{corresponds to} \quad (X_{i,j})_{\substack{i=1,\dots,k_1 \\ j=1,\dots,k_1}} \cup (Y_{i,j})_{\substack{i=1,\dots,k_1 \\ j=1,\dots,m}} \\ (Q_{2,i})_i \quad &\text{corresponds to} \quad (Q_{s,i,j})_{\substack{s=1,\dots,q \\ i=k_1+1,\dots,k \\ j=1,\dots,m}} \\ (X_{2,i})_i \quad &\text{corresponds to} \quad (X_{i,j})_{\substack{i=k_1+1,\dots,k \\ j=k_1+1,\dots,k}} \cup (Y_{i,j})_{\substack{i=k_1+1,\dots,k \\ j=1,\dots,m}} \\ R \quad &\text{corresponds to} \quad \tilde{R}. \end{aligned}$$

Condition $\star$ is satisfied thanks to the induction hypothesis for $k_1 < k$ and $k_2 < k$.

Second, let $1 \le j_1,\dots,j_k \le k$ be positive integers in $\{1,\dots,k\}$. These integers are fixed in all this second step. Let us show that R contains no monomial of the form $U_{s_1,1,j_1}\cdots U_{s_k,1,j_k}$. Let us write R as a sum $R = \tilde{R} + \hat{R}$, with $\tilde{R}$ containing the monomials of the above form, and $\hat{R}$ the other monomials. We remark that all monomials in $\tilde{R}((Q_{s,i,j})_{s,i,j})$ are multiple of $Y_{i_1,j_1}\cdots Y_{i_k,j_k}$ (for some $i_1,\dots,i_k$), while monomials in $\hat{R}((Q_{s,i,j})_{s,i,j})$ are not. Since $R((Q_{s,i,j})_{s,i,j}) = 0$, $\tilde{R}((Q_{s,i,j})_{s,i,j}) = 0$. We now just need to prove that $\tilde{R} = 0$.

We order monomials of $\mathscr{R}$ using the product order on $\{Y_{i,j}\}_{i,j} \times \{X_{\ell,i,j}\}_{\ell,i,j}$, with the lexicographic order on $\{Y_{i,j}\}$ corresponding to the lexicographic order on (i,j), and with the order on $\{X_{\ell,i,j}\}$ corresponding to the lexicographic order on (i,j,ℓ):

$$\begin{aligned} Y_{i,j} < Y_{i',j'} \quad &\iff \quad \left| \begin{array}{l} i < i' \\ \text{or } i = i' \text{ and } j < j' \end{array} \right. \\ X_{\ell,i,j} < X_{\ell',i',j'} \quad &\iff \quad \left| \begin{array}{l} i < i' \\ \text{or } i = i' \text{ and } j < j' \\ \text{or } (i,j) = (i',j') \text{ and } \ell < \ell' \end{array} \right. \end{aligned}$$

Now, we set $X_{\ell,i,j}$ to 0 for all $\ell = 1,\dots,n$ and $i \neq j$ and $i \neq 1$. We also set $X_{\ell,1,i} = X_{\ell,i,i}$. For the sake of simplicity, in this step, we write $X_{\ell,1,i} = X_{\ell,i,i} = X_{\ell,i}$, and $\overrightarrow{X}_i = (X_{1,i},\dots,X_{n,i})$. Concretely, this means that:

$$\boldsymbol{A}_\ell = \begin{pmatrix} X_{\ell,1} & X_{\ell,2} & X_{\ell,3} & \dots & X_{\ell,k} \\ 0 & X_{\ell,2} & 0 & \dots & 0 \\ 0 & 0 & X_{\ell,3} & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & \dots & \dots & 0 & X_{\ell,k} \end{pmatrix}.$$

Then, we get (easily by induction):

$$P_s(\vec{\boldsymbol{A}}) = \begin{pmatrix} P_s(\vec{X}_1) & \mathrm{LT}(P_s(\vec{X}_2))+\dots & \mathrm{LT}(P_s(\vec{X}_3))+\dots & \dots & \mathrm{LT}(P_s(\vec{X}_k))+\dots \\ 0 & P_s(\vec{X}_2) & 0 & \dots & 0 \\ 0 & 0 & P_s(\vec{X}_3) & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & \dots & \dots & 0 & P_s(\vec{X}_k) \end{pmatrix},$$

where $\mathrm{LT}(P_s(\vec{X}_i))+\dots$ corresponds to a polynomial with leading term $\mathrm{LT}(P_s(\vec{X}_i))$. Thus, we have:

$$Q_{s,1,j} = P_s(\vec{X}_1)\cdot Y_{1,j} + (\mathrm{LT}(P_s(\vec{X}_2))+\dots)\cdot Y_{2,j} + \dots + (\mathrm{LT}(P_s(\vec{X}_k))+\dots)Y_{k,j} \quad (12)$$

Let us now suppose by contradiction that $\tilde{R}\neq 0$. Let $U_{s_1,1,j_1}\cdots U_{s_k,1,j_k}$ be the monomial of $\tilde{R}$, for which the tuple $(s_{\sigma(k)},\dots,s_{\sigma(1)})$ is the highest for the lexicographic order, where σ is a permutation of $\{1,\dots,k\}$ such that $s_{\sigma(k)}\geq\dots\geq s_{\sigma(1)}$. When $\tilde{R}$ is evaluated on $(Q_{s,i,j})_{s,i,j}$ this monomial corresponds to $Q_{s_1,1,j_1}\cdots Q_{s_k,1,j_k}$. From Equation (12), we get that the latter expression contains the following monomial:

$$M = P_{s_{\sigma(1)}}(\vec{X}_1)\cdot Y_{1,j_{\sigma(1)}}\cdot \mathrm{LM}(P_{s_{\sigma(2)}}(\vec{X}_2))\cdot Y_{2,j_{\sigma(2)}}\cdots \mathrm{LM}(P_{s_{\sigma(k)}}(\vec{X}_k))\cdot Y_{k,j_{\sigma(k)}}.$$

We just need to prove that this monomial M does not appear in any other polynomial $Q_{s'_1,1,j_1}\cdots Q_{s'_k,1,j_k}$, with $(s'_{\sigma'(k)},\dots,s'_{\sigma'(1)})$ lower or equal to $(s_{\sigma(k)},\dots,s_{\sigma(1)})$ for the lexicographic order (we write it $(s'_{\sigma'(k)},\dots,s'_{\sigma'(1)})\preceq(s_{\sigma(k)},\dots,s_{\sigma(1)})$) and $U_{s'_1,1,j'_k}\cdots U_{s'_k,1,j'_k}\neq U_{s_1,1,j_1}\cdots U_{s_k,1,j_k}$, where σ' is defined similarly to σ. This will implies that $\tilde{R}((Q_{s,i,j})_{s,i,j})\neq 0$ (as it contains the above monomial M which does not get canceled out by other terms), which is impossible.

Let us suppose that $Q_{s'_1,1,j_1}\cdots Q_{s'_k,1,j_k}$ contains the monomial M, with $(s'_{\sigma'(k)},\dots,s'_{\sigma'(1)})\preceq(s_{\sigma(k)},\dots,s_{\sigma(1)})$. We first remark that none of the terms in the "left" part of $Q_{s,i,j}$:

$$P_s(\vec{X}_1)\cdot Y_{1,j} + (\mathrm{LT}(P_s(\vec{X}_2))+\dots)\cdot Y_{2,j} + \dots + (\mathrm{LT}(P_s(\vec{X}_{k-1}))+\dots)Y_{k-1,j}$$

contain monomials multiple of $X_{\ell,i,j}$ for $i\leq k-1$ (from the definition of the monomial order on $\mathscr{R}$). The monomial $\mathrm{LM}(P_{s_{\sigma(k)}}(\vec{X}_k))\cdot Y_{k,j_{\sigma(k)}}$ divides the monomial M and can only come from the "right" part of one $Q_{s'_r,1,j_r}$, because of $Y_{k-1,j}$ which is only present one time in the monomial M. As in addition, $s'_{\sigma'(1)}\leq\dots\leq s'_{\sigma'(k)}\leq s_{\sigma(k)}$ (because $(s'_{\sigma'(k)},\dots,s'_{\sigma'(1)})\preceq(s_{\sigma(k)},\dots,s_{\sigma(1)})$), we get that $s_{\sigma'(k)}=s_{\sigma(k)}$ and $j_{\sigma'(k)}=j_{\sigma(k)}$. We can continue like that by induction and prove that $s'_{\sigma'(k-1)}=s_{\sigma(k-1)}$, $j_{\sigma'(k-1)}=j_{\sigma(k-1)}$, …, $s'_{\sigma'(1)}=s_{\sigma(1)}$, and $j_{\sigma'(1)}=j_{\sigma(1)}$. We finally get that $U_{s'_1,1,j'_k}\cdots U_{s'_k,1,j'_k}\neq U_{s_1,1,j_1}\cdots U_{s_k,1,j_k}$. That was we wanted to prove.

Third, let us conclude by showing all the other cases come down to the first two cases after performing some permutation. More precisely, let σ be a permutation of $\{1,\dots,k\}$. Let $\Sigma\in\mathbb{Z}_p^{k\times k}$ be the corresponding permutation matrix: $\Sigma_{i,j}=1$ if and only if $\sigma(j)=i$, and $\Sigma_{i,j}=0$ otherwise. We also set:

$$\begin{aligned} a'_{\ell,i,j} &= X'_{\ell,i,j} = X_{\ell,\sigma(i),\sigma(j)} \\ b'_{i,j} &= Y'_{i,j} = Y'_{\sigma(i),j} \end{aligned}$$

so that:

$$\begin{aligned} \boldsymbol{A}' &= \Sigma^{-1}\cdot\boldsymbol{A}\cdot\Sigma \\ \boldsymbol{B}' &= \Sigma^{-1}\cdot\boldsymbol{B} \\ P_s(\boldsymbol{A})\cdot\boldsymbol{B} &= \Sigma\cdot P_s(\boldsymbol{A}')\cdot\boldsymbol{B}' \\ Q_{s,i,j} &= Q_{s,\sigma(i),j}((X_{\ell,i,j}\leftarrow X'_{\ell,i,j})_{\ell,i,j},(Y_{i,j}\leftarrow Y'_{i,j})_{i,j}) \end{aligned}$$

Let R' be the polynomial R where $U_{s,i,j}$ is replaced by $U_{s,\sigma(i),j}$. We have:

$$\begin{aligned} R((Q_{s,i,j})) &= R((Q_{s,\sigma(i),j}((X_{\ell,i,j}\leftarrow X'_{\ell,i,j})_{\ell,i,j},(Y_{i,j}\leftarrow Y'_{i,j})_{i,j}))_{s,i,j}) \\ &= R'((Q_{s,i,j})_{s,i,j})((X_{\ell,i,j}\leftarrow X'_{\ell,i,j})_{\ell,i,j},(Y_{i,j}\leftarrow Y'_{i,j})_{i,j}). \end{aligned}$$

Therefore, as $R((Q_{s,i,j})_{s,i,j}) = 0$, we have $R'((Q_{s,i,j})_{s,i,j}) = 0$. And it is clear that $R = 0$ if and only if $R' = 0$.

Let us now show that R contain no monomial $M = U_{s_1,i_1,j_1} \cdots U_{s_{k'},i_{k'},j_{k'}}$, with $k' \leq k$. This will prove that $R = 0$. To do that, let us suppose by contradiction that R contains such monomial M. We consider two cases:

- if $k = k'$ and $i_1 = \cdots = i_k$, then choose σ to be an arbitrary permutation such that $\sigma(i_1) = 1$. We know that the corresponding polynomial R' then contains the monomial $M' = U_{s_1,1,j_1} \cdots U_{s_k,k,j_k}$. But that is impossible according to the second step, as $R'((Q_{s,i,j})_{s,i,j}) = 0$.

- otherwise, let $k_1 = |\{p | i_p = i_1\}|$ be the number of "i" indices equal to i_1. We know that $k_1 < k$. Let $k_2 = k' - k_1$ and let (I_1, I_2) be a partition of $\{1, \ldots, k\}$ such that $|I_1| = k_1$, $|I_2| = k - k_1 \geq k_2$, $i_1 \in I_1$, and for all $i_p \neq i_1$, $i_p \in I_2$. Such a partition exist because there are only k_2 values $i_p \neq i_1$. Then, let σ be an arbitrary permutation such that $\sigma(I_1) = \{1, \ldots, k_1\}$ and $\sigma(I_2) = \{k_1 + 1, \ldots, k\}$. Finally, we remark that the corresponding polynomial R' contains the monomial

$$U_{s_1,i'_1,j_1} \cdots U_{s_{k_1},i'_{k_1},j_{k_1}} \cdot U_{s_{k_1+1},i'_{k_1+1},j_{k_1+1}} \cdots U_{s_{k_1+k_2},i'_{k_1+k_2},j_{k_1+k_2}},$$

  such that:

$$i'_1, \ldots, i'_{k_1} \in \{1, \ldots, k_1\} \qquad\qquad i'_{k_1+1}, \ldots, i'_{k_1+k_2} \in \{k_1 + 1, \ldots, k\},$$

  where $i'_p = \sigma(i_p)$. But the first step of our proof show it is impossible.

This concludes the proof. □

E.3 Security of $\mathcal{E}_{k,d}$-MDDH

Similarly to the proof of Theorem 3 of [EHK+13] and the proof for uber 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 k between entries of $\boldsymbol{\Gamma}$ and $\boldsymbol{Z}$, both when $\boldsymbol{Z} = \boldsymbol{\Gamma} \cdot \boldsymbol{W}$ and when $\boldsymbol{Z} = \boldsymbol{U}$, with

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

Indeterminates are entries of $\boldsymbol{A}_1$ and $\boldsymbol{B}$ ($a_{1,i,j}$, $b_{i,j}$, for $i = 1, \ldots, k$, $j = 1, \ldots, k$), entries of $\boldsymbol{W}$ (w_i, for $i = 1, \ldots, k$), and entries of $\boldsymbol{U}$ ($u_{i,j}$, for $i = 1, \ldots, k(d+1)$, $j = 1, \ldots, k$). The polynomial independence follows from Lemma E.5, with $n = 1$, $q = d + 1$, and $P_s = T_1^{s-1}$, for $s = 1, \ldots, d + 1$. □