Algebraic XOR-RKA-Secure Pseudorandom Functions from Post-Zeroizing Multilinear Maps

Michel Abdalla[1,2], Fabrice Benhamouda[3], and Alain Passelègue[4]

[1] Département d'informatique de l'ENS
École normale supérieure, CNRS, PSL Research University, 75005 Paris, France
michel.abdalla@ens.fr
[2] INRIA
[3] IBM Research, Yorktown Heights, USA
fabrice.benhamouda@normalesup.org
[4] UCLA, Los Angeles, USA
alapasse@gmail.com

Abstract. Due to the vast number of successful related-key attacks against existing block-ciphers, related-key security has become a common design goal for such primitives. In these attacks, the adversary is not only capable of seeing the output of a function on inputs of its choice, but also on related keys. At Crypto 2010, Bellare and Cash proposed the first construction of a pseudorandom function that could provably withstand such attacks based on standard assumptions. Their construction, as well as several others that appeared more recently, have in common the fact that they only consider linear or polynomial functions of the secret key over complex groups. In reality, however, most related-key attacks have a simpler form, such as the XOR of the key with a known value. To address this problem, we propose the first construction of RKA-secure pseudorandom function for XOR relations. Our construction relies on multilinear maps and, hence, can only be seen as a feasibility result. Nevertheless, we remark that it can be instantiated under two of the existing multilinear-map candidates since it does not reveal any encodings of zero. To achieve this goal, we rely on several techniques that were used in the context of program obfuscation, but we also introduce new ones to address challenges that are specific to the related-key-security setting.

Keywords. Pseudorandom Functions, Related-Key Security, Multilinear Maps, Post-Zeroizing Constructions.

1 Introduction

Context. Most of the security models used to prove the security of cryptographic schemes usually assume that an adversary has only a black-box access to the cryptosystem. In particular, the adversary has no information about the secret key, nor can it modify the latter. Unfortunately, it has been shown that this is not always true in practice. For instance, an adversary may learn information from physical measures such as the running time of the protocol or its energy consumption, or may also be able to inject faults in the cryptosystem. In the specific case of fault attacks, in addition of possibly being able to learn partial information about the key, the adversary may be able to force the cryptosystem to run with a different, but related, secret key. Then, by observing the outcomes of the cryptosystem with this new related key, an adversary may be able to break it. Such an attack is called a related-key attack (RKA) and has often been used against concrete blockciphers [Knu93, Bih94, BDK05, BDK+10].

Formalization of RKA security. Following the seminal cryptanalytic works by Biham and Knudsen, theoreticians have defined new security models in order to capture such attacks. In 2003, Bellare and Kohno formalized the foundations for RKA security [BK03]. Specifically, let $F\colon\ \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ be a pseudorandom function and let $\Phi \subseteq \mathsf{Fun}(\mathcal{K}, \mathcal{K})$ be a set of functions on the key space $\mathcal{K}$, called a class of related-key deriving (RKD) functions.We say that F is Φ-RKA secure if it is hard to distinguish an oracle which, on input a pair $(\phi, x) \in \Phi \times \mathcal{D}$, outputs $F(\phi(K), x)$, from an oracle which, on the same input pair, outputs $G(\phi(K), x)$, where $K \in \mathcal{K}$ is a random target key and $G\colon\ \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ is a random function.

Existing Constructions. Building provably RKA-secure pseudorandom functions in non-idealized modelhas been a long-standing open question until the work of Bellare and Cash in 2010 [BC10]. Their construction is adapted from the Naor-Reingold PRF [NR97] and is obtained by applying a generic framework to it. This generic framework has led to other constructions based under different assumptions [LMR14] and been recently extended to circumvent its limitations [ABPP14, ABP15]. However, despite being simple and elegant, the existing frameworks crucially rely on the algebraic structure of the pseudorandom functions and thus only allow to build RKA-secure pseudorandom functions for algebraic classes of RKD functions. More precisely, existing constructions use a key whose components are elements in $\mathbb{Z}_p$, and are proven secure against an attacker that is given the capability to perform operations (additions, multiplications, or even polynomial evaluation) modulo p, with p super-polynomial in the security parameter for instantiations over cyclic or multilinear groups [BC10, LMR14, ABPP14, ABP15], or p polynomial but still much larger than 2, for instantiations based on lattices [LMR14, BP14].

Unfortunately, the algebraic classes of RKD functions above are not very natural as they seem difficult to implement from the perspective of an attacker. Moreover, they also do not seem to match attacks against concrete blockciphers such as AES (e.g., [BDK05, BDK+10]). To address these shortcomings, we focus in this paper on the XOR class of RKD functions, which seems more relevant for practice, as suggested by Bellare and Kohno [BK03]. In this class, which corresponds to the class of functions $\Phi_\oplus = \{\phi_s\colon\ K \in \{0,1\}^k \mapsto K \oplus s \in \{0,1\}^k \mid s \in \{0,1\}^k\}$ with $\mathcal{K} = \{0,1\}^k$ being the keyspace, the adversary is allowed to flip bits of the secret key.

In the context of pseudorandom functions, there have been a few proposals for protecting against weak versions of XOR-related-key attacks. In [AW14], for instance, Applebaum and Widder proposed several schemes that can provably resist XOR attacks by restricting the capabilities of the adversary. In particular, they consider models where the adversary only uses a bounded number of related keys or where it only performs random XOR operations. In [JW15], Jafargholi and Wichs proposed constructions of pseudorandom functions based on continuous non-malleable codes that can resist several forms of related-key attacks, including XOR-related-key attacks. Their solutions, however, only guarantee pseudorandomness under the original key and not under both the original and related keys as in standard notions of related-key security. Another advantage of our construction compared to theirs is that the key in our case is just a bit string. Lastly, to the best of our knowledge, building a pseudorandom function that provably resists XOR-related-key attacks in non-idealized modelsstill remains a major open problem.

In the random oracle model, there is a straightforward construction of a pseudorandom function secure against XOR-related-key attacks: $F(K, x) = H(K \| x)$, with H being a hash function modeled as a random oracle, and $\|$ being the string concatenation operator. However, from a theoretical perspective, having a construction in the random oracle model does not provide any guarantee that the primitive can be realized under non-interactive (and falsifiable) assumptions. Furthermore, we would like to point out that we do not know of any construction of a pseudorandom function secure against XOR-related-key attacks in the generic multilinear group model.

Our Contributions. In this paper, we provide the first provably secure construction of an RKA-secure pseudorandom function for XOR relations. To achieve this goal, our construction departs significantly from previous RKA-secure pseudorandom functions in that the secret key no longer lies in an algebraic group. As in prior constructions (e.g., [BC10, ABPP14, ABP15, LMR14]), our new scheme also requires public parameters, such as generators of a cyclic group, which cannot be tampered with. However, unlike these constructions, the secret key in our scheme is just a bit string, whose values are used to select a subset of the common parameters which will be used in the evaluation of the pseudorandom function.[5] The evaluation is performed using an asymmetric multilinear map [GGH13a, CLT13, GGH15].

In particular, we prove our construction under two non-interactive assumptions, which do not reveal encodings of zero (in the generic multilinear map model) and therefore are plausible under some current instantiations of multilinear maps. To the best of our knowledge, this is the first construction using multilinear maps with such a level of security. In [BMSZ16], Badrinarayanan et al. were the first to take into account zeroizing attacks (i.e., attacks using encodings of zero) against current multilinear maps constructions into their constructions and to propose a scheme which is proven not to reveal encodings of zero, in the generic multilinear map model. But, contrary to us, they did not provide a proof of their scheme under a non-interactive assumption.

[5] Please note that these common parameters are public and can therefore be stored in a non-secure read-only part of the memory which can potentially more easily be made tamper-proof.

Overview of our techniques. As mentioned above, our construction is based on asymmetric multilinear maps [GGH13a, CLT13, GGH15]. Informally speaking, a multilinear map is a generalization of bilinear maps. It allows to "encode" scalars a (in some finite field $\mathbb{Z}_p$) into some encodings $[a]_{\mathcal{S}}$ with respect to an index set (also called an index) $\mathcal{S} \subseteq \mathscr{U}$, where $\mathcal{S}$ indicates the level of the encoding $[a]_{\mathcal{S}}$ and $\mathscr{U}$ denotes the top-level index set. We can add two elements $[a]_{\mathcal{S}}$ and $[b]_{\mathcal{S}}$ belonging to the same index set $\mathcal{S}$ to obtain $[a+b]_{\mathcal{S}}$. We can also multiply elements $[a]_{\mathcal{S}_1}$ and $[b]_{\mathcal{S}_2}$ to compute $[a \cdot b]_{\mathcal{S}_1 \cup \mathcal{S}_2}$ to obtain the encoding of $a \cdot b$ with respect to the index set $\mathcal{S}_1 \cup \mathcal{S}_2$, as long as the two index sets are disjoint ($\mathcal{S}_1 \cap \mathcal{S}_2 = \emptyset$). Finally, it is possible to test whether an element at level $\mathscr{U}$ is an encoding of 0.

Let k and n be the lengths of the secret key K and input x, respectively, and let $\mathscr{U} = \{1, \dots, k+n\}$. Next, let $\{a_{i,b}\}_{i \in [k], b \in \{0,1\}}$ and $\{c_{j,b}\}_{j \in [n], b \in \{0,1\}}$ be random scalars in $\mathbb{Z}_p$ and let $\hat{a}_{i,b} = [a_{i,b}]_{\{i\}}$ be the encoding of $a_{i,b}$ at index level $\{i\}$ and $\hat{c}_{j,b} = [c_{j,b}]_{\{j+k\}}$ be the encoding of $c_{j,b}$ at index level $\{j+k\}$. The starting point of our construction is the function

$$F_{\mathsf{pp}}(K, x) = \left[\prod_{i=1}^{k} a_{i,K_i} \prod_{j=1}^{n} c_{j,x_j} \right]_{\mathscr{U}} ,$$

where the public parameters pp include $\{\hat{a}_{i,b}\}_{i \in [k], b \in \{0,1\}}$ and $\{\hat{c}_{j,b}\}_{j \in [n], b \in \{0,1\}}$ as well as the public parameters of the multilinear map.

Since the encodings of the scalars $a_{i,b}$ and $c_{j,b}$ are included in pp, it is not hard to see that the user in possession of the secret K can efficiently evaluate this function at any point x by computing the multilinear map function with the help of the encodings $\hat{a}_{i,K_i}$ and $\hat{c}_{j,x_j}$. To prove it secure and be able to instantiate the scheme with existing multilinear map candidates, however, is not straightforward as one needs to show that the adversary cannot mix too many existing encodings and create an encoding of zero.

While a proof in the generic multilinear map model [Sho97, GGH$^+$13b, BR14, BGK$^+$14, Zim15] is possible, we would like to rely on non-interactive complexity assumptions over multilinear maps which are likely to hold in existing multilinear map constructions. To achieve this goal, we change the way in which the index sets are defined using techniques from program obfuscation [GGH$^+$13b, BGK$^+$14, Zim15]. More precisely, we make use of the notion of strong straddling sets [BGK$^+$14, MSW14, Zim15, BMSZ16], which informally allows to partition a set into two disjoint sets of subsets so that they cannot be mixed. As in [MSW14, Zim15, BMSZ16], we first construct strong straddling set systems over sets $\mathscr{S}_i$ of k^2 fresh symbols, for $i \in \{0, \dots, k\}$ and use $\mathscr{S}_i$ for $i \geq 1$ to prevent the adversary from mixing an exponential number of inputs. In addition to that, and unlike [MSW14, Zim15, BMSZ16], we use $\mathscr{S}_0$ in the proof to prevent the adversary from mixing an internal private representation of the key (used by the reduction) with the parameters.

As we show in Section 4, the resulting scheme can be proven secure in the generic multilinear map model. In particular, we are actually able to prove that no polynomial-time adversary can construct a (non-trivial) encoding of zero.[6] In addition to that, as we show in Section 5, one benefit of using (strong) straddling sets is that it allows us to prove the security of our construction under non-interactive assumptions and avoid the use of idealized models.[7] Finally, we also prove the plausibility of these new assumptions in Appendix C and Appendix D by showing that they hold in the generic multilinear map model.

We would like to stress that the security proof in Section 5 is a much stronger result qualitatively than a direct proof in the generic multilinear map model since the latter model is only used to show that our (non-interactive) assumptions are plausible.

Concrete Instantiations. The security of our scheme relies on two new non-interactive assumptions on multilinear maps. However, contrary to most classical assumptions (such as the Decisional Diffie-Hellman assumption for low-level group elements) or the multilinear subgroup assumption used to construct witness encryption and indistinguishable obfuscation in [GLW14, GLSW15], our new assumptions do not reveal any encoding of zero. More precisely, similarly to what Badrinarayanan et al. did in [BMSZ16], we show in the ideal multilinear map model, that given an instance for one of our assumptions, the adversary cannot construct any encoding of zero (even at the top level). In particular, this implies that our assumptions holds in the hybrid graded encoding model [GMS16], as this model just restricts the shape of the encodings of zero that the adversary can construct, while we prove that the adversary cannot construct any such encoding.

Our assumptions are therefore not broken by most of the attacks against multilinear maps [GGH13a, CHL$^+$15, CGH$^+$15, HJ16, CLLT16] including the recent annihilation attacks [MSZ16, CGH17], as these attacks need encodings of zero at least at the top level. Hence, while the GGH13 and CLT15 multilinear map candidates [GGH13a, CLT15] might not be used for our construction because of recent attacks without encodings of zero [CFL$^+$16, ABD16, CJL16], our assumptions are plausible when implemented with the GGH15[8] or the CLT13 multilinear map candidates [GGH15, CLT13].

Additional Related Work. In addition to the work mentioned above, a few other constructions of pseudorandom functions against related-key attacks for linear and polynomial functions have been proposed in [BLMR13, LMR14]. While their RKA-secure pseudorandom functions also require multilinear maps, their security proofs are based on assumptions which reveal encodings of zero and hence are subject to the multiple attacks mentioned above. Moreover, these works do not consider XOR-related-key attacks.

[6] Note that, to prove that two games are indistinguishable in the generic multilinear map model, it suffices to prove that the adversary cannot generate encodings of zeros in either game as it would only obtain random handles which carry no information.

[7] The security result in Section 4 is therefore implied by the security result in Section 5. Section 4 should be seen as a warm-up for Section 5.

[8] The GGH15 multilinear map is defined over a graph. In our case as in the case of indistinguishable obfuscation candidate, a single chain can be used as a graph. See [GGH15, Section 5.2].

Related-key security with respect to XOR relations has also been considered in the context of Even-Mansour ciphers [CS15,FP15,Men16,WLZZ16]. Unlike our work, which aims to prove security under well specified non-interactive complexity assumptions, all these works rely on idealized models, which we want to avoid.

In [FX15], the authors proposed efficient constructions of identity-based encryption and key encapsulation schemes that remain secure against related-key attacks for a large class of functions, which include XOR relations. While their results are interesting, we remark that achieving RKA security for randomized primitives appears to be significantly easier than for deterministic ones, as already noted by Bellare and Cash [BC10].

Finally, it is worth mentioning here that, due to the results of Bellare, Cash, and Miller [BCM11], RKA security for pseudorandom functions can be transferred to several other primitives, including identity-based encryption, signatures, and chosen-ciphertext-secure public-key encryption.

Organization. The rest of the paper is organized as follows. Section 2 presents standard definitions of (related-key) pseudorandom functions and multilinear maps. It also introduces the generic multilinear map model and the notion of straddling set system. Section 3 explains our construction of XOR-RKA PRF. Section 4 provides a first security proof of our new scheme in the generic multilinear map model. It also shows that is not feasible for an adversary to generate non-trivial encodings of zero. Finally, Section 5 describes our main result, which is a proof of security for our new construction under two new non-interactive assumptions. The formal proof of security, as well as the proofs of our assumptions are detailed in the appendix. In particular, we prove that our assumptions are not only secure in the generic multilinear map model, but also that it is not feasible for an adversary to generate (non-trivial) encodings of zero. Hence, our assumptions are plausible given some current instantiations of multilinear maps.

2 Definitions

2.1 Notation and Games

Notation. We denote by κ the security parameter. Let $F\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ be a function that takes a key $K \in \mathcal{K}$ and an input $x \in \mathcal{D}$ and returns an output $F(K,x) \in \mathcal{R}$. The set of all functions $F\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ is then denoted by $\mathsf{Fun}(\mathcal{K},\mathcal{D},\mathcal{R})$. Likewise, $\mathsf{Fun}(\mathcal{D},\mathcal{R})$ denotes the set of all functions mapping $\mathcal{D}$ to $\mathcal{R}$. If S is a set, then we denote by $s \overset{\$}{\leftarrow} S$ 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 $|x|$, x_i its i-th bit, so $x \in \{0,1\}^{|x|}$ and $x = x_1 \,\|\, \ldots \,\|\, x_n$.

Games [BR06]. Most of our definitions and proofs use the code-based game-playing framework, in which a game has an **Initialize** procedure, procedures to respond to adversary oracle queries, and a **Finalize** procedure. To execute a game G with an adversary $\mathscr{A}$, we proceed as follows. First, **Initialize** is executed and its outputs become the input of $\mathscr{A}$. When $\mathscr{A}$ executes, its oracle queries are answered by the corresponding procedures of G. When $\mathscr{A}$ terminates, its outputs become the input of **Finalize**. The output of the latter, denoted $\mathrm{G}^{\mathscr{A}}$ is called the output of the game, and we let "$\mathrm{G}^{\mathscr{A}} \Rightarrow 1$" denote the event that this game output takes the value 1. The running time of an adversary by convention is the worst case time for the execution of the adversary with any of the games defining its security, so that the time of the called game procedures is included.

2.2 Pseudorandom Functions

Our definitions of pseudorandom functions and related-key secure pseudorandom functions include a Setup algorithm that is used to generate public parameters. For classical PRFs, the public parameters could actually just be included in the key. However, to our knowledge, all the known proven RKA-secure PRFs [BC10, ABPP14, LMR14, ABP15], use public parameters. Contrary to the key itself, the public parameters cannot be modified by the related-key deriving function. In our case, the Setup algorithm is specified explicitly in the construction for clarity.

PRFs [GGM86, BC10]. Consider a pseudorandom function $F_{\mathsf{pp}} \colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ with public parameters pp. The advantage of an adversary $\mathscr{A}$ in attacking the standard PRF security of a function F_{pp} is defined via

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

Game $\mathrm{PRFReal}_{F_{\mathsf{pp}}}$ first runs the Setup algorithm to generate the public parameters pp which it outputs. It then picks $K \xleftarrow{\$} \mathcal{K}$ at random, and responds to oracle query $\mathbf{Fn}(x)$ via $F_{\mathsf{pp}}(K, x)$. Game $\mathrm{PRFRand}_{F_{\mathsf{pp}}}$ runs Setup and outputs the public parameters pp. It then picks $f \xleftarrow{\$} \mathsf{Fun}(\mathcal{D}, \mathcal{R})$ and responds to oracle query $\mathbf{Fn}(x)$ via $f(x)$.

RKA-PRFs [BK03, BC10]. Consider a pseudorandom function $F_{\mathsf{pp}} \colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ with public parameters pp. Let $\Phi \subseteq \mathsf{Fun}(\mathcal{K}, \mathcal{K})$; the members of Φ are called RKD (Related-Key Deriving) functions. An adversary is said to be Φ-restricted if its oracle queries (ϕ, x) satisfy $\phi \in \Phi$. The advantage of a Φ-restricted adversary $\mathscr{A}$ in attacking the RKA-PRF security of F_{pp} is defined via

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

Game $\mathrm{RKPRFReal}_{F_{\mathsf{pp}}}$ first runs the Setup algorithm to generate the public parameters which it outputs. It then picks a key $K \xleftarrow{\$} \mathcal{K}$ at random, and responds to oracle query $\mathbf{RKFn}(\phi, x)$ via $F_{\mathsf{pp}}(\phi(K), x)$. Game $\mathrm{RKPRFRand}_{F_{\mathsf{pp}}}$ runs Setup to generate the public parameters pp and outputs them. It then picks $G \xleftarrow{\$} \mathsf{Fun}(\mathcal{K}, \mathcal{D}, \mathcal{R})$ and $K \xleftarrow{\$} \mathcal{K}$ at random, and responds to oracle query $\mathbf{RKFn}(\phi, x)$ via $G(\phi(K), x)$. We say that F_{pp} is a Φ-RKA-secure PRF if for any Φ-restricted adversary, its advantage in attacking the RKA-PRF security is negligible.

XOR-RKA-PRFs. Let $F_{\mathsf{pp}}\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ be a pseudorandom function with public parameters pp and $\mathcal{K} = \{0,1\}^k$ for some integer $k \geq \kappa$. We say that F_{pp} is XOR-RKA-secure if it is a $\Phi_\oplus$-RKA-secure PRF according to the above definition, where $\Phi_\oplus = \{\phi_s\colon\ K \in \{0,1\}^k \mapsto K \oplus s \in \{0,1\}^k \mid s \in \{0,1\}^k\}$.

2.3 Multilinear Maps

We informally introduced multilinear maps in the introduction. Let us now introduce formal definitions, following the notations of [Zim15].

Definition 1 [Formal Symbol] A *formal symbol* is a bitstring in $\{0,1\}^*$. Distinct variables denote distinct bitstrings, and we call a *fresh* formal symbol any bitstring in $\{0,1\}^*$ that has not already been assigned to a formal symbol.

Definition 2 [Index Sets] An *index set* (also called *index*)is a set of formal symbols.

Definition 3 [Multilinear Map] A multilinear map is a tuple of six algorithms ($\mathsf{MM.Setup}, \mathsf{MM.Encode}, \mathsf{MM.Add}, \mathsf{MM.Mult}, \mathsf{MM.ZeroTest}, \mathsf{MM.Extract}$) with the following properties:

– $\mathsf{MM.Setup}$ takes as inputs the security parameter κ in unary and an index set $\mathscr{U}$, termed the *top-level index set*, and generates public parameters $\mathsf{mm.pp}$, secret parameters $\mathsf{mm.sp}$, and a prime number p;
– $\mathsf{MM.Encode}$ takes as inputs secret parameters $\mathsf{mm.sp}$, a scalar $x \in \mathbb{Z}_p$, and an index set $\mathcal{S} \subseteq \mathscr{U}$ and outputs:

$$\mathsf{MM.Encode}(\mathsf{mm.sp}, x, \mathcal{S}) \to [x]_{\mathcal{S}} \ ;$$

For the index set $\mathcal{S} = \emptyset$, $[x]_\emptyset$ is simply the scalar $x \in \mathbb{Z}_p$.
– $\mathsf{MM.Add}$ takes as inputs public parameters $\mathsf{mm.pp}$ and two encodings with same index set $\mathcal{S} \subseteq \mathscr{U}$ and outputs:

$$\mathsf{MM.Add}(\mathsf{mm.pp}, [x]_{\mathcal{S}}, [y]_{\mathcal{S}}) \to [x+y]_{\mathcal{S}} \ ;$$

– $\mathsf{MM.Mult}$ takes as inputs public parameters $\mathsf{mm.pp}$ and two encodings with index sets $\mathcal{S}_1, \mathcal{S}_2 \subseteq \mathscr{U}$ respectively and outputs:

$$\mathsf{MM.Mult}(\mathsf{mm.pp}, [x]_{\mathcal{S}_1}, [y]_{\mathcal{S}_2}) \to \begin{cases} [xy]_{\mathcal{S}_1 \cup \mathcal{S}_2} & \text{if } \mathcal{S}_1 \cap \mathcal{S}_2 = \emptyset \\ \perp & \text{otherwise} \end{cases} \ ;$$

– $\mathsf{MM.ZeroTest}$ takes as inputs public parameters $\mathsf{mm.pp}$ and a *top-level* encoding (with index set $\mathscr{U}$) and outputs:

$$\mathsf{MM.ZeroTest}(\mathsf{mm.pp}, [x]_{\mathcal{S}}) \to \begin{cases} \text{“zero”} & \text{if } \mathcal{S} = \mathscr{U} \text{and } x = 0 \\ \text{“non-zero”} & \text{otherwise} \end{cases} \ ;$$

– $\mathsf{MM.Extract}$ takes public parameters $\mathsf{mm.pp}$ and a top-level encoding $[x]_{\mathscr{U}}$ as inputs and outputs a canonical and randomrepresentation of $[x]_{\mathscr{U}}$.

Remark 4 The $\mathsf{MM.Extract}$ algorithm is needed for our pseudorandom function to be deterministic with all currently known instantiations of multilinear maps [GGH13a, CLT13, GGH15, CLT15]. Indeed, in these instantiations, the same group element has many different representations, and the extraction procedure enables to extract a unique representation from any top-level group element (i.e., of index $\mathscr{U}$).

This extraction is necessary for our proof under non-interactive assumptions in Section 5 to work. For our proof in the generic multilinear map model, this is not required. For this reason, our generic multilinear map model does not support extraction for the sake of simplicity. Actually, this only strengthens the result, as before extraction, the adversary still has to possibility to add top-level group elements while extracted values are not necessarily homomorphic.

Conventions. In order to ease the reading, we adopt the following conventions in the rest of the paper:

– Scalars are noted with lowercase letter, e.g. $a, b, \dots$
– Encodings are noted either as their encoding at index set $\mathcal{S}$, $[a]_{\mathcal{S}}$ or simply with a hat, when the index set is clear from the context, e.g. $\hat{a}, \hat{b}, \dots$ In particular, $\hat{a}$ is an encoding of the scalar a.
– Index sets as well as formal variables are noted with uppercase letters, e.g. $X, S, \mathscr{S}, \dots$.
– We denote by $S_1 \cdot S_2$ or $S_1 S_2$ the union of sets S_1 and S_2. This notation implicitly assumes that the two sets are disjoint. If S_1 is an element, then $S_1 \cdot S_2$ stands for $\{S_1\} \cdot S_2$.
– The top-level index set is refered as $\mathscr{U}$.

We also naturally extend these notations when clear from the context, so for instance $\hat{a} + \hat{b} = \mathsf{MM.Add}(\mathsf{mm.pp}, \hat{a}, \hat{b})$ and $\hat{a} \cdot \hat{b} = \mathsf{MM.Mult}(\mathsf{mm.pp}, \hat{a}, \hat{b})$.

2.4 Generic Multilinear Map Model

Our construction is proven secure under two non-interactive assumptions. One is very classical and is a variant of DDH. The other is relatively simple but is not classical and we prove it in the generic multilinear map model to show its plausibility. That is why we need to introduce the generic multilinear map model, in addition to the fact that we also prove in this model that our construction does not enable the adversary to produce encodings of zero.

The generic multilinear map model is similar to the generic group model [Sho97]. Roughly speaking, the adversary has only the capability to apply operations (add, multiply, and zero-test) of the multilinear map to encodings. A scheme is secure in the generic multilinear map model if for any adversary breaking the real scheme, there is a generic adversary that breaks a modified scheme in which encodings are replaced by fresh nonces, called *handles*, that it can supply to a stateful oracle $\mathscr{M}$, defined as follows:

Definition 5 [Generic Multilinear Map Oracle] A *generic multilinear map oracle* is a stateful oracle $\mathscr{M}$ that responds to queries as follows:

- On a query $\mathsf{MM.Setup}(1^\kappa, \mathscr{U})$, $\mathscr{M}$ generates a prime number p and parameters $\mathsf{mm.pp}, \mathsf{mm.sp}$ as fresh nonces chosen uniformly at random from $\{0,1\}^\kappa$. It also initializes an internal table $T \leftarrow []$ that it uses to store queries and handles. It finally returns $(\mathsf{mm.pp}, \mathsf{mm.sp}, p)$ and set internal state so that subsequent $\mathsf{MM.Setup}$ queries fail.
- On a query $\mathsf{MM.Encode}(z, x, \mathcal{S})$, with $z \in \{0,1\}^\kappa$ and $x \in \mathbb{Z}_p$, it checks that $z = \mathsf{mm.sp}$ and $\mathcal{S} \subseteq \mathscr{U}$ and outputs $\perp$ if the check fails, otherwise it generates a fresh handle $h \overset{\$}{\leftarrow} \{0,1\}^\kappa$, adds $h \mapsto (x, \mathcal{S})$ to T, and returns h.
- On a query $\mathsf{MM.Add}(z, h_1, h_2)$, with $z, h_1, h_2 \in \{0,1\}^\kappa$, it checks that $z = \mathsf{mm.pp}$, that h_1 and h_2 are handles in T which are mapped to values $(x_1, \mathcal{S}_1)$ and $(x_2, \mathcal{S}_2)$ such that $\mathcal{S}_1 = \mathcal{S}_2 = \mathcal{S} \subseteq \mathscr{U}$, and returns $\perp$ if the check fails. If it passes, it generates a fresh handle $h \overset{\$}{\leftarrow} \{0,1\}^\kappa$, adds $h \mapsto (x_1 + x_2, \mathcal{S})$ to T, and returns h.
- On a query $\mathsf{MM.Mult}(z, h_1, h_2)$, with $z, h_1, h_2 \in \{0,1\}^\kappa$, it checks $z = \mathsf{mm.pp}$, that h_1 and h_2 are handles in T which are mapped to values $(x_1, \mathcal{S}_1)$ and $(x_2, \mathcal{S}_2)$ such that $\mathcal{S}_1 \cup \mathcal{S}_2 \subseteq \mathscr{U}$ and $\mathcal{S}_1 \cap \mathcal{S}_2 = \emptyset$, and returns $\perp$ if the check fails. If it passes, it generates a fresh handle $h \overset{\$}{\leftarrow} \{0,1\}^\kappa$, adds $h \mapsto (x_1 x_2, \mathcal{S}_1 \cup \mathcal{S}_2)$ to T, and returns h.
- On a query $\mathsf{MM.ZeroTest}(z, h)$, with $z, h \in \{0,1\}^\kappa$, it checks $z = \mathsf{mm.pp}$, that h is a handle in T such that it is mapped to a value $(x, \mathscr{U})$, and returns $\perp$ if the check fails. If it passes, it returns "zero" if $x = 0$ and "non-zero" otherwise.

Remark 6 In order to ease the reading, we actually use a slightly different and more intuitive characterization of the generic multilinear map oracle in our proofs. Informally, instead of considering encodings as nonces,we consider these as formal polynomials (that can be computed easily), whose formal variables are substituted with their join value distribution from the real game. In our construction, formal variables are $\hat{a}_{i,b}, \hat{c}_{j,b}, \hat{z}_{i_1,i_2,b_1,b_2}$ —please refer to the construction in Section 3 for details. This variant characterization follows the formalization from [Zim15, Appendix B], please refer to this section for more formal definitions. =

2.5 Actual Instantiations

While Definition 3 is a very natural definition and is what we actually would like as a multilinear map, up to now, we still do not know any such construction. Known constructions [GGH13a, CLT13, GGH15, CLT15] of multilinear maps are actually "noisy" variants of our formal definition. That is, each encoding includes a random error term, and similarly to what happens for lattice-based constructions, this error term grows when performing operations (addition or multiplication). Eventually, this error term becomes too big and the $\mathsf{MM.ZeroTest}$ can no longer recover the correct answer. This noise implicitly restricts the number of operations that can be performed. Intuitively, in current constructions, the errors are added when performing an addition and multiplied when performing a multiplication. However, the fact that current instantiations are noisy does not pose any problem regarding our construction, as the number of operations for evaluating our pseudorandom function is fixed and independent from the instantiation of the multilinear map.

2.6 Straddling Sets

Our construction and its proofs use strong straddling sets [BGK+14, MSW14, Zim15, BMSZ16], in order to prevent the adversary from mixing too many encodings and creating encodings of zero. We recall their definition below. We first recall that, for a set $\mathscr{S}$, we say that $\{S_1, \ldots, S_k\}$, for some integer k, is a partition of $\mathscr{S}$, if and only if $\cup_{i=1}^{k} S_i = \mathscr{S}$, $S_i \neq \emptyset$ and $S_i \cap S_j = \emptyset$, for any $1 \leq i, j \leq k$, $i \neq j$.

Definition 7 [(Strong) Straddling Set System] For $k \in \mathbb{N}$, a *k-straddling set system* over a set $\mathscr{S}$ consists of two partitions $S_0 = \{S_{0,1}, \ldots, S_{0,k}\}$ and $S_1 = \{S_{1,1}, \ldots, S_{1,k}\}$ of $\mathscr{S}$ such that the following holds: for any $\mathscr{T} \subseteq \mathscr{S}$, if T_0, T_1 are distinct subsequences of $S_{0,1}, \ldots, S_{0,k}, S_{1,1}, \ldots, S_{1,k}$ such that T_0 and T_1 are partitions of $\mathscr{T}$, then $\mathscr{T} = \mathscr{S}$ and $T_0 = S_b$ and $T_1 = S_{1-b}$ for some $b \in \{0,1\}$.

Moreover, we say that S_0, S_1 is a *strong* k-straddling set system if for any $1 \leq i, j \leq k$, $S_{0,i} \cap S_{1,j} \neq \emptyset$.

A strong k-straddling set system is clearly also a k-straddling set system. Intuitively, a k-straddling set system ensures that the only two solutions to build a partition of $\mathscr{S}$ from combining sets in S_0 or S_1 are to use either every element in S_0 or every element in S_1.

We are only using strong straddling set systems in this paper, for the sake of simplicity. However, we only rely on the straddling set property in all our proofs, except the proof of one of our non-interactive assumptions, namely the Sel-Prod assumption. Let us now recall the construction of strong straddling set systems from [MSW14]. For the sake of completeness, we also recall the construction of straddling set systems from [BGK+14] in Appendix A.

Construction 8 [Constructions of Strong Straddling Set Systems [MSW14]]. Let k be a fixed integer and let $\mathscr{S} = \{1, \ldots, k^2\}$. Then the following partitions $S_b = (S_{b,1}, \ldots, S_{b,k})$, for $b \in \{0,1\}$, form a strong k-straddling set system over $\mathscr{S}$:

$$S_{0,i} = \{k(i-1)+1, k(i-1)+2, \ldots, ki\}$$
$$S_{1,i} = \{i, k+i, 2k+i, \ldots, k(k-1)+i\}\ .$$

This construction naturally extends to any set with k^2 elements.

Strong straddling set systems also satisfy the following lemma. Please refer to [BMSZ16] for proofs of constructions and lemma.

Lemma 9 *Let S_0, S_1 be a strong k-straddling set system over a set $\mathcal{U}$. Then for any $\mathcal{T} \subsetneq \mathcal{U}$ that can be written as a disjoint union of sets from S_0, S_1, there is a unique $b \in \{0,1\}$ such that $\mathcal{T} = \cup_{i \in I} S_{b,i}$ for some $I \subseteq \{1, \ldots, k\}$.*

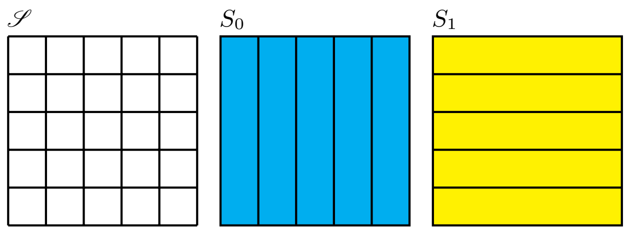

Fig. 1. Construction of strong 5-straddling set systems

3 Our Construction

Let us now describe our construction of an XOR-RKA-secure pseudorandom function, for security parameter κ, key set $\mathcal{K} = \{0,1\}^k$, with $k = 2\kappa$, and domain $\mathcal{D} = \{0,1\}^n$ for some integer n.

3.1 Intuition

Construction Overview. The starting point of our construction is the Naor-Reingold pseudorandom function, defined as $\mathsf{NR} : (\vec{a}, x) \in \mathbb{Z}_p^{2n} \times \{0,1\}^n \mapsto g^{\prod_{i=1}^{n} a_{i,x_i}}$, where $\vec{a}$ is the secret key. As we are interested in XOR relations, we want the key to be a bitstring. A simple solution is to tweak this construction by considering the function $f_{\vec{a},\vec{c}} : (K, x) \in \{0,1\}^k \times \{0,1\}^n \mapsto g^{\prod_{i=1}^{k} a_{i,K_i} \cdot \prod_{j=1}^{n} c_{i,x_i}}$, with $\vec{a} \in \mathbb{Z}_p^k$ and $\vec{c} \in \mathbb{Z}_p^n$. It is easy to see that without knowing $\vec{a}$ nor $\vec{c}$, the outputs of this function are computationally indistinguishable from random (they correspond to NR evaluations with key $(\vec{a}, \vec{c})$ and on input (K, x)). However, given a key $K \in \{0,1\}^k$, one needs the values $\vec{a}, \vec{c}$ in order to be able to evaluate this function, so these values need to be made public. Then, it becomes very easy, even without knowing K, to distinguish this function from a random one.

That is why we use a multilinear map: this allows us to publicly reveal low-level encodings of elements in $\vec{a}$ and $\vec{c}$. These encodings let anyone evaluate the function on any key K and any input x, while keeping the outputs of the function computationally indistinguishable from random to an adversary that does not know the secret key K. Formally, we let $\mathscr{U} = \{1, \dots, k+n\}$ be the set of indices for a multilinear map, $(a_{i,b})_{i \in \{1,\dots,k\}, b \in \{0,1\}}$ and $(c_{j,b})_{j \in \{1,\dots,n\}, b \in \{0,1\}}$ be random scalars in $\mathbb{Z}_p$ and $\hat{a}_{i,b} = [a_{i,b}]_{\{i\}}$ be an encoding of $a_{i,b}$ at index index i and $\hat{c}_{j,b} = [c_{j,b}]_{\{j+k\}}$ be an encoding of $c_{j,b}$ at index level $j + k$. We then consider the function:

$$F_{\mathsf{pp}}(K, x) = \left[\prod_{i=1}^{k} a_{i,K_i} \prod_{j=1}^{n} c_{j,x_j}\right]_{\mathscr{U}} = \prod_{i=1}^{k} \hat{a}_{i,K_i} \cdot \prod_{j=1}^{n} \hat{c}_{j,x_j} \ ,$$

with public parameters pp including the public parameters of the multilinear map as well as $\{\hat{a}_{i,b}\}_{i \in \{1,\dots,k\}, b \in \{0,1\}}$ and $(\hat{c}_{j,b})_{j \in \{1,\dots,n\}, b \in \{0,1\}}$.

This construction can be easily proven to be an XOR-RKA secure pseudorandom function in the generic multilinear map model, and it is also easy to show that it does not let an adversary create encodings of zero. However, it seems very hard to prove that this construction is secure under a non-interactive assumption. Hence, we modify this construction by using a more complex set of indices and straddling sets. While this makes the proof in the generic multilinear map model a bit harder, this allows us to prove the security of our construction under non-interactive assumptions, whose hardness seems plausible even with current instantiations of multilinear maps. In particular, we prove in Appendices C and D that these assumptions are secure in the generic multilinear map model and do not let an adversary generate (non-trivial) encodings of zero.

Proof Overview. In the proof, we need to show that an oracle $(s, x) \mapsto F_{\mathsf{pp}}(K \oplus s, x)$ (where K is chosen secretly uniformly at random) looks indistinguishable from a random oracle. We first remark that we can write $F_{\mathsf{pp}}(K \oplus s, x)$ as:

$$F_{\mathsf{pp}}(K \oplus s, x) = \prod_{i=1}^{k} \hat{\gamma}_{i,s_i} \cdot \prod_{j=1}^{n} \hat{c}_{j,x_j} \quad ,$$

where $\hat{\gamma}_{i,b} = [a_{i,K_i \oplus b}]_{\{i\}}$ is an encoding of $a_{i,K_i \oplus b}$ for $i \in \{1, \dots, k\}$ Thus, instead of using $K \in \{0,1\}^k$ as the key, we can use an alternative private representation: $(\hat{\gamma}_{i,b})_{i \in \{1,\dots,k\}}$.

The main idea of our reduction is to be able to replace this private representation of the key, by completely random group elements, independent of the public parameters. We remark that the function

$$((\hat{\gamma}_{i,b})_{i \in \{1,\dots,k\}}, s) \mapsto \prod_{i=1}^{k} \hat{\gamma}_{i,s_i} \quad ,$$

is a slight variant of the Naor-Reingold pseudorandom function with $(\hat{\gamma}_{i,b})_{i \in \{1,\dots,k\}}$ being the key and $s \in \{0,1\}^k$ the input. It is actually possible to prove it is a pseudorandom function under a DDH-like assumption, and from that to prove that our construction is XOR-RKA-secure.

Unfortunately, it is obviously not true that given the public parameters $(\hat{a}_{i,b})_{i \in \{1,\dots,k\}, b \in \{0,1\}}$ (which are encodings of $a_{i,b}$), the real group elements $(\hat{\gamma}_{i,b})_{i \in \{1,\dots,k\}}$ (which are encodings $a_{i,K_i \oplus b}$) are indistinguishable from encodings of independent uniform values: it is straightforward to check that $\hat{a}_{i,b}$ corresponds to the same scalar as $\gamma_{\hat{i},b}$ or $\gamma_{i,\hat{1}-b}$ (depending whether $K_i = 0$ or 1), using $\mathsf{MM.ZeroTest}$ (and after multiplying these group elements by the same group elements to get a top-level encodings). Our solution is to use more complex index sets based on strong straddling sets to solve this issue.

The first step consists in using a first strong k-straddling set $\mathscr{S}_0$: we use the indices of the first partition for $\hat{a}_{i,0}$ and $\hat{a}_{i,1}$ (for each i, $\hat{a}_{i,0}$ and $\hat{a}_{i,1}$ get the same index), and the indices of the second partition for $\hat{\gamma}_{i,0}$ and $\hat{\gamma}_{i,1}$. This prevents the adversary from comparing one group element $\hat{a}_{i,b}$ with a group element $\hat{\gamma}_{i,b'}$ directly. But this is not sufficient, as the adversary still could check whether:

$$\prod_{i=1}^{k}(\hat{a}_{i,0} + \hat{a}_{i,1}) = \prod_{i=1}^{k}(\hat{\gamma}_{i,0} + \hat{\gamma}_{i,1}) \enspace ,$$

for example. When $(\hat{\gamma}_{i,b})_{i \in \{1,\dots,k\}}$ are correctly generated, the equality is satisfied, while otherwise, it is not with overwhelming probability. More generally, the adversary can generate expression which contains an exponential number of monomials when expanded. We do not know how to prove anything reasonable on these expressions, so instead, we are using k additional strong k-straddling sets to prevent this from happening, in a similar way as they are used in [Zim15].

3.2 Actual Construction

Index set. First, similarly to [Zim15], for each $i \in \{0,\dots,k\}$, we construct a strong k-straddling set system over a set $\mathscr{S}_i$ of $2k-1$ fresh formal symbols. We denote by $S_{i,b}$ the two partitions forming each of this straddling set, for $b \in \{0,1\}$, and by $S_{i,b,j}$ their elements, for $1 \le j \le k$. We also define:

$$\mathsf{BitCommit}_{i,b} = S_{i,b,i} \qquad \mathsf{BitFill}_{i_1,i_2,b_1,b_2} = S_{i_1,b_1,i_2} \cdot S_{i_2,b_2,i_1}$$

for any $i, i_1, i_2 \in \{1,\dots,k\}$ and $b, b_1, b_2 \in \{0,1\}$. Intuitively, the straddling set systems $\mathscr{S}_i$ for $i \ge 1$ play the same role as in [Zim15] (preventing the adversary from mixing an exponential number of inputs), while $\mathscr{S}_0$ is used in the proof to prevent the adversary from mixing the private representation of the key with the public parameters.

Let X_j be fresh formal symbols for $j \in \{1,\dots,n\}$. We then define the top-level index set as follows:

$$\mathscr{U} = \prod_{i=0}^{k} \mathscr{S}_i \prod_{j=1}^{n} X_j \enspace .$$

Setup. The algorithm Setup first generates the parameters $(\mathsf{mm.pp}, \mathsf{mm.sp}, p)$ for the multilinear map by running $\mathsf{MM.Setup}(1^\kappa, \mathscr{U})$. Then it generates the following elements:

$$\begin{aligned}
a_{i,b} &\overset{\$}{\leftarrow} \mathbb{Z}_p && \text{for } i \in \{1,\dots,k\} \text{ and } b \in \{0,1\}\\
\hat{a}_{i,b} &\leftarrow [a_{i,b}]_{S_{0,0,i}\mathsf{BitCommit}_{i,b}} && \text{for } i \in \{1,\dots,k\} \text{ and } b \in \{0,1\}\\
c_{j,b} &\overset{\$}{\leftarrow} \mathbb{Z}_p && \text{for } j \in \{1,\dots,n\} \text{ and } b \in \{0,1\}\\
\hat{c}_{j,b} &\leftarrow [c_{j,b}]_{X_j} && \text{for } j \in \{1,\dots,n\} \text{ and } b \in \{0,1\}\\
\hat{z}_{i_1,i_2,b_1,b_2} &\leftarrow [1]_{\mathsf{BitFill}_{i_1,i_2,b_1,b_2}} && \text{for } i_1, i_2 \in \{1,\dots,k\} \text{ and } b_1, b_2 \in \{0,1\}\\
& && \text{with } i_1 < i_2 \,,
\end{aligned}$$

and outputs the following parameters:

$$\mathsf{pp} = \left(\mathsf{mm.pp}, (\hat{a}_{i,b})_{i,b}, (\hat{c}_{j,b})_{j,b}, (\hat{z}_{i_1,i_2,b_1,b_2})_{i_1,i_2,b_1,b_2}\right) .$$

Intuitively, $\mathsf{BitCommit}_{i,b} = S_{i,b,i}$ is associated to the (public) encoding used to evaluate the function if the i-th bit of the key is b. By definition of a straddling set, the only way to reach the top-level $\mathscr{U}$, which contains $\mathscr{S}_i$, once we have used an encoding with index $S_{i,b,i}$ is to use every index $S_{i,b,j}$ with $j \neq i$. These is done by multiplying the terms $\hat{z}_{i,j,K_i,K_j}$. Therefore, using $K_i = b$ is like "committing" to the partition $S_{i,b}$ of $\mathscr{S}_i$, and terms $\hat{z}_{i,j,K_i,K_j}$ are then used to "fill" this partition.

Remark 10 For the sake of simplicity, we set $\hat{z}_{i_1,i_2,b_1,b_2}$ to be encodings of 1, but one could also simply set it to encodings of a random value, as soon as they are all encodings of the same value.

Evaluation. The output of the PRF on a key $K \in \{0,1\}^k$ and an input $x \in \{0,1\}^n$ is

$$F_{\mathsf{pp}}(K,x) = \mathsf{MM.Extract}\left(\prod_{i=1}^{k} \hat{a}_{i,K_i} \prod_{j=1}^{n} \hat{c}_{j,x_j} \prod_{i_1=1}^{k} \prod_{i_2=i_1+1}^{k} \hat{z}_{i_1,i_2,K_{i_1},K_{i_2}}\right) .$$

We can re-write it as:

$$F_{\mathsf{pp}}(K,x) = \mathsf{MM.Extract}\left(\left[\prod_{i=1}^{k} a_{i,K_i} \prod_{j=1}^{n} c_{j,x_i}\right]_{\mathscr{U}}\right) .$$

Extraction. As explained in Remark 4, the role of extraction ($\mathsf{MM.Extract}$) is dual. First, it ensures the correctness of the PRF, as in currently known instantiations of multilinear maps [GGH13a, CLT13, GGH15, CLT15], a scalar has many different encodings. Second, it is used in our proof of security under non-interactive assumptions in Section 5, as in the security proof we change the way the group element

$$\prod_{i=1}^{k} \hat{a}_{i,K_i} \prod_{j=1}^{n} \hat{c}_{j,x_j} \prod_{i_1=1}^{k} \prod_{i_2=i_1+1}^{k} \hat{z}_{i_1,i_2,K_{i_1},K_{i_2}}$$

is computed. We indeed recall that due to the fact that a scalar has many different encodings, any group element (as the above one) leaks information on the exact computation used to obtain it, instead of just depending on its discrete logarithm. The usual way to solve this issue is to randomize the resulting group element using encodings of zero. However, in this paper, we do not want to use any encoding of zeros, hence the requirement for this extraction. For the proof in the generic multilinear map model in Section 4, this is not an issue, and we just ignore the extraction (see Remark 4).

4 Security in the Generic Multilinear Map Model

In this section, we prove the security of our construction in the generic multilinear map model. As already explained at the end of Section 3, we suppose in this section that no extraction is performed. We actually even prove that no polynomial-time adversary can construct a (non-trivial) encoding of zero, in any of the two experiments RKPRFReal and RKPRFRand, with non-negligible probability. This implies, in particular, that these two experiments cannot be distinguished by a polynomial-time adversary, in the generic multilinear map model, as an adversary only sees handles which only leak information when two of them correspond to the same (top-level) element.

This section is mainly a warm-up to familiarize with the model, since the fact that we prove our construction under some assumptions that are proven secure in the generic multilinear map model and proven to not let an adversary generate encodings of zero also implies the results below. However, it is very simple to modify the proof below in order to prove the security of the simplified construction proposed in Section 3.1, which is of independent interest.

We first need to formally define the notion of (non-trivial) encoding of zero. We follow the definition of Badrinarayanan et al. [BMSZ16].

Definition 11 [(Non-trivial) encoding of zero] An adversary $\mathscr{A}$ in the generic multilinear map model with multilinear map oracle $\mathscr{M}$ returns a *(non-trivial) encoding of zero* if it returns a handle h (output by $\mathscr{M}$) such that h corresponds to the element 0 in $\mathscr{M}$'s table and the polynomial corresponding to the handle is not identically null.

Theorem 12 (Impossibility of constructing encodings of zero) *In the generic multilinear map model with oracle $\mathscr{M}$, for any adversary $\mathscr{A}$ making at most $q_{\mathscr{M}}$ queries to the oracle $\mathscr{M}$ and $q_{\boldsymbol{RKFn}}$ queries to the oracle $\boldsymbol{RKFn}$, we have:*

$$\Pr\left[\mathrm{PRFReal}^{\mathscr{A}}_{F_{\mathsf{pp}}} \Rightarrow \textit{an encoding of 0}\right] \le q_{\mathscr{M}}\left(\frac{q_{\boldsymbol{RKFn}}}{2^k} + \frac{k+n}{p}\right)$$

and

$$\Pr\left[\mathrm{PRFRand}^{\mathscr{A}}_{F_{\mathsf{pp}}} \Rightarrow \textit{an encoding of 0}\right] \le q_{\mathscr{M}}\frac{k+n}{p} \enspace .$$

Proof (Theorem 12). We first introduce a technical lemma, whose proof is given in Appendix B.

Lemma 13 *Let k and n be two positive integers. Let $\mathscr{U}$ be the index defined in Section 3. Let $\hat{z}_{i_1,i_2,b_1,b_2} = [1]_{\mathsf{BitFill}_{i_1,i_2,b_1,b_2}}$ for $1 \le i_1 < i_2 \le k$ and $b_1, b_2 \in \{0,1\}$. Let Z_1 and Z_2 be two subsets of $\{(i_1,i_2) \mid 1 \le i_1 < i_2 \le k\} \times \{0,1\}^2$. If $t_1 = \prod_{(i_1,i_2,b_1,b_2)\in Z_1} \hat{z}_{i_1,i_2,b_1,b_2}$ and $t_2 = \prod_{(i_1,i_2,b_1,b_2)\in Z_2} \hat{z}_{i_1,i_2,b_1,b_2}$ have the same index set, then $Z_1 = Z_2$.*

We need to show that the adversary cannot generate a non-trivial encoding of zero.

$\mathrm{RKPRFRand}_{F_{\mathsf{pp}}}^{\mathscr{A}}$. We start by proving it in the game $\mathrm{RKPRFRand}_{F_{\mathsf{pp}}}^{\mathscr{A}}$. In this game, except for $\hat{z}_{i_1,i_2,b_1,b_2}$, all the handles the adversary sees correspond to fresh new formal variables, as the oracle **RKFn** only returns fresh new formal variables (of index $\mathscr{U}$). The only polynomials the adversary can generate are therefore of the form:

$$P = \sum_{\ell=1}^{L} Q_\ell \prod_{(i_1,i_2,b_1,b_2)\in Z_\ell} \hat{z}_{i_1,i_2,b_1,b_2} \quad ,$$

where Q_ℓ are polynomials over all the elements except $\hat{z}_{i_1,i_2,b_1,b_2}$, and Z_ℓ are distinct subsets of $\{(i_1,i_2) \mid 1 \le i_1 < i_2 \le k\} \times \{0,1\}^2$ (L might be exponential in $q_{\mathscr{M}}$, but that does not matter for what follows).

Let us now show that if P is not the zero polynomial, then when replacing $\hat{z}_{i_1,i_2,b_1,b_2}$ by 1, the resulting polynomial is still a non-zero polynomial. From Lemma 13, one can assume that elements $\prod_{(i_1,i_2,b_1,b_2)\in Z_\ell} \hat{z}_{i_1,i_2,b_1,b_2}$ all have distinct indices. Therefore, the polynomials Q_ℓ all have distinct indices too. No monomial in two different Q_ℓ of the sum $\sum_\ell Q_\ell$ (when forgetting the indices) can therefore cancel out, otherwise this would mean that the adversary can construct two equal monomials (without $\hat{z}_{i,b}$) with two different indices. This is impossible as, except for $\hat{z}_{i,b}$, two distinct handles correspond to two fresh variables (or in other words, all the handles except $\hat{z}_{i,b}$ are encodings of scalars chosen uniformly and independently at random).

We therefore simulate the oracle $\mathscr{M}$ as follows: we do everything as normal, but we make the zero-testing oracle always output "non-zero" except when its input corresponds to the zero polynomial. The Schwarz-Zippel lemma ensures that any non-zero polynomial of degree at most $k+n$ and whose variables are fixed to uniformly random values in $\mathbb{Z}_p$ does not evaluate to zero, except with probability at most $(k+n)/p$. In other words, the zero-testing oracle outputs "zero" on a non-zero polynomial with probability at most $(k+n)/p$, as this polynomial remains non-zero and has degree at most $(k+n)$, when we replace $\hat{z}_{i_1,i_2,b_1,b_2}$ by 1. As we can suppose that the zero-testing oracle is queried with the output of the adversary without loss of generality, using at most $q_{\mathscr{M}}$ hybrid games (replacing one-by-one every output of the zero-testing oracle with "non-zero") we get that:

$$\Pr\left[\mathrm{PRFRand}_{F_{\mathsf{pp}}}^{\mathscr{A}} \Rightarrow \text{an encoding of } 0\right] \le q_{\mathscr{M}} \frac{k+n}{p} \quad .$$

$\mathrm{RKPRFReal}_{F_{\mathsf{pp}}}^{\mathscr{A}}$. Let us now look at the game $\mathrm{RKPRFReal}_{F_{\mathsf{pp}}}^{\mathscr{A}}$. The analysis is more complicated as the adversary has access to new formal variables

$$\hat{y}_{s,x} = F_{\mathsf{pp}}(K \oplus s, x)$$

returned by queries $\mathbf{RKFn}(\phi_s, x)$.

We use the same simulator as in the previous case. We need to show that if a polynomial P produced by the adversary is not zero, it remains non-zero when $\hat{z}_{i_1,i_2,b_1,b_2}$ is replaced by 1 and $\hat{y}_{s,x}$ is replaced by its value.

We first consider the case where P is not a top-level polynomial. In this case, P cannot contain these new variables $\hat{y}_{s,x}$ as these variables are top-level. Then, as in $\mathrm{RKPRFRand}^{\mathscr{A}}_{F_{\mathsf{pp}}}$, the zero-testing oracle outputs "non-zero" except with probability at most $(k+n)/p$.

Let us now suppose that P is a top-level polynomial. This polynomial has the form:

$$P = \sum_{\ell=1}^{L} Q_\ell \prod_{i=1}^{k} \hat{a}_{i,K'_{\ell,i}} \prod_{i_1=1}^{k} \prod_{i_2=i_1+1}^{k} \hat{z}_{i_1,i_2,K'_{\ell,i_1},K'_{\ell,i_2}} + \sum_{j=1}^{q'} \lambda_j \hat{y}_{s_j,x_j} ,$$

with L being a non-negative integer (possibly exponential in $q_{\mathscr{M}}$), Q_ℓ being non-zero polynomials in the formal variables $\hat{c}_{j,b}$, K'_ℓ being distinct bitstrings in $\{0,1\}^k$ (chosen by the adversary), q' an integer less or equal to $q_{\mathbf{RKFn}}$, $(s_1,x_1),\dots,(s_{q'},x_{q'})$ queries to **RKFn**, and λ_j some scalar in $\mathbb{Z}_p$. Indeed, the adversary can ask for an encoding of any polynomial of the form $Q_\ell \prod_{i=1}^{k} \hat{a}_{i,K'_{\ell,i}}$, and by definition of straddling set systems, the unique way to obtain a top-level encoding from such a polynomial is by multiplying it with an encoding of $\prod_{i_1=1}^{k} \prod_{i_2=i_1+1}^{k} \hat{z}_{i_1,i_2,K'_{\ell,i_1},K'_{\ell,i_2}}$.

Let us suppose that P is not zero but becomes zero when $\hat{z}_{i_1,i_2,b_1,b_2}$ is replaced by 1 and $\hat{y}_{s,x}$ is replaced by its value. In this case, in particular, the first monomial (for any order) of the term

$$Q_1 \prod_{i=1}^{k} \hat{a}_{i,K'_{1,i}} \prod_{i_1=1}^{k} \prod_{i_2=i_1+1}^{k} \hat{z}_{i_1,i_2,K'_{1,i_1},K'_{1,i_2}}$$

necessarily needs to be canceled out by some $\hat{y}_{s_j,x_j}$. The probability over $K \overset{\$}{\leftarrow} \{0,1\}^k$ that this happens is at most:

$$\Pr\left[\, \exists j' \in \{1,\dots,q'\},\ K'_1 = K \oplus s_{j'} \,\right] \le \frac{q'}{2^k} \le \frac{q_{\mathbf{RKFn}}}{2^k} \quad .$$

As before, thanks to the Schwarz-Zippel lemma, we get that the zero-testing oracle outputs "zero", on input a non-zero polynomial, with probability at most:

$$\frac{q_{\mathbf{RKFn}}}{2^k} + \frac{k+n}{p} \quad .$$

This concludes the proof of Theorem 12. ☐

Remark 14 We never use the properties of the straddling set system $\mathscr{S}_0$ in this proof. These properties are only used in our proof under non-interactive assumptions in Section 5.

We obtain the following immediate corollary.

Corollary 15 (Security in the generic multilinear map model) *Let $\mathscr{A}$ be an adversary in the generic multilinear map model with oracle $\mathscr{M}$ against the*

XOR-RKA security of the PRF F defined in Section 3. If $\mathscr{A}$ makes at most $q_{\mathscr{M}}$ queries to the oracle $\mathscr{M}$ and $q_{\boldsymbol{RKFn}}$ queries to the oracle ***RKFn****, then:*

$$\mathbf{Adv}^{\mathsf{prf\text{-}rka}}_{\Phi_\oplus, F_{\mathsf{pp}}}(\mathscr{A}) \leq \frac{q_{\mathscr{M}} q_{\boldsymbol{RKFn}}}{2^k} + \frac{2q_{\mathscr{M}}(k+n)}{p} \enspace .$$

Proof (Corollary 15). We just consider an intermediate game where we simulate everything as before except the zero-testing oracle which always outputs "non-zero" unless its input is zero, as a polynomial. This game is indistinguishable from both $\mathrm{RKPRFReal}^{\mathscr{A}}_{F_{\mathsf{pp}}}$ and $\mathrm{RKPRFRand}^{\mathscr{A}}_{F_{\mathsf{pp}}}$ according to Theorem 12 (up to the bounds in this lemma). Corollary 15 follows. □

5 Security Under Non-Interactive Assumptions

In this section, we show that our construction is an XOR-RKA PRF under two non-interactive assumptions defined below.

5.1 Assumptions

We use two assumptions that we call the (k, n, X, Y)-$\mathsf{XY\text{-}DDH}$ assumption, which is roughly a generalization of the standard DDH assumption, and the (k, n)-$\mathsf{Sel\text{-}Prod}$ assumption. We show in Appendices C and D that both these assumptions are secure in the generic multilinear map model, and even that an adversary against these assumptions cannot generate encodings of zero. As explained in the "concrete instantiations" paragraph of Section 1, contrary to most assumptions considered on multilinear maps (e.g., classical DDH-like assumptions and the multilinear subgroup assumption [GLW14, GLSW15]), these assumptions are therefore plausible at least with two current instantiations of multilinear maps [CLT13, GGH15].

To ensure the impossibility of generating encodings of zero, in these two assumptions, we restrict the adversary's capabilities as follows: it is only provided parameters $\mathsf{mm.pp}$, so it can only run $\mathsf{MM.Add}$, $\mathsf{MM.Mult}$ and $\mathsf{MM.ZeroTest}$ (and of course use the elements generated by the assumption), but we do not allow the adversary to generate new encodings of a chosen scalar. In particular, this forces us to let the assumption contain the group elements $\hat{z}_{i_1,i_2,b_1,b_2}$. It is straightforward to get rid of these additional elements by allowing the adversary to generate any element of the multilinear map, at the cost of getting an implausible assumption under current instantiations of multilinear maps.

Finally, our assumption implicitly contains a list $\mathcal{L}$ of a polynomial number of encodings of independent uniform random values at non-zero index, index being implicit parameters of the assumption. We could avoid this artifact with the previous proposition as well, or by giving a sufficient number of encodings of 0 and 1, but once again, in that case, the assumption would most likely not hold with currently known multilinear maps instantiations. We believe this is a small price to pay to get plausible assumptions, as the resulting assumptions are still non-interactive.

We insist on the fact that the encodings in $\mathcal{L}$ are encodings of independent uniformly random scalars. At least in the generic multilinear group model, our assumptions hold whatever the list of indices of these encodings is. We do not have any constraint on this list of indices.

Definition 16 [(k,n,X,Y)-XY-DDH] Let k and n be two positive integers. Let X and Y be two non-empty and disjoint indices in the index set $\mathscr{U}$ of our construction in Section 3. The advantage of an adversary $\mathscr{D}$ against the (k,n,X,Y)-XY-DDH problem is:

$$\mathbf{Adv}^{(k,n,X,Y)\text{-}\mathsf{XY\text{-}DDH}}(\mathscr{D}) = \Pr\left[\,(k,n,X,Y)\text{-}\mathsf{XY\text{-}DDH\text{-}L}^{\mathscr{D}} \Rightarrow 1\,\right] - \Pr\left[\,(k,n,X,Y)\text{-}\mathsf{XY\text{-}DDH\text{-}R}^{\mathscr{D}} \Rightarrow 1\,\right],$$

where the games (k,n,X,Y)-XY-DDH-L$^{\mathscr{D}}$ and (k,n,X,Y)-XY-DDH-R$^{\mathscr{D}}$ are defined in Fig. 2. The (n,k,X,Y)-XY-DDH assumption holds when this advantage is negligible for any polynomial-time adversary $\mathscr{D}$.

This assumption is very close to the classical DDH assumption with indices, with two main differences: the presence of elements $\hat{z}_{i_1,i_2,b_1,b_2}$ which are necessary to prove our construction and the implicit presence of encodings of random values at non-zero indices (list $\mathcal{L}$ described previously) instead of a polynomial number of encodings of 0 and 1. Without the elements $\hat{z}_{i_1,i_2,b_1,b_2}$, the proof of this assumption in the generic multilinear map model would be completely straightforward. The difficulty of the proof is to deal with these elements.

In the security proof of our construction, this assumption is used in a similar way as the DDH assumption in the proof of the Naor-Reingold PRF.

Definition 17 [(k,n)-Sel-Prod] Let k and n be two positive integers. The advantage of an adversary $\mathscr{D}$ against the (k,n,X,Y)-Sel-Prod problem is:

$$\mathbf{Adv}^{(k,n)\text{-}\mathsf{Sel\text{-}Prod}}(\mathscr{D}) = \Pr\left[\,(k,n)\text{-}\mathsf{Sel\text{-}Prod\text{-}L}^{\mathscr{D}} \Rightarrow 1\,\right] - \Pr\left[\,(k,n)\text{-}\mathsf{Sel\text{-}Prod\text{-}R}^{\mathscr{D}} \Rightarrow 1\,\right],$$

where the games (k,n)-Sel-Prod-L$^{\mathscr{D}}$ and (k,n)-Sel-Prod-R$^{\mathscr{D}}$ are defined in Fig. 2. The (n,k)-Sel-Prod assumption holds when this advantage is negligible for any polynomial-time adversary $\mathscr{D}$.

Intuitively, this assumption states that, given a low-level encodings of $a_{i,0}$ and $a_{i,1}$ at indices $S_{0,i}$ from the first partition of the straddling set $\mathscr{S}$, where $i \in \{1,\ldots,k\}$, then it is hard to distinguish low-level encodings of $\gamma_{i,0} = a_{i,K_i}$ and $\gamma_{i,1} = a_{i,1-K_i}$ (where K_i is a random bit) at indices $S_{1,i}$ from encodings of fresh random values (at the same index). The hardness of this assumption crucially relies on two facts: First, one can only compare top-level encodings, and thus, the only way to compare elements whose index is in the first partition of $\mathscr{S}$ with elements whose index is in the second partition is to combine k such elements to reach the same index. Second, the latter can be hard only if k, the

```
$(k, n, X, Y)$-XY-DDH-L$^{\mathscr{D}}$
proc Initialize
$(\mathsf{mm.sp}, \mathsf{mm.pp}) \xleftarrow{\$} \mathsf{MM.Setup}(1^\kappa, \mathscr{U})$
$d_0, d_1, e \xleftarrow{\$} \mathbb{Z}_p$
$\hat{d}_0 \leftarrow [d_0]_X$;        $\hat{d}_1 \leftarrow [d_1]_X$
$\hat{e}_0 \leftarrow [ed_0]_{XY}$;       $\hat{e}_1 \leftarrow [ed_1]_{XY}$
For $i_1, i_2 \in \{1, \ldots, k\}$ and $b_1, b_2 \in \{0, 1\}$
    $\hat{z}_{i_1,i_2,b_1,b_2} \leftarrow [1]_{\mathsf{BitFill}_{i_1,i_2,b_1,b_2}}$
Return $(\mathcal{L}, \mathsf{mm.pp}, \hat{d}_0, \hat{d}_1, \hat{e}_0, \hat{e}_1,$
                                        $(\hat{z}_{i_1,i_2,b_1,b_2}))$
proc Finalize$(b)$
Return $b$
```

```
$(k, n, X, Y)$-XY-DDH-R$^{\mathscr{D}}$
proc Initialize
$(\mathsf{mm.sp}, \mathsf{mm.pp}) \xleftarrow{\$} \mathsf{MM.Setup}(1^\kappa, \mathscr{U})$
$d_0, d_1, e_0, e_1 \xleftarrow{\$} \mathbb{Z}_p$
$\hat{d}_0 \leftarrow [d_0]_X$;        $\hat{d}_1 \leftarrow [d_1]_X$
$\hat{e}_0 \leftarrow [e_0]_{XY}$;       $\hat{e}_1 \leftarrow [e_1]_{XY}$
For $i_1, i_2 \in \{1, \ldots, k\}$ and $b_1, b_2 \in \{0, 1\}$
    $\hat{z}_{i_1,i_2,b_1,b_2} \leftarrow [1]_{\mathsf{BitFill}_{i_1,i_2,b_1,b_2}}$
Return $(\mathcal{L}, \mathsf{mm.pp}, \hat{d}_0, \hat{d}_1, \hat{e}_0, \hat{e}_1,$
                                        $(\hat{z}_{i_1,i_2,b_1,b_2}))$
proc Finalize$(b)$
Return $b$
```

```
$(k, n)$-Sel-Prod-L$^{\mathscr{D}}$
proc Initialize
$(\mathsf{mm.sp}, \mathsf{mm.pp}) \xleftarrow{\$} \mathsf{MM.Setup}(1^\kappa, \mathscr{U})$
$K \xleftarrow{\$} \{0, 1\}^k$
For $i \in \{1, \ldots, k\}$ and $b \in \{0, 1\}$
    $a_{i,b} \xleftarrow{\$} \mathbb{Z}_p$;        $\gamma_{i,K_i \oplus b} \leftarrow a_{i,b}$
    $\hat{a}_{i,b} \leftarrow [a_{i,b}]_{S_{0,0,i}\mathsf{BitCommit}_{i,b}}$
    $\hat{\gamma}_{i,K_i \oplus b} \leftarrow [\gamma_{i,K_i \oplus b}]_{S_{0,1,i}\mathsf{BitCommit}_{i,K_i \oplus b}}$
For $i_1, i_2 \in \{1, \ldots, k\}$ and $b_1, b_2 \in \{0, 1\}$
    $\hat{z}_{i_1,i_2,b_1,b_2} \leftarrow [1]_{\mathsf{BitFill}_{i_1,i_2,b_1,b_2}}$
Return $(\mathcal{L}, \mathsf{mm.pp}, (\hat{a}_{i,b}), (\hat{\gamma}_{i,b}),$
                                        $(\hat{z}_{i_1,i_2,b_1,b_2}))$
proc Finalize$(b)$
Return $b$
```

```
$(k, n)$-Sel-Prod-R$^{\mathscr{D}}$
proc Initialize
$(\mathsf{mm.sp}, \mathsf{mm.pp}) \xleftarrow{\$} \mathsf{MM.Setup}(1^\kappa, \mathscr{U})$

For $i \in \{1, \ldots, k\}$ and $b \in \{0, 1\}$
    $a_{i,b} \xleftarrow{\$} \mathbb{Z}_p$;        $\gamma_{i,b} \xleftarrow{\$} \mathbb{Z}_p$
    $\hat{a}_{i,b} \leftarrow [a_{i,b}]_{S_{0,0,i}\mathsf{BitCommit}_{i,b}}$
    $\hat{\gamma}_{i,b} \leftarrow [\gamma_{i,b}]_{S_{0,1,i}\mathsf{BitCommit}_{i,b}}$
For $i_1, i_2 \in \{1, \ldots, k\}$ and $b_1, b_2 \in \{0, 1\}$
    $\hat{z}_{i_1,i_2,b_1,b_2} \leftarrow [1]_{\mathsf{BitFill}_{i_1,i_2,b_1,b_2}}$
Return $(\mathcal{L}, \mathsf{mm.pp}, (\hat{a}_{i,b}), (\hat{\gamma}_{i,b}),$
                                        $(\hat{z}_{i_1,i_2,b_1,b_2}))$
proc Finalize$(b)$
Return $b$
```

Fig. 2. Games defining the advantage of an adversary $\mathscr{D}$ against the XY-DDH and Sel-Prod problems.

size of the partitions, is big. Indeed, assume $k = 2$, then one can just guess K_1 and K_2 and check if the relation carries on between encodings of $a_{i,b}$ and encodings of $\gamma_{i,b}$. Therefore, we show that if k is big enough, this assumption holds in the generic multilinear map model.

As explained in Section 3.1, this assumption is used to switch from the key K to a private independent key represented by the encodings $\gamma_{i,b}$. More precisely, under this assumption, we can replace the encodings $\hat{a}_{i,K_i \oplus s_i}$ at index from the first partition of the straddling set $\mathscr{S}_0$, used in the computation of the output with relation s, to encodings of uniformly random scalars at index from the second partition of $\mathscr{S}_0$. In particular, doing this change, we no longer need to know the key K to simulate correctly the output, but only the relations s for each query.

Remark 18 For the sake of simplicity, we do not explicitly specify the noise level in our assumptions. It can easily be made to work with our proof.

5.2 Security of our Construction

In this whole section, we set $\mathscr{S} = \prod_{i=0}^{k} \mathscr{S}_i$ and $\mathscr{S}' = \prod_{i=1}^{k} \mathscr{S}_i$, so $\mathscr{S} = \mathscr{S}_0 \cdot \mathscr{S}'$. Please refer to Section 3 for notation.

Theorem 19 (Security under non-interactive assumptions) *Let $\mathscr{A}$ be a polynomial-time adversary against the XOR-RKA security of the PRF F defined in Section 3. We suppose that $\mathscr{A}$ makes at most $q_{\boldsymbol{RKFn}}$ queries to the oracle* ***RKFn****. We can define an adversary $\mathscr{D}$ against the (k,n)-$\mathsf{Sel\text{-}Prod}$ problem, $(k-1)$ adversaries $\mathscr{B}_{i'}$ against the $(k, n, \mathscr{S}' \prod_{i=1}^{i'-1} S_{0,1,i}, S_{0,1,i'})$-$\mathsf{XY\text{-}DDH}$ problem for $i' \in \{2, \ldots, k\}$, and n adversaries $\mathscr{C}_{j'}$ against the $(k, n, \mathscr{S} \prod_{j=1}^{j'-1} X_j, X_{j'})$-$\mathsf{XY\text{-}DDH}$ problem for $j' \in \{1, \ldots, n\}$, such that:*

$$\begin{aligned}\mathbf{Adv}^{\mathsf{prf\text{-}rka}}_{\Phi_\oplus, F_{\mathsf{pp}}}(\mathscr{A}) \le \mathbf{Adv}^{(k,n)\text{-}\mathsf{Sel\text{-}Prod}}(\mathscr{D}) + \\ \sum_{i'=2}^{k} q_{\boldsymbol{RKFn}} \cdot \mathbf{Adv}^{(k,n,\mathscr{S}' \prod_{i=1}^{i'-1} S_{0,1,i}, S_{0,1,i'})\text{-}\mathsf{XY\text{-}DDH}}(\mathscr{B}_{i'}) + \\ \sum_{j'=1}^{n} q_{\boldsymbol{RKFn}} \cdot \mathbf{Adv}^{(k,n,\mathscr{S} \prod_{j=1}^{j'-1} X_j, X_{j'})\text{-}\mathsf{XY\text{-}DDH}}(\mathscr{C}_{j'}) \enspace .\end{aligned}$$

Furthermore, all these adversaries run in polynomial time (their running time is approximately the same as $\mathscr{A}$).

Below, we provide a sketch of the proof. The full proof is given in Appendix E.

Sketch of proof. The proof follows a sequence of hybrid games. The first hybrid corresponds exactly to $\mathrm{RKPRFReal}_F^{\mathscr{A}}$, while the last game corresponds to $\mathrm{RKPRFRand}_F^{\mathscr{A}}$. Here is how we proceed. First, instead of computing the output using encodings $\hat{a}_{i,b}$ of $a_{i,b}$ with index $S_{0,0,i}\mathsf{BitCommit}_{i,b}$, we use encodings $\hat{\gamma}_{i,b}$ of $a_{i,K_i \oplus b}$ with index $S_{0,1,i}\mathsf{BitCommit}_{i,K_i \oplus b}$. That is, we use the second partition

$S_{0,1}$ of the straddling set $\mathscr{S}_0$ instead of the first one ($S_{0,0}$) to reach top-level index (which contains $\mathscr{S}_0$). Also, we now compute the output using only the relation s instead of the key K. More precisely, the output on a query (s, x) is computed as:

$$\mathsf{MM.Extract}\left(\prod_{i=1}^{k} \hat{\gamma}_{i,s_i} \prod_{j=1}^{n} \hat{c}_{j,x_j} \prod_{i_1=1}^{k} \prod_{i_2=i_1+1}^{k} \hat{z}_{i_1,i_2,K_{i_1},K_{i_2}}\right),$$

which can be computed without knowing K. This does not change anything regarding the output (thanks to the extraction), so these two games are indistinguishable.

However, using the (k,n)-Sel-Prod assumption, we can now switch the encodings $\hat{\gamma}_{i,b}$ to encodings of fresh random scalars $\gamma_{i,b} \in \mathbb{Z}_p$. The rest of the proof is very similar to the proof of the Naor-Reingold pseudorandom function. We do $k+n$ hybrid games, where in the j-th hybrid, we just switch products of encodings $\prod_{i=1}^{j} \hat{\gamma}_{i,s_i}$ to encodings of uniformly fresh random values using the XY-DDH assumption with the proper indices. These modifications are done in a lazy fashion to obtain a polynomial-time reduction.

Acknowledgements

The first author was supported in part by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013 Grant Agreement 339563 – CryptoCloud). The second author was supported by the Defense Advanced Research Projects Agency (DARPA) and Army Research Office (ARO) under Contract No. W911NF-15-C-0236. The third author was supported in part from a DARPA/ARL SAFEWARE award, NSF Frontier Award 1413955, NSF grants 1619348, 1228984, 1136174, and 1065276, BSF grant 2012378, a Xerox Faculty Research Award, a Google Faculty Research Award, an equipment grant from Intel, and an Okawa Foundation Research Grant. This material is based upon work supported by the Defense Advanced Research Projects Agency through the ARL under Contract W911NF-15-C-0205. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, or the U.S. Government.

References

ABD16. Martin R. Albrecht, Shi Bai, and Léo Ducas. A subfield lattice attack on overstretched NTRU assumptions - cryptanalysis of some FHE and graded encoding schemes. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 153–178. Springer, Heidelberg, August 2016.

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.

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.

AW14. Benny Applebaum and Eyal Widder. Related-key secure pseudorandom functions: The case of additive attacks. Cryptology ePrint Archive, Report 2014/478, 2014. http://eprint.iacr.org/2014/478.

BC10. Mihir Bellare and David Cash. Pseudorandom functions and permutations provably secure against related-key attacks. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 666–684. Springer, Heidelberg, August 2010.

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

BDK05. Eli Biham, Orr Dunkelman, and Nathan Keller. Related-key boomerang and rectangle attacks. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 507–525. Springer, Heidelberg, May 2005.

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

BGK$^+$14. Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Protecting obfuscation against algebraic attacks. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 221–238. Springer, Heidelberg, May 2014.

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

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

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

BMSZ16. Saikrishna Badrinarayanan, Eric Miles, Amit Sahai, and Mark Zhandry. Post-zeroizing obfuscation: New mathematical tools, and the case of evasive circuits. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 764–791. Springer, Heidelberg, May 2016.

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

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.

BR14. Zvika Brakerski and Guy N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 1–25. Springer, Heidelberg, February 2014.

CFL$^+$16. Jung Hee Cheon, Pierre-Alain Fouque, Changmin Lee, Brice Minaud, and Hansol Ryu. Cryptanalysis of the new CLT multilinear map over the integers. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 509–536. Springer, Heidelberg, May 2016.

CGH$^+$15. Jean-Sébastien Coron, Craig Gentry, Shai Halevi, Tancrède Lepoint, Hemanta K. Maji, Eric Miles, Mariana Raykova, Amit Sahai, and Mehdi Tibouchi. Zeroizing without low-level zeroes: New MMAP attacks and their limitations. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 247–266. Springer, Heidelberg, August 2015.

CGH17. Yilei Chen, Craig Gentry, and Shai Halevi. Cryptanalyses of candidate branching program obfuscators. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part III*, volume 10212 of *Lecture Notes in Computer Science*, pages 278–307, 2017.

CHL$^+$15. Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehlé. Cryptanalysis of the multilinear map over the integers. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 3–12. Springer, Heidelberg, April 2015.

CJL16. Jung Hee Cheon, Jinhyuck Jeong, and Changmin Lee. An algorithm for ntru problems and cryptanalysis of the ggh multilinear map without a low-level encoding of zero. *LMS Journal of Computation and Mathematics*, 19(A):255–266, 2016.

CLLT16. Jean-Sébastien Coron, Moon Sung Lee, Tancrède Lepoint, and Mehdi Tibouchi. Cryptanalysis of GGH15 multilinear maps. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 607–628. Springer, Heidelberg, August 2016.

CLT13. Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 476–493. Springer, Heidelberg, August 2013.

CLT15. Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. New multilinear maps over the integers. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 267–286. Springer, Heidelberg, August 2015.

CS15. Benoit Cogliati and Yannick Seurin. On the provable security of the iterated Even-Mansour cipher against related-key and chosen-key attacks. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 584–613. Springer, Heidelberg, April 2015.

FP15. Pooya Farshim and Gordon Procter. The related-key security of iterated Even-Mansour ciphers. In Gregor Leander, editor, *FSE 2015*, volume 9054 of *LNCS*, pages 342–363. Springer, Heidelberg, March 2015.

FX15. Eiichiro Fujisaki and Keita Xagawa. Efficient RKA-secure KEM and IBE schemes against invertible functions. In Kristin E. Lauter and Francisco Rodríguez-Henríquez, editors, *LATINCRYPT 2015*, volume 9230 of *LNCS*, pages 3–20. Springer, Heidelberg, August 2015.

GGH13a. Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 1–17. Springer, Heidelberg, May 2013.

GGH$^+$13b. Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, October 2013.

GGH15. Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multilinear maps from lattices. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 498–527. Springer, Heidelberg, March 2015.

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

GLSW15. Craig Gentry, Allison Bishop Lewko, Amit Sahai, and Brent Waters. Indistinguishability obfuscation from the multilinear subgroup elimination assumption. In Venkatesan Guruswami, editor, *56th FOCS*, pages 151–170. IEEE Computer Society Press, October 2015.

GLW14. Craig Gentry, Allison B. Lewko, and Brent Waters. Witness encryption from instance independent assumptions. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 426–443. Springer, Heidelberg, August 2014.

GMS16. Sanjam Garg, Pratyay Mukherjee, and Akshayaram Srinivasan. Obfuscation without the vulnerabilities of multilinear maps. Cryptology ePrint Archive, Report 2016/390, 2016. http://eprint.iacr.org/2016/390.

HJ16. Yupu Hu and Huiwen Jia. Cryptanalysis of GGH map. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 537–565. Springer, Heidelberg, May 2016.

JW15. Zahra Jafargholi and Daniel Wichs. Tamper detection and continuous non-malleable codes. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part I*, volume 9014 of *LNCS*, pages 451–480. Springer, Heidelberg, March 2015.

Knu93. Lars R. Knudsen. Cryptanalysis of LOKI91. In Jennifer Seberry and Yuliang Zheng, editors, *AUSCRYPT'92*, volume 718 of *LNCS*, pages 196–208. Springer, Heidelberg, December 1993.

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

Men16. Bart Mennink. XPX: Generalized tweakable even-mansour with improved security guarantees. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 64–94. Springer, Heidelberg, August 2016.

MSW14. Eric Miles, Amit Sahai, and Mor Weiss. Protecting obfuscation against arithmetic attacks. Cryptology ePrint Archive, Report 2014/878, 2014. http://eprint.iacr.org/2014/878.

MSZ16. Eric Miles, Amit Sahai, and Mark Zhandry. Annihilation attacks for multilinear maps: Cryptanalysis of indistinguishability obfuscation over GGH13. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 629–658. Springer, Heidelberg, August 2016.

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.

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

WLZZ16. Peng Wang, Yuling Li, Liting Zhang, and Kaiyan Zheng. Related-key almost universal hash functions: Definitions, constructions and applications. In Thomas Peyrin, editor, *FSE 2016*, volume 9783 of *LNCS*, pages 514–532. Springer, Heidelberg, March 2016.

Zim15. Joe Zimmerman. How to obfuscate programs directly. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 439–467. Springer, Heidelberg, April 2015.

A Construction of Straddling Set Systems

Construction 20 [Constructions of Straddling Set Systems [BGK+14]]. Let k be a fixed integer and let $\mathscr{S} = \{1, \ldots, 2k-1\}$. Then, the following partitions form a k-straddling set system over $\mathscr{S}$:

$$S_0 = (S_{0,1}, \ldots, S_{0,k}) = (\{1\}, \{2,3\}, \{4,5\}, \ldots, \{2k-4, 2k-3\}, \{2k-2, 2k-1\})$$
$$S_1 = (S_{1,1}, \ldots, S_{1,k}) = (\{1,2\}, \{3,4\}, \ldots, \{2k-3, 2k-2\}, \{2k-1\}) \,.$$

This construction naturally extends to any set with $2k-1$ elements.

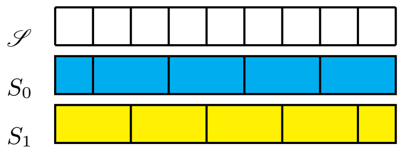

Fig. 3. Construction of 5-straddling set systems

B Proof of Lemma 13

Proof (Lemma 13). Let T denote the index set of t_1 and t_2. For $i = 1, \ldots, k$, one can intersect T with $\mathscr{S}_i$. As T cannot contain $\mathscr{S}_i$ (since it cannot contain

$S_{i,b,i}$ for any $b \in \{0,1\}$ as it is not contained in any index set of any $\hat{z}_{i_1,i_2,b_1,b_2}$), we have $T \cap \mathscr{S}_i \neq \mathscr{S}_i$. Therefore, for any $(i_1, i_2, b_1, b_2) \in Z_1$ with $i_1 < i_2$, as S_{i_1,b_1,i_2} is in the index set of t_1 and $T \cap \mathscr{S}_{i_1} \neq \mathscr{S}_{i_1}$, there exists b' such that $S_{i_1,b_1,i_2,b'}$ is in Z_2 (so that S_{i_1,b_1,i_2} is in the index set of t_2), by definition of a straddling set system. Then, there are two possibilities: either $b' = b_2$ and $(i_1, i_2, b_1, b_2) \in Z_2$, or $b' = 1 - b_2$. In the latter case, this means that S_{i_2,b_2,i_1} is contained in the index set of t_1 and $S_{i_2,1-b_2,i_1}$ is contained in the index set of t_2. As $T \cap \mathscr{S}_{i_2} \neq \mathscr{S}_{i_2}$, this contradicts the fact that t_1 and t_2 have the same index set. Lemma 13 immediately follows. □

C Security of the **XY-DDH** Assumption in the Generic Multilinear Map Model

Theorem 21 (Impossibility of constructing encodings of zero) *Let k, n be two positive integers. Let X and Y be two non-empty and disjoint indices from the index set $\mathscr{U}$ of our construction in Section 3. In the generic multilinear map model with oracle $\mathscr{M}$, for any adversary $\mathscr{D}$ making at most $q_{\mathscr{M}}$ queries to the oracle $\mathscr{M}$, we have:*

$$\Pr\left[(k,n,X,Y)\text{-}\mathsf{XY\text{-}DDH\text{-}L}^{\mathscr{D}} \Rightarrow \textit{an encoding of 0}\right] \le q_{\mathscr{M}} \frac{|\mathscr{U}|}{p}$$

and

$$\Pr\left[(k,n,X,Y)\text{-}\mathsf{XY\text{-}DDH\text{-}R}^{\mathscr{D}} \Rightarrow \textit{an encoding of 0}\right] \le q_{\mathscr{M}} \frac{|\mathscr{U}|}{p} \enspace .$$

Proof (Theorem 21). We consider the two cases separately.

(k,n,X,Y)-XY-DDH-R$^{\mathscr{D}}$. Except for $\hat{z}_{i_1,i_2,b_1,b_2}$, all the handles the adversary sees corresponds to fresh variables: $\hat{d}_0, \hat{d}_1, \hat{e}_0, \hat{e}_1$, in addition to the ones it can generate using the list $\mathcal{L}$. We can therefore conclude exactly as in the case $\mathrm{RKPRFRand}_F^{\mathscr{A}}$ of the proof of Theorem 12. The only difference is that the maximum degree of the polynomials the adversary can create is at most $|\mathscr{U}|$ (instead of $k+n$), as the adversary does not have access to elements of index $\emptyset$ (i.e., scalars).

(k,n,X,Y)-XY-DDH-L$^{\mathscr{D}}$. Similarly to the case $\mathrm{RKPRFReal}_F^{\mathscr{A}}$ of the proof of Theorem 12, we just need to prove that any non-zero polynomial P remains non-zero when it remains non-zero when $\hat{z}_{i_1,i_2,b_1,b_2}$ is replaced by 1 and $\hat{e}_0$ and $\hat{e}_1$ are replaced respectively by ed_0 and ed_1 (where e is a fresh formal variable).

Such a non-zero polynomial P can be written $Q_0\hat{d}_0 + Q_1\hat{d}_1 + Q_2\hat{e}_0 + Q_3\hat{e}_1$, where Q_0, Q_1, Q_2, Q_3 are four polynomials over all the formal variables except $\hat{d}_0, \hat{d}_1, \hat{e}_0, \hat{e}_1$, because indices prevent the multiplication of two of the variables $\hat{d}_0, \hat{d}_1, \hat{e}_0, \hat{e}_1$. Furthemore at least one of the polynomials Q_0, Q_1, Q_2, Q_3 is non-zero, and it remains non-zero when $\hat{z}_{i_1,i_2,b_1,b_2}$ is replaced by 1 (with a proof similar to the one of the case $\mathrm{RKPRFRand}_F^{\mathscr{A}}$ of the proof of Theorem 12, using Lemma 13). Thus, even when replacing $\hat{z}_{i_1,i_2,b_1,b_2}$ by 1 and $(\hat{e}_0, \hat{e}_1)$ by (ed_0, ed_1), P remains non-zero.

This concludes the proof. □

Similarly to Corollary 15, we have the following corollary.

Corollary 22 (Security in the generic multilinear map model) *Let k, n be two positive integers. Let X and Y be two non-empty and disjoint indices in the index set $\mathscr{U}$ of our construction in Section 3. In the generic multilinear map model with oracle $\mathscr{M}$, for any adversary $\mathscr{D}$ making at most $q_{\mathscr{M}}$ queries to the oracle $\mathscr{M}$, we have:*

$$\mathbf{Adv}^{(k,n,X,Y)\text{-}\mathsf{XY\text{-}DDH}}(\mathscr{D}) \leq 2q_{\mathscr{M}} \frac{|\mathscr{U}|}{p} \ .$$

Proof (Corollary 22). We just consider an intermediate game where we simulate everything as before except the zero-testing oracle which always outputs "non-zero" unless its input is zero, as a polynomial. Corollary 22 easily follows. □

D Security of the Sel-Prod Assumption in the Generic Multilinear Map Model

The proof of the Sel-Prod assumption is subtle and first requires the introduction of the notion of profiles and some lemmata. As mentioned in the body of the paper, for this part, we will consider that the straddling set systems used in the construction are strong straddling set systems.

D.1 Proof Ingredient: Index Sets and Profiles

We adapt the proof in [Zim15, Section 3.4] to our case. In the whole section, monomials and polynomials are over the formal variables $\hat{a}_{i,b}, \hat{\gamma}_{i,b}, \hat{c}_{j,b}, \hat{z}_{i_1,i_2,b_1,b_2}$, for $i, i_1, i_2 \in \{1, \dots, k\}$, $j \in \{1, \dots, n\}$, $b, b_1, b_2 \in \{0, 1\}$. Furthermore, we use the index set $\mathscr{U}$ defined in Section 3. We also write $\hat{a}^0_{i,b} = \hat{a}_{i,b}$ and $\hat{a}^1_{i,b} = \hat{\gamma}_{i,b}$, for $i \in \{1, \dots, k\}$ and $b \in \{0, 1\}$.

Definition 23 [Profile of a monomial] Let t be a monomial. For any $i \in \{1, \dots, k\}$ and any bit $b \in \{0, 1\}$, t *incorporates* b as its i-th bit if and only if t contains one of the following formal variables: $\hat{a}^{b'}_{i,b}, \hat{z}_{i,i',b,b'}, \hat{z}_{i',i,b',b}$ for $i' \in \{1, \dots, k\}$ and $b' \in \{0, 1\}$. The *profile* of t is the tuple $\mathsf{prof}(t) = ((\mathsf{prof}(t))_1, \dots, (\mathsf{prof}(t))_k) \in \{0, 1, *\}^k$ such that $\mathsf{prof}(t)_i = b_i = b$ if t incorporates b as its i-th bit, and $\mathsf{prof}(t)_i = b_i = *$ if t does not incorporate 0 nor 1 as its i-th bit.

Please note that since we use strong straddling set systems, t *cannot incorporate both* 0 and 1 as its i-th bit.

Definition 24 [Partial profile, conflict, and merge] Let t_1 and t_2 be two monomials. We say that:

- the profile $\mathsf{prof}(t_1)$ is *partial* if $\mathsf{prof}(t_1)_i = *$ for some $i \in \{1, \dots, k\}$;
- the profiles $\mathsf{prof}(t_1)$ and $\mathsf{prof}(t_2)$ *conflict* if there exists $i \in \{1, \dots, k\}$ such that $\mathsf{prof}(t_1)_i, \mathsf{prof}(t_2)_i \in \{0, 1\}$ and $\mathsf{prof}(t_1)_i \neq \mathsf{prof}(t_2)_i$.

We also define the *merge* of two *non-conflicting* profiles $\mathsf{prof}(t_1)$ and $\mathsf{prof}(t_2)$ as the tuple $(b_1, \dots, b_n)$ such that:

$$b_i = \begin{cases} \mathsf{prof}(t_1)_i & \text{if } \mathsf{prof}(t_1)_i \in \{0,1\} \\ \mathsf{prof}(t_2)_i & \text{if } \mathsf{prof}(t_2)_i \in \{0,1\} \\ * & \text{otherwise.} \end{cases}$$

Definition 25 [Profile of a polynomial] Let P be a polynomial. The *profile* of a polynomial P is the set $\{\mathsf{prof}(t) \mid t \text{ is in the formal expansion of } P\}$.

Let us now characterize the polynomials that appears in $\mathsf{Sel\text{-}Prod}$.

Lemma 26 (Characterization of polynomials in $\mathsf{Sel\text{-}Prod}$) *Let $\mathscr{A}$ denote an adversary in the generic multilinear map model, making at most q queries to the multilinear map oracle $\mathscr{M}$, and returning a handle h corresponding to some polynomial P. Then, P satisfies one of these two conditions:*

1. $\mathsf{prof}(P)$ *is a singleton:*
   (a) *if the index of P does not contain $\mathscr{S}_0$, there exists a polynomial Q in the formal variables $\hat{c}_{i,b}$ for $i \in \{1, \dots, n\}$ and $b \in \{0,1\}$, together with some bit $\beta \in \{0,1\}$, some tuple $K' \in \{0,1,*\}^k$, and some set $Z \subseteq \{(i_1, i_2) \mid 1 \le i_1 < i_2 \le k\} \times \{0,1\}^2$ such that:*
   $$P = Q \cdot \prod_{\substack{i=1 \\ K'_i \neq *}}^{k} \hat{a}^{\beta}_{i,K'_i} \cdot \prod_{(i_1,i_2,b_1,b_2) \in Z} \hat{z}_{i_1,i_2,b_1,b_2} \;;$$
   (b) *if the index of P does contain $\mathscr{S}_0$, there exist two polynomials Q_0 and Q_1 in the formal variables $\hat{c}_{i,b}$ for $i \in \{1, \dots, n\}$ and $b \in \{0,1\}$, some tuple $K' \in \{0,1\}^k$, and some set $Z \subseteq \{(i_1, i_2) \mid 1 \le i_1 < i_2 \le k\} \times \{0,1\}^2$ such that:*
   $$P = \left(Q_0 \cdot \prod_{i=1}^{k} \hat{a}^{0}_{i,K'_i} + Q_1 \cdot \prod_{i=1}^{k} \hat{a}^{1}_{i,K'_i} \right) \cdot \prod_{(i_1,i_2,b_1,b_2) \in Z} \hat{z}_{i_1,i_2,b_1,b_2} \;.$$
2. $\mathsf{prof}(P)$ *is a set containing at least 2 and at most q non-partial profiles $\mathsf{prof}(P) = \{K'_1, \dots, K'_{q'}\}$, with $q' \le q$:*
   (a) *if the index of P does not contain $\mathscr{S}_0$, there exist polynomials $Q_1, \dots, Q_{q'}$, $q' \ge 2$, $I \subseteq \{1, \dots, k\}$, sets $Z_1, \dots, Z_{q'}$, and a bit β such that:*
   $$P = \sum_{\ell=1}^{q'} Q_\ell \cdot \prod_{i \in I} \hat{a}^{\beta}_{i,K'_{\ell,i}} \prod_{(i_1,i_2,b_1,b_2) \in Z_\ell} \hat{z}_{i_1,i_2,b_1,b_2} \;;$$
   (b) *if the index of P does contain $\mathscr{S}_0$, there exist polynomials $Q_{1,0}, Q_{1,1}, \dots, Q_{q',0}, Q_{q',1}$ in the formal variables $\hat{c}_{i,b}$ for $i \in \{1, \dots, n\}$, sets $Z_1, \dots, Z_{q'}$, and $b \in \{0,1\}$, such that:*
   $$P = \sum_{\ell=1}^{q'} \left(Q_{\ell,0} \cdot \prod_{i=1}^{k} \hat{a}^{0}_{i,K'_{\ell,i}} + Q_{\ell,1} \cdot \prod_{i=1}^{k} \hat{a}^{1}_{i,K'_{\ell,i}} \right) \cdot \prod_{(i_1,i_2,b_1,b_2) \in Z_\ell} \hat{z}_{i_1,i_2,b_1,b_2} \;.$$

In addition, when the index set of P does not contain any $\mathscr{S}_i$ for $i \geq 1$, then necessarily $q' = 1$.

Furthermore $\mathsf{prof}(P)$ *and the above decompositions of P can be efficiently computed (i.e., in a time polynomial in κ and q).*

The proof is by induction over the expression of the polynomials formed by the adversary. We first need to introduce some intermediate lemmata.

Lemma 27 (Impossibility to add partial terms) *Let t_1, t_2 be two monomials. If* $\mathsf{prof}(t_1)$ *or* $\mathsf{prof}(t_2)$ *is partial and* $\mathsf{prof}(t_1) \neq \mathsf{prof}(t_2)$*, then t_1 and t_2 do not have the same index set.*

Proof (Lemma 27). We can assume without loss of generality that, for a fixed bit $b \in \{0,1\}$ and some $i \in \{1,\dots,k\}$, $\mathsf{prof}(t_1)_i = b$ and $\mathsf{prof}(t_2)_i \neq b$. Hence, as $\mathsf{prof}(t_2)_i \neq b$ and as $\mathsf{prof}(t_2) \neq \bot$, t_2 cannot contain a factor of $\hat{a}_{i,b}^{b'}$, $\hat{z}_{i,i',b,b'}$ or $\hat{z}_{i',i,b',b}$ for any $i' \in \{1,\dots,k\}$ and $b, b' \in \{0,1\}$. We now prove Lemma 27 by contradiction. The possible cases are the following:

- Suppose t_1 contains a factor of $\hat{a}_{i,b}^{b'}$. Then its index set was formed using $\mathsf{BitCommit}_{i,b} = S_{i,b,i}$. Since t_2 does not contain a factor of $\hat{a}_{i,b}^{b'}$, its index set was not formed using $\mathsf{BitCommit}_{i,b} = S_{i,b,i}$. Therefore, if t_1 and t_2 have the same index set, by definition of a straddling set, this index set has to contain the whole set $\mathscr{S}_i$. However, for every $i' \in \{1,\dots,k\}$ with $i \neq i'$, the only encodings that contain $S_{i,b,i'}$, for any $b \in \{0,1\}$, are $\hat{z}_{i,i',b,b'}$ and $\hat{z}_{i',i,b',b}$ for some $b' \in \{0,1\}$. This implies that t_1 contains a factor of $\hat{z}_{i,i',b,b'}$ or $\hat{z}_{i',i,b',b}$ for some $b' \in \{0,1\}$, for every $i' \in \{1,\dots,k\}$ and similarly, that t_2 contains a factor of $\hat{z}_{i,i',1-b,b'}$ or $\hat{z}_{i',i,b',1-b}$ for some $b' \in \{0,1\}$ and for every $i' \in \{1,\dots,k\}$. This contradicts the fact that at least one of the two profiles is partial.
- Suppose t_1 does not contain any factor of $\hat{a}_{i,b}^{b'}$ but does contain a factor of $\hat{z}_{i,i',b,b'}$ or $\hat{z}_{i',i,b',b}$ for some $i' \in \{1,\dots,k\}$ and some $b \in \{0,1\}$. Then, its index set was formed using $\mathsf{BitFill}_{i,i',b,b'} = S_{i,b,i'} S_{i',b',i}$ or $\mathsf{BitFill}_{i',i,b',b} = S_{i',b',i} S_{i,b,i'}$, so in particular using $S_{i,b,i'}$. Then, one can apply the same argument than in the previous case to exclude this case.

Lemma 27 immediately follows. □

Lemma 28 *Let Q_1, Q_2 be two polynomials in the formal variables $\hat{c}_{i,b}$, for $i \in \{1,\dots,n\}$ and $b \in \{0,1\}$, β_1, β_2 two bits, K'_1, K'_2 two tuples in $\{0,1,*\}^k$, and Z_1, Z_2 two subsets of $\{(i_1,i_2) \mid 1 \leq i_1 < i_2 \leq k\} \times \{0,1\}^2$. Let P_1 and P_2 be the two polynomials defined by*

$$P_\ell = Q_\ell \prod_{\substack{i=1 \\ K'_{\ell,i} \neq *}}^{k} \hat{a}_{i,K'_{\ell,i}}^{\beta_\ell} \prod_{(i_1,i_2,b_1,b_2) \in Z_\ell} \hat{z}_{i_1,i_2,b_1,b_2} \, ,$$

for $\ell \in \{1,2\}$. If the index of P_1 and P_2 is the same but does not contain $\mathscr{S}_0$ nor any $\mathscr{S}_i$ for $i \geq 1$, then $K'_1 = K'_2$ and $Z_1 = Z_2$. The latter condition (on the index not containing any $\mathscr{S}_i$ for $i \geq 1$) is in particular satisfied when the profile of P_1 (and so of P_2 too) contains a partial profile.

Proof (Lemma 28). The proof is similar to that of Lemma 27. The fact that the index of P_1 (and P_2) does not contain $\mathscr{S}_0$ guarantees that there exists i such that $K'_{1,i} = *$. Thanks to the definition of a straddling set, this means that both polynomials contain exactly the same sets from the two partitions $\{S_{0,0,i}\}_i$ and $\{S_{0,1,i}\}_i$ of $\mathscr{S}_0$ and then, thanks to Lemma 9 $K'_{1,i} \neq *$ if and only if $K'_{2,i} \neq *$ and $\beta_1 = \beta_2$. Moreover, for any i such that $K'_{1,i} \neq *$, assume $K'_{1,i} = b$. Then the index set of P_1 is formed using $S_{i,b,i}$, but since $\mathscr{S}_i$ is not contained in it, we have immediately $K'_{2,i} = b$. Therefore, $K'_1 = K'_2$. Similarly, we must have $Z_1 = Z_2$, and Lemma 28 immediately follows. □

Lemma 29 *Let $Q_{1,0}, Q_{1,1}, Q_{2,0}, Q_{2,1}$ be four polynomials in the formal variables $\hat{c}_{i,b}$, for $i \in \{1,\dots,k\}$ and $b \in \{0,1\}$, K'_1, K'_2 two tuples in $\{0,1,*\}^k$, and Z_1, Z_2 two subsets of $\{(i_1,i_2) \mid 1 \leq i_1 < i_2 \leq k\} \times \{0,1\}^2$. Let P_1 and P_2 be the two polynomials defined by*

$$P_\ell = \left(Q_{\ell,0} \prod_{\substack{i=1\\ K'_{\ell,i} \neq *}}^{k} \hat{a}^0_{i,K'_{\ell,i}} + Q_{\ell,1} \prod_{\substack{i=1\\ K'_{\ell,i} \neq *}}^{k} \hat{a}^1_{i,K'_{\ell,i}} \right) \cdot \prod_{(i_1,i_2,b_1,b_2) \in Z_\ell} \hat{z}_{i_1,i_2,b_1,b_2} ,$$

for $\ell \in \{1,2\}$. If the index of P_1 and P_2 is the same but does not contain any $\mathscr{S}_i$ for $i \geq 1$, then $K_1 = K_2$ and $Z_1 = Z_2$. The latter condition (on the index not containing any $\mathscr{S}_i$ for $i \geq 1$) is in particular satisfied when the profile of P_1 (and so of P_2 too) contains a partial profile.

Proof (Lemma 29). This lemma is almost the same as the proof of the previous lemma. Please note that since terms $\hat{a}^0_{i,b}$ and $\hat{a}^1_{i,b}$ have an index containing $S_{0,0,i}$ and $S_{0,1,i}$ respectively, the only way the above polynomials are well-defined is if either their index contains the whole $\mathscr{S}_0$ (and thus $K'_\ell \in \{0,1\}^k$) or nothing from $\mathscr{S}_0$ (and thus $K'_\ell = *^k$). The rest of the proof is exactly the same as the previous proof. □

We now have every tool to prove Lemma 26.

Proof (Lemma 26). In this proof, the integers i, i_1, i_2 are in $\{1,\dots,k\}$ and b is a bit. We set $\mathscr{S}' = \prod_{i=1}^k \mathscr{S}_i$. Furthermore:

- Q and Q_ℓ always denote polynomials in the formal variables $\hat{c}_{i,b}$,
- β and β_ℓ always denot bit strings in $\{0,1\}^k$,
- K' and K'_ℓ always denote tuples in $\{0,1,*\}^k$, and
- Z and Z_ℓ always denote subsets of $\{(i_1,i_2) \mid 1 \leq i_1 < i_2 \leq k\} \times \{0,1\}^2$.

We prove the Lemma 29 by induction on the sequence of formal polynomials P formed by the adversary $\mathscr{A}$ via oracle queries. Let P denote a newly formed polynomial. We consider the following cases:

- Suppose $P = t$ with t being a monomial: clearly, $\mathsf{prof}(P)$ is a singleton containing either a partial profile or a non-partial profile. Moreover, as a monomial, it is straightforward that it has the expected form detailed in the statement of Lemma 26.
- Suppose $P = P_1 + P_2$ with P_1, P_2 some polynomials already formed. We have two possible cases:
  1. $\mathsf{prof}(P_1)$ or $\mathsf{prof}(P_2)$ is a singleton containing one partial profile. We have necessarily $\mathsf{prof}(P_1) = \mathsf{prof}(P_2)$, otherwise P_1 and P_2 cannot be added via Lemma 27, since there must exist two monomials t_1 and t_2 in the formal expansion of P_1 and P_2 respectively such that $\mathsf{prof}(t_1) \neq \mathsf{prof}(t_2)$ and with $\mathsf{prof}(t_1)$ or $\mathsf{prof}(t_2)$ being partial. Then $\mathsf{prof}(P) = \mathsf{prof}(P_1) = \mathsf{prof}(P_2)$ and we conclude by applying lemmata 28 or 29 that P remains of the form described in cases 1a or 1b of Lemma 26.
  2. Both $\mathsf{prof}(P_1)$ and $\mathsf{prof}(P_2)$ are sets containing at least 2 non-partial profiles. Then $\mathsf{prof}(P) = \mathsf{prof}(P_1) \cup \mathsf{prof}(P_2)$ and does not contain any partial profile. Furthermore, if the index of P (which is the same as the one of P_1 and P_2) contains $\mathscr{S}_i$ for some $i \geq 1$, it is straightforward that P has the expected form, with $q' = |\mathsf{prof}(P)| \leq |\mathsf{prof}(P_1)| + |\mathsf{prof}(P_2)|$. It remains to show that if the index of P does not contain any $\mathscr{S}_i$ for $i \geq 1$, P has the expected form with $q' = 1$. By hypothesis induction, as the indexes of P_1 and P_2 do not contain any $\mathscr{S}_i$ for $i \geq 1$, then depending on whether these indexes do or do not contain $\mathscr{S}_0$, either there exist two polynomials Q_1 and Q_2, two tuples K'_1 and K'_2, a bit β, and two sets Z_1 and Z_2, such that $\mathsf{prof}(P_1) = \{K'_1\}$ and $\mathsf{prof}(P_2) = \{K'_2\}$, such that:
  $$P_\ell = Q_\ell \prod_{\substack{i=1 \\ K'_{\ell,i} \neq *}}^{k} \hat{a}^{\beta}_{i,K'_{\ell,i}} \prod_{(i_1,i_2,b_1,b_2) \in Z_\ell} \hat{z}_{i_1,i_2,b_1,b_2} \quad ,$$
  for $\ell \in \{1,2\}$, or there exist four polynomials $Q_{1,0}, Q_{1,1}, Q_{2,0}$ and $Q_{2,1}$, two tuples K'_1 and K'_2, and two sets Z_1 and Z_2, such that $\mathsf{prof}(P_1) = \{K'_1\}$ and $\mathsf{prof}(P_2) = \{K'_2\}$, such that:
  $$P_\ell = \left(Q_{\ell,0} \prod_{\substack{i=1 \\ K'_{\ell,i} \neq *}}^{k} \hat{a}^{0}_{i,K'_{\ell,i}} + Q_{\ell,1} \prod_{\substack{i=1 \\ K'_{\ell,i} \neq *}}^{k} \hat{a}^{1}_{i,K'_{\ell,i}} \right) \cdot \prod_{(i_1,i_2,b_1,b_2) \in Z_\ell} \hat{z}_{i_1,i_2,b_1,b_2} \quad ,$$
  for $\ell \in \{1,2\}$. According to Lemma 28 or Lemma 29 respectively, this implies that $K'_1 = K'_2$ and $Z_1 = Z_2$ and this clearly implies that $P = P_1 + P_2$ has the expected form.
- Suppose $P = P_1 \cdot P_2$ with P_1, P_2 some formal polynomials already formed. Once again, we have two possible cases:

1. $\mathsf{prof}(P_1)$ or $\mathsf{prof}(P_2)$ is a singleton. Without loss of generality, let us assume that $\mathsf{prof}(P_1)$ is a singleton. Then, $\mathsf{prof}(P)$ is the set containing the merge of the profiles of P_2 with the profile of P_1 and P has the expected form.
2. Assume now that both profiles are sets containing at least 2 non-partial profiles. Thanks to the definition of strong straddling set systems, P_1 can only contain monomials whose index sets complement those of monomials contained in P_2, and the claim follows.

□

D.2 Security of the Sel-Prod Assumption

Theorem 30 (Impossibility of constructing encodings of zero) *Let k, n be two positive integers. In the generic multilinear map model with oracle $\mathscr{M}$, for any adversary $\mathscr{D}$ making at most $q_{\mathscr{M}}$ queries to the oracle $\mathscr{M}$, we have:*

$$\Pr\left[\,(k,n)\text{-}\mathsf{Sel\text{-}Prod\text{-}L}^{\mathscr{D}} \Rightarrow \textit{an encoding of } 0\right] \leq q_{\mathscr{M}}\left(\frac{q_{\mathscr{M}}}{2^k} + \frac{k+n}{p}\right)$$

and

$$\Pr\left[\,(k,n)\text{-}\mathsf{Sel\text{-}Prod\text{-}R}^{\mathscr{D}} \Rightarrow \textit{an encoding of } 0\right] \leq \frac{q_{\mathscr{M}}^2}{2^k} + \frac{2q_{\mathscr{M}}(k+n)}{p} \quad .$$

Proof (Theorem 30). Once again, we consider the two cases separately.

(k,n)-Sel-Prod$^{\mathscr{D}}$. The proof is similar to the case $\mathrm{RKPRFRand}_F^{\mathscr{A}}$ of the proof of Theorem 12 and to the case (k,n,X,Y)-XY-DDH-R$^{\mathscr{D}}$ of the proof of Theorem 21.

(k,n)-Sel-Prod$^{\mathscr{D}}$. Similarly to the case $\mathrm{RKPRFReal}_F^{\mathscr{A}}$ of the proof of Theorem 12, we just need to prove that any non-zero polynomial P remains non-zero when $\hat{z}_{i_1,i_2,b_1,b_2}$ is replaced by 1 and $\hat{\gamma}_{i,b} = \hat{a}^1_{i,b}$ is replaced by $a_{i,K_i \oplus b}$ where K is a random bit string in $\{0,1\}$. We call this replacement: *the partial evaluation.*

According to Lemma 26, we have four possible cases (we use the notation of this lemma):

- P has the following form:

$$P = Q \cdot \prod_{\substack{i=1 \\ K'_i \neq *}}^{k} \hat{a}^{\beta}_{i,K'_i} \cdot \prod_{(i_1,i_2,b_1,b_2)\in Z} \hat{z}_{i_1,i_2,b_1,b_2} \quad .$$

  In this case, the polynomial clearly remains non-zero after the partial evaluation.
- P has the following form:

$$P = \left(Q_0 \cdot \prod_{i=1}^{k} \hat{a}^0_{i,K'_i} + Q_1 \cdot \prod_{i=1}^{k} \hat{a}^1_{i,K'_i}\right) \cdot \prod_{(i_1,i_2,b_1,b_2)\in Z} \hat{z}_{i_1,i_2,b_1,b_2} \quad .$$

In this case, if this polynomial P is zero after the partial evaluation, then for any $i \in \{1,\dots,k\}$, $a_{i,K'_i} = a_{i,K_i \oplus K'_i}$, i.e., $K' = K \oplus K'$, or in other words $K = 0$. This happens with probability $1/2^k$.

– P has the following form:

$$P = \sum_{\ell=1}^{q'} Q_\ell \cdot \prod_{i \in I} \hat{a}^{\beta}_{i,K'_{\ell,i}} \prod_{(i_1,i_2,b_1,b_2)\in Z_\ell} \hat{z}_{i_1,i_2,b_1,b_2} \enspace ;$$

Once again, it is clear that this polynomial remains non-zero after the partial evaluation.

– P has the following form:

$$P \;=\; \sum_{\ell=1}^{q'} \left(Q_{\ell,0} \cdot \prod_{i=1}^{k} \hat{a}^0_{i,K'_{\ell,i}} + Q_{\ell,1} \cdot \prod_{i=1}^{k} \hat{a}^1_{i,K'_{\ell,i}} \right) \cdot \prod_{(i_1,i_2,b_1,b_2)\in Z_\ell} \hat{z}_{i_1,i_2,b_1,b_2} \enspace .$$

Let us suppose that P becomes the zero polynomial after the partial evaluation. The first (for any order) monomial of the term

$$Q_{1,0} \cdot \prod_{i=1}^{k} \hat{a}^0_{i,K'_{1,i}} \prod_{(i_1,i_2,b_1,b_2)\in Z_\ell} \hat{z}_{i_1,i_2,b_1,b_2}$$

needs to be canceled by one of the monomials of the terms for some ℓ:

$$Q_{\ell,1} \cdot \prod_{i=1}^{k} \hat{a}^1_{i,K'_{\ell,i}} \prod_{(i_1,i_2,b_1,b_2)\in Z_\ell} \hat{z}_{i_1,i_2,b_1,b_2} \enspace .$$

This implies that for some $\ell \in \{1,\dots,q'\}$, $K'_1 = K \oplus K'_\ell$. This happens with probability at most $q'/2^k$.

Therefore the probability that a non-zero polynomial P generated by the adversary $\mathscr{D}$ remains non-zero when $\hat{z}_{i_1,i_2,b_1,b_2}$ is replaced by 1 and $\hat{\gamma}_{i,b}$ is replaced by $a_{i,K_i \oplus b}$ where K is a random bit string in $\{0,1\}$, is at most $q_{\mathscr{M}}^2/2^k$ (as the adversary $\mathscr{D}$ generates at most $q_{\mathscr{M}}$ polynomials). We conclude as usual using the Schwarz-Zippel lemma and a hybrid argument. □

Similarly to Corollary 15 and Corollary 31, we have the following corollary.

Corollary 31 (Security in the generic multilinear map model) *Let k, n be two positive integers. In the generic multilinear map model with oracle $\mathscr{M}$, for any adversary $\mathscr{D}$ making at most $q_{\mathscr{M}}$ queries to the oracle $\mathscr{M}$, we have:*

$$\mathbf{Adv}^{(k,n)\text{-}\mathsf{Sel}\text{-}\mathsf{Prod}}(\mathscr{D}) \le 2q_{\mathscr{M}} \frac{|\mathscr{U}|}{p} \enspace .$$

Proof (Corollary 31). We just consider an intermediate game where we simulate everything as before except the zero-testing oracle which always outputs "non-zero" unless its input is zero, as a polynomial. Corollary 31 easily follows. □

E Security of Our Construction Under Non-Interactive Assumptions (Theorem 19)

Proof (Theorem 19). The proof is based on the games in Fig. 4 and Fig. 5. We consider the games in this order:

$$\mathrm{G}_0, \mathrm{G}_1, \mathrm{G}_2, \mathrm{G}_3, \mathrm{G}_{4,2}, \dots, \mathrm{G}_{4,k}, \mathrm{G}_{5,1}, \dots, \mathrm{G}_{5,n} \ .$$

For the sake of simplicity, and as already mentioned in Remark 18, we assume that the multilinear map is clean and the representation is unique. In particular, these assumptions allow us to avoid using an extractor and to assume that two equivalent games are perfectly indistinguishable (and not only statistically indistinguishable). Our proof could be easily adapted to avoid this assumption.

Game G_0**.** This game exactly corresponds to $\mathrm{RKPRFReal}_F^{\mathscr{A}}$:

$$\Pr\left[\mathrm{RKPRFReal}_F^{\mathscr{A}} \Rightarrow 1\right] = \Pr\left[\,\mathrm{G}_0 \Rightarrow 1\,\right] \ .$$

Game G_1**.** In this game, outputs of $\mathbf{RKFn}(\phi_s, x)$ are generated using $\hat{\gamma}_{i,s_i}$ instead of $\hat{a}_{i,K_i \oplus s_i}$, where:

$$\hat{a}_{i,b} = [a_{i,b}]_{S_{0,0,i}\mathsf{BitCommit}_{i,b}}$$
$$\hat{\gamma}_{i,b} = [a_{i,K_i \oplus b}]_{S_{0,1,i}\mathsf{BitCommit}_{i,b}} \ .$$

This game is perfectly indistinguishable from the previous one, so:

$$\Pr\left[\,\mathrm{G}_0 \Rightarrow 1\,\right] = \Pr\left[\,\mathrm{G}_1 \Rightarrow 1\,\right] \ .$$

Game G_2**.** In this game, $\hat{\gamma}_{i,b}$ are now chosen uniformly at random and independently of the public parameters $\hat{a}_{i,b}$. It is immediate to see that games G_1 and G_2 are indistinguishable under the (k,n)-$\mathsf{Sel\text{-}Prod}$ assumption. More precisely, we can construct an adversary $\mathscr{D}$ running in time similar to $\mathscr{A}$ such that:

$$\Pr\left[\,\mathrm{G}_1 \Rightarrow 1\,\right] - \Pr\left[\,\mathrm{G}_2 \Rightarrow 1\,\right] \le \mathbf{Adv}^{(k,n)\text{-}\mathsf{Sel\text{-}Prod}}(\mathscr{D}) \ .$$

Game G_3**.** In this game, we just slightly change the indices of $\hat{\gamma}_{1,0}$ and $\hat{\gamma}_{1,1}$, so that combining elements $\hat{\gamma}_{i,b}$ and $\hat{c}_{j,x_j}$ directly leads to a top-level encoding (without any need of the $\hat{z}_{i_1,i_2,b_1,b_2}$'s). As elements $\hat{\gamma}_{1,0}$ and $\hat{\gamma}_{1,1}$ are never directly revealed to the adversary, this game is perfectly indistinguishable from the previous one, so:

$$\Pr\left[\,\mathrm{G}_2 \Rightarrow 1\,\right] = \Pr\left[\,\mathrm{G}_3 \Rightarrow 1\,\right] \ .$$

We also call this game: Game $\mathrm{G}_{4,0}$.

Game $\mathrm{G}_{4,i'}$**.** In this game, $\prod_{i=1}^{i'} \hat{\gamma}_{i,s_i}$ are replaced by independent random elements for each tuple $(s_1, \dots, s_{i'}) \in \{0,1\}^{i'}$ queried by the adversary (so only for polynomially many such tuples). Game $\mathrm{G}_{4,i'}$ is indistinguishable from

```
proc Initialize // $G_0$
$(\mathsf{mm.sp}, \mathsf{mm.pp}) \xleftarrow{\$} \mathsf{MM.Setup}(1^\kappa, \mathscr{U})$
$K \xleftarrow{\$} \{0,1\}^k$
For $i \in \{1,\ldots,k\}$ and $b \in \{0,1\}$
    $a_{i,b} \xleftarrow{\$} \mathbb{Z}_p$
    $\hat{a}_{i,b} \leftarrow [a_{i,b}]_{S_{0,0,i}\mathsf{BitCommit}_{i,b}}$
For $j \in \{1,\ldots,n\}$ and $b \in \{0,1\}$
    $c_{j,b} \xleftarrow{\$} \mathbb{Z}_p$
    $\hat{c}_{j,b} \leftarrow [c_{i,b}]_{X_i}$
For $i_1, i_2 \in \{1,\ldots,k\}$ and $b_1, b_2 \in \{0,1\}$
    $\hat{z}_{i_1,i_2,b_1,b_2} \leftarrow [1]_{\mathsf{BitFill}_{i_1,i_2,b_1,b_2}}$
$\mathsf{pp} \leftarrow (\mathsf{mm.pp}, (\hat{a}_{i,b}), (\hat{c}_{j,b}), (\hat{z}_{i_1,i_2,b_1,b_2}))$
Return $\mathsf{pp}$

proc RKFn$(\phi_s, x)$ // $G_0$
$\hat{y}' \leftarrow \prod_{i=1}^{k} \hat{a}_{i,K_i \oplus s_i} \prod_{j=1}^{n} \hat{c}_{j,x_j}$
$\hat{y} \leftarrow \hat{y}' \prod_{i_1=1}^{k} \prod_{i_2=i_1+1}^{k} \hat{z}_{i_1,i_2,K_{i_1}\oplus s_{i_1},K_{i_2}\oplus s_{i_2}}$
Return $\mathsf{MM.Extract}(\hat{y})$

proc Finalize$(b)$ // All games
Return $b$
```

```
proc Initialize // $G_1$
$(\mathsf{mm.sp}, \mathsf{mm.pp}) \xleftarrow{\$} \mathsf{MM.Setup}(1^\kappa, \mathscr{U})$
$K \xleftarrow{\$} \{0,1\}^k$
For $i \in \{1,\ldots,k\}$ and $b \in \{0,1\}$
    $a_{i,b} \xleftarrow{\$} \mathbb{Z}_p$
    $\hat{a}_{i,b} \leftarrow [a_{i,b}]_{S_{0,0,i}\mathsf{BitCommit}_{i,b}}$
    $\gamma_{i,K_i \oplus b} \leftarrow a_{i,b}$
    $\hat{\gamma}_{i,K_i \oplus b} \leftarrow [\gamma_{i,K_i \oplus b}]_{S_{0,1,i}\mathsf{BitCommit}_{i,K_i \oplus b}}$
For $j \in \{1,\ldots,n\}$ and $b \in \{0,1\}$
    $c_{j,b} \xleftarrow{\$} \mathbb{Z}_p$
    $\hat{c}_{j,b} \leftarrow [c_{i,b}]_{X_i}$
For $i_1, i_2 \in \{1,\ldots,k\}$ and $b_1, b_2 \in \{0,1\}$
    $\hat{z}_{i_1,i_2,b_1,b_2} \leftarrow [1]_{\mathsf{BitFill}_{i_1,i_2,b_1,b_2}}$
$\mathsf{pp} \leftarrow (\mathsf{mm.pp}, (\hat{a}_{i,b}), (\hat{c}_{j,b}), (\hat{z}_{i_1,i_2,b_1,b_2}))$
Return $\mathsf{pp}$

proc RKFn$(\phi_s, x)$ // $G_1$
$\hat{y}' \leftarrow \prod_{i=1}^{k} \hat{\gamma}_{i,s_i} \prod_{j=1}^{n} \hat{c}_{j,x_j}$
$\hat{y} \leftarrow \hat{y}' \prod_{i_1=1}^{k} \prod_{i_2=i_1+1}^{k} \hat{z}_{i_1,i_2,s_{i_1},s_{i_2}}$
Return $\mathsf{MM.Extract}(\hat{y})$
```

```
proc Initialize // $G_2$
$(\mathsf{mm.sp}, \mathsf{mm.pp}) \xleftarrow{\$} \mathsf{MM.Setup}(1^\kappa, \mathscr{U})$
For $i \in \{1,\ldots,k\}$ and $b \in \{0,1\}$
    $a_{i,b} \xleftarrow{\$} \mathbb{Z}_p$
    $\hat{a}_{i,b} \leftarrow [a_{i,b}]_{S_{0,0,i}\mathsf{BitCommit}_{i,b}}$
    $\gamma_{i,b} \xleftarrow{\$} \mathbb{Z}_p$
    $\hat{\gamma}_{i,b} \leftarrow [\gamma_{i,b}]_{S_{0,1,i}\mathsf{BitCommit}_{i,b}}$
For $j \in \{1,\ldots,n\}$ and $b \in \{0,1\}$
    $c_{j,b} \xleftarrow{\$} \mathbb{Z}_p$
    $\hat{c}_{j,b} \leftarrow [c_{i,b}]_{X_i}$
For $i_1, i_2 \in \{1,\ldots,k\}$ and $b_1, b_2 \in \{0,1\}$
    $\hat{z}_{i_1,i_2,b_1,b_2} \leftarrow [1]_{\mathsf{BitFill}_{i_1,i_2,b_1,b_2}}$
$\mathsf{pp} \leftarrow (\mathsf{mm.pp}, (\hat{a}_{i,b}), (\hat{c}_{j,b}), (\hat{z}_{i_1,i_2,b_1,b_2}))$
Return $\mathsf{pp}$

proc RKFn$(\phi_s, x)$ // $G_2$
$\hat{y}' \leftarrow \prod_{i=1}^{k} \hat{\gamma}_{i,s_i} \prod_{j=1}^{n} \hat{c}_{j,x_j}$
$\hat{y} \leftarrow \hat{y}' \prod_{i_1=1}^{k} \prod_{i_2=i_1+1}^{k} \hat{z}_{i_1,i_2,s_{i_1},s_{i_2}}$
Return $\mathsf{MM.Extract}(\hat{y})$
```

```
proc Initialize // $G_3$
$(\mathsf{mm.sp}, \mathsf{mm.pp}) \xleftarrow{\$} \mathsf{MM.Setup}(1^\kappa, \mathscr{U})$
For $i \in \{1,\ldots,k\}$ and $b \in \{0,1\}$
    $a_{i,b} \xleftarrow{\$} \mathbb{Z}_p$
    $\hat{a}_{i,b} \leftarrow [a_{i,b}]_{S_{0,0,i}\mathsf{BitCommit}_{i,b}}$
    $\gamma_{i,b} \xleftarrow{\$} \mathbb{Z}_p$
    If $i = 1$
        $\hat{\gamma}_{i,b} \leftarrow [\gamma_{i,b}]_{S_{0,1,i}\mathscr{S}'}$
    Else
        $\hat{\gamma}_{i,b} \leftarrow [\gamma_{i,b}]_{S_{0,1,i}}$
For $j \in \{1,\ldots,n\}$ and $b \in \{0,1\}$
    $c_{j,b} \xleftarrow{\$} \mathbb{Z}_p$
    $\hat{c}_{j,b} \leftarrow [c_{i,b}]_{X_i}$
For $i_1, i_2 \in \{1,\ldots,k\}$ and $b_1, b_2 \in \{0,1\}$
    $\hat{z}_{i_1,i_2,b_1,b_2} \leftarrow [1]_{\mathsf{BitFill}_{i_1,i_2,b_1,b_2}}$
$\mathsf{pp} \leftarrow (\mathsf{mm.pp}, (\hat{a}_{i,b}), (\hat{c}_{j,b}), (\hat{z}_{i_1,i_2,b_1,b_2}))$
Return $\mathsf{pp}$

proc RKFn$(\phi_s, x)$ // $G_3$
$\hat{y} \leftarrow \prod_{i=1}^{k} \hat{\gamma}_{i,s_i} \prod_{j=1}^{n} \hat{c}_{j,x_j}$
Return $\mathsf{MM.Extract}(\hat{y})$
```

Fig. 4. Games G_0, G_1, G_2, G_3 used in the proof of Theorem 19

Game $\mathrm{G}_{4,i'-1}$, under the $(k, n, \mathscr{S}' \prod_{i=1}^{i'-1} S_{0,1,i}, S_{0,1,i'})$-XY-DDH assumption. More precisely, we can construct an adversary $\mathscr{B}_{i'}$ running in time similar to $\mathscr{A}$ such that:

$$\begin{aligned}\Pr\left[\,\mathrm{G}_{4,i'-1} \Rightarrow 1\,\right] - \Pr\left[\,\mathrm{G}_2 \Rightarrow 1\,\right] \\ \leq q_{\mathbf{RKFn}} \cdot \mathbf{Adv}^{(k,n,\mathscr{S}' \prod_{i=1}^{i'-1} S_{0,1,i}, S_{0,1,i'})\text{-}\mathsf{XY\text{-}DDH}}(\mathscr{B}_{i'}) \;. \quad (1)\end{aligned}$$

The reduction is immediate from the definition of the $(k, n, \mathscr{S}' \prod_{i=1}^{i'-1} S_{0,1,i}, S_{0,1,i'})$-XY-DDH assumption. We also call Game $\mathrm{G}_{4,k}$, Game $\mathrm{G}_{5,0}$.

Game $\mathrm{G}_{5,j'}$**.** In this game, $\prod_{i=1}^{k} \hat{\gamma}_{i,s_i} \prod_{j=1}^{j'} \hat{b}_{i,x_i}$ are also replaced by independent random elements for each tuple $(s_1, \dots, s_k, x_1, \dots, x_{j'}) \in \{0,1\}^{k+j'}$ queried by the adversary (so only for polynomially many such tuples). Game $\mathrm{G}_{5,j'}$ is indistinguishable from Game $\mathrm{G}_{5,j'-1}$, under the $(k, n, \mathscr{S} \prod_{j=1}^{j'-1} X_j, X_{j'})$-XY-DDH assumption. More precisely, we can construct an adversary $\mathscr{C}_{j'}$ running in time similar to $\mathscr{A}$ such that:

$$\Pr\left[\,\mathrm{G}_{4,i'-1} \Rightarrow 1\,\right] - \Pr\left[\,\mathrm{G}_2 \Rightarrow 1\,\right] \leq q_{\mathbf{RKFn}} \cdot \mathbf{Adv}^{(k,n,\mathscr{S} \prod_{j=1}^{j'-1} X_j, X_{j'})\text{-}\mathsf{XY\text{-}DDH}}(\mathscr{C}_{j'}) \;.$$

Once again, the proof is immediate from the definition of the $(k, n, \mathscr{S} \prod_{j=1}^{j'-1} X_j, X_{j'})$-XY-DDH assumption.

Finally, we remark that Game $\mathrm{G}_{5,n}$ is exactly $\mathrm{RKPRFRand}_F^{\mathscr{A}}$ and therefore:

$$\Pr\left[\,\mathrm{G}_{5,n} \Rightarrow 1\,\right] = \Pr\left[\,\mathrm{RKPRFRand}_F^{\mathscr{A}} \Rightarrow 1\,\right] \;.$$

By summing all the previous bounds, we obtain the following bound:

$$\begin{aligned}\mathbf{Adv}^{\mathsf{prf\text{-}rka}}_{\Phi_\oplus, F_{\mathsf{pp}}}(\mathscr{A}) \leq \mathbf{Adv}^{(k,n)\text{-}\mathsf{Sel\text{-}Prod}}(\mathscr{D}) + \\ \sum_{i'=2}^{k} q_{\mathbf{RKFn}} \cdot \mathbf{Adv}^{(k,n,\mathscr{S}' \prod_{i=1}^{i'-1} S_{0,1,i}, S_{0,1,i'})\text{-}\mathsf{XY\text{-}DDH}}(\mathscr{B}_{i'}) + \\ \sum_{j'=1}^{n} q_{\mathbf{RKFn}} \cdot \mathbf{Adv}^{(k,n,\mathscr{S} \prod_{j=1}^{j'-1} X_j, X_{j'})\text{-}\mathsf{XY\text{-}DDH}}(\mathscr{C}_{j'}) \;,\end{aligned}$$

where the running time of every adversary is approximately the same as the running time of $\mathscr{A}$. Theorem 19 immediately follows. □

```
proc Initialize // $\mathrm{G}_{4,i'}$
$(\mathsf{mm.sp}, \mathsf{mm.pp}) \xleftarrow{\$} \mathsf{MM.Setup}(1^\kappa, \mathscr{U})$
$T \leftarrow$ empty table
For $i \in \{1, \dots, k\}$ and $b \in \{0, 1\}$
    $a_{i,b} \xleftarrow{\$} \mathbb{Z}_p$
    $\hat{a}_{i,b} \leftarrow [a_{i,b}]_{S_{0,0,i}\mathsf{BitCommit}_{i,b}}$
    $\gamma_{i,b} \xleftarrow{\$} \mathbb{Z}_p$
    If $i = 1$
        $\hat{\gamma}_{i,b} \leftarrow [\gamma_{i,b}]_{S_{0,1,i}\mathscr{S}'}$
    Else
        $\hat{\gamma}_{i,b} \leftarrow [\gamma_{i,b}]_{S_{0,1,i}}$
For $j \in \{1, \dots, n\}$ and $b \in \{0, 1\}$
    $c_{j,b} \xleftarrow{\$} \mathbb{Z}_p$
    $\hat{c}_{j,b} \leftarrow [c_{i,b}]_{X_i}$
For $i_1, i_2 \in \{1, \dots, k\}$ and $b_1, b_2 \in \{0, 1\}$
    $\hat{z}_{i_1,i_2,b_1,b_2} \leftarrow [1]_{\mathsf{BitFill}_{i_1,i_2,b_1,b_2}}$
$\mathsf{pp} \leftarrow (\mathsf{mm.pp}, (\hat{a}_{i,b}), (\hat{c}_{j,b}), (\hat{z}_{i_1,i_2,b_1,b_2}))$
Return $\mathsf{pp}$

proc RKFn$(\phi_s, x)$ // $\mathrm{G}_{4,i'}$
If $T[(s_1, \dots, s_{i'-1})] \neq \perp$
    $r \xleftarrow{\$} \mathbb{Z}_p$
    $T[(s_1, \dots, s_{i'-1})] \leftarrow [r]_{\mathscr{S}' \prod_{i=1}^{i'-1} S_{0,1,i}}$
$\hat{y} \leftarrow T[(s_1, \dots, s_{i'-1})] \cdot \prod_{i=i'}^{k} \hat{\gamma}_{i,s_i} \prod_{j=1}^{n} \hat{c}_{j,x_j}$
Return $\mathsf{MM.Extract}(\hat{y})$
```

```
proc Initialize // $\mathrm{G}_{5,j'}$
$(\mathsf{mm.sp}, \mathsf{mm.pp}) \xleftarrow{\$} \mathsf{MM.Setup}(1^\kappa, \mathscr{U})$
$T \leftarrow$ empty table
For $i \in \{1, \dots, k\}$ and $b \in \{0, 1\}$
    $a_{i,b} \xleftarrow{\$} \mathbb{Z}_p$
    $\hat{a}_{i,b} \leftarrow [a_{i,b}]_{S_{0,0,i}\mathsf{BitCommit}_{i,b}}$
    $\gamma_{i,b} \xleftarrow{\$} \mathbb{Z}_p$
    If $i = 1$
        $\hat{\gamma}_{i,b} \leftarrow [\gamma_{i,b}]_{S_{0,1,i}\mathscr{S}'}$
    Else
        $\hat{\gamma}_{i,b} \leftarrow [\gamma_{i,b}]_{S_{0,1,i}}$
For $j \in \{1, \dots, n\}$ and $b \in \{0, 1\}$
    $c_{j,b} \xleftarrow{\$} \mathbb{Z}_p$
    $\hat{c}_{j,b} \leftarrow [c_{i,b}]_{X_i}$
For $i_1, i_2 \in \{1, \dots, k\}$ and $b_1, b_2 \in \{0, 1\}$
    $\hat{z}_{i_1,i_2,b_1,b_2} \leftarrow [1]_{\mathsf{BitFill}_{i_1,i_2,b_1,b_2}}$
$\mathsf{pp} \leftarrow (\mathsf{mm.pp}, (\hat{a}_{i,b}), (\hat{c}_{j,b}), (\hat{z}_{i_1,i_2,b_1,b_2}))$
Return $\mathsf{pp}$

proc RKFn$(\phi_s, x)$ // $\mathrm{G}_{5,j'}$
If $T[(s, x_1, \dots, x_{j'-1})] \neq \perp$
    $r \xleftarrow{\$} \mathbb{Z}_p$
    $T[(s, x_1, \dots, x_{j'-1})] \leftarrow [r]_{\mathscr{S} \prod_{j=1}^{j'-1} X_j}$
$\hat{y} \leftarrow T[(s, x_1, \dots, x_{j'-1})] \cdot \prod_{j=j'}^{n} \hat{c}_{j,x_j}$
Return $\mathsf{MM.Extract}(\hat{y})$
```

Fig. 5. Games $\mathrm{G}_{4,i'}, \mathrm{G}_{5,j'}$ used in the proof of Theorem 19 (when T is a table, $T[x]$ is the value of T at index x if this value exists, or $\perp$ otherwise)