On the Impossibility of Structure-Preserving Deterministic Primitives

Masayuki Abe[1], Jan Camenisch[2], Rafael Dowsley[3], and Maria Dubovitskaya[2,4]

[1] NTT Corporation, Japan, abe.masayuki@lab.ntt.co.jp
[2] IBM Research - Zurich, Switzerland, {jca, mdu}@zurich.ibm.com
[3] Karlsruhe Institute of Technology, Germany, rafael.dowsley@kit.edu
[4] ETH Zurich, Switzerland, dumaria@inf.ethz.ch

Abstract. Complex cryptographic protocols are often constructed in a modular way from primitives such as signatures, commitments, and encryption schemes, verifiable random functions, etc. together with zero-knowledge proofs ensuring that these primitives are properly orchestrated by the protocol participants. Over the past decades a whole framework of discrete logarithm based primitives has evolved. This framework, together with so-called generalized Schnorr proofs, gave rise to the construction of many efficient cryptographic protocols.

Unfortunately, the non-interactive versions of Schnorr proofs are secure only in the random oracle model, often resulting in protocols with unsatisfactory security guarantees. Groth and Sahai have provided an alternative non-interactive proof system (GS-proofs) that is secure in the standard model and allows for the "straight line" extraction of witnesses. Both these properties are very attractive, in particular if one wants to achieve composable security. However, GS-proofs require bilinear maps and, more severely, they are proofs of knowledge only for witnesses that are group elements. Thus, researchers have set out to construct efficient cryptographic primitives that are compatible with GS-proofs, in particular, primitives that are structure-preserving, meaning that their inputs, outputs, and public keys consist only of source group elements. Indeed, structure-preserving signatures, commitments, and encryption schemes have been proposed. Although deterministic primitives such as (verifiable) pseudo-random functions or verifiable unpredictable functions play an important role in the construction of many cryptographic protocols, no structure-preserving realizations of them are known so far.

As it turns out, this is no coincidence: in this paper we show that it is impossible to construct *algebraic* structure-preserving deterministic primitives that provide provability, uniqueness, and unpredictability. This includes verifiable random functions, unique signatures, and verifiable unpredictable functions as special cases. The restriction of structure-preserving primitives to be algebraic is natural, in particular as otherwise it is not possible to prove with GS-proofs that an algorithm has been run correctly. We further extend our negative result to pseudorandom functions and deterministic public key encryption as well as non-strictly structure-preserving primitives, where target group elements are also allowed in their ranges and public keys.

Keywords. Verifiable random functions, unique signatures, structure-preserving primitives, Groth-Sahai proofs.

1 Introduction

Most practical cryptographic protocols are built from cryptographic primitives such as signature, encryption, and commitments schemes, pseudorandom functions, and zero-knowledge (ZK) proofs. Thereby the ZK proofs often "glue" different building blocks together by proving relations among their inputs and outputs. The literature provides a fair number of different

cryptographic primitives (e.g., CL-signatures [20, 21], Pedersen Commitments [49], ElGamal and Cramer-Shoup encryption [30, 27], verifiable encryption of discrete logarithms [23], verifiable pseudo-random functions [29]) that are based on the discrete logarithm problem and that together with so-called generalized Schnorr protocols [50, 18] provide a whole framework for the construction of practical protocols. Examples of such constructions include anonymous credential systems [19, 6], oblivious transfer with access control [15], group signatures [10, 43], or e-cash [17]. The non-interactive versions of generalized Schnorr protocols are secure only in the random oracle model as they are obtained via the Fiat-Shamir heuristic [31] and it is well known the random oracles cannot be securely instantiated [25]. Consequently, many protocols constructed from this framework unfortunately are secure only in the random oracle model.

A seminal step towards a framework allowing for security proofs in the *standard model* was therefore the introduction of the so-called GS-proofs by Groth and Sahai [36]. These are efficient non-interactive proofs of knowledge or language membership and are secure in the standard model. They make use of bilinear maps to verify statements and therefore are limited to languages of certain types of equations, including systems of pairing product and multi exponentiation equations. In particular, GS-proofs are proofs of *knowledge* only for witnesses that are group elements but not for exponents. Thus, it is unfortunately not possible to use GS-proofs as a replacement for generalized Schnorr proofs in the "discrete logarithm based framework of cryptographic primitives" described earlier. To alleviate this, the research community has engaged on a quest for alternative cryptographic primitives that are *structure-preserving*, i.e., for which the public keys, inputs, and output consist of (source) group elements and the verification predicate is a conjunction of pairing product equations, thus making the primitives "GS-proof compatible" and enabling a similar, GS-proof-based, framework for the construction of complex cryptographic protocols. Such a framework is especially attractive because GS-proofs are "on-line" extractable, a property that is essential for the construction of UC-secure [24] protocols.

Structure-preserving realizations exist for primitives such as signature schemes [3, 4, 38, 13, 2], commitment schemes [3, 5], and encryption schemes [16]. However, so far no *structure-preserving* constructions are known for important primitives including pseudorandom functions (PRF) [34, 28], verifiable unpredictable functions (VUF) [46], verifiable random functions (VRF) [46, 40], simulatable verifiable random functions (VRF) [26], unique signatures (USig) [35, 46, 45], and deterministic encryption (DE) [9, 12] despite the fact that these primitives are widely employed in the literature. Examples include efficient search on encrypted data [12] from deterministic encryption; micropayments [48] from unique signatures; resettable zero-knowledge proofs [47], updatable zero-knowledge databases [44], and verifiable transaction escrow schemes [42] from verifiable random functions. PRFs together with a proof of correct evaluation have been used to construct compact e-cash [17], keyword search [32], set intersection protocols [37], and adaptive oblivious transfer protocols [22, 14, 41]. We further refer to Abdalla et al. [1] and Hohenberger and Waters [40] for a good overview of applications of VRFs .

Our Results. In this paper we show that it is no coincidence that no structure-preserving constructions of PRF, VRF, VUF, USig, and DE are known: it is in fact impossible to construct them with *algebraic* algorithms. To this end, we provide a generic definition of a secure Structure-Preserving Deterministic Primitive (SPDP) and show that such a primitive cannot be built using algebraic operations only. The latter is a very reasonable restriction, indeed all constructions of structure-preserving primitives known to date are algebraic. We then show that PRF, VRF, VUF, and USig are special cases of a SPDP. We further extend our results to de-

terministic encryption and to "non-strictly" structure-preserving primitives which are allowed to have target group elements in their public keys and ranges. Regarding the latter, we show that such primitives cannot be constructed for asymmetric bilinear maps and that the possible constructions for symmetric maps are severely restricted in the operations they can use.

Let us point out that of course our results do not rule out the possibility of constructing efficient protocols from GS-proofs and non-structure-preserving primitives. Indeed a couple of such protocols are known where although some of the inputs include exponents (e.g., x) it turned out to be sufficient if only knowledge of a group elements (e.g., g^x) is proved. Examples here include the construction of a compact e-cash scheme [7] from the Dodis-Yampolskiy VRF [29] and of a so-called F-unforgeable signature scheme [6] and its use in the construction of anonymous credentials.

Related Work. Some impossibility results and lower bounds for structure-preserving primitives are known already. Abe et al. [4] show that a signature from a structure-preserving signature scheme must consist of at least three group elements when the signature algorithm is algebraic. They also give constructions meeting this bound. Lower bounds for structure-preserving commitment schemes are presented by Abe, Haralambiev and Ohkubo [5]. They show that a commitment cannot be shorter than the message and that verifying the opening of a commitment in a symmetric bilinear group setting requires evaluating at least two independent pairing product equations. They also provide optimal constructions that match these lower bounds.

To the best of our knowledge, there are no results about the (im)possibility of structure-preserving *deterministic* primitives.

Paper Organization. In Section 2 we specify our notation, define the syntax and security properties of an algebraic structure-preserving deterministic primitive, and show that such primitives are impossible to construct. In Section 3 we present some generalizations to the non-strictly structure-preserving case. Then, in Section 4, we show how our result can be applied to structure-preserving PRF, VRF, VUF, and unique signatures. Section 5 is devoted to the impossibility results for structure-preserving deterministic encryption. Finally, Section 6 concludes the paper and points to open problems and possible future research directions.

2 Definitions and Impossibility Results for Algebraic Structure-Preserving Deterministic Primitives

2.1 Preliminaries

Notation. We say that a function is *negligible* in the security parameter λ if it is asymptotically smaller than the inverse of any fixed polynomial in λ. Otherwise, the function is said to be *non-negligible* in λ. We say that an event happens with *overwhelming* probability if it happens with probability $p(\lambda) \geq 1 - \mathsf{negl}(\lambda)$, where $\mathsf{negl}(\lambda)$ is a negligible function of λ.

We denote by $Y \xleftarrow{\$} \mathsf{F}(X)$ a *probabilistic* algorithm that on input X outputs Y. A similar notation $Y \leftarrow \mathsf{F}(X)$ is used for a *deterministic* algorithm with input X and output Y. We abbreviate polynomial time as PT.

We use an upper-case, multiplicative notation for group elements. By $\mathbb{G}_1$ and $\mathbb{G}_2$ we denote source groups and by $\mathbb{G}_T$ a target group. Let $\mathcal{G}$ be a bilinear group generator that takes as input a security parameter 1^λ and outputs the description of a bilinear group $\Lambda = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G_1, G_2)$ where $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are groups of prime order p, e is an efficient, non-degenerated bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, and G_1 and G_2 are generators of

the groups $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. We denote by $\Lambda^* = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ the description Λ without the group generators. By Λ_{sym} we denote the symmetric setting where $\mathbb{G}_1 = \mathbb{G}_2$ and $G_1 = G_2$. In the symmetric setting we simply write $\mathbb{G}$ for both $\mathbb{G}_1$ and $\mathbb{G}_2$, and G for G_1 and G_2.

We also denote the set of all possible vectors of group elements from both $\mathbb{G}_1$ and $\mathbb{G}_2$ as $\{\mathbb{G}_1, \mathbb{G}_2\}^*$, and from $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ as $\{\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T\}^*$. For example, if $H_1 \in \mathbb{G}_1, H_2 \in \mathbb{G}_2$ then $(H_2, H_1) \in \{\mathbb{G}_1, \mathbb{G}_2\}^*$ and $(H_1^a, H_2^b, H_2^c, H_1^d) \in \{\mathbb{G}_1, \mathbb{G}_2\}^*$ for $a, b, c, d \in \mathbb{Z}_p$.

Algebraic Algorithms. For a bilinear group Λ generated by $\mathcal{G}$, an algorithm Alg that takes group elements $(X_1, \ldots, X_n)$ as input and outputs a group element Y is called algebraic if Alg always "knows" a representation of Y with respect to $(X_1, \ldots, X_n)$, i.e., if there is a corresponding extractor algorithm Ext that outputs $(c_1, \ldots, c_n)$ such that $Y = \prod X_i^{c_i}$ holds for all inputs and outputs of Alg. We consider this property with respect to the source groups only. A formal definition for the minimal case where Alg takes group elements from only one group $\mathbb{G}$ and outputs one element of this group is provided below.

Definition 1 (Algebraic Algorithm). *Let* Alg *be a probabilistic PT algorithm that takes as an input a bilinear group description Λ generated by $\mathcal{G}$, a tuple of group elements $(X_1, \ldots, X_n) \in \mathbb{G}^n$ for some $n \in \mathbb{N}$, and some auxiliary string $aux \in \{0,1\}^*$ and outputs a group element Y and a string ext. The algorithm* Alg *is algebraic with respect to $\mathcal{G}$ if there is a probabilistic PT extractor algorithm* Ext *that takes the same input as* Alg *(including the random coins) and generates output $(c_1, \ldots, c_n, ext)$ such that for all $\Lambda \xleftarrow{\$} \mathcal{G}(1^\lambda)$, all polynomial sizes n, all $(X_1, \ldots, X_n) \in \mathbb{G}^n$ and all auxiliary strings aux the following inequality holds over the choice of the coins r:*

$$\mathsf{Pr}\left[\begin{array}{l}(Y, ext) \leftarrow \mathsf{Alg}(\Lambda^*, X_1, \ldots, X_n, aux; r)\ ; \\ (c_1, \ldots, c_n, ext) \leftarrow \mathsf{Ext}(\Lambda^*, X_1, \ldots, X_n, aux; r)\end{array} \middle| \ Y \neq \prod X_i^{c_i}\right] \leq \mathsf{negl}(\lambda).$$

It is straightforward to extend this definition to algorithms that output multiple elements of the groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of Λ. We note that all known constructions of structure-preserving primitives are algebraic in the sense defined here. Indeed if the considered algorithms were non-algebraic one could no longer prove their correct execution with GS-proofs.

One may see a similarity between the above definition and the knowledge of exponent assumption (KEA) [11] as both involve an extractor. We, however, emphasize that the algebraic algorithm definition characterizes honest algorithms, whereas the KEA is an assumption on adversaries.

2.2 Definitions of Structure-Preserving Deterministic Primitives

We define the syntax of a structure-preserving deterministic primitive (SPDP). An SPDP consists of the tuple of the following algorithms: $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Comp}, \mathsf{Prove}, \mathsf{Verify})$. Besides the parameters generation (Setup), key generation (KeyGen), and main computation function (Comp), it includes proving (Prove) and verification (Verify) algorithms that guarantee that the output value was computed correctly using Comp. We call it provability property. It captures the verifiability notion of some deterministic primitives such as verifiable random functions, unique signatures, and verifiable unpredictable functions. Furthermore, for the deterministic primitives that do not have an inherent verification property such as pseudorandom functions and deterministic encryption, it covers their widely used combination with non-interactive proof systems. Indeed, one of the main advantages of the structure-preserving primitives and one of the reasons to construct those is their compatibility with the existing non-interactive zero-knowledge proof systems.

Definition 2 (Provable Structure-Preserving Deterministic Primitive). *Let $\mathcal{G}$ be a bilinear group generator that takes as an input a security parameter 1^λ and outputs a description of a bilinear group $\Lambda = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G_1, G_2)$. Let $\mathcal{SK}, \mathcal{PK}, \mathcal{X}, \mathcal{Y}, \mathcal{P}$ be a secret key space, a public key space, a domain, a range, and a proof space, respectively. Let $\mathsf{F} : \mathcal{SK} \times \mathcal{X} \to \mathcal{Y}$ be a family of deterministic PT computable functions. A primitive $\mathfrak{P} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Comp}, \mathsf{Prove}, \mathsf{Verify})$ that realizes F is called a Structure-Preserving Deterministic Primitive with respect to $\Lambda, \mathcal{SK}, \mathcal{PK}, \mathcal{X}, \mathcal{Y}$, and $\mathcal{P}$ if:*

- $\mathcal{PK}, \mathcal{X}, \mathcal{Y}, \mathcal{P} \subset \{\mathbb{G}_1, \mathbb{G}_2\}^*$. *Namely, the public key space, the domain, range and the proof space consist only of the source group elements.*
- $CP \xleftarrow{\$} \mathsf{Setup}(\Lambda)$ *is a probabilistic algorithm that takes as input the group description Λ and outputs the common parameters CP. Without loss of generality we assume $\Lambda \in CP$.*
- $(PK, SK) \xleftarrow{\$} \mathsf{KeyGen}(CP)$ *is a probabilistic key generation algorithm that takes as input the common parameters and outputs a public key $PK \in \mathcal{PK}$ and a secret key $SK \in \mathcal{SK}$. It is assumed without loss of generality that PK includes CP, and SK includes PK.*
- $Y \leftarrow \mathsf{Comp}(X, SK)$ *is a deterministic algorithm that takes $X \in \mathcal{X}$ and a secret key SK as input and outputs $Y \in \mathcal{Y}$.*
- $\Pi \xleftarrow{\$} \mathsf{Prove}(X, SK)$ *is a probabilistic algorithm that takes X, SK as input and outputs a proof $\Pi \in \mathcal{P}$ for relation $Y = \mathsf{Comp}(X, SK)$.*
- $0/1 \leftarrow \mathsf{Verify}(X, Y, \Pi, PK)$ *is a deterministic verification algorithm that takes $(X \in \mathcal{X}, Y \in \mathcal{Y}, \Pi \in \mathcal{P}, PK \in \mathcal{PK})$ as input and accepts or rejects the proof that Y was computed correctly. The verification operations are restricted to the group operations and evaluation of pairing product equations (PPE), which for a bilinear group Λ and for group elements $A_1, A_2, \ldots \in \mathbb{G}_1, B_1, B_2, \ldots \in \mathbb{G}_2$ contained in X, Y, Π, PK and constants $c_{11}, c_{12}, \ldots \in \mathbb{Z}_p$ are equations of the form:*

$$\prod_i \prod_j e(A_i, B_j)^{c_{ij}} = 1.$$

The following properties are required from a provable *structure-preserving deterministic primitive:*

1. ***Uniqueness:*** *For all $\lambda, \Lambda, CP \xleftarrow{\$} \mathsf{Setup}(\Lambda)$, there are no values $(PK, X, Y, Y', \Pi, \Pi')$ such that $Y \neq Y'$ and $\mathsf{Verify}(X, Y, \Pi, PK) = \mathsf{Verify}(X, Y', \Pi', PK) = 1$.*
2. ***Provability:*** *For all $\lambda, \Lambda, CP \xleftarrow{\$} \mathsf{Setup}(\Lambda)\,;\ (PK, SK) \xleftarrow{\$} \mathsf{KeyGen}(CP)\,;\ X \in \mathcal{X}\,;\ Y \leftarrow \mathsf{Comp}(X, SK)\,;\ \Pi \xleftarrow{\$} \mathsf{Prove}(X, SK)$ it holds that $\mathsf{Verify}(X, Y, \Pi, PK) = 1$.*

Now, we define two security properties. The *unpredictability* property states that no PT adversary can predict the output value Y for a fresh input X after having called the Comp and Prove oracles with inputs that are different from X. The *pseudorandomness* property states that no PT adversary can distinguish the output value Y from a random value.

Definition 3 (Unpredictability). *A Structure-Preserving Deterministic Primitive $\mathfrak{P}$ is unpredictable if for all probabilistic PT algorithms $\mathcal{A}$*

$$\Pr\left[\begin{array}{l} CP \xleftarrow{\$} \mathsf{Setup}(\Lambda)\,; \\ (PK, SK) \xleftarrow{\$} \mathsf{KeyGen}(CP)\,; \\ (X, Y) \leftarrow \mathcal{A}^{\mathsf{Comp}(\cdot, SK), \mathsf{Prove}(\cdot, SK)}(PK) \end{array} \middle| \begin{array}{l} Y = \mathsf{Comp}(X, SK) \wedge \\ X \notin S \end{array} \right] \leq \mathsf{negl}(\lambda)$$

where S is the set of inputs queried to the oracles Comp *and* Prove.

Definition 4 (Pseudorandomness). *A Structure-Preserving Deterministic Primitive $\mathfrak{P}$ is pseudorandom if for all probabilistic PT distinguishers $\mathcal{D} = (\mathcal{D}_1, \mathcal{D}_2)$*

$$\mathsf{Pr}\left[\begin{array}{l} CP \xleftarrow{\$} \mathsf{Setup}(\Lambda)\,;\ (PK, SK) \xleftarrow{\$} \mathsf{KeyGen}(CP)\,; \\ (X, st) \leftarrow {\mathcal{D}_1}^{\mathsf{Comp}(\cdot,SK),\mathsf{Prove}(\cdot,SK)}(PK)\,; \\ Y_{(0)} \leftarrow F_{SK}(X)\,;\ Y_{(1)} \xleftarrow{\$} \mathcal{Y}_\lambda\,;\ b \xleftarrow{\$} \{0,1\}\,; \\ b' \xleftarrow{\$} {\mathcal{D}_2}^{\mathsf{Comp}(\cdot,SK),\mathsf{Prove}(\cdot,SK)}(Y_{(b)}, st) \end{array}\middle|\ \begin{array}{l} b = b' \wedge \\ X \notin S \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

where S is the set of queries to the oracles Comp *and* Prove*.*

One can see that a provable SPDP having the unpredictability property is a structure-preserving verifiable unpredictable function (VUF), and a provable SPDP with the pseudorandomness property is a structure-preserving verifiable random function (VRF).

2.3 Inexistence of Structure-Preserving Verifiable Unpredictable Functions

Now, we prove that a structure-preserving VUF as defined in the previous section cannot exist. Namely, we show that a provable SPDP cannot unpredictable according to Definition 3 because of its uniqueness property.

Theorem 1. *Let $\mathcal{G}$ be a bilinear group generator that takes as an input a security parameter 1^λ and outputs a description of bilinear groups $\Lambda = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G_1, G_2)$. Let $\mathfrak{P} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Comp}, \mathsf{Prove}, \mathsf{Verify})$ be a Provable Structure-Preserving Deterministic Primitive as in Definition 2. Suppose that the discrete logarithm problem is hard in the groups $\mathbb{G}_1, \mathbb{G}_2$ of Λ and let* KeyGen*,* Comp*, and* Prove *be restricted to the class of algebraic algorithms over Λ. Then $\mathfrak{P}$ is not unpredictable according to Definition 3.*

Proof. For simplicity, we first consider a symmetric bilinear setting ($\Lambda = \Lambda_{\mathsf{sym}}$), where $\mathcal{PK}$, $\mathcal{X}, \mathcal{Y}, \mathcal{P} \subset \{\mathbb{G}\}^*$. Furthermore, we consider the input X to consist only of a single group element. We then show that the same result holds for the input being a tuple of group elements from the group $\mathbb{G}$ and also in the asymmetric setting, for both Type 2 pairings (where an efficiently computable homomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$ exists and there is no efficiently computable homomorphism from $\mathbb{G}_1$ to $\mathbb{G}_2$), and Type 3 pairings (where there are no efficiently computable homomorphisms between $\mathbb{G}_1$ and $\mathbb{G}_2$) [33].

The outline of the proof is as follows. First, in Lemma 1 we show that for any $\mathfrak{P}$ that is provable and unique as specified in Definition 2, the output of Comp must have a particular format, namely $\mathsf{Comp}(X, SK) = (G^{a_1}X^{b_1}, \ldots, G^{a_\ell}X^{b_\ell})$ for constants $a_1, \ldots, a_\ell, b_1, \ldots, b_\ell \in \mathbb{Z}_p$. Then, in Lemma 2, we prove that if the output of Comp has this format then the unpredictability property from Definition 3 does not hold for $\mathfrak{P}$. This means that a structure-preserving VUF cannot exist.

Lemma 1. *Let $\mathfrak{P} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Comp}, \mathsf{Prove}, \mathsf{Verify})$ be a Structure-Preserving Deterministic Primitive such that* KeyGen, Comp*, and* Prove *are algebraic algorithms over Λ. If the discrete-logarithm problem is hard in the base group of Λ and $\mathfrak{P}$ meets the provability and uniqueness property as defined in Definition 2, then with an overwhelming probability it holds that $\mathsf{Comp}(X, SK) = (Y_1, \ldots, Y_\ell) = (G^{a_1}X^{b_1}, \ldots, G^{a_\ell}X^{b_\ell})$ for constants $a_1, \ldots, a_\ell, b_1, \ldots, b_\ell \in \mathbb{Z}_p$.*

Proof. Fix $(PK, SK) \xleftarrow{\$} \mathsf{KeyGen}(CP)$, where $PK \subset \{\mathbb{G}\}^*$. Let $x \xleftarrow{\$} \mathbb{Z}_p, X = G^x$.

Notice that because Comp, Prove and KeyGen are algebraic algorithms, their outputs can be expressed as

$$\mathsf{Comp}(X, SK) = Y = (Y_1, \ldots, Y_\ell) \text{ with } Y_i = G^{a_i} X^{b_i},$$

$$\mathsf{Prove}(X, SK) = \Pi = (\Pi_1, \ldots, \Pi_n) \text{ with } \Pi_j = G^{u_j} X^{v_j}, \text{ and}$$

$$PK = (S_1, \ldots, S_m) \text{ with } S_f = G^{s_f},$$

where $a_i = H_{1,i}(X, SK), b_i = H_{2,i}(X, SK), v_j = H_{3,j}(X, SK; r), u_j = H_{4,j}(X, SK; r)$, $H_{\ell,m}$ are arbitrary functions, and r is the randomness used by the Prove algorithm. We note that a_i, b_i, u_j, v_j can depend on X in an arbitrary manner, but since Comp and Prove are algebraic, one can extract a_i, b_i, u_j, v_j as values from $\mathbb{Z}_p$ using the extractors of algorithms Comp and Prove.

Second, we recall that according to Definition 2 the verification algorithm consists of pairing product equations (PPE). Let the k-th PPE used in the verification algorithm be

$$\prod_{f=1}^{m} e\left(S_f, X^{c_{k,1,f}} \prod_{t=1}^{m} S_t^{c_{k,2,f,t}} \prod_{i=1}^{\ell} Y_i^{c_{k,3,f,i}} \prod_{j=1}^{q} \Pi_j^{c_{k,4,f,j}}\right) \prod_{q=1}^{n} e\left(\Pi_q, \prod_{j=1}^{n} \Pi_j^{c_{k,5,q,j}}\right)$$
$$e\left(X, X^{c_{k,6}} \prod_{i=1}^{\ell} Y_i^{c_{k,7,i}} \prod_{j=1}^{q} \Pi_j^{c_{k,8,j}}\right) \cdot \prod_{w=1}^{\ell} e\left(Y_w, \prod_{i=1}^{\ell} Y_i^{c_{k,9,w,i}} \prod_{j=1}^{n} \Pi_j^{c_{k,10,m,j}}\right) = 1.$$

The intuition behind the proof is the following. We note that Comp should perform the computation without necessarily knowing the discrete logarithm of the input - otherwise one can use Comp to solve the discrete logarithm for the input X. Now, one can see that the relation in the exponents of the k-th PPE for the tuple (X, Y, Π, PK) induce a polynomial $Q_k(x)$ in the discrete logarithm x. Basically, we can re-write the k-th PPE as $e(G, G)^{Q_k(x)} = 1$. So, first, we prove that $Q_k(x)$ is a trivial function, otherwise it is possible to solve the discrete logarithm problem for the given X by solving Q_k. Second, we show that if Q_k is trivial, then by the uniqueness property a_i and b_i are constants. Let a_i, b_i, u_j, v_j be the values computed for one specific $X : Y_i = G^{a_i} X^{b_i}$, $\Pi_j = G^{u_j} X^{v_j}$, and $\mathsf{Verify}(PK, X, Y, \Pi) = 1$. Proposition 2 shows that these values can be reused to compute a correct $\tilde{Y}$ for any other $\tilde{X} \in \mathcal{X}$. So, if $\tilde{Y}_i$ is computed as $G^{a_i} \tilde{X}^{b_i}$ and $\tilde{\Pi}_j$ as $G^{u_j} \tilde{X}^{v_j}$, instead of using the normal computation procedures, then $(\tilde{X}, \tilde{Y}, \tilde{\Pi})$ are also accepted by the verification algorithm due to the triviality of Q_k. Then, from the uniqueness property, it follows that these a_i, b_i are the only valid values, i.e. constants.

Now we provide the proof in detail. First, we prove that all polynomials Q_k induced by the verification PPEs, as described above, are constants with overwhelming probability.

Proposition 1. *If the discrete logarithm problem in the base group of Λ is hard, then Q_k is a trivial function.*

Proof. The proof is done by constructing a reduction algorithm R that takes as an input a group description $\Lambda_{\mathsf{sym}} = (p, \mathbb{G}, \mathbb{G}_T, e, G)$ generated by a group generator $\mathcal{G}(1^\lambda)$ and a random element $X \in \mathbb{G}$ and outputs $x \in \mathbb{Z}_p$ that satisfies $X = G^x$ with a high probability.

The reduction algorithm R works as follows. It first takes Λ as an input and sets the common parameters $CP = \Lambda$. R then runs $\mathsf{KeyGen}(CP)$, $\mathsf{Comp}(X, SK)$, and $\mathsf{Prove}(X, SK)$ with the given X. It also runs the corresponding extractors for KeyGen, Comp, and Prove. The extractor for KeyGen outputs representations s_f that satisfy $S_f = G^{s_f}$ with overwhelming probability. Similarly, the extractor for Comp outputs representations a_i, b_i such that $Y_i = G^{a_i} X^{b_i}$, and the extractor for Prove outputs u_j, v_j such that $\Pi_j = G^{u_j} X^{v_j}$ as concrete values in $\mathbb{Z}_p$.

This set of extracted exponents s_f, a_i, b_i, u_j, v_j induce a quadratic equation Q_k in the exponents of the k-th pairing product verification equation (PPE). Let us call the variable of this exponent equation $\tilde{x}$, then we can write the k-th PPE as $e(G,G)^{Q_k(\tilde{x})} = 1$. Given the representations, R can compute $Q_k(\tilde{x}) : d_2\tilde{x}^2 + d_1\tilde{x} + d_0 = 0$ in $\mathbb{Z}_p$. The condition that $Q_k(\tilde{x})$ is non-trivial guarantees that $d_2 \neq 0$ or $d_1 \neq 0$. But then R can solve $Q_k(\tilde{x})$ for $\tilde{x}$ with standard algebra. Due to the provability property, x is one of the possible solutions to $\tilde{x}$. So if the equation is non-trivial, then we can solve this equation on $\tilde{x}$ and obtain the discrete logarithm of $X : \tilde{x} = x$ with a high probability. Therefore, if the discrete logarithm problem is hard in the base group of Λ, Q_k must be trivial.

Now we show that if Q_k is trivial then by the provability and uniqueness properties a_i and b_i are constants.

Proposition 2. *Fix (PK, SK, X) and let $a_i \leftarrow H_{1,i}(X, SK)$, $b_i \leftarrow H_{2,i}(X, SK)$, $u_j \leftarrow H_{3,j}(X, SK, r)$ and $v_j \leftarrow H_{4,j}(X, SK, r)$. If all the relations in the exponents of the PPEs are trivial, then, for any $\tilde{X} \in \mathbb{G}$, $\tilde{Y} = (\tilde{Y}_1, \ldots, \tilde{Y}_\ell)$ with $\tilde{Y}_i = G^{a_i}\tilde{X}^{b_i}$ and $\tilde{\Pi} = (\tilde{\Pi}_1, \ldots, \tilde{\Pi}_n)$ with $\tilde{\Pi}_j = G^{u_j}\tilde{X}^{v_j}$, it holds that $(\tilde{X}, \tilde{Y}, \tilde{\Pi}, PK)$ will be accepted by the verification algorithm.*

Proof. Consider fixed (PK, SK, X), any $\tilde{X} \in \mathbb{G}$, and $\tilde{Y}$ and $\tilde{\Pi}$ computed from $\tilde{X}$ as specified in the proposition. Note that the verification algorithm only evaluates PPEs and performs group memberships tests. First, all group memberships tests are clearly successful for the above tuple $(\tilde{X}, \tilde{Y}, \tilde{\Pi}, PK)$. Since all polynomials Q_k are trivial and due to the way in which $\tilde{Y}, \tilde{\Pi}$ are defined, it holds that the result of evaluating the k-th PPE will be the same for any tuple $(\tilde{X}, \tilde{Y}, \tilde{\Pi}, PK)$. Therefore, $\mathsf{Verify}(\tilde{X}, \tilde{Y}, \tilde{\Pi}, PK)$ should output the same value for every $\tilde{X} \in \mathbb{G}$. Now, considering the case where $\tilde{X} = X$ we have that $\tilde{Y} = (\tilde{Y}_1, \ldots, \tilde{Y}_\ell)$ with $\tilde{Y}_i = G^{a_i}X^{b_i}$ and $\tilde{\Pi} = (\tilde{\Pi}_1, \ldots, \tilde{\Pi}_n)$ with $\tilde{\Pi}_j = G^{u_j}X^{v_j}$. But due to the correctness of the extractors of Comp and Prove, these $\tilde{Y}$ and $\tilde{\Pi}$ are exactly the outputs of $\mathsf{Comp}(X, SK)$ and $\mathsf{Prove}(X, SK)$. Therefore, by the provability property, it holds that $\mathsf{Verify}(X, \tilde{Y}, \tilde{\Pi}, PK) = 1$ for $\tilde{X} = X$; and thus, for any $\tilde{X} \in \mathbb{G}$, $\mathsf{Verify}(\tilde{X}, \tilde{Y}, \tilde{\Pi}, PK) = 1$ also.

Now, for an arbitrary $\tilde{X} \in \mathbb{G}$, consider the tuple $(PK, SK, \tilde{X}, \tilde{Y}, \tilde{\Pi}, a_1, \ldots, a_\ell, b_1, \ldots, b_\ell)$ defined above. $\tilde{\Pi}$ is valid proof for $(\tilde{X}, \tilde{Y})$, and thus the uniqueness property guarantees that there is no other $\hat{Y} \neq \tilde{Y}$ for which there is a valid proof that $\hat{Y}$ is the output corresponding to $\tilde{X}$. But the provability property guarantees that for $(\tilde{X}, \mathsf{Comp}(\tilde{X}, SK))$ there is a valid proof of correctness. Hence, for any $\tilde{X} \in \mathbb{G}$, it holds that

$$\mathsf{Comp}(\tilde{X}, SK) = \tilde{Y} = (\tilde{Y}_1, \ldots, \tilde{Y}_\ell) = (G^{a_1}\tilde{X}^{b_1}, \ldots, G^{a_\ell}\tilde{X}^{b_\ell}),$$

where $a_1, \ldots, a_\ell, b_1, \ldots, b_\ell$ are constants in $\mathbb{Z}_p$ for a fixed SK.

Lemma 2. *Suppose that $\mathfrak{P} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Comp}, \mathsf{Prove}, \mathsf{Verify})$ is a provable Structure-Preserving Deterministic Primitive such that $\mathsf{Comp}(X, SK) = (Y_1, \ldots, Y_\ell) = (G^{a_1}X^{b_1}, \ldots, G^{a_\ell}X^{b_\ell})$ for constants $a_1, \ldots, a_\ell, b_1, \ldots, b_\ell \in \mathbb{Z}_p$. Then $\mathfrak{P}$ does not satisfy the unpredictability requirement from Definition 3.*

Proof. Pick $\hat{X}, \tilde{X}$ and define $\overline{X}$, such that $\overline{X} = \hat{X}^2/\tilde{X} \notin \{\hat{X}, \tilde{X}\}$. Then an adversary that learns

$$\begin{aligned}\mathsf{Comp}(\hat{X}, SK) = \quad & (\hat{Y}_1, \ldots, \hat{Y}_\ell) = \quad (G^{a_1}\hat{X}^{b_1}, \ldots, G^{a_\ell}\hat{X}^{b_\ell}) \text{ and} \\ \mathsf{Comp}(\tilde{X}, SK) = \quad & (\tilde{Y}_1, \ldots, \tilde{Y}_\ell) = \quad (G^{a_1}\tilde{X}^{b_1}, \ldots, G^{a_\ell}\tilde{X}^{b_\ell})\end{aligned}$$

can compute the value of $\mathsf{Comp}(\overline{X}, SK)$ as:

$$\left(\frac{\hat{Y}_1^2}{\tilde{Y}_1}, \ldots, \frac{\hat{Y}_\ell^2}{\tilde{Y}_\ell}\right) = \left(\frac{G^{2a_1}\hat{X}^{2b_1}}{G^{a_1}\tilde{X}^{b_1}}, \ldots, \frac{G^{2a_\ell}\hat{X}^{2b_\ell}}{G^{a_\ell}\tilde{X}^{b_\ell}}\right) =$$

$$\left(G^{a_1}\left(\frac{\hat{X}^2}{\tilde{X}}\right)^{b_1}, \ldots, G^{a_\ell}\left(\frac{\hat{X}^2}{\tilde{X}}\right)^{b_\ell}\right) = \left(G^{a_1}\overline{X}^{b_1}, \ldots, G^{a_\ell}\overline{X}^{b_\ell}\right) = \mathsf{Comp}(\overline{X}, SK),$$

and therefore the function is not unpredictable.

X is a tuple of group elements. Both Lemmas 1 and 2 can be easily modified to the case where X consists of more than one (say m) group element as follows. The reduction algorithm after receiving the discrete logarithm challenge X_1 will choose $m-1$ random exponents $x_2, \ldots, x_m$ and fix X_i as G^{x_i} for $i = 2, \ldots, m$. Then the lemmas use the first group element X_1 in the place of the original X. Note that in the computation of the Y_j and Π_j the exponents corresponding to $X_2, \ldots, X_m$ can essentially be incorporated into $H_{1,j}(X, SK)$ and $H_{3,j}(X, SK)$ since the prover knows $x_2, \ldots, x_m$. If the quadratic equations $Q_k(\tilde{x}_1)$ in the exponents of the PPEs are not trivial, then the first element of the input X can be used to solve the discrete logarithm problem; otherwise, supposing that the uniqueness and provability properties hold, the elements of the output will be of the form $\tilde{Y}_i = G^{a_i}\tilde{X}_1^{\ b_i}$ (for the fixed values $x_2, \ldots, x_m$) and this can be used to break the unpredictability by asking two queries in which only the first elements of the inputs are different (i.e., $\hat{X}_1$ and $\tilde{X}_1$) and then learning the output corresponding to a third input which has $\overline{X}_1 = \hat{X}_1^2/\tilde{X}_1$ and the remaining elements equal to the ones of the oracle queries.

Asymmetric bilinear groups setting. Lemmas 1 and 2 can be generalized to the asymmetric setting as well. We consider both Type 2 and Type 3 pairings. The case where there are efficiently computable homomorphisms in both directions can be reinterpreted as a symmetric setting [33]. If $\mathcal{X}_\lambda$ consists of t group elements, we choose $t-1$ random exponents $x_2, \ldots, x_t$ and fix X_i as $G_1^{x_i}$ if the i-th input element is in group $\mathbb{G}_1$ or $G_2^{x_i}$ if the i-th input element is in group $\mathbb{G}_2$. Then either some quadratic equation $Q_k(\tilde{x}_1)$ in the exponents of the PPEs is not trivial in x_1 and this can be used to solve the discrete logarithm problem in the base group in which X_1 is contained, or one of the three security properties (provability, uniqueness and unpredictability) does not hold.

In the case of Type 3 pairings, where there are no efficiently computable homomorphisms between the groups, each Y_j (let $\mathbb{G}_c$ denote the group in which it is and G_c its generator) is of the form

$$Y_j = G_c^{H_{1,j}(X,SK)} X_1^{H_{2,j}(X,SK)}$$

(where $H_{2,j}(X, SK) = 0$ if X_1 and Y_j are not in the same group) and each Π_j (that is in the group $\mathbb{G}_c$) is of the form $\Pi_j = G_c^{H_{3,j}(X,SK)} X_1^{H_{4,j}(X,SK)}$ (where $H_{4,j}(X, SK) = 0$ if X_1 and Π_j are not in the same group), in both cases with the exponents corresponding to $X_2, \ldots, X_t$ incorporated into $H_{1,j}(X, SK)$ and $H_{3,j}(X, SK)$. Then the argument continues as in the previous cases.

In the case of Type 2 pairings, there is an efficiently computable homomorphism $\phi : \mathbb{G}_2 \to \mathbb{G}_1$. Then an element Y_j of the output (or an element Π_j of the proof) that is in the group $\mathbb{G}_1$ can depend on both group generators and on X_1 or its mapping $\phi(X_1)$ into $\mathbb{G}_1$.

I.e., if $X_1 \in \mathbb{G}_1$, Y_j and Π_j have the form:

$$Y_j = G_1^{H_{1,j}(X,SK)} X_1^{H_{2,j}(X,SK)} \phi(G_2)^{H_{5,j}(X,SK)};$$

$$\Pi_j = G_1^{H_{3,j}(X,SK)} X_1^{H_{4,j}(X,SK)} \cdot \phi(G_2)^{H_{6,j}(X,SK)};$$

or if $X_1 \in \mathbb{G}_2$, Y_j and Π_j have the form:

$$Y_j = G_1^{H_{1,j}(X,SK)} \phi(X_1)^{H_{2,j}(X,SK)} \phi(G_2)^{H_{5,j}(X,SK)};$$

$$\Pi_j = G_1^{H_{3,j}(X,SK)} \phi(X_1)^{H_{4,j}(X,SK)} \phi(G_2)^{H_{6,j}(X,SK)}.$$

Then we should have $H_{1,j}(X, SK) = a_j$, $H_{2,j}(X, SK) = b_j$ and $H_{5,j}(X, SK) = z_j$ for constants a_j, b_j, z_j if the provability and uniqueness hold. But in this case the unpredictability is broken in the same way as before.

Putting Lemmas 1 and 2 together completes the proof of Theorem 1.

3 Impossibility Results for "Non-strictly" Structure-Preserving Primitives

One can see that the definition above only captures so-called "strictly" structure-preserving primitives, i.e. $\mathcal{PK}$ and $\mathcal{Y}$ must contain only source group elements. Let us discuss the case of structure-preserving primitives that also have target group elements in their public key space and/or range. A target group element can be represented by 2 source group elements using pairing randomization techniques [3] or even deterministically, by fixing the "randomization" exponents. This means that the provability property can be preserved. Now, the question is: if the uniqueness property holds is the output unpredictable according to the definition above? In this section, we show that our impossibility result can be extended to some cases of "non-strictly" structure-preserving primitives, formally defined below:

Definition 5 ("Non-strictly" Structure-Preserving Deterministic Primitive). *Let $\mathcal{G}$ be a bilinear group generator that takes as an input a security parameter 1^λ and outputs a description of bilinear groups $\Lambda = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G_1, G_2)$. Let $\mathcal{SK}, \mathcal{PK}, \mathcal{X}, \mathcal{Y}, \mathcal{P}$ be the secret key space, public key space, domain, range, and the proof space, respectively. Let $\mathfrak{P} =$ (Setup, KeyGen, Comp, Prove, Verify) be a Structure-Preserving Deterministic Primitive as defined in Def. 2, except that the range of* Comp *and* KeyGen *can contain also target group elements ($\mathcal{Y}, \mathcal{PK} \subset \{\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T\}^*$). Then the primitive $\mathfrak{P}$ is called a "non-strictly" structure-preserving deterministic primitive.*

First, we extend the notion of the algebraic algorithms from Definition 1 to operate in all groups of Λ. We provide a formal definition below.

Definition 6 (Algebraic Algorithms over Λ). *Let* Alg *be a probabilistic PT algorithm that takes as an input a bilinear group description Λ generated by $\mathcal{G}$, a tuple of group elements $(X_{1,1}, \ldots, X_{1,n}) \in \{\mathbb{G}_1\}^n, (X_{2,1}, \ldots, X_{2,m}) \in \{\mathbb{G}_2\}^m$ for some $n, m \in \mathbb{N}$, and some auxiliary string $aux \in \{0,1\}^*$ and outputs group elements $Y \in \mathbb{G}_1, W \in \mathbb{G}_2, Z \in \mathbb{G}_T$ and a string ext. The algorithm* Alg *is algebraic with respect to $\mathcal{G}$ if there is a probabilistic PT extractor algorithm* Ext *that takes the same input as* Alg *(including the random coins) and generates output $(c = (c_1, \ldots, c_n), d = (d_1, \ldots, d_m), f = (f_1, \ldots, f_{nm}), ext)$ such that for all $\Lambda \xleftarrow{\$} \mathcal{G}(1^\lambda)$, all polynomial sized n, m, all $(X_{1,1}, \ldots, X_{1,n}) \in \{\mathbb{G}_1\}^n, (X_{2,1}, \ldots, X_{2,m}) \in \{\mathbb{G}_2\}^m$ and*

all auxiliary strings aux the following inequality holds over the choice of the coins r and for $X = (X_{1,1}, \ldots, X_{1,n}, X_{2,1}, \ldots, X_{2,m})$:

$$\mathsf{Pr}\left[\begin{array}{l}(Y,W,Z,ext) \leftarrow \mathsf{Alg}(\Lambda^*, X, aux; r)\,; \\ (c,d,f,ext) \leftarrow \mathsf{Ext}(\Lambda^*, X, aux; r)\end{array} \middle| \begin{array}{l} Y \neq \prod_{i=1}^{n} X_{1,i}^{c_i} \vee \\ W \neq \prod_{j=1}^{m} X_{2,j}^{d_j} \vee \\ Z \neq \prod_{i=1}^{n}\prod_{j=1}^{m} \\ \qquad e(X_{1,i}, X_{2,j})^{f_{(j-1)n+i}} \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Similarly to Definition 1, this definition can be extended to algorithms that output multiple elements of the groups of Λ. Then one can use the extractors of KeyGen and Comp to also extract representations for target group elements output by them.

Now, as we discussed in Section 2, pairing randomization techniques allow us to preserve the provability property. Here we show that if the uniqueness property (according to Definition 2) holds, then the unpredictability property does not hold in the asymmetric setting and also in some cases for the symmetric setting.

Theorem 2. *Let $\mathfrak{P} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Comp}, \mathsf{Prove}, \mathsf{Verify})$ be a "non-strictly" structure-preserving deterministic primitive as defined in Definition 5. Suppose that the discrete logarithm problem is hard in the source groups of the asymmetric bilinear groups Λ and let* KeyGen*,* Comp*, and* Prove *be restricted to the class of algebraic algorithms over Λ. Then $\mathfrak{P}$ is not unpredictable according to Definition 3.*

Proof. The outline of the proof is the same as the one for Theorem 1. First, in Lemma 3 we show that for any $\mathfrak{P}$ that is provable and has the uniqueness property as specified in Definition 2, the output of Comp must have a particular format, namely $\mathsf{Comp}(X, SK) = (Y_1, \ldots, Y_\ell, W_1, \ldots, W_n, Z_1, \ldots, Z_m)$ with $Y_i = G_1^{a_{1,i}} X_1^{b_{1,i}}, W_i = G_2^{a_{2,i}} X_2^{b_{2,i}}$, and $Z_i = e(G_1, G_2)^{a'_i} e(X_1, G_2)^{b'_i} e(G_1, X_2)^{c'_i} e(X_1, X_2)^{d'_i}$ for $X_1, G_1 \in \mathbb{G}_1, X_2, G_2 \in \mathbb{G}_2$ and for constants $a_{j,i}, b_{j,i}, a'_i, b'_i, c'_i, d'_i \in \mathbb{Z}_p$. Then in Lemma 4 we prove that if the output of Comp has this format then the unpredictability property from Definition 3 does not hold for $\mathfrak{P}$, and thus the latter cannot exist.

We note that Lemma 3 holds for both symmetric and asymmetric settings. Lemma 4, however, holds only for the asymmetric setting, thus the result of this theorem holds only for the asymmetric setting.

Lemma 3. *If the discrete-logarithm problem is hard in the source groups of Λ, $\mathfrak{P}$ has the provability the uniqueness properties (Definition 2), then with an overwhelming probability it holds that $\mathsf{Comp}(X, SK) = (Y_1, \ldots, Y_\ell, W_1, \ldots, W_n, Z_1, \ldots, Z_m)$ with $Y_i = G_1^{a_{1,i}} X_1^{b_{1,i}}, W_i = G_2^{a_{2,i}} X_2^{b_{2,i}}$, and $Z_i = e(G_1, G_2)^{a'_i} e(X_1, G_2)^{b'_i} e(G_1, X_2)^{c'_i} \cdot e(X_1, X_2)^{d'_i}$ for $X_1, G_1 \in \mathbb{G}_1$; $X_2, G_2 \in \mathbb{G}_2$; and for constants $a_{j,i}, b_{j,i}, a'_i, b'_i, c'_i, d'_i \in \mathbb{Z}_p$.*

Proof. Similarly to Lemma 1, we start with the symmetric setting ($\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}$) and a single group element as an input, for simplicity. Fix $(PK, SK) \xleftarrow{\$} \mathsf{KeyGen}(CP)$, where a public key consists of both source and target group elements: $PK \subset \{\mathbb{G}, \mathbb{G}_T\}^*$. Let $x \xleftarrow{\$} \mathbb{Z}_p, X = G^x$.

First, since Comp is deterministic and KeyGen, Comp and Prove are all algebraic algorithms, without loss of generality, their outputs can be expressed as

$$\mathsf{Comp}(X, SK) = Y = (Y_1, \ldots, Y_\ell, Z_1, \ldots, Z_{\ell'}) \text{ with } Y_i = G^{a_{1,i}} X^{b_{1,i}},$$
$$Z_i = e(G^{a_i} X^{b_i}, G^{c_i} X^{d_i}) = e(G, G)^{a'_i} e(X, G)^{b'_i} e(G, X)^{c'_i} e(X, X)^{d'_i};$$

$$\mathsf{Prove}(X, SK) = \Pi = (\Pi_1, \ldots, \Pi_n) \text{ with } \Pi_j = G^{u_j} X^{v_j},$$

$$PK = (S_1, \ldots S_m, T_1, \ldots T_{m'}) \text{ with } S_f = G^{s_f}, T_{f'} = G^{t_{f'}};$$

where $a_i = H_{a,i}(X, SK), b_i = H_{b,i}(X, SK), a'_i = H_{a',i}(X, SK), b'_i = H_{b',i}(X, SK), c'_i = H_{c',i}(X, SK), d'_i = H_{d',i}(X, SK), v_j = H_{v,j}(X, SK, r), u_j = H_{u,j}(X, SK, r)$, and $H_{*,*}$ are arbitrary functions. We note that $a_i, b_i, a'_i, b'_i, c'_i, d'_i, u_j, v_j$ can depend on X in an arbitrary manner, but since Comp and Prove are algebraic, one can extract $a_i, b_i, a'_i, b'_i, c'_i, d'_i, u_j, v_j$ as values from $\mathbb{Z}_p$ using the extractors of algorithms KeyGen, Comp and Prove.

Second, we recall that according to Definition 2 the verification algorithm consists of pairing product equations (PPE). Let the k-th PPE from the verification algorithm be

$$e\left(X, X^{c_{k,6}} \prod_{i=1}^{\ell} Y_i^{c_{k,7,i}} \prod_{j=1}^{q} \Pi_j^{c_{k,8,j}}\right) \cdot \prod_{w=1}^{\ell} e\left(Y_w, \prod_{i=1}^{\ell} Y_i^{c_{k,9,w,i}} \prod_{j=1}^{n} \Pi_j^{c_{k,10,m,j}}\right) \prod_{i=1}^{\ell'} Z_i^{c_{k,i}}$$

$$\prod_{f=1}^{m} e\left(S_f, X^{c_{k,1,f}} \prod_{t=1}^{m} S_t^{c_{k,2,f,t}} \prod_{i=1}^{\ell} Y_i^{c_{k,3,f,i}} \prod_{j=1}^{q} \Pi_j^{c_{k,4,f,j}}\right) \prod_{q=1}^{n} e\left(\Pi_q, \prod_{j=1}^{n} \Pi_j^{c_{k,5,q,j}}\right) = T_k.$$

The proof works very similarly to Lemma 1. One can see that the relation in the exponents of the k-th PPE for the tuple (X, Y, Π, PK) induce a polynomial $Q_k(x)$ in the discrete logarithm x. Basically, we can re-write the k-th PPE as $e(G, G)^{Q_k(x)} = 1$. So, first, we prove that $Q_k(x)$ is a trivial function, otherwise it is possible to solve the discrete logarithm problem for the given X by solving Q_k. Second, if Q_k is trivial, then by the uniqueness property $a_i, b_i, a'_i, b'_i, c'_i, d'_i$ are constants. Let $a_i, b_i, a'_i, b'_i, c'_i, d'_i, u_j, v_j$ be the correct values computed for one specific X : $Y_i = G^{a_{1,i}} X^{b_{1,i}}$, $Z_i = e(G, G)^{a'_i} \cdot e(X, G)^{b'_i} e(G, X)^{c'_i} e(X, X)^{d'_i}$, $\Pi_j = G^{u_j} X^{v_j}$, and $\mathsf{Verify}(PK, X, Y, \Pi) = 1$. Proposition 2 shows that these values can be reused to compute a correct $\tilde{Y}$ for any other $\tilde{X} \in \mathcal{X}$. So if $\tilde{Y}_i$ is computed as $G^{a_{1,i}} \tilde{X}^{b_{1,i}}$, $\tilde{Z}_i = e(G, G)^{a'_i} e(\tilde{X}, G)^{b'_i} e(G, \tilde{X})^{c'_i} e(\tilde{X}, \tilde{X})^{d'_i}$, and $\tilde{\Pi}_j$ as $G^{u_j} \tilde{X}^{v_j}$, instead of using the normal computation procedures, then $(\tilde{X}, \tilde{Y}, \tilde{\Pi})$ are also accepted by the verification algorithm due to the triviality of Q_k. Then, from the uniqueness property, it follows that these $a_i, b_i, a'_i, b'_i, c'_i, d'_i$ are the only valid values, i.e. constants.

Similarly to Lemma 1 the proof above can be extended to the asymmetric setting and to the case when the input consists of a tuple of group elements.

Lemma 4. *Let Λ be a description of asymmetric bilinear groups. Let $\mathfrak{P} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Comp}, \mathsf{Prove}, \mathsf{Verify})$ be a "non-strictly" structure-preserving deterministic primitive as defined in Definition 5. If $\mathfrak{P}$ is provable and unique according to Definition 2 then $\mathfrak{P}$ does not satisfy the unpredictability requirement of Definition 3.*

Proof. We consider an input X consisting of a single element from the first source group $X_1 \in \mathbb{G}_1$ and a single element from the second source group $X_2 \in \mathbb{G}_2$: $X = (X_1, X_2)$. Applying Lemma 3, the output of Comp looks as follows, without loss of generality:

$$\mathsf{Comp}(X, SK) = (Y_1, \ldots, Y_\ell, W_1, \ldots, W_{\ell''}, Z_1, \ldots, Z_{\ell'}) \text{ with } Y_i = G_1^{a_{1,i}} X_1^{b_{1,i}},$$

$$W_i = G_2^{a_{2,i}} X_2^{b_{2,i}}, Z_i = e(G_1^{a_i} X_1^{b_i}, G_2^{c_i} X_2^{d_i}) =$$

$$e(G_1, G_2)^{a'_i} e(X_1, G_2)^{b'_i} e(G_1, X_2)^{c'_i} e(X_1, X_2)^{d'_i}.$$

Pick $\hat{X}_1, \tilde{X}_1 \in \mathbb{G}_1, X_2 \in \mathbb{G}_2$, set $\hat{X} = (\hat{X}_1, X_2), \tilde{X} = (\tilde{X}_1, X_2)$ and define $\overline{X}$, such that $\overline{X}_1 = \hat{X}_1^2/\tilde{X}_1 \notin \{\hat{X}_1, \tilde{X}_1\}, \overline{X}_2 = X_2$. As we proved in Lemma 2, the unpredictability does not hold for source group elements. Now we show that with this choice of the input the adversary can also compute the target group elements of the output. For simplicity, let us now consider the output consist only of the target group elements.

An adversary that learns $\mathsf{Comp}(\hat{X}, SK) = (\hat{Z}_1, \ldots, \hat{Z}_{\ell'})$ with $\hat{Z}_i = e(G_1^{a_i}\hat{X}_1^{b_i}, G_2^{c_i}\hat{X}_2^{d_i}) = e(G_1, G_2)^{a'_i} e(\hat{X}_1, G_2)^{b'_i} e(G_1, \hat{X}_2)^{c'_i} e(\hat{X}_1, \hat{X}_2)^{d'_i}$ and $\mathsf{Comp}(\tilde{X}, SK) = (\tilde{Z}_1, \ldots, \tilde{Z}_{\ell'})$ with $\tilde{Z}_i = e(G_1^{a_i}\tilde{X}_1^{b_i}, G_2^{c_i}\tilde{X}_2^{d_i}) = e(G_1, G_2)^{a'_i} e(\tilde{X}_1, G_2)^{b'_i} e(G_1, \tilde{X}_2)^{c'_i} e(\tilde{X}_1, \tilde{X}_2)^{d'_i}$ can already compute the value of $\mathsf{Comp}(\overline{X}, SK) = (\overline{Z}_1, \ldots, \overline{Z}_{\ell'})$ as $\left(\frac{\hat{Z}_1^2}{\tilde{Z}_1}, \ldots, \frac{\hat{Z}_{\ell'}^2}{\tilde{Z}_{\ell'}}\right)$, because we have that

$$\frac{\hat{Z}_i^2}{\tilde{Z}_i} = \frac{e(G_1, G_2)^{2a'_i} e(\hat{X}_1, G_2)^{2b'_i} e(G_1, \hat{X}_2)^{2c'_i} e(\hat{X}_1, \hat{X}_2)^{2d'_i}}{e(G_1, G_2)^{a'_i} e(\tilde{X}_1, G_2)^{b'_i} e(G_1, \tilde{X}_2)^{c'_i} e(\tilde{X}_1, \tilde{X}_2)^{d'_i}} =$$

$$e(G_1, G_2)^{a'_i} \cdot e\left(\frac{\hat{X}_1^2}{\tilde{X}}, G_2\right)^{b'_1} \cdot e\left(G_1, \overline{X}_2\right)^{c'_1} \cdot e\left(\frac{\hat{X}_1^2}{\tilde{X}_1}, \overline{X}_2\right)^{d'_1} = \overline{Z}_i.$$

Therefore, $\mathfrak{P}$ is not unpredictable for the target group elements either.

One can see that for the symmetric setting the result above holds only if there is no element $e(X_1, X_2)^{d'_i}$ in the output. I.e., since $X_1 = X_2 = X$, when X appears on both sides of the pairing, the relation will not be linear - X^2 will induce the power of 4 in the output: $e(X, X)^4$.

Corollary 1. *Let $\mathfrak{P} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Comp}, \mathsf{Prove}, \mathsf{Verify})$ be "non-strictly" struc-ture-preserving deterministic primitive as defined in Definition 5. Suppose that the discrete logarithm problem is hard in the source groups of the symmetric bilinear groups Λ_{sym} and let* KeyGen, Comp, *and* Prove *be restricted to the class of algebraic algorithms over Λ. If the* Comp *algorithm does not have the output of the form $e(X, X)^{d'_i}$, then $\mathfrak{P}$ is not unpredictable according to Definition 3.*

Finally, allowing just a public key to contain target group elements would also induce the impossibility result in both symmetric and asymmetric settings (see the following corollary).

Corollary 2. *Let $\mathcal{G}$ be a bilinear group generator that takes as an input a security parameter 1^λ and outputs a description of bilinear groups $\Lambda = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G_1, G_2)$. Let $\mathcal{SK}, \mathcal{PK}, \mathcal{X}, \mathcal{Y}, \mathcal{P}$ be a secret key space, public key space, domain, range and a proof space, respectively. Let $\mathfrak{P} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Comp}, \mathsf{Prove}, \mathsf{Verify})$ be a Structure-Preserving Deterministic Primitive as defined in Definition 2, except that the public key can contain also target group elements ($\mathcal{PK} \in \{\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T\}^*$). Then $\mathfrak{P}$ is not unpredictable according to Definition 3.*

One can see that if Comp does not contain target group elements, but the public key does, then the result follows from Lemmas 1 and 2 with a slight modification of Lemma 1. Namely, in this case KeyGen is algebraic according to Definition 6 and one can use its extractor to compute the exponents for both source and target group elements of the public key.

4 Impossibility Results for Structure-Preserving PRF, VRF and Unique Signatures

In this section we show how the definition of an abstract provable structure-preserving deterministic primitive (SPDP) given in Section 2 relates to the definitions of structure-preserving

verifiable random function (VRF), and unique signatures (USig). We show that the security properties of an SPDP are necessary conditions for any VRF or USig to be secure.[5] We also discuss how the SPDP definition relates to structure-preserving PRF.

We recall the standard definitions of PRF and USig with a slight adaptation to our notation in Appendix A. Here we only explain how the requirements for structure-preserving variants of these primitives are captured by our SPDP definition.

4.1 Impossibility of Structure-Preserving Unique Signatures.

A unique signature scheme consists of the setup Setup, key generation KeyGen, signing Sign and verification Verify algorithms as formally defined below:

Definition 7 (Unique Signatures [45]). *A function family $\sigma : \mathcal{SK} \times \mathcal{X} \to \mathcal{Y}$ is an unique signature scheme (USig) if there exists probabilistic PT algorithms* Setup *and* KeyGen*, and deterministic PT algorithms* Sign *and* Verify *(in case* Verify *is probabilistic, the adjustment to the definition is straightforward) such that:*

- $CP \xleftarrow{\$} \mathsf{Setup}(\Lambda)$ *is a common parameter generation algorithm that takes as input a group description Λ and outputs the common parameters CP.*
- $(PK, SK) \xleftarrow{\$} \mathsf{KeyGen}(CP)$ *is a key generation algorithm that takes as input the common parameters CP and outputs a public key PK and the corresponding secret key SK.*
- $Y \leftarrow \mathsf{Sign}(X, SK)$ *is a deterministic algorithm that takes as input $X \in \mathcal{X}, \mathcal{SK} \in \mathcal{SK}$ and outputs the signature $Y = \sigma_{SK}(X) \in \mathcal{Y}$.*
- $0/1 \leftarrow \mathsf{Verify}(X, Y, PK)$ *is a verification algorithm that takes as input a public key PK, $X \in \mathcal{X}$, $Y \in \mathcal{Y}$ and verifies whether $Y = \sigma_{SK}(X)$.*

The following properties are required from an unique signature scheme:

1. ***Uniqueness of the signature:*** *There are no values (PK, X, Y, Y') such that $Y \neq Y'$ and $\mathsf{Verify}(X, Y, PK) = \mathsf{Verify}(X, Y', PK) = 1$.*
2. ***Security:*** *For all probabilistic PT adversaries $\mathcal{A}$:*

$$\mathsf{Pr}\left[\begin{array}{l} CP \xleftarrow{\$} \mathsf{Setup}(\Lambda)\,; \\ (PK, SK) \xleftarrow{\$} \mathsf{KeyGen}(CP)\,; \\ (X, Y) \xleftarrow{\$} \mathcal{A}^{\mathsf{Sign}(\cdot, SK)}(PK) \end{array} \middle| \; \mathsf{Verify}(X, Y, PK) = 1 \wedge X \notin S \right] \leq \mathsf{negl}(\lambda)$$

where S is the set of queries to the oracle Sign*.*

Goldwasser and Ostrovsky [35] proposed a relaxed definition for USig. Namely, they require a proof that the signature is correct as an additional input to the verification algorithm. This proof is also an output of the signing algorithm together with the signature, but it might not be unique. This definition is sufficient to construct a VRF from USig [46].

Applying the definition of an SPDP to the context of unique signatures, one can see that Comp is the signing algorithm, and that the Prove algorithm does not exist, which is equivalent to a Prove algorithm that always returns an empty string. From the security point of view, the uniqueness of the unique signatures according to Definition 7 is the same as in Definition 2. Now we see the match for the unpredictability property.

[5] Note that the requirements are necessary conditions, but maybe not sufficient conditions, e.g., in the case of VRF pseudorandomness is a stronger requirement than unpredictability.

One can see that in the security game from Definition 7 an adversary can output a forgery Y that passes the verification equation, but it can be computed in an arbitrary manner. However, in the unpredictability game from Definition 3 a forgery must be computed using the Comp algorithm. But because of the provability and uniqueness properties the former condition ($\mathsf{Verify}(X, Y, PK) = 1$) actually implies the latter one ($Y = \mathsf{Comp}(X, SK)$). Therefore, the unpredictability property from Definition 3 is equivalent to the Security property of USig described above. Thus, the following corollary holds:

Corollary 3. *Assuming the hardness of the discrete logarithm problem in the base groups of Λ, there is no unique signature that is algebraic and secure.*

4.2 Impossibility of Structure-Preserving Verifiable Random Functions.

The syntax of a verifiable random function (VRF) follows our generic definition of an SPDP: VRF consists of $\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Comp}, \mathsf{Prove}$, and Verify algorithms. Structure-preserving VRF has the same restriction on the public key space, domain, range and proof space as an SPDP, namely, they consist only of source group elements. But the security requirements for a VRF are slightly different. Instead of unpredictability, VRF has a pseudorandomness property (see Definition 4).

Lemma 5. *If a verifiable random function is pseudorandom according to Definition 4 then its translation to a generic deterministic primitive satisfies unpredictability as defined in Definition 3.*

Proof. The distinguisher $\mathcal{D} = (\mathcal{D}_1, \mathcal{D}_2)$ of the pseudorandomness from Definition 4 can use the adversary $\mathcal{A}$ that breaks the unpredictability according to Definition 3. $\mathcal{D}_1$ executes a copy of $\mathcal{A}$ internally and forwards the oracle queries/answers appropriately. If $\mathcal{A}$ produces an output pair (X, Y) where Y is an output value for a fresh input X that was not queried to the oracle before, then $\mathcal{D}_1$ uses X as his output and forwards Y to $\mathcal{D}_2$ who uses Y to distinguish if the returned challenge $Y_{(b)}$ is a random value or the output of the real function. If no such pair (X, Y) is produced by $\mathcal{A}$, then $\mathcal{D}$ makes a random guess. □

Given the above, the impossibility of an SPDP that provides unpredictability implies the impossibility of a VRF that provides pseudorandomness:

Corollary 4. *Assuming the hardness of the discrete logarithm problem in the base groups of Λ, there is no verifiable random function that is algebraic and secure.*

One can see that this result also rules out the construction of a structure-preserving simulatable VRF (sVRF) [26], which is a special case of a VRF with the public parameters (see Definition 1 from [26]) and is a key building block in some e-cash schemes [7].

4.3 Impossibility of Structure-Preserving Pseudorandom Functions.

The standard definition of a PRF (Definition 11 in Appendix A) does not feature Prove and Verify algorithms. However, the reason one wants a PRF to be structure-preserving is that one can use GS-proofs so that one party can prove to another party that the Comp algorithm was followed as prescribed. As we mentioned before, one of the examples of using PRF coupled with non-interactive zero-knowledge (NIZK) proofs is in e-cash systems [17, 7].

This approach essentially adds Prove and Verify algorithms to the definition of a PRF. We formalize it later in this section. First, note that adding a proof that a PRF was computed correctly does not result in a VRF as in the latter case there is a public key and one wants to verify that the VRF was really computed with a specific public key. Whereas here one is interested in proving that the PRF was correctly computed w.r.t. any secret key, which is a weaker requirement. We are thus interested in the question of whether it is possible to construct a PRF for which one can prove the correctness of computation with GS-proofs. Or, more generally, with NIZK proofs that use only PPE for verification. With this in mind, we define a variant of Definition 2 and Definition 4, which are extensions of PRF definitions with Prove and Verify algorithms and further have Setup generate parameters for NIZK proofs. Note that one can of course always trivially prove that a PRF was computed correctly without using NIZK proofs by just revealing the secret key (as is for instance done in the Hohenberger-Waters signature scheme [39]). Formally, this proof method follows the definition we give, nevertheless, it is easy to see that a straightforward adaptation of our impossibility proof rules out the existence of PRFs (or even functions which instead of being pseudorandom are only unpredictable) that are algebraic and secure according to the suitable modification of Definition 2 and Definition 4 to allow the verification algorithm to use SK.

Non-Interactive Zero-Knowledge Proof System. First, we define a Non-Interactive Zero-Knowledge Proof System. Let R be an efficiently computable binary relation. For pairs $(W, S) \in R$ we call S the statement and W the witness. Let $\mathcal{L}$ be the language consisting of statements in R.

Definition 8 (Non-Interactive Zero-Knowledge Proof System (NIZK)). *Let $\mathcal{G}$ be a bilinear group generator that takes as an input a security parameter 1^λ and outputs a description of a bilinear group $\Lambda = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G_1, G_2)$. The non-interactive zero-knowledge proof system for a language $\mathcal{L}$ consists of the following algorithms and protocols:*

- $CP \xleftarrow{\$} \mathsf{Setup}_{nizk}(\Lambda)$*: On input Λ, it outputs the common parameters (CP) for the proof system.*
- $\Pi \xleftarrow{\$} \mathsf{Prove}_{nizk}(CP, W, S)$*: On input the common parameters CP, a statement S, and a witness W, it generates a zero-knowledge proof that the witness W satisfies the statement S.*
- $0/1 \leftarrow \mathsf{Verify}_{nizk}(CP, \Pi, S)$*: On input a statement S and a proof Π, it outputs* 1 *if Π is valid, and* 0 *otherwise.*

In this work we refer to Groth-Sahai proofs [36] as the instantiation of the NIZK proof system.

Theorem 3. *[36] The Groth-Sahai ZK proof system is a non-interactive zero-knowledge (NIZK) proof system with perfect correctness, perfect soundness and composable zero-knowledge for satisfiability of a set of equations over a bilinear group where the K-linear assumption holds.*

We refer to [36] for detailed security definitions and proofs. We also note that the results from this section and Section 5 hold for non-interactive witness-indistinguishable proofs as well that can be also instantiated with NIWI proofs by Groth and Sahai ([36]).

Combining Structure-Preserving PRF with a NIZK Proof System. Below we provide a formal definition for the construction of a PRF coupled with NIZK proofs, where verification operations are restricted to checking group membership and evaluating pairing product equations. Note that $\mathsf{Prove}_{\mathsf{nizk}}$ and $\mathsf{Verify}_{\mathsf{nizk}}$ algorithms take NIZK parameters and a proof statement as an input as well. Since in our case the statement is always a correctness of Comp algorithm, for consistency of notation with Definition 2 we omit the statement input.

In order to distinguish functions and variables with the same name among different primitives, we may give subscripts that represents the primitive in obvious manner. For instance $\mathsf{Comp}_{\mathsf{prf}}$ denote Comp of the PRF in mind.

Definition 9 (Structure-Preserving Pseudorandom Function with a Proof of Computation Correctness). *Let $\mathcal{G}$ be a bilinear group generator that takes as an input a security parameter 1^λ and outputs a description of a bilinear group $\Lambda = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G_1, G_2)$. A structure-preserving pseudorandom function with a proof of computation correctness with respect to Λ is a set of the following algorithms:*

- $CP \xleftarrow{\$} \mathsf{Setup}(\Lambda)$*: Run* $CP_{prf} \xleftarrow{\$} \mathsf{Setup}_{prf}(\Lambda)$ *and* $CP_{nizk} \xleftarrow{\$} \mathsf{Setup}_{nizk}(\Lambda)$*, and return* $CP = (CP_{prf}, CP_{nizk})$.
- $(PK, SK) \xleftarrow{\$} \mathsf{KeyGen}(CP)$*: Run* $SK \xleftarrow{\$} \mathsf{KeyGen}_{prf}(CP_{prf})$*. Set an empty string to* PK*. Return* (PK, SK).
- $Y \leftarrow \mathsf{Comp}(X, SK)$*: Run* $Y \leftarrow \mathsf{Comp}_{prf}(X, SK)$*. Return* Y.
- $\Pi \xleftarrow{\$} \mathsf{Prove}(X, SK)$ *: Run* $Y \leftarrow \mathsf{Comp}_{prf}(X, SK)$ *and* $\Pi \xleftarrow{\$} \mathsf{Prove}_{nizk}(CP_{nizk}, SK, (X, Y))$*. (We consider* Prove_{nizk} *for the relation* $R = \{(SK, (X, Y)) : Y = F_{SK}(X)\}$*, where* (X, Y) *is the proof statement and* SK *is the witness.) Return* Π.
- $0/1 \leftarrow \mathsf{Verify}(X, Y, \Pi)$ *: Run* $b \leftarrow \mathsf{Verify}_{nizk}(CP_{nizk}, \Pi, (Y, X))$ *and return* b.

We now show that the above primitive provides provability, uniqueness and pseudorandomness based on the security of underlying PRF and NIZK.

Lemma 6. *The above pseudorandom function with a proof of computation correctness is a provable structure-preserving deterministic primitive defined in Def. 2 and provides unpredictability according to Definition 3 if the underlying PRF is pseudorandom and NIZK is correct and sound.*

Proof. Syntactical consistency can be verified by inspection. We focus on the security properties. First of all, provability holds from correctness of NIZK as Verify is identical to $\mathsf{Verify}_{\mathsf{nizk}}$. Uniqueness holds due to the soundness of NIZK and the fact that a PRF is deterministic. Namely, if (SK, X, Y) satisfies the relation defined by the PRF, (SK, X, Y') for $Y \neq Y'$ does not satisfy the relation since a PRF is deterministic. Thus, by the soundness of NIZK, there is no Π' that is accepted by the verification algorithm for (SK, X, Y'). The pseudorandomness holds due to the definition of a PRF (see Definition 11). The unpredictability follows from it due to the same reason as stated in Lemma 5. □

We observe, that similarly to a pair VRF-VUF, where the pseudorandomness implies unpredictability, one can follow Lemma 6 and show that any unpredictable function can be coupled with NIZK to get SPDP with required properties.

Corollary 5. *Assuming the hardness of the discrete logarithm problem in the base groups of Λ, there is no triple of pseudorandom function (or even functions that are only unpredictable), and prove and verification algorithms that is algebraic and satisfies Definition 9.*

The Corollary follows by a trivial adaptation of the proof in the context of Definition 9. As discussed in Section 4.2, if the adversary can break the unpredictability property and compute the output value himself, then he can obviously break the pseudorandomness property.

5 Impossibility Results for Structure-Preserving Deterministic Encryption

Since deterministic encryption (DE) does not fit into Definition 2 both from the syntax and security perspective, we discuss it separately here. DE consists of the following algorithms: $\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}$ (see Definition 12 from Appendix A). A structure-preserving encryption scheme has public keys, messages, and ciphertexts that consist entirely of source group elements. Moreover, the encryption and decryption algorithms perform only group and bilinear map operations.

Following Definition 2, one can view Comp as the encryption algorithm Enc from Definition 12. If we add Prove and Verify algorithms, then Prove will output a proof of the correct computation of the encryption algorithm. Below we provide a formal definition for the DE with this in mind, similarly as we did for the PRF in Section 4.3. We note that even though we assume the instantiation of Prove and Verify with GS-proofs our result holds in general for proofs that require only PPE for verification.

Definition 10 (Structure-Preserving Deterministic Encryption with a Proof of Encryption Correctness). *Let $\mathcal{G}$ be a bilinear group generator that takes as an input a security parameter 1^λ and outputs a description of a bilinear group $\Lambda = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G_1, G_2)$. Structure-Preserving Deterministic Encryption with a NIZK proof of encryption correctness with respect to Λ is a tuple of the following PT algorithms:*

- $CP \xleftarrow{\$} \mathsf{Setup}(\Lambda)$*: Run* $CP_{de} \xleftarrow{\$} \mathsf{Setup}_{de}(\Lambda)$ *and* $CP_{nizk} \xleftarrow{\$} \mathsf{Setup}_{nizk}(\Lambda)$*, and return* $CP = (CP_{de}, CP_{nizk})$.
- $(PK, SK) \xleftarrow{\$} \mathsf{KeyGen}(CP)$*: Run* $(SK, PK) \xleftarrow{\$} \mathsf{KeyGen}_{de}(CP_{de})$*. Return* (PK, SK).
- $Y \leftarrow \mathsf{Comp}(X, SK)$*: Compute* PK *from* SK*. Run* $Y_{de} \leftarrow \mathsf{Enc}(X, PK)$*. Return* $Y = (Y_{de}, PK)$.
- $\Pi \xleftarrow{\$} \mathsf{Prove}(X, SK)$: *Compute* PK *from* SK*, run* $Y_{de} \leftarrow \mathsf{Enc}(X, PK)$ *and* $\Pi \xleftarrow{\$} \mathsf{Prove}_{nizk}(CP_{nizk}, X, (Y, PK))$*. (We consider* Prove_{nizk} *for the relation* $R = \{(X, (Y_{de}, PK)) : Y_{de} = \mathsf{Enc}(X, PK)\}$*, where* (Y_{de}, PK) *is the proof statement and* X *is the witness.) Return* Π.
- $0/1 \leftarrow \mathsf{Verify}(X, Y, \Pi)$: *Run* $b \leftarrow \mathsf{Verify}_{nizk}(CP_{nizk}, \Pi, Y)$ *and return* b.

Note that X is not really referred in Verify but it is anyway consistent to the syntax of an SPDP.

We would like to point out that since Comp, as an encryption algorithm, takes a public key as an input, no unpredictability-style property can hold in this case. Thus, we cannot require both uniqueness and unpredictability properties (see Definitions 2 and 3) to hold for the primitive from the Definition 10 to be secure. Nevertheless, we show that the latter primitive, which is provable and unique, cannot exist.

First, we show that two security properties, provability and uniqueness, of an SPDP hold for the above DE coupled with (GS) NIZK-proofs. Formally:

Lemma 7. *The above deterministic encryption scheme with a proof of encryption correctness has the provability and uniqueness properties defined in Definition 2, if NIZK is correct and sound.*

The proof of the above lemma is the same as that for Lemma 6 with obvious modification and thus omitted.

Theorem 4. *Assuming the hardness of the discrete logarithm problem in the base groups of Λ, there is no algebraic structure-preserving deterministic encryption scheme, which is secure and where encryption can be verified by an algorithm, which takes X, Y, Π, PK as input and only performs group operations and PPE evaluations.*

Proof. As we mentioned before, by the definition of DE and the correctness of decryption, the uniqueness property holds if we consider Enc to be the Comp algorithm. According to Lemma 1 the ciphertext that encrypts a group element X looks as follows: $\mathsf{Comp}(X, PK) = Y = (G^{a_1}X^{b_1}, \ldots, G^{a_\ell}X^{b_\ell})$, where $a_1, \ldots, a_\ell, b_1, \ldots, b_\ell$ are constants in $\mathbb{Z}_p$, and G is a group generator. To encrypt X, it is obvious that G^{a_i} and $b_i, i = 1, \ldots, \ell$ should be efficiently derivable from the public key. This means that the ciphertext can be decrypted using the public key.

6 Conclusion

In this paper we proved that it is impossible to construct algebraic structure-preserving VRF, VUF and USig. It is also shown that PRF and DE coupled with non-interactive proof system cannot be structure-preserving, either. We further extend our results to "non-strictly" structure preserving primitives, which are allowed to have target group elements in their public keys and ranges. Regarding the latter, we show that such primitives cannot be constructed for asymmetric bilinear maps and that the possible constructions for symmetric maps are severely restricted on the operation they can use.

Although our results are restricted to the class of algebraic algorithms, all known constructions of structure-preserving primitives consist of algebraic algorithms. Finding constructions of secure structure-preserving algorithms that allow non-algebraic operations but whose correctness of computation still can be verified using a system of PPE is an interesting problem. We also would like to point out that it might be possible to extend our impossibility result to the quasi-deterministic case where the uniqueness condition can be relaxed to have at most $poly(\lambda)$ output values corresponding to each input value.

Finally, we note that the deterministic primitives might exist in a restricted form, where only one query to the oracle is allowed. Namely, one-time deterministic primitives might still be possible in the world of structure-preserving cryptography.

Acknowledgements

The authors would like to thank Kristiyan Haralambiev for the useful discussions and the anonymous reviewers for their helpful comments and suggestions. The research leading to these results was supported in part by the European Community's Seventh Framework Programme for the projects ABC4Trust (grant agreement no. 257782) and PERCY (grant agreement no. 321310).

References

1. M.Abdalla, D.Catalano, and D.Fiore. Verifiable random functions from identity-based key encapsulation. In *EUROCRYPT 2009*, *LNCS*. Springer, April 2009.
2. M.Abe, M.Chase, B.David, M.Kohlweiss, R.Nishimaki, and M.Ohkubo. Constant-size structure-preserving signatures: Generic constructions and simple assumptions. *ASIACRYPT 2012, LNCS*.

3. M.Abe, G.Fuchsbauer, J.Groth, K.Haralambiev, and M.Ohkubo. Structure-preserving signatures and commitments to group elements. In *CRYPTO 2010*, *LNCS*. Springer, August 2010.
4. M.Abe, J.Groth, K.Haralambiev, and M.Ohkubo. Optimal structure-preserving signatures in asymmetric bilinear groups. In *CRYPTO 2011*, *LNCS*. Springer, August 2011.
5. M.Abe, K. Haralambiev, and M.Ohkubo. Group to group commitments do not shrink. In *EUROCRYPT 2012*, *LNCS*. Springer, April 2012.
6. M.Belenkiy, M. Chase, M.Kohlweiss, and A.Lysyanskaya. P-signatures and noninteractive anonymous credentials. In *TCC 2008*, volume 4948 of *LNCS*. Springer, March 2008.
7. M.Belenkiy, M.Chase, M.Kohlweiss, and A.Lysyanskaya. Compact e-cash and simulatable VRFs revisited. In *PAIRING 2009*, *LNCS*. Springer, August 2009.
8. M.Bellare, A.Boldyreva, and A.O'Neill. Deterministic and efficiently searchable encryption. In *CRYPTO 2007*, *LNCS*. Springer, August 2007.
9. M.Bellare, M.Fischlin, A.O'Neill, and T.Ristenpart. Deterministic encryption: Definitional equivalences and constructions without random oracles. In *CRYPTO 2008*, *LNCS*. Springer, August 2008.
10. M.Bellare, D.Micciancio, and B.Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *EUROCRYPT 2003*, *LNCS*. Springer, May 2003.
11. Mihir Bellare and Adriana Palacio. The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In *CRYPTO 2004*, *LNCS*. Springer, August 2004.
12. A.Boldyreva, S.Fehr, and A.O'Neill. On notions of security for deterministic encryption, and efficient constructions without random oracles. In *CRYPTO 2008*, *LNCS*. Springer, August 2008.
13. J.Camenisch, M.Dubovitskaya, and K.Haralambiev. Efficient structure-preserving signature scheme from standard assumptions. In *SCN 12*, *LNCS*. Springer, September 2012.
14. J.Camenisch, M.Dubovitskaya, and G.Neven. Oblivious transfer with access control. In *ACM CCS 09*. ACM Press, November 2009.
15. J.Camenisch, M.Dubovitskaya, G.Neven, and G. M. Zaverucha. Oblivious transfer with hidden access control policies. In *PKC 2011*, *LNCS*. Springer, March 2011.
16. J.Camenisch, K.Haralambiev, M.Kohlweiss, J.Lapon, and V.Naessens. Structure preserving CCA secure encryption and applications. In *ASIACRYPT 2011*,*LNCS*. Springer, December 2011.
17. J.Camenisch, S.Hohenberger, and A.Lysyanskaya. Compact e-cash. In *EUROCRYPT 2005*, *LNCS*. Springer, May 2005.
18. J.Camenisch, A.Kiayias, and M.Yung. On the portability of generalized schnorr proofs. In *EUROCRYPT 2009*, *LNCS*. Springer, April 2009.
19. J.Camenisch and A.Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *EUROCRYPT 2001*,*LNCS*. Springer, May 2001.
20. J.Camenisch and A.Lysyanskaya. A signature scheme with efficient protocols. In *SCN 02*, *LNCS*. Springer, September 2002.
21. J.Camenisch and A.Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *CRYPTO 2004*, volume 3152 of *LNCS*. Springer, August 2004.
22. J.Camenisch, G.Neven, and A.Shelat. Simulatable adaptive oblivious transfer. In *EUROCRYPT 2007*, *LNCS*. Springer, May 2007.
23. J.Camenisch and V.Shoup. Practical verifiable encryption and decryption of discrete logarithms. In*CRYPTO 2003*, *LNCS*. Springer, August 2003.
24. R.Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*. IEEE Computer Society Press, October 2001.
25. R.Canetti, O.Goldreich, and S.Halevi. The random oracle methodology, revisited (preliminary version). In *30th ACM STOC*, pages 209–218. ACM Press, May 1998.
26. M.Chase and A.Lysyanskaya. Simulatable VRFs with applications to multi-theorem NIZK. In *CRYPTO 2007*, *LNCS*. Springer, August 2007.
27. R.Cramer and V.Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167–226, 2003.
28. Y.Dodis. Efficient construction of (distributed) verifiable random functions. In *PKC 2003*,*LNCS*. Springer, January 2003.

29. Y.Dodis and A.Yampolskiy. A verifiable random function with short proofs and keys. In *PKC 2005*, *LNCS*. Springer, January 2005.
30. T.ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *CRYPTO'84*, *LNCS*. Springer, August 1985.
31. A.Fiat and A.Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO'86*, *LNCS*. Springer, August 1987.
32. M. J. Freedman, Y.Ishai, B.Pinkas, and O.Reingold. Keyword search and oblivious pseudorandom functions. In *TCC 2005*, *LNCS*. Springer, February 2005.
33. S. D. Galbraith, K. G. Paterson, and N. P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.
34. O.Goldreich, S. Goldwasser, and S.Micali. How to construct random functions. In *25th FOCS*. IEEE Computer Society Press, October 1984.
35. S.Goldwasser and R.Ostrovsky. Invariant signatures and non-interactive zero-knowledge proofs are equivalent (extended abstract). In *CRYPTO'92*, *LNCS*. Springer, August 1993.
36. J.Groth and A.Sahai. Efficient non-interactive proof systems for bilinear groups. In *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, April 2008.
37. C.Hazay and Y.Lindell. Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries. In *TCC 2008*, *LNCS*. Springer, March 2008.
38. D.Hofheinz and T.Jager. Tightly secure signatures and public-key encryption. In *CRYPTO 2012*, *LNCS*. Springer, August 2012.
39. S.Hohenberger and B.Waters. Short and stateless signatures from the RSA assumption. In *CRYPTO 2009*, *LNCS*. Springer, August 2009.
40. S.Hohenberger and B.Waters. Constructing verifiable random functions with large input spaces. In *EUROCRYPT 2010*, *LNCS*. Springer, May 2010.
41. S.Jarecki and X.Liu. Efficient oblivious pseudorandom function with applications to adaptive OT and secure computation of set intersection. In *TCC 2009*, *LNCS*. Springer, March 2009.
42. S.Jarecki and V.Shmatikov. Handcuffing big brother: an abuse-resilient transaction escrow scheme. In *EUROCRYPT 2004*, *LNCS*. Springer, May 2004.
43. A.Kiayias and M.Yung. Group signatures with efficient concurrent join. In *EUROCRYPT 2005*, *LNCS*. Springer, May 2005.
44. M.Liskov. Updatable zero-knowledge databases. In *ASIACRYPT 2005*,*LNCS*. Springer, 2005.
45. A.Lysyanskaya. Unique signatures and verifiable random functions from the DH-DDH separation. In *CRYPTO 2002*, *LNCS*. Springer, August 2002.
46. S.Micali, M. O. Rabin, and S.l P. Vadhan. Verifiable random functions. In *40th FOCS*. IEEE Computer Society Press, October 1999.
47. S.Micali and L.Reyzin. Soundness in the public-key model. In *CRYPTO '01*, *LNCS*. Springer, 2001.
48. S.Micali and R. L. Rivest. Micropayments revisited. In *CT-RSA 2002*, *LNCS*. Springer, 2002.
49. T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO'91*, *LNCS*. Springer,1992.
50. C.-P.Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3), 1991.

A Definitions of PRF and DE

In this section we recall the definitions of the cryptographic primitives that we are concerned with, i.e., pseudorandom functions and deterministic encryption. To make the notation consistent throughout the paper we slightly adjust the original definitions of the primitives described in this section. Also, for all primitives we assume that a group description is a public parameter and is given as input to a setup algorithm.

For all definitions we consider the following. Let $\mathcal{G}$ be a bilinear group generator that takes as input a security parameter 1^λ and outputs a description of bilinear groups $\Lambda = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G_1, G_2)$. Let $\mathcal{SK}, \mathcal{X}, \mathcal{Y}, \mathcal{P}$ be the secret key space, public key space, domain, range and a proof space, respectively.

A.1 Pseudorandom Functions

Pseudorandom functions (PRF) were introduced in [34]. Below we give an adaptation of the original definition to the notation used in this paper.

Definition 11 (Pseudorandom Function). *A function family $F : \mathcal{SK} \times \mathcal{X} \to \mathcal{Y}$ is called a pseudorandom function (PRF) if there are probabilistic PT algorithms* Setup *and* KeyGen *and a deterministic PT algorithm* Comp *such that:*

- $CP \xleftarrow{\$} \mathsf{Setup}(\Lambda)$ *is an algorithm that takes as input a group description Λ and outputs the common parameters CP.*
- $SK \xleftarrow{\$} \mathsf{KeyGen}(CP)$ *is an algorithm that takes as input the common parameters CP and outputs a (secret) key $SK \in \mathcal{SK}$.*
- $Y \leftarrow \mathsf{Comp}(X, SK)$ *is a deterministic algorithm that takes as input $X \in \mathcal{X}$ and $SK \in \mathcal{SK}$ and outputs the function value $Y = F_{SK}(X) \in \mathcal{Y}$.*

The following property is required from a PRF:
Pseudorandomness: *For all probabilistic PT distinguishers $\mathcal{D} = (\mathcal{D}_1, \mathcal{D}_2)$ we have*

$$\mathsf{Pr}\left[\begin{array}{l} CP \xleftarrow{\$} \mathsf{Setup}(\Lambda)\,;\ SK \xleftarrow{\$} \mathsf{KeyGen}(CP)\,; \\ (X, st) \leftarrow {\mathcal{D}_1}^{\mathsf{Comp}(\cdot, SK)}(CP)\,; \\ Y_{(0)} \leftarrow \mathsf{F}_{SK}(X)\,;\ Y_{(1)} \xleftarrow{\$} \mathcal{Y}\,;\ b \xleftarrow{\$} \{0,1\}\,; \\ b' \xleftarrow{\$} {\mathcal{D}_2}^{\mathsf{Comp}(\cdot, SK)}(Y_{(b)}, st) \end{array} \middle| \; b = b' \wedge X \notin S \right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)\ ,$$

where S is the set of queries to the oracle Comp.

A.2 Deterministic Encryption

Deterministic encryption was introduced by was introduced by Bellare, Boldyreva, and ONeill in [8]. Here we provide a slightly adapted definition of DE.

Definition 12 (Deterministic Encryption). *A function family $F : \mathcal{PK} \times \mathcal{X} \to \mathcal{Y}$ is called a structure-preserving deterministic encryption (SPDE) if there are probabilistic PT algorithms* Setup *and* KeyGen*, and a deterministic PT algorithms* Enc *amd* Dec *such that:*

- $CP \xleftarrow{\$} \mathsf{Setup}(\Lambda)$ *is a probabilistic algorithm that takes as input the security parameter and outputs the common parameters CP that consists of the group description $\Lambda = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G_1, G_2)$ generated by $\mathcal{G}(1^\lambda)$ and possibly also constants in $\mathbb{Z}_p$.*
- $(PK, SK) \xleftarrow{\$} \mathsf{KeyGen}(CP)$ *is a probabilistic key generation algorithm that takes as input the common parameters and outputs a public key PK and a secret key SK. It is assumed without loss of generality that SK includes PK.*
- $Y \leftarrow \mathsf{Enc}(X, PK)$ *is a deterministic algorithm that takes as input $X \in \mathcal{X}$ and a public key PK and outputs a ciphertext $Y \in \mathcal{Y}$.*
- $X \leftarrow \mathsf{Dec}(Y, SK)$ *is a deterministic algorithm that takes as input a ciphertext $Y \in \mathcal{Y}$ and a secret key SK and outputs a plaintext $X \in \mathcal{X}$.*

Intuitively, the security notion for deterministic encryption, called a PRIV game, that was introduced in [8], states that it should be hard to guess any public key independent information of a list of messages given their encryptions, as long as the list has component-wise high min-entropy. Or, in other words, the adversary should not be able to distinguish ciphertexts that correspond to messages that come from two message distributions with high min-entropy. We refer the reader to [8] for the formal definition of the game.