Adaptive-Secure VRFs with Shorter Keys from Static Assumptions

Răzvan Roşie

ENS, CNRS, INRIA and PSL Research University, Paris, France
razvan.rosie@ens.fr

Abstract. Verifiable random functions are pseudorandom functions producing publicly verifiable proofs for their outputs, allowing for efficient checks of the correctness of their computation. In this work, we introduce a new computational hypothesis, the n-Eigen-Value assumption, which can be seen as a relaxation of the U_n-MDDH assumption, and prove its equivalence with the n-Rank assumption. Based on the newly introduced computational hypothesis, we build the core of a verifiable random function having an exponentially large input space and reaching adaptive security under a static assumption. The final construction achieves shorter public and secret keys compared to the existing schemes reaching the same properties.

Keywords: verifiable random function, Matrix-DDH, eigenvalue assumption.

1 Introduction

The notion of a pseudorandom function (PRF), introduced in the seminal work of Goldreich, Goldwasser and Micali [GGM86], is a foundational building block in theoretical cryptography. A PRF is a *keyed* functionality guaranteeing the randomness of its output under various assumptions. PRFs found applications in the construction of both symmetric and public-key primitives. Since the inception of their investigation, various number-theoretical constructions targeted efficiency [NR04,Lys02] or enhancing the security guarantees [BMR10]. Recent developments of PRFs include works on key-homomorphic PRFs [BLMR13,BPR12,BV15] or functional PRFs and their variants [BGI14,SW14].

A stronger, related and theoretically relevant concept, the notion of verifiable random function (VRF), has been introduced by Micali, Rabin and Vadhan [MRV99] in 1999. A verifiable random function behaves similarly to its simpler pseudorandom counterpart, but in addition to its output value y, it also creates publicly verifiable proofs π allowing efficient verification of the correctness of the computation. Among their proposed applications, Micali and Rivest mention simple, non-interactive lottery systems [MR02], while Micali and Reyzin found applications in reducing the number of rounds to 3 in zero-knowledge proofs [MR04].

More recently, VRFs found more *practical applications* in preventing zone enumeration attacks against the DNS Security Extensions (DNSSEC). Originally, the Domain Name System (DNS) was not designed to handle denial-of-service (DOS) attacks, an issue DNSSEC attempts to solve. However, so called *zone enumeration* attacks are still possible even against DNSSEC. To give a flavour, a query for the IP corresponding to the domain `www.mddh.com` can be answered positively by the zone server, which also signs the mapping between the domain name and the IP. Similar queries (i.e. for the IP of `eigenvalue.mddh.com`) may be answered negatively. Negative queries are the sensitive issues related to DNSSEC: trivial solutions of replying to every single query or issuing non-existence message for all lexicographical combinations are simply not feasible. RFC4034 (NSEC3) [LMR+05] tackles this problem by preparing a lexicographically ordered array of the names under the current zone and providing the "neighbours" for queries with negative replies. Still, in such a way, attackers can fully learn the (names, IP) pairs under the zone. The work of Goldberg et al. [GNP+15] proposes a new solution (NSEC5) that prevents zone-enumeration attacks in DNSSEC. Preventing zero-enumeration attacks follows from the *privacy* property of the NSEC5 construction, which is achieved by the means of a VRF.

The first construction of a VRFs was introduced in [MRV99] and was based on the RSA $s(k)$-hardness assumption, obtaining a scheme with unrestricted input lengths. Since then, various works targeted constructions of VRFs that

accomplish (1) adaptive security [HW10], (2) security under standard assumption [DY05] and (3) exponentially large input spaces [ACF09,HW10]. However, the realization of VRFs that simultaneously achieve these three requirements, but without having to rely on a q-type assumption has been proven a difficult task until the recent work of Hofheinz and Jager [HJ16]. The impeding constraint in achieving adaptive security resides in the lack of techniques for removing the q-type assumptions from the security proofs. As described by [Che06], q-type assumptions get stronger with the increase of the parameter q. Recently, an interesting, lattice-based approach has been recently proposed in [Yam17]: although the computational hypothesis used is still not static, we point out the reduced sizes of the proofs and secret keys obtained in the constructions introduced in [Yam17]. This work targets VRFs under static assumptions using the framework introduced in [HJ16].

The VRF by Hofheinz and Jager. The core of the construction by Hofheinz and Jager is inspired by the scheme introduced by Lysyanskaya in [Lys02], where the output corresponding to $x \in \{0,1\}^k$ is defined as $y = g^{\prod_{i=1}^{k} a_{i,x_i}}$. The novel technique presented in [HJ16] consists in replacing the set of (uniform) unidimensional exponents $a_{i,0/1}$ with matrix exponents. The crux point is to benefit from the algebraic properties enabled by the chain of matrix multiplications (linear maps in this case) in order to remove the need for q-type assumptions. The full realization of the VRF in [HJ16] involves an extra *generic* step of post-processing the vectorial output of the matrix construction via a final multiplication with a randomness extractor. The size of matrices and the issued proofs (proportional to the binary length of the input) constitutes the main downside of the construction introduced by Hofheinz and Jager. Therefore, an interesting open problems resides in obtaining constructions with shorter parameters or proofs.

Our contributions. A first contribution is the introduction of a novel computational assumption (the n-Eigen-Value assumption), that we prove equivalent to the n-Rank assumption (Section 3). Informally, it states that given an encoding of a uniform $n \times n$ matrix having (at least) one eigenvalue in $\mathbb{Z}_p$, a computationally bounded adversary cannot distinguish between an encoding of its eigenvalue ($[\lambda] := g^{\lambda}$) and an encoding of a uniformly sampled element. Finally, we provide a VRF based on the framework introduced in [HJ16] and prove its adaptive security under the aforementioned assumption. In essence, we adopt the same methodology of obtaining the output through matrix multiplications. In terms of comparison with the previous construction, we are able to reduce the size of the keys, by eliminating n elements from each pair of $n \times n$ matrix needed. For efficient implementation over bilinear maps, where the dimension of the matrices is 3×3, the result translates in reducing the size of the *secret* and *verification* keys by a factor of one sixth.

Our technique. We give a brief overview of the proof technique we use and explain the difficulty encountered while attempting to reduce the dimensions of the matrix-based keys even with a constant factor. For brevity, consider a simplified version of our construction having the output generated as: $g^{\mathbf{v}^\top} = g^{\mathbf{u}^\top \prod_{i=1}^{k} (\mathbf{M}_i - \mathbf{P}_{i,x_i})}$. The crucial property we want to achieve is a linear mapping between $\mathbf{v}^{(j-1)} = \mathbf{u}^\top \cdot \prod_{i=1}^{j-1} (\mathbf{M}_i - \mathbf{P}_{i,x_i})$ and $\mathbf{v}^{(j)} = \mathbf{u}^\top \cdot \prod_{i=1}^{j} (\mathbf{M}_i - \mathbf{P}_{i,x_i})$ through the means of $\mathbf{M}_j - \mathbf{P}_{j,x_j}$. If $\mathbf{v}^{(j-1)}$ belongs to a pre-established subspace $\mathscr{S}_{j-1}$, we intend to map it to $\mathscr{S}_j$ by multiplication with $\mathbf{M}_j - \mathbf{P}_{j,x_j}$; otherwise ($\mathbf{v}^{(j-1)} \notin \mathscr{S}_{j-1}$), the matrix multiplication should guarantee that $\mathbf{v}^{(j)} \notin \mathscr{S}_j$. To ensure this, we consider an **orthogonal component** $\mathbf{n} \perp \mathscr{S}_{j-1}$ and attempt to "hide" it in $\mathbf{P}_{j,0}$ or $\mathbf{P}_{j,1}$ (in fact in their difference depending on the challenge $X^{(0)}$). Thus if $\mathbf{v}^{(j-1)} \in \mathscr{S}_{j-1}$, then the inner-product $\mathbf{n}^\top \cdot \mathbf{v}^{(j-1)} = 0$ (and $\neq 0$ for $\mathbf{v}^{(j-1)} \notin \mathscr{S}_{j-1}$).

The second idea that we will apply is to consider a representation of $\mathbf{n}$ where its last entry is 1; stated differently $\mathbf{n}^\top = (\alpha_1, \ldots, \alpha_{n-1}, 1)$. Setting the last component to 1 enables us to save a row per index in the matrix construction we provide. In some sense, the main difficulty in going beyond one row elimination consists in the number of constant components we can obtain in $\mathbf{n}$. The basic construction is presented in Section 4, while an adaptive construction is given in Section 5. Finally, we simply get the VRF through the generic transform introduced in [HJ16] (stated in Theorem 3, Appendix A.3).

2 Preliminaries

We denote by $s \xleftarrow{\$} S$ the fact that s is picked uniformly at random from a finite set S. Variables in bold capital letters stand for matrices (e.g. $\mathbf{M}$) while bold lowercase letters represent vectors (e.g. $\mathbf{u}$). A subscript i on a vector $\mathbf{u}$ (e.g. $\mathbf{u}_i$) stands for the i-th component of the vector. An analogue convention is used for matrices. By $[a] := g^a$ we denote

the "encoding of an element" w.r.t. a group generator $g \in \mathbb{G}$, while through $[\mathbf{M}]$ and $[\mathbf{u}]$, we denote the encodings of a matrix, respectively vector. $\overline{\mathbf{W}}$ denotes the matrix formed by the top $n-1$ rows of a matrix $\mathbf{W}$ of size $n \times n$. When working with a family of vectors $\mathbf{v}$, we use the upper script to distinguish between them: $\mathbf{v}^{(0)}, \mathbf{v}^{(1)}, \ldots$. We abuse notation and extend it to bilinear maps by writing $e([\mathbf{A}], [\mathbf{B}]) = e(g,g)^{\mathbf{A}\cdot\mathbf{B}} = [\mathbf{A}\cdot\mathbf{B}]$ to denote the matrix obtained after multiplying the exponents and getting as a result the pairing of entries. By $C(\mathbf{A})$, we denote the columnspace of a matrix $\mathbf{A}$, and by $C(\mathbf{A}^\top)$, we denote its rowspace. We denote the security parameter by $\lambda \in \mathbb{N}$ and we assume it is given to all algorithms in the unary representation 1^λ. We regard an algorithm as being randomized (unless stated) and being modeled by stateless Turing machine. PPT as usual stands for "probabilistic polynomial-time." Given a randomized algorithm $\mathscr{A}$ we denote the action of running $\mathscr{A}$ on input(s) $(1^\lambda, x_1, \ldots)$ with uniform random coins r and assigning the output(s) to $(y_1, \ldots)$ by $(y_1, \ldots) \xleftarrow{\$} \mathscr{A}(1^\lambda, x_1, \ldots; r)$. We denote the set of all negligible functions by NEGL. With $\bar{x} \prec x$, we denote a bitstring prefix.

2.1 Definitions

We recall the standard definition of a VRF and the novel notion of verifiable vector hash function from [HJ16]. Certified Bilinear Group Generators are presented in Appendix A.1.

Definition 1 (Verifiable Random Function). *A* verifiable random function *consists of a tuple of three PPT algorithms* $\mathsf{VRF} := (\mathsf{VRF.Gen}, \mathsf{VRF.Eval}, \mathsf{VRF.Vfy})$*, defined as:*

- $\mathsf{VRF.Gen}$ *takes as input the security parameter* 1^λ *(in unary), and outputs a secret key* sk *together with a verification key* vk*;*
- $\mathsf{VRF.Eval}$ *takes as inputs a secret key* sk *and a string* $X \in \{0,1\}^k$ *and outputs a function value* $Y \in \mathscr{Y}$ *(where* $\mathscr{Y}$ *is a finite set) and a proof* π*;*
- $\mathsf{VRF.Vfy}$ *takes as inputs a verification key* vk*, a string* $X \in \{0,1\}^k$*, a value* $Y \in \mathscr{Y}$*, and a proof* π*, and outputs a bit.*

Moreover, we require the following three properties to hold:

- Correctness*:* $\Pr\left[\mathsf{VRF.Vfy}(vk, Y, X, \pi) = 1 \;\middle|\; \begin{array}{l}(sk, vk) \xleftarrow{\$} \mathsf{VRF.Gen}(1^\lambda) \\ (Y, \pi) \xleftarrow{\$} \mathsf{VRF.Eval}(sk, X)\end{array}\right] = 1$
- Unique Provability*: for any* (vk, sk) *(even maliciously generated) and any* $X \in \{0,1\}^k$*, there* does not exist *any* (Y_0, π_0, Y_1, π_1) *such that* $Y_0 \neq Y_1$ *and* $\mathsf{VRF.Vfy}(vk, X, Y_0, \pi_0) = \mathsf{VRF.Vfy}(vk, X, Y_1, \pi_1) = 1$
- Pseudorandomness*: for any PPT adversary* $\mathscr{A} = (\mathscr{A}_0, \mathscr{A}_1)$*, its advantage:*

$$\mathbf{Adv}^{\mathsf{vrf}}_{\mathsf{VRF}}(\mathscr{A}, \lambda) := 2 \cdot \Pr\left[\mathsf{ExpVRF}^{\mathscr{A}}_{\mathsf{VRF}}(\lambda) = 1\right] - 1 \in \mathrm{NEGL}$$

is negligible, where $\mathsf{ExpVRF}^{\mathscr{A}}_{\mathsf{VRF}}$ *is defined in Figure 1, and where the adversary never repeats a query twice (in particular, it cannot query the challenge* X^* *to the oracle* $\mathcal{O}$*). We define the security experiments for both* selective *and* adaptive *security — selective security being similar except the challenge* X^* *is chosen by the adversary before seeing a verification key.*

Vector hash functions (VHFs) are extensions of programmable hash functions [HK08], with outputs represented vectorially. Programmable hash functions are number theoretic hashes that work in two indistinguishable modes: using standard keys, the hash behaves "normally", but under trapdoor keys, the output follows a particular algebraic specification.

Definition 2 (Verifiable Vector Hash Function). *A* verifiable vector hash function *consists in a tuple of three algorithms* $\mathsf{VHF} = (\mathsf{VHF.Gen}, \mathsf{VHF.Eval}, \mathsf{VHF.Vfy})$ *defined as:*

- $\mathsf{VHF.Gen}$ *takes as input a certified bilinear group* Π *for a security parameter* 1^λ*, and outputs an evaluation key and a verification key* (ek, vk)*;*
- $\mathsf{VHF.Eval}$ *takes as input an evaluation key* ek *and an input* X *and outputs a vector of group encodings* $[\mathbf{v}] \in \mathbb{G}^n$ *together with a proof of correctness* $\pi \in \{0,1\}^*$*;*
- $\mathsf{VHF.Vfy}$ *takes as input a verification key* vk*, an input* X*, a vector* $[\mathbf{v}]$*, and a proof* π*, and outputs a bit* b*,*

$\underline{\mathsf{ExpVRF}^{\mathscr{A}}_{\mathsf{VRF}}(1^\lambda)}$:
$\boxed{(X^*, st) \xleftarrow{\$} \mathscr{A}_0(1^\lambda)}$
$(sk, vk) \xleftarrow{\$} \mathsf{VRF.Gen}(1^\lambda)$
$b \xleftarrow{\$} \{0,1\}$
$\boxed{\boxed{(X^*, st) \xleftarrow{\$} \mathscr{A}_0^{\mathcal{O}(\cdot)}(vk)}}$
$Y^* \leftarrow \mathsf{Chal}(X^*, b)$
$b' \xleftarrow{\$} \mathscr{A}_1^{\mathcal{O}(\cdot)}(st, Y^*)$
Return $b = b'$

$\underline{\mathcal{O}(X)}$:
$(Y, \pi) \leftarrow \mathsf{VRF.Eval}(sk, X)$
Return (Y, π)

$\underline{\mathsf{Chal}(X^*, b)}$:
If $b = 0$ then
$(Y^*, \pi) \leftarrow \mathsf{VRF.Eval}(sk, X^*)$
Else
$Y^* \xleftarrow{\$} \mathscr{Y}$
Return Y^*

$\underline{\boxed{\mathsf{Sel}}\,\mathsf{TrapInd}^{\mathscr{A}}_{\mathsf{VHF}}(\lambda)}$:
$b \xleftarrow{\$} \{0,1\}$
$\Pi \xleftarrow{\$} \mathsf{GrpGen}(1^\lambda)$
$\boxed{(X^{(0)}, st) \xleftarrow{\$} \mathscr{A}_0(1^\lambda)}$
$(vk_0, ek_0) \xleftarrow{\$} \mathsf{VHF.Gen}(\Pi)$
$\mathbf{B} \xleftarrow{\$} \mathrm{GL}_n(\mathbb{Z}_p)$
$(vk_1, ek_1) \xleftarrow{\$} \mathsf{VHF.TrapGen}(\Pi, [\mathbf{B}], \boxed{X^{(0)}})$
$b' \xleftarrow{\$} \mathscr{A}^{\mathcal{O}_b(\cdot), \boxed{\boxed{\mathcal{O}_{\mathsf{check}}(\cdot)}}}(st, vk_b)$
Return $b = b'$

$\underline{\mathcal{O}_0(X)}$:
$([\mathbf{v}], \pi) \xleftarrow{\$} \mathsf{VHF.Eval}(ek_0, X)$
Return $([\mathbf{v}], \pi)$

$\underline{\mathcal{O}_1(X)}$:
$([\boldsymbol{\beta}], \pi) \leftarrow \mathsf{VHF.TrapEval}(ek_1, X)$
$[\mathbf{v}] := [\mathbf{B}] \cdot [\boldsymbol{\beta}]$
Return $([\mathbf{v}], \pi)$

$\underline{\mathcal{O}_{\mathsf{check}}(X)}$:
$([\boldsymbol{\beta}], \pi) \leftarrow$
$\mathsf{VHF.TrapEval}(ek_1, X)$
$(\beta_1, \ldots, \beta_n) := \boldsymbol{\beta}$
If $\beta_n \neq 0$ then Return 1
Else Return 0

Fig. 1: The experiment (game) defining the pseudorandomness of a VRF (top). The experiment and oracles defining indistinguishability for selective/adaptive programmable VHFs (down). A boxed value is included in the selective game, double-boxed in the adaptive game.

and such that they satisfy the following two properties, termed correctness *and* unique provability*:*

– *For* ***correctness****, we require that:*

$$\Pr\left[\mathsf{VHF.Vfy}(vk, [\mathbf{v}], X, \pi) = 1; \left| \begin{array}{l} \Pi \xleftarrow{\$} \mathsf{GrpGen}(1^\lambda) \wedge \\ (vk, ek) \leftarrow \mathsf{VHF.Gen}(\Pi) \wedge \\ ([\mathbf{v}], \pi) \leftarrow \mathsf{VHF.Eval}(ek, X) \end{array} \right.\right] = 1\,. \tag{1}$$

– ***Unique provability*** *requires that, for any verification key vk and any input X, there does not exist any tuple $([\mathbf{v}]_1, \pi_{\mathbf{v}}, [\mathbf{w}], \pi_{\mathbf{w}})$ with $[\mathbf{v}] \neq [\mathbf{w}]$ and such that:*

$$\mathsf{VHF.Vfy}(vk, X, [\mathbf{v}], \pi_{\mathbf{v}}) = \mathsf{VHF.Vfy}(vk, X, [\mathbf{w}], \pi_{\mathbf{w}}) = 1\,. \tag{2}$$

We require that a VHF reaches selective/adaptive programmability with respect to trapdoor procedures (to be used in security proofs) as introduced in [HJ16]. We also denote them as PVHFs (programmable VHFs). The adaptive programmability is similar and is defined in Definition 4.

Definition 3 (Selective Programmability). *A verifiable vector hash fuction* VHF = (VHF.Gen, VHF.Eval, VHF.Vfy) *is* selectively programmable *if there exist two PPT algorithms* (VHF.TrapGen, VHF.TrapEval)*:*

– VHF.TrapGen *takes as input $\Pi \xleftarrow{\$} \mathsf{GrpGen}(1^\lambda)$, a matrix $[\mathbf{B}] \in \mathbb{G}^{n \times n}$, and $X^{(0)} \in \{0,1\}^k$ and outputs a verification key vk and a trapdoor evaluation key td;*
– VHF.TrapEval *takes as input a trapdoor evaluation key and a string $X \in \{0,1\}^k$, and outputs a vector $\boldsymbol{\beta} \in \mathbb{Z}_p^n$ and a proof $\pi \in \{0,1\}^*$,*

such that the following three properties hold:

– Correctness*:*

$$\Pr\left[\mathsf{VHF.Vfy}(vk,[\mathbf{v}],X,\pi)=1 \;\middle|\; \begin{array}{l}\Pi \xleftarrow{\$} \mathsf{GrpGen}(1^\lambda)\wedge\\ (vk,td)\xleftarrow{\$}\mathsf{VHF.TrapGen}(\Pi,[\mathbf{B}],X^{(0)})\wedge\\ (\boldsymbol{\beta},\pi)\xleftarrow{\$}\mathsf{VHF.TrapEval}(td,X)\wedge[\mathbf{v}]:=[\mathbf{B}]\cdot\boldsymbol{\beta}\end{array}\right]=1 \tag{3}$$

– Well-Distributed Outputs*: for $q=q(\lambda)$ a polynomial, there exists a polynomial* poly *such that for any $X^{(0)},X^{(1)},\ldots,X^{(q)}\in(\{0,1\}^k)^{q+1}$ with $X^{(i)}\neq X^{(0)}$ for all $i\in\{1,\ldots,q\}$, we have:*

$$\Pr\left[\begin{array}{l}\beta_n^{(0)}\neq 0\wedge\beta_n^{(i)}=0;\\ \forall i=1,\ldots,q\end{array}\;\middle|\; \begin{array}{l}\Pi\xleftarrow{\$}\mathsf{GrpGen}(1^\lambda)\wedge\mathbf{B}\xleftarrow{\$}\mathrm{GL}_n(\mathbb{Z}_p)\wedge\\ (vk,td)\xleftarrow{\$}\mathsf{VHF.TrapGen}(\Pi,[\mathbf{B}],X^{(0)})\wedge\\ (\boldsymbol{\beta}^{(i)},\pi)\xleftarrow{\$}\mathsf{VHF.TrapEval}(td,X^{(i)})\end{array}\right]\geq\frac{1}{poly(k)} \tag{4}$$

where $\beta_n^{(i)}$ denotes the n-th component of $\boldsymbol{\beta}^{(i)}$.

– Indistinguishability: *for any PPT adversary $\mathscr{A}=(\mathscr{A}_0,\mathscr{A}_1)$, its advantage:*

$$\mathbf{Adv}^{\mathsf{sel\text{-}ind\text{-}vhf}}_{\mathsf{VHF}}(\mathscr{A},\lambda):=2\cdot\Pr\left[\mathsf{SelTrapInd}^{\mathscr{A}}_{\mathsf{VHF}}(\lambda)=1\right]-1$$

is negligible, where experiment $\mathsf{SelTrapInd}_{\mathsf{VHF}}$ *is defined in Figure 1. Intuitively, we require that verification keys generated using* VHF.Gen *are indistinguishable from those generated using* VHF.TrapGen.

Definition 4 (Adaptive Programmability). *A verifiable vector hash function* VHF = (VHF.Gen, VHF.Eval, VHF.Vfy) *is* adaptively *programmable, if algorithms* (VHF.TrapGen, VHF.TrapEval)*, defined as above except that* VHF.TrapGen *takes as input only* Π *and* $[\mathbf{B}]$ *(and no longer an input $X^{(0)}$), exist, and satisfy the above correctness, well-distribution, and such that the advantage of any PPT adversary $\mathscr{A}$, defined as:*

$$\mathbf{Adv}^{\mathsf{ind\text{-}vhf}}_{\mathsf{VHF}}(\mathscr{A},\lambda):=2\cdot\Pr\left[\mathsf{TrapInd}^{\mathscr{A}}_{\mathsf{VHF}}(\lambda)=1\right]-1\ ,$$

is negligible, where experiment $\mathsf{SelTrapInd}_{\mathsf{VHF}}$ *is defined in Figure 1.*

3 The Eigen Value Assumption

In this section, we propose a new assumption, termed the n-Eigen-Value assumption, which we prove to be equivalent to the n-Rank assumption, whose definition is also recalled below. The purpose of this assumption is to offer more flexibility than the n-Rank assumption; we then use the n-Eigen-Value to prove the security of our construction (which then holds under the standard n-Rank assumption). Before giving the formal definition of our assumption, let us recall the definition of the n-Rank assumption in a group $\mathbb{G}$:

Definition 5 (n-Rank Assumption). *Let $\mathbf{M}_i$ denote a $n\times n$ matrix ($n\geq 2$) of rank i sampled uniformly at random from $\{\mathbf{W}|\mathbf{W}\in\mathbb{Z}_p^{n\times n}\wedge\mathrm{RANK}(\mathbf{W})=i\}$. Let $\mathscr{A}$ be any* PPT *adversary. Then the advantage of $\mathscr{A}$ against the n-Rank problem in a group $\mathbb{G}$, defined as:*

$$\mathbf{Adv}^{n\text{-}\mathsf{rank}}_{\mathbb{G}}(\mathscr{A},\lambda):=\Pr\left[\mathscr{A}\left([\mathbf{M}_n],1^\lambda\right)=1\right]-\Pr\left[\mathscr{A}\left([\mathbf{M}_{n-1}],1^\lambda\right)=1\right]\ ,$$

is negligible.

Remark 1 (Computational vs Decisional n-Rank). Note that Definition 5 implicitly makes the n-Rank assumption "computational": distinguishing between the two ranks implicitly gives the rank of the matrix.

The natural n-Eigen-Value counterpart of the n-Rank assumption, roughly saying that $\left([\mathbf{M}], [\lambda]\right) \approx_c \left([\mathbf{M}], [\$]\right)$ — the encoding of an eigenvalue for a randomly sampled matrix $\mathbf{M}$ is indistinguishable from the encoding of a random element, needs a more careful definitional setting. This happens because not every $\mathbf{M}$ defined over $\mathbb{Z}_p^{n\times n}$ has its eigenvalues belonging to $\mathbb{Z}_p$. Concretely, the (monic) characteristic polynomial of $\mathbf{M}$ is irreducible with probability:

$$\mu = \frac{1}{n} + \mathbf{O}\left(p^{-n/2} \cdot \sqrt{n}\right). \tag{5}$$

For large p, μ approaches $1/n$. Thus, the probability that a random matrix $\mathbf{M}$ has eigenvalues in $\mathbb{Z}_p$ is simply $1-1/n$. Still, practical applications would benefit for small values of n, and the aforementioned probability in Equation (5) is significant, so we employ a different strategy.

Based on the previous observation, we now define the n-Eigen-Value assumption as follows: let $\mathbf{A}, \mathbf{B}$ be two matrices of ranks respectively $n-1, n$, and $\mathbf{L}$ be randomly sampled from $\mathbb{Z}_p^{n\times n}$ (which has rank n with overwhelming probability). We set $\mathbf{M}_{n-1} \leftarrow \mathbf{L}\cdot\mathbf{A} + \lambda\cdot\mathbf{I}_n$ and $\mathbf{M}_n \leftarrow \mathbf{B}$. We now claim that the two distributions — $\left([\mathbf{M}_{n-1}], [\lambda]\right)$ and $\left([\mathbf{M}_n], [\$]\right)$ — are indeed computationally indistinguishable.

Definition 6 (n-Eigen-Value)**.** *Let* $\mathbf{A} \xleftarrow{\$} \left\{\mathbf{W} \middle| \mathbf{W} \in \mathbb{Z}_p^{n\times n} \wedge \mathit{rank}(\mathbf{W}) = n-1\right\}$, $\mathbf{B} \xleftarrow{\$} \mathbb{Z}_p^{n\times n}$ *and* $\mathbf{L} \xleftarrow{\$} \mathrm{GL}_n(\mathbb{Z}_p)$. *Let* $\mathscr{A}$ *be a* PPT *algorithm. The advantage of any* PPT *adversary* $\mathscr{A}$ *against the* n-Eigen-Value *problem in a group* $\mathbb{G}$*, defined as:*

$$\mathbf{Adv}_{\mathbb{G}}^{n\text{-ev}}(\mathscr{A}, \lambda) := \Pr\left[\mathscr{A}\left([\mathbf{M}], [\lambda], 1^\lambda\right) = 1\right] - \Pr\left[\mathscr{A}\left([\mathbf{N}], [r], 1^\lambda\right) = 1\right],$$

is negligible, where $\mathbf{M} \leftarrow \mathbf{L}\cdot\mathbf{A} + \lambda\cdot\mathbf{I}_n$, $\mathbf{N} \leftarrow \mathbf{B}$ *and* $r \xleftarrow{\$} \mathbb{Z}_p$.

Theorem 1. *The* n-Rank *assumption holds in* $\mathbb{G}$ *if and only if the* n-Eigen-Value *assumption holds in* $\mathbb{G}$.

Proof. We prove this statement by proving both implications separately.

n-Rank $\Rightarrow$ n-Eigen-Value. Let $\mathscr{A}$ be an adversary against the n-Eigen-Value problem in $\mathbb{G}$. We then build an adversary $\mathscr{B}$ against the n-Rank problem in $\mathbb{G}$ as follows: $\mathscr{B}$ is given a matrix of group elements $[\mathbf{M}]$ with $\mathbf{M}$ being of rank $n-1$ or n. It then picks uniformly at random an invertible matrix $\mathbf{L} \xleftarrow{\$} \mathrm{GL}_n(\mathbb{Z}_p)$, a scalar $\lambda \xleftarrow{\$} \mathbb{Z}_p$ and computes $[\mathbf{B}] = [\mathbf{M}\cdot\mathbf{L} + \lambda\cdot\mathbf{I}_n]$. Finally, it sends $([\mathbf{B}], [\lambda])$ to $\mathscr{A}$. When $\mathscr{A}$ halts with some bit b, $\mathscr{B}$ outputs b.

Then, assuming $\mathbf{M}$ is a rank n matrix, $\det(\mathbf{B} - \lambda\cdot\mathbf{I}_n) = \det(\mathbf{M}\cdot\mathbf{L}) \neq 0$, since $\mathbf{M}$ and $\mathbf{L}$ are rank n matrices, and then λ is not an eigenvalue of $\mathbf{B}$, and $\mathscr{B}$ simulates precisely the setting $\left([\mathbf{B}], [r]\right)$ (with overwhelming probability). Now, if $\mathbf{M}$ is a rank $n-1$ matrix, so is $\mathbf{M}\cdot\mathbf{L}$, and then $\det(\mathbf{B} - \lambda\cdot\mathbf{I}_n) = 0$, which implies that λ is an eigen value of $\mathbf{B}$. Then, $\mathscr{B}$ simulates exactly the setting $\left([\mathbf{B}], [\lambda]\right)$.

n-Eigen-Value $\Rightarrow$ n-Rank. Let now $\mathscr{A}$ denote an adversary against the n-Rank problem in $\mathbb{G}$. Then we build an adversary $\mathscr{B}$ against the n-Eigen-Value problem in $\mathbb{G}$ as follows: $\mathscr{B}$ is given a tuple $([\mathbf{M}], [\lambda])$ with λ being either random or an eigenvalue of $\mathbf{M}$ (for the second case, $\mathbf{M} = \mathbf{L}\cdot\mathbf{A} + \lambda\cdot\mathbf{I}_n$). Then, $\mathscr{B}$ simply computes $[\mathbf{M} - \lambda\cdot\mathbf{I}_n]$ and sends it to $\mathscr{A}$. When $\mathscr{A}$ halts with some bit b, $\mathscr{B}$ outputs b.

Indeed, if λ is not an eigenvalue and $\mathbf{M}$ is uniformly sampled, $\mathbf{M} - \lambda\cdot\mathbf{I}_n$ is just a uniform matrix in $\mathbb{Z}_p^{n\times n}$, which is then of rank n with overwhelming probability, and $\mathscr{B}$ simulates correctly the case where the input matrix for the n-Rank adversary has rank n. However, if λ is indeed an eigenvalue, then $\det(\mathbf{M} - \lambda\cdot\mathbf{I}_n) = 0$, which implies that $\mathbf{M} - \lambda\cdot\mathbf{I}_n$ has rank at most $n-1$ (rank $n-1$ with overwhelming probability). In this case, $\mathscr{B}$ simulates the case where the input matrix to the n-Rank solver has rank $n-1$.

This concludes the proof of Theorem 1. □

4 A new programmable vector hash function construction

Verifiable random functions can be build generically on top of programmable verifiable vector hash functions. We recall their transform allowing to obtain PVHF-based VRFs in Appendix A.3. In what follows, we contribute by introducing a new PVHF construction with smaller public and secret keys compared to the construction proposed in [HJ16], and relying on a new technique.

4.1 A new PVHF construction

For $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ a symmetric pairing with $|\mathbb{G}| = p$ (p prime), let $\mathscr{D}_0$ and $\mathscr{D}_1$ be the following "pattern" matrix distributions:

$$\mathscr{D}_0 = \begin{pmatrix} d_{1,1} & d_{1,2} & \dots & d_{1,n} \\ 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \dots & \vdots \\ 0 & 0 & \dots & 0 \end{pmatrix} \quad \mathscr{D}_1 = \begin{pmatrix} 0 & 0 & \dots & 0 \\ d_{2,1} & d_{2,2} & \dots & d_{2,n} \\ \dots & \dots & \dots & \dots \\ d_{n-1,1} & d_{n-1,2} & \dots & d_{n-1,n} \\ 0 & 0 & \dots & 1 \end{pmatrix} \quad \text{, where } d_{i,j} \xleftarrow{\$} \mathbb{Z}_p \text{ , } i \in [n-1], j \in [n].$$

$\underline{\mathsf{VHF.Gen}(1^\lambda)}$:
$\mathbf{u} \xleftarrow{\$} \mathbb{Z}_p^n$
$(\mathbf{M}_i, \mathbf{P}_{i,0}, \mathbf{P}_{i,1}) \xleftarrow{\$} \mathbb{Z}_p^{n\times n} \times \mathscr{D}_0 \times \mathscr{D}_1, \forall i \in [k-1]$
$(\mathbf{M}_{k,0}, \mathbf{M}_{k,1}) \xleftarrow{\$} \mathbb{Z}_p^{n\times n} \times \mathbb{Z}_p^{n\times n}$
$vk := \left([\mathbf{u}], \{[\mathbf{M}_i, \mathbf{P}_{i,b}, \mathbf{M}_{k,b}]\}_{i\in[k-1], b\in\{0,1\}}\right)$
$sk := \left(\mathbf{u}, \{\mathbf{M}_i, \mathbf{P}_{i,b}, \mathbf{M}_{k,b}\}_{i\in[k-1], b\in\{0,1\}}\right)$
Return (vk, sk)

$\underline{\mathsf{VHF.Eval}(x, sk)}$:
$[\mathbf{v}] := [\mathbf{u}^\top \cdot \mathbf{M}_x]$
$\pi := \left\{[\mathbf{u}^\top \cdot \mathbf{M}_{\bar{x}}] : \forall \bar{x} \prec x\right\}$
Return $([\mathbf{v}], \pi)$

$\underline{\mathsf{VRF.Vfy}(x, vk, [y], \pi)}$:
check $e([\mathbf{u}^\top \cdot \mathbf{M}_{\bar{x}}^{(i)}], [\mathbf{1}]) \stackrel{?}{=} e([\mathbf{M}_{i,x_i}], [\mathbf{u}^\top \cdot \mathbf{M}_{\bar{x}}^{(i-1)}]), \forall i \in [k]$

Fig. 2: The underlying VHF construction, used to construct a selective PVHF for inputs of length k. We use the following notation $\mathbf{M}_x := \prod_{i=1}^{k-1} \left(\mathbf{M}_i - \mathbf{P}_{i,x_i}\right) \cdot \mathbf{M}_{k,x_k}$. We **emphasize** the representation size of $\mathbf{P}_{i,0}, \mathbf{P}_{i,1}$ is n^2-n, which translates in 6 elements for efficient construction using 3×3 matrices, thus obtaining a more efficient construction in terms of both public and secret keys when comparing to the one in [HJ16].

Overview of the construction. We define a VHF construction in Figure 2, show its correctness and unique provability, then introduce the trapdoor algorithms in Section 4.2 and prove programmability.

- The VHF.Gen algorithm of the proposed scheme generates a set of uniform matrices $\left(\mathbf{M}_j, \mathbf{P}_{j,0}, \mathbf{P}_{j,1}\right), \forall j \in [k\text{-}1]$, sampled according to the appropriate distributions. The set of "plain" matrices form the secret key sk, while the vk is set to be their encodings.
- The VHF.Eval procedure under sk, corresponding to $x \in \{0,1\}^k$ outputs:

$$\left[\mathbf{u}^\top \cdot \left(\prod_{i=1}^{k-1} \left(\mathbf{M}_i - \mathbf{P}_{i,x_i}\right)\right) \cdot \mathbf{M}_{k,x_k}\right]$$

  In some sense, our evaluation procedure is similar to the one described in [Lys02], but we use a matrix multiplication construction rather than a unidimensional product.
- The VHF.Vfy procedure checks if all the pairings of (1) the vectors constituting the proof π, with (2) the matrices forming the public key, are correct.

Fine tunning the VHF *to achieve well-distributed outputs.* We also justify the need for uniform matrices corresponding to position k. Well-distributed outputs, as formulated in Definition 3 (Figure 1), enforces the last subspace ($\mathscr{W}$) we work to be the one spanned by the first $n-1$ canonical vectors. The reason for this constraint, resides in the proof for pseudorandomness: the outcome of the trapdoor evaluation is multiplied by a (n-LIN-distributed - cf. [EHK+13], page 9) matrix $\mathbf{B}$. To cope with this subtlety, we "fine-tune" the PVHF, by doing the following change: we replace the matrices in the last position $\left(\mathbf{M}_k, \mathbf{P}_{k,0}, \mathbf{P}_{k,1}\right)$ with two "proper", random matrices $\left(\mathbf{M}_{k,0}, \mathbf{M}_{k,1}\right)$.

4.1.1 Correctness and unique provability.

Correctness (1) follows immediately from the construction. Unique provability (2) follows from the deterministic evaluation. We prove it formally below:

Lemma 1. *The* VHF *construction presented in Figure 2 achieves unique provability, according to Definition 2.*

Proof. Suppose there exists a public key vk, a strings X of length k, and a tuple $([\mathbf{v}], \pi, [\mathbf{w}], \phi)$ with $[\mathbf{v}] \neq [\mathbf{w}]$ and $[\mathbf{v}], [\mathbf{w}] \in \mathbb{G}^n$ such that: $\mathsf{VHF.Vfy}(vk, X, [\mathbf{v}], \pi) = \mathsf{VHF.Vfy}(vk, X, [\mathbf{w}], \phi) = 1$.

To prove uniqueness (2), we use of the properties of the construction. Suppose there exists a vector index $j \in [n]$ such that $[\mathbf{v}_j] \neq [\mathbf{w}_j]$. We expand $[\mathbf{v}_j] \neq [\mathbf{w}_j]$ as follows:

$$[\mathbf{v}_j] \neq [\mathbf{w}_j] \iff \left(\mathbf{u}^\top \cdot \prod_{i=1}^{k-1} (\mathbf{M}_i - \mathbf{P}_{i,x_i}) \cdot \mathbf{M}_{k,x_k}\right)_j \neq \left(\mathbf{u}^\top \cdot \prod_{i=1}^{k-1} (\mathbf{M}_i - \mathbf{P}_{i,x_i}) \cdot \mathbf{M}_{k,x_k}\right)_j \tag{6}$$

The expanded equation shows the output values are uniquely determined by the secret key and the input string x, in a fully deterministic procedure. Hence, from Equation (6), it follows the unique provability property. □

4.2 Proving Selective Programmability

As usual, provable security is achieved via a reduction to a computationally hard problem; in our case, we rely on the n-Eigen-Value assumption (detailed in Section 3), stating that $([\mathbf{A}], [\lambda]) \approx_c ([\mathbf{B}], [\$])$ — the distribution of encoded matrices with their eigenvalues is indistinguishable from the uniform distribution defined over all encoded matrices and some random group elements.

Overview. To prove selective programmability (Definition 3) w.r.t. a challenge $X^{(0)}$ for the VHF in Figure 2, we *gradually* replace the relevant rows in $\mathbf{P}_{j,0}, \mathbf{P}_{j,1}$ with vectors lying in the rowspace defined by the challenge n-Eigen-Value tuple $\left([\mathbf{A}], [e]\right)$ (i.e the eigenspace of $[\mathbf{A} - e \cdot \mathbf{I}_n]$). Among these vectors, we attempt to "hide" the orthogonality components for the rowspace defined by the matrices corresponding to position j-1 (Figure 3). We also replace $\mathbf{M}_j$ such that $\left(\mathbf{M}_j - \mathbf{P}_{j,X_j^{(0)}}\right)$ linearly maps vectors between rowspaces defined by $\left(\mathbf{M}_{j-1} - \mathbf{P}_{j-1,1-X_{j-1}^{(0)}}\right)$ and $\left(\mathbf{M}_j - \mathbf{P}_{j,1-X_j^{(0)}}\right)$. Then, given the k pairs of matrices $\{(\mathbf{M}_1 - \mathbf{P}_{1,0}, \mathbf{M}_1 - \mathbf{P}_{1,1}), \ldots, (\mathbf{M}_{k-1} - \mathbf{P}_{k-1,0}, \mathbf{M}_{k-1} - \mathbf{P}_{k-1,1}), (\mathbf{M}_{k,0}, \mathbf{M}_{k,1})\}$, the value $\mathbf{u}^\top \cdot \mathbf{M}_{X^{(0)}}$ completely determines $\mathbf{u}$ (since $\mathbf{M}_{X^{(0)}}$ has full rank w.h.p.), whereas for any $x \neq X^{(0)}$, $\mathbf{M}_x$ is a matrix of rank $n-1$ (because for some j, $\mathbf{M}_j - \mathbf{P}_{j,1-X_j^{(0)}}$ has rank $n-1$ and it's a term in the product represented by $\mathbf{M}_x$), so $\mathbf{u}^\top \cdot \mathbf{M}_x$ *loses* information about $\mathbf{u}$.

The trapdoor mode algorithms VHF.TrapGen and VHF.TrapEval are described below:

1. VHF.TrapGen sets all the matrices $(\mathbf{M}_j, \mathbf{P}_{j,0}, \mathbf{P}_{j,1})$ such that:
   – it first generates, uniformly at random, a subspace of dimension $n-1$, denoted $\mathscr{S}_0$ and a uniform $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_p^n$, which with overwhelming probability will not belong to $\mathscr{S}_0$.
   – it incrementally constructs the challenge-related matrices such that $\mathbf{M}_j - \mathbf{P}_{j,1-X^{(0)}}$ defines a rowspace of rank $n-1$, denoted $\mathscr{S}_j$. Also, $\mathbf{M}_j - \mathbf{P}_{j,X^{(0)}}$ defines a **linear map** between the subspaces $\mathscr{S}_{j-1}$ and $\mathscr{S}_j$; formally $\left(\mathbf{M}_j - \mathbf{P}_{j,X^{(0)}}\right) : \mathscr{S}_{j-1} \to \mathscr{S}_j$.

   Achieving the linear map property is more subtle, and requires a careful programming of the matrices $\mathbf{M}_j, \mathbf{P}_{j,0}, \mathbf{P}_{j,1}$. The VHF.TrapGen begins by sampling $\mathbf{L}_j \xleftarrow{\$} \mathbb{Z}_p^{n \times n}$. It also samples $\mathbf{S}_j \xleftarrow{\$} \mathbb{Z}_p^{(n-1) \times n}$ such that its rowspace defines $\mathscr{S}_j$. A **normal vector** $(\alpha_1, \ldots, \alpha_{n-1}, 1)$, orthogonal on $\mathscr{S}_{j-1}$ is also computed. Finally the matrices can be set as in Figure 3:

   The **intuition** behind this setting is given by the orthogonal component. The value of the inner product between $\mathbf{v}^{(j-1)}$ and the orthogonal component on $\mathscr{S}_{j-1}$ will enable/disable the membership of $\mathbf{v}^{(j)}$ to $\mathscr{S}_j$: a zero inner product means that $\mathbf{v}^{(j-1)} \in \mathscr{S}_{j-1}$, otherwise $\mathbf{v}^{(j-1)} \notin \mathscr{S}_j$. Finally $\mathbf{M}_{k,1-X^{(0)}}$ is sampled from $\mathscr{S}_k$, while $\mathbf{M}_{k,X^{(0)}}$ is set to map $\mathscr{S}_{k-1} \xrightarrow{C^{-1}} \mathscr{W} \xrightarrow{C'} \mathscr{S}_k$, where $\mathscr{W}$ is the subspace spanned by the first n-1 vectors from the canonical basis.
2. VHF.TrapEval behaves exactly as the VHF.Eval with the noticeable difference, that for a trapdoor algorithm, the resulting vector is multiplied with the matrix $\mathbf{B}$, as specified in Figure 1.

We now prove *well-distributed* outputs and *indistinguishability*.

$$\mathbf{P}_{j,1} = \begin{pmatrix} 0 & 0 & \dots & 0 \\ s_{2,1}^{(j)} & s_{2,2}^{(j)} & \dots & s_{2,n}^{(j)} - \alpha_2 \\ \dots & \dots & \dots & \dots \\ s_{n-1,1}^{(j)} & s_{n-1,2}^{(j)} & \dots & s_{n-1,n}^{(j)} - \alpha_{n-1} \\ 0 & 0 & \dots & -1 \end{pmatrix} \qquad \mathbf{P}_{j,0} = \begin{pmatrix} s_{1,1}^{(j)} & s_{1,2}^{(j)} & \dots & s_{1,n}^{(j)} + \alpha_1 \\ 0 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 \end{pmatrix}$$

$$\mathbf{M}_j = \mathbf{L}_j \cdot \begin{pmatrix} \mathbf{S}_j \\ \mathbf{s}_j^\top \cdot \mathbf{S}_j \end{pmatrix} + \mathbf{P}_{j,1-X_j^{(0)}}, \forall j \in [k-1]$$

$$\mathbf{M}_{k,1-X_k^{(0)}} = \mathbf{L}_k \cdot \begin{pmatrix} \mathbf{S}_k \\ \mathbf{s}_k^\top \cdot \mathbf{S}_k \end{pmatrix} \qquad \mathbf{M}_{k,1-X_k^{(0)}} = \begin{pmatrix} \mathbf{L}_k^{'} \cdot \mathbf{S}_{k-1} \\ \mathbf{r}^{'} \end{pmatrix}^{-1} \cdot \begin{pmatrix} \mathbf{L}_k^{''} \cdot \mathbf{S}_k \\ \mathbf{r}^{''} \end{pmatrix}$$

Fig. 3: Matrices $\mathbf{M}_j, \mathbf{P}_{j,0}, \mathbf{P}_{j,1}$ for $j \in [k\text{-}1]$ generated for the trapdoor mode. The matrices $\mathbf{M}_{k,0}, \mathbf{M}_{k,1}$ are generated as in [HJ16].

4.2.1 Well-distributed outputs

Lemma 2. *The* PVHF *construction presented in Figure 2 has well-distributed outputs according to Definition 3.*

Proof. The triple $\left(\mathbf{M}_j, \mathbf{P}_{j,0}, \mathbf{P}_{j,1}\right)_{j \in [k-1]}$ — as defined in Figure 3 — can be used to construct matrices with the properties enumerated below:

1. *Property 1.* $\mathbf{M}_j - \mathbf{P}_{j,X_j^{(0)}} = \mathbf{L}_j^{'} \cdot \begin{pmatrix} \mathbf{S}_j \\ \mathbf{s}^\top \cdot \mathbf{S}_j \end{pmatrix} + (-1)^{X_j^{(0)}} \cdot \mathbf{N}$, where $\mathbf{N}$ is the **zero matrix** except the last column being set to $(\alpha_1, \alpha_2, \dots, \alpha_{n-1}, 1)$. (A detailed proof is worked out in Appendix B).
2. *Property 2.* $\mathbf{M}_j - \mathbf{P}_{j,1-X_j^{(0)}} = \mathbf{L}_j \cdot \begin{pmatrix} \mathbf{S}_j \\ \mathbf{s}^\top \cdot \mathbf{S}_j \end{pmatrix}$.

An *invariant* is used to easily encapsulate the well-distributed outputs property:

> $\exists$ matrices $\mathbf{S}_1, \dots, \mathbf{S}_k \in \mathbb{Z}_p^{(n-1)\times n}$ of rank $n-1$ such that for each $x \in \{0,1\}^j$ not a prefix of $X^{(0)}$, $\mathbf{M}_x$ lies in $\mathscr{S}_j$, where $C\left(\mathbf{S}_j^\top\right) = \mathscr{S}_j$.

We demonstrate the invariant is preserved under the setting of matrices shown in Figure 3 by proving a set of simple lemmata. Essentially, the first one ensures that $\left(\mathbf{M}_j - \mathbf{P}_{j,1-X_j^{(0)}}\right) : \mathbb{Z}_p^n \to \mathscr{S}_j$. The second and third lemmas state that $\mathbf{M}_j - \mathbf{P}_{j,X_j^{(0)}}$ will map vectors from $\mathscr{S}_{j-1} \to \mathscr{S}_j$ for $j \in [k\text{-}1]$. Full proofs are given in Appendix C.

– **Lemma 3.** *Let* $\mathbf{S} \xleftarrow{\$} \mathbb{Z}_p^{(n-1)\times n}$, $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_p^n$. *Let* $\mathbf{b} \in \mathbb{Z}_p^n$. *Then* $\exists \mathbf{a} \in \mathbb{Z}_p^n$ *s.t.* $\mathbf{b}^\top \cdot \left(\mathbf{M}_j - \mathbf{P}_{j,1-X_j^{(0)}}\right) = \mathbf{a}^\top \cdot \begin{pmatrix} \mathbf{S}_j \\ \mathbf{s}_j^\top \cdot \mathbf{S}_j \end{pmatrix}$.

– **Lemma 4.** *Let* $\mathbf{S} \xleftarrow{\$} \mathbb{Z}_p^{(n-1)\times n}$ *and* $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_p^n$. *Let* $\mathbf{v} \in \mathbb{Z}_p^n$ *and* $\mathbf{b}^\top = \mathbf{v}^\top \cdot \begin{pmatrix} \mathbf{S}_{j-1} \\ \mathbf{s}_{j-1}^\top \cdot \mathbf{S}_{j-1} \end{pmatrix}$. *Then* $\exists \mathbf{a} \in \mathbb{Z}_p^n$ *such that* $\mathbf{b}^\top \cdot \left(\mathbf{M}_j - \mathbf{P}_{j,1-X_j^{(0)}}\right) = \mathbf{a}^\top \cdot \begin{pmatrix} \mathbf{S}_j \\ \mathbf{s}_j^\top \cdot \mathbf{S}_j \end{pmatrix}$.

– **Lemma 5.** *Let* $\mathbf{S} \xleftarrow{\$} \mathbb{Z}_p^{(n-1)\times n}$ *and* $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_p^n$. $\forall \mathbf{v} \in \mathbb{Z}_p^n$ *let* $\mathbf{b}^\top \neq \mathbf{v}^\top \cdot \begin{pmatrix} \mathbf{S}_{j-1} \\ \mathbf{s}_{j-1}^\top \cdot \mathbf{S}_{j-1} \end{pmatrix}$. *Then* $\forall \mathbf{a} \in \mathbb{Z}_p^n$ *we have* $\mathbf{b}^\top \cdot \left(\mathbf{M}_j - \mathbf{P}_{j,X_j^{(0)}}\right) \neq \mathbf{a}^\top \cdot \begin{pmatrix} \mathbf{S}_j \\ \mathbf{s}_j^\top \cdot \mathbf{S}_j \end{pmatrix}$.

Finally, care is needed because the special way of sampling $\mathbf{M}_{k,0}$ and $\mathbf{M}_{k,1}$. Their trapdoor form is given in Figure 3 and the argument for linear maps is the one we mentioned before. Moreover, according to the Definition 3, one needs to ensure that $\mathscr{S}_k = \mathscr{W}$, which implies a special form for $\mathbf{S}_k$ used to instantiate $\left(\mathbf{M}_{k,0}, \mathbf{M}_{k,1}\right)$. Finally, the invariant immediately follows from the lemmas, and therefore we obtain *well-distributed outputs* for the construction in Figure 2. □

4.2.2 Indistinguishability proof. The procedures VHF.Gen and VHF.TrapGen used in the indistinguishability security experiment (Definition 3) are given in Figure 4.

$\underline{\mathsf{VHF.Gen}(\Pi)}$:	$\underline{\mathsf{VHF.TrapGen}(\Pi, [\mathbf{B}], X^{(0)})}$:
$\mathbf{u} \xleftarrow{\$} \mathbb{Z}_p^n$	$\mathbf{u} \xleftarrow{\$} \mathbb{Z}_p^n$
$(\mathbf{A}_i, \mathbf{B}_{i,0}, \mathbf{B}_{i,1}) \xleftarrow{\$} \mathbb{Z}_p^{n\times n} \times \mathscr{D}_0 \times \mathscr{D}_1$, $i \in [k-1], b \in \{0,1\}$	$(\mathbf{M}_i, \mathbf{P}_{i,0}, \mathbf{P}_{i,1})$ are sampled according to Figure 10, Appendix D
$(\mathbf{D}_{k,0}, \mathbf{D}_{k,1}) \xleftarrow{\$} \mathbb{Z}_p^{n\times n} \times \mathbb{Z}_p^{n\times n}$	$(\mathbf{M}_{k,0}, \mathbf{M}_{k,1})$ are sampled according to Figure 10, Appendix D
$vk := \left([\mathbf{u}], \{[\mathbf{A}_i], [\mathbf{B}_{i,b}], [\mathbf{D}_{k,b}]\}_{i\in[k-1], b\in\{0,1\}}\right)$	$vk := \left([\mathbf{u}], \{[\mathbf{M}_i], [\mathbf{P}_{i,b}], [\mathbf{M}_{k,b}]\}_{i\in[k-1], b\in\{0,1\}}, \right)$
$sk := \left(\mathbf{u}, \{\mathbf{A}_i, \mathbf{B}_{i,b}, \mathbf{D}_{k,b}\}_{i\in[k-1], b\in\{0,1\}}\right)$	$td := \left(\mathbf{u}, \{\mathbf{M}_i, \mathbf{P}_{i,b}, \mathbf{M}_{k,b}\}_{i\in[k-1], b\in\{0,1\}}, \right)$
Return (vk, sk)	Return (vk, td)
$\underline{\mathbf{Eval}_0(x)}$:	$\underline{\mathbf{Eval}_1(x)}$:
$\mathbf{v}^\top := \mathbf{u}^\top \cdot \prod_{i=1}^{k-1} (\mathbf{A}_i - \mathbf{B}_{i,x_i}) \cdot \mathbf{D}_{k,x_k}$	$\mathbf{v}^\top := \mathbf{u}^\top \cdot \prod_{i=1}^{k-1} (\mathbf{M}_i - \mathbf{P}_{i,x_i}) \cdot \mathbf{M}_{k,x_k}$
Return $[\mathbf{v}]$	Return $[\mathbf{B} \cdot \mathbf{v}]$

Fig. 4: The VHF.Gen and VHF.TrapGen procedures and the oracles used by the indistinguishability security experiment defined in Figure 1. We provide explicit forms for the two evaluation oracles.

Theorem 2. *The* PVHF *construction presented in Figure 2 is indistinguishable according to Definition 3.*

Proof. *Intuitively*, we build the proof on transitions based on indistinguishability between **hybrid games** (Figure 5, left side). $\mathbf{Game}_0$ will correspond to the setting where the matrices used by the VHF.TrapGen are sampled via the VHF.Gen procedure. A transition between the hybrids j-1 and j consists in replacing the first j pairs of matrices in the public key with ones sampled according to VHF.TrapGen. To facilitate the transition from $\mathbf{Game}_j$ to $\mathbf{Game}_{j+1}$ we introduce an **intermediate hybrid** security experiment, defined in Figure 5 (right side). Lemmata 6, 7 are used to prove the transitions between $\mathbf{Game}_j \rightarrow \mathbf{Game}_{j,A}$ and $\mathbf{Game}_{j,A} \rightarrow \mathbf{Game}_{j+1}$.

Lemma 6. *For $j \in [k-2]$, a* PPT *adversary $\mathscr{A}$ distinguishes between $\mathbf{Game}_j$ and $\mathbf{Game}_{j,A}$ with negligible advantage:* $\Pr\left[\mathbf{Game}_j^{\mathscr{A}} = 1\right] - \Pr\left[\mathbf{Game}_{j,A}^{\mathscr{A}} = 1\right] \in \text{NEGL}$.

Proof (Lemma 6). We prove an adversary cannot detect the transition between $\mathbf{Game}_j$ and $\mathbf{Game}_{j+1}$, via the intermediate sub-hybrid game we have introduced. The transition between $\mathbf{Game}_j$ and $\mathbf{Game}_{j,A}$ relies on a statistical argument. In $\mathbf{Game}_j$, the tuple $(\mathbf{M}_{j+1}, \mathbf{P}_{j+1,0}, \mathbf{P}_{j+1,1})$ corresponding to the position j+1 are sampled uniformly from $\mathbb{Z}_p^{n\times n} \times \mathscr{D}_0 \times \mathscr{D}_1$. In $\mathbf{Game}_{j,A}$, we sample the matrices in position $j+1$ using the trapdoor distribution. The indistinguishability follows from the fact that in $\mathbf{Game}_{j,A}$ (Figure 5) , all the entries in $\mathbf{P}_{j+1,0}, \mathbf{P}_{j+1,1}$ are sampled uniformly at random; *observe* the entries in the last columns of $\mathbf{P}_{j+1,0}, \mathbf{P}_{j+1,1}$ consists in the sum (difference) of randomly sampled $r_{i,n} + \alpha_i$. Thus, the orthogonality components $(\alpha_1, \ldots, \alpha_{n-1}, 1)$ are indistinguishable from uniform elements, which implies that $\Pr\left[\mathbf{Game}_j^{\mathscr{A}} = 1\right] - \Pr\left[\mathbf{Game}_{j,A}^{\mathscr{A}} = 1\right] \in \text{NEGL}$. □

Lemma 7. *For $j \in [k\text{-}2]$, a* PPT *adversary distinguishes between $\mathbf{Game}_{j,A}$ and $\mathbf{Game}_{j+1}$ with negligible advantage:* $\Pr\left[\mathbf{Game}_{j,A}^{\mathscr{A}} = 1\right] - \Pr\left[\mathbf{Game}_{j+1}^{\mathscr{A}} = 1\right] \leq \mathbf{Adv}_{\mathbb{G}}^{n\text{-ev}}(\mathscr{A}, \lambda)$.

Proof (Lemma 7). One can observe that our trapdoor construction differs from the one introduced in [HJ16] in the sense that instead of using two matrices, we use a matrix and two vectors (when $n = 3$), in a sense aiming to compress as much info as possible in the two row vectors.

*Reduction to the n-*Eigen-Value *assumption.* We take the contrapositive. Let $\mathscr{A}$ be a PPT adversary having a non-negligible advantage in distinguishing between $\mathbf{Game}_{j,A}$ and $\mathbf{Game}_{j+1}$. We build an adversary $\mathscr{A}'$ that wins the n-Eigen-Value game with the same probability. The n-Eigen-Value game commences by sampling a bit b'. $\mathscr{A}'$ is

$\underline{\mathsf{VHF.TrapGen}(\Pi,[\mathbf{B}],X^{(0)},1^\lambda)}$: // $\mathbf{Game}_j$

$\mathbf{u} \xleftarrow{\$} \mathbb{Z}_p^n$

$(\mathbf{A}_i,\mathbf{B}_{i,0},\mathbf{B}_{i,1}) \xleftarrow{\$} \mathbb{Z}_p^{n\times n}\times\mathscr{D}_0\times\mathscr{D}_1$

$(\mathbf{M}_i,\mathbf{P}_{i,0},\mathbf{P}_{i,1}), \forall i\in[j]$ are sampled as in Figure 10

$(\mathbf{M}_{k,0},\mathbf{M}_{k,1}) \xleftarrow{\$} \mathbb{Z}_p^{n\times n}\times\mathbb{Z}_p^{n\times n}$

$vk := \Big([\mathbf{u}],\{[\mathbf{M}_i],[\mathbf{P}_{i,b}]\}_{i\in[j],b\in\{0,1\}},$
$\{[\mathbf{A}_i],[\mathbf{B}_{i,b}],[\mathbf{M}_{k,b}]\}_{i\in[k-1]-[j],b\in\{0,1\}}\Big)$

$td := \Big(\mathbf{u},\{\mathbf{M}_i,\mathbf{P}_{i,b}\}_{i\in[j],b\in\{0,1\}},$
$\{\mathbf{A}_i,\mathbf{B}_{i,b},\mathbf{M}_{k,b}\}_{i\in[k-1]-[j],b\in\{0,1\}}\Big)$

Return (vk,td)

$\underline{\mathbf{Eval}_1(x)}$: // $\mathbf{Game}_j$

$\mathbf{v}^\top = \mathbf{u}^\top\cdot\prod_{i=1}^{j}(\mathbf{M}_i-\mathbf{P}_{i,x_i})\cdot$
$\prod_{i=j+1}^{k-1}(\mathbf{A}_i-\mathbf{B}_{i,x_i})\cdot\mathbf{M}_{k,x_k}$

Return $[\mathbf{B}\cdot\mathbf{v}]$

$\underline{\mathsf{VHF.TrapGen}(\Pi,[\mathbf{B}],X^{(0)},1^\lambda)}$: // $\mathbf{Game}_{j,A}$

$\mathbf{u} \xleftarrow{\$} \mathbb{Z}_p^n$

$(\mathbf{A}_i,\mathbf{B}_{i,0},\mathbf{B}_{i,1}) \xleftarrow{\$} \mathbb{Z}_p^{n\times n}\times\mathscr{D}_0\times\mathscr{D}_1$

$\big(\mathbf{M}_i,\mathbf{P}_{i,0},\mathbf{P}_{i,1}\big)_{i\le j}$ are sampled according to Figure 10

$$\mathbf{P}_{j+1,0} = \begin{pmatrix} r_{1,1} & r_{1,2} & \dots & r_{1,n}+\alpha_1 \\ 0 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 \end{pmatrix}$$

$$\mathbf{P}_{j+1,1} = \begin{pmatrix} 0 & 0 & \dots & 0 \\ r_{2,1} & r_{2,2} & \dots & r_{2,n}-\alpha_2 \\ \dots & \dots & \dots & \dots \\ r_{n-1,1} & r_{n-1,2} & \dots & r_{n-1,n}-\alpha_{n-1} \\ 0 & 0 & \dots & -1 \end{pmatrix}$$

$\mathbf{M}_{j+1} \xleftarrow{\$} \mathbb{Z}_p^{n\times n}$

$\big(\mathbf{M}_{k,0},\mathbf{M}_{k,1}\big) \xleftarrow{\$} \mathbb{Z}_p^{n\times n}\times\mathbb{Z}_p^{n\times n}$

$vk := \Big([\mathbf{u}],\{[\mathbf{M}_i],[\mathbf{P}_{i,b}]\}_{i\in[j+1],b\in\{0,1\}},$
$\{[\mathbf{A}_i],[\mathbf{B}_{i,b}],[\mathbf{M}_{k,b}]\}_{i\in[k-1]-[j+1],b\in\{0,1\}},\Big)$

$td := \Big(\mathbf{u},\{\mathbf{M}_i,\mathbf{P}_{i,b}\}_{i\in[j+1],b\in\{0,1\}},$
$\{\mathbf{A}_i,\mathbf{B}_{i,b},\mathbf{M}_{k,b}\}_{i\in[k-1]-[j+1],b\in\{0,1\}}\Big)$

Return (vk,td)

$\underline{\mathbf{Eval}_1(x)}$: // $\mathbf{Game}_{j,A}$

$\mathbf{v}^\top = \mathbf{u}^\top\cdot\prod_{i=1}^{j+1}(\mathbf{M}_i-\mathbf{P}_{i,x_i})\cdot$
$\prod_{i=j+2}^{k-1}(\mathbf{A}_i-\mathbf{B}_{i,x_i})\cdot\mathbf{M}_{k,x_k}$

Return $[\mathbf{B}\cdot\mathbf{v}]$

Fig. 5: We describe the VHF.TrapGen and VHF.TrapEval procedures used during the indistinguishability experiment defined in Figure 1 for $\mathbf{Game}_{j\in[k-1]}$ (left) and $\mathbf{Game}_{j,A}$ (right). VHF.Gen and VHF.Eval are identical to ones described in Figure 4.

given as input a pair $([\mathbf{A}],[e_b])$; e_0 stands for an eigenvalue for $\mathbf{A}$ and e_1 for a uniform element. We provide the internal working of $\mathscr{A}'$ (Figure 6) in what follows:

- Input: given a tuple $([\mathbf{A}],[e_b])$, we set the $\mathbf{M}_i,\mathbf{P}_{i,0},\mathbf{P}_{i,1}, i\in[j]$ as in the $Game_{j,A}$. We embed the challenge in $\mathbf{M}_{j+1},\mathbf{P}_{j+1,0},\mathbf{P}_{j+1,1}$ as depicted in Figure 3.
- $\mathscr{A}'$ is able to construct the evaluation oracle, given that it knows the values of the matrices used, except the ones corresponding to index j+1 (which are encoded). This means that $\mathscr{A}'$ is be able to interact with the evaluation oracle.

Correctness of the simulation:

- if e_b is an eigenvalue for $\mathbf{A}$, then the matrices $\mathbf{M}_{j+1},\mathbf{P}_{j+1,0},\mathbf{P}_{j+1,1}$ are distributed as in $\mathbf{Game}_{j+1}$.
- if e_b is not an eigenvalue for $\mathbf{A}$, but a uniformly sampled element from $\mathbb{Z}_p$, the matrices are distributed exactly as in $\mathbf{Game}_{j,A}$, the argument relying on the fact that $(\mathbf{A}-e_b\cdot\mathbf{I}_n)$ is a full rank matrix:

Algorithm $\mathscr{A}'\big(([\mathbf{A}],[e_b]),j,\lambda\big)$:

1. $\left[\begin{pmatrix}\mathbf{S}_{j+1}\\ \mathbf{s}_{j+1}^\top\cdot\mathbf{S}_{j+1}\end{pmatrix}\right]\leftarrow[\mathbf{A}-e_b\cdot\mathbf{I}_n]$ (we abuse notation — the last row matching the form $\mathbf{s}_{j+1}^\top\cdot\mathbf{S}_{j+1}$)
2. set $\big(td,vk\big)$ as in $Game_{j,A}$; set $\big([\mathbf{M}_{j+1}],[\mathbf{P}_{j+1,0}],[\mathbf{P}_{j+1,1}]\big)$ as in Figure 3
3. $\mathscr{A}'$ builds $\mathbf{Eval}(\cdot)$ as follows:
   3.1. $\mathscr{A}'$ knows all the matrices, except the one on position $j+1$ (built from challenge)
   3.2. $\mathscr{A}'$ computes $\mathbf{Eval}(x)=\big[\mathbf{u}^\top\cdot\prod_{i=1}^{k-1}\big(\mathbf{M}_i-\mathbf{P}_{i,x_i}\big)\cdot\mathbf{M}_{k,x_k}\big]=[\mathbf{v}]$ as follows:
      3.2.1. compute "in plain" $\mathbf{v}^{(j)^\top}=\mathbf{u}^\top\cdot\prod_{i=1}^{j}\big(\mathbf{M}_i-\mathbf{P}_{i,x_i}\big)$
      3.2.2. compute $[\mathbf{v}^{(j+1)^\top}]=[\mathbf{v}^{(j)^\top}\cdot\big(\mathbf{M}_{j+1}-\mathbf{P}_{j+1,x_{j+1}}\big)]$
      3.2.3. compute $[\mathbf{v}^\top]=[\mathbf{v}^{(k)^\top}]=[\mathbf{v}^{(j+1)^\top}]\cdot\prod_{i=j+2}^{k-1}\big(\mathbf{M}_i-\mathbf{P}_{i,x_i}\big)\cdot\mathbf{M}_{k,x_k}$
4. $b\leftarrow\mathscr{A}^{\mathbf{Eval}(\cdot)}$; return b

Fig. 6: The reduction algorithm used in the proof of Lemma 7.

– $\mathbf{P}_{j+1,0}=\begin{pmatrix}s_{1,1}^{(j+1)} & s_{1,2}^{(j+1)} & \dots & s_{1,n}^{(j+1)}+\alpha_1\\ 0 & 0 & \dots & 0\\ \dots & \dots & \dots & \dots\\ 0 & 0 & \dots & 0\end{pmatrix}$ is indistinguishable from its equivalent from $\mathbf{Game}_{j,A}$, due to the fact that $(s_{1,1}^{(j+1)},s_{1,2}^{(j+1)},\dots,s_{1,n}^{(j+1)})$ are uniform elements.

– $\mathbf{P}_{j+1,1}=\begin{pmatrix}0 & 0 & \dots & 0\\ s_{2,1}^{(j+1)} & s_{2,2}^{(j+1)} & \dots & s_{2,n}^{(j+1)}-\alpha_2\\ \dots & \dots & \dots & \dots\\ s_{n-1,1}^{(j+1)} & s_{n-1,2}^{(j+1)} & \dots & s_{n-1,n}^{(j+1)}-\alpha_{n-1}\\ 0 & 0 & \dots & -1\end{pmatrix}$ is indistinguishable from its equivalent from $\mathbf{Game}_{j,A}$, due to the fact that $\{s_{i,j}\}$ are uniform elements.

– $\mathbf{M}_{j+1}=\mathbf{L}_{j+1}\cdot\big(\mathbf{A}-e_b\cdot\mathbf{I}_n\big)+\mathbf{P}_{j,1-X_j^{(0)}}$. We argue for pseudorandomness based on the fact that $\mathbf{L}_{j+1}$ is a uniformly sampled matrix which randomizes the left side, and the result of the multiplication with the full rank matrix $\mathbf{A}-e_b\cdot\mathbf{I}_n$ is also uniform. Thus $\mathbf{M}_{j+1}$ is a uniform matrix if e_b is not an eigenvalue for $\mathbf{A}$.

– Finally, if the adversary $\mathscr{A}$ can distinguish the way $\mathbf{M}_j,\mathbf{P}_{j+1,0},\mathbf{P}_{j+1,1}$ were set up, based on $\mathbf{A}$ and e_b, then it can break the n-Eigen-Value assumption.

□

Hence, the advantage of an adversary distinguishing between $\mathbf{Game}_0$ and $\mathbf{Game}_{k-1}$ is bounded by:

$$\Pr\left[\mathbf{Game}_0^{\mathscr{A}}=1\right]-\Pr\left[\mathbf{Game}_{k-1}^{\mathscr{A}}=1\right]\leq(k-1)\cdot\mathbf{Adv}_{\mathbb{G}}^{n\text{-ev}}(\mathscr{A},\lambda)\ .$$

Final games. In $\mathbf{Game}_k$ we change the setup for $\big(\mathbf{M}_{k,0},\mathbf{M}_{k,1}\big)$. We rely on the same technique as in [HJ16]: given a challenge tuple $([\mathbf{A}],[e_b])$, we set $\left[\mathbf{M}_{k,x_k^*}\right]=[\mathbf{A}-e_b\cdot\mathbf{I}_n]$ and $\left[\mathbf{M}_{k,1-x_k^*}\right]=\mathbf{C}_k^{-1}\cdot[\mathbf{C}_k']$, where the forms of $\mathbf{C}_k$ and $\mathbf{C}_k'$ are depicted in Figure 8. Pictorially, one can visualize $\mathbf{C}_k^{-1}$ mapping $\mathscr{S}_{k-1}\to\mathscr{W}$ and $\mathbf{C}_k'$ mapping $\mathscr{W}\to\mathscr{S}_k$. The simulator knows $\mathbf{C}_k^{-1}$ and can construct $[\mathbf{C}_k']$ based on the challenge $([\mathbf{A}],[e_b])$ (by sampling uniformly $n-1$ encodings of vectors in the subspace $\mathscr{S}_k$ defined by $\mathbf{A}-e_b\cdot\mathbf{I}_n$). e_b not an eigenvalue implies a uniformly sampled $\mathbf{C}'$, and thus we are in the setting corresponding to $\mathbf{Game}_{k-1}$. On the other hand, if e_b is an eigenvalue, then we are in the setting corresponding to $\mathbf{Game}_k$.

The final transition, from $\mathbf{Game}_k\to\mathbf{Game}_{k+1}$, consists in embedding $\mathbf{B}$ (Definition 3) in $\mathbf{M}_{k,0}$ and $\mathbf{M}_{k,1}$. This is done by setting $\mathscr{S}_k$ as $\mathscr{W}$ and multiplying the resulting $\mathbf{M}_{k,0},\mathbf{M}_{k,1}$ with a uniformly sampled $\mathbf{B}$, which guarantees indistinguishability from random. $\mathbf{B}$ is to be used in Theorem 3 to embed the n-Lin assumption and prove the

pseudorandomness of the final (generic) VRF construction. Thus:

$$\Pr\left[\mathbf{Game}_0^{\mathscr{A}}=1\right]-\Pr\left[\mathbf{Game}_{k+1}^{\mathscr{A}}=1\right]=k\cdot\mathbf{Adv}_{\mathbb{G}}^{n\text{-ev}}(\mathscr{A},\lambda)+\mathbf{O}\big(k\cdot n/p\big)\ ^{1}$$

Finally, this completes the proof of Theorem 2, and shows the VHF is selectively programmable. Adaptive programmability is similar and shown in Section 5. □

5 Adaptive programmable PVHFs

The adaptive PVHF is obtained on top of the selective one defined in Figure 2, with the additional change that we make use of an admissible hash function AHF (Definition 8, Appendix A.2). AHFs, introduced in [BB04], and utilized in [CHKP10,AFL12], are useful tools for doing the "jump" from selective to adaptive security. In terms of the PVHF construction introduced in Section 4.1, the main change consists in hashing the inputs via an AHF. The resulting strings are used to feed the VHF.Eval, VHF.TrapEval algorithms defined for the PVHF. A partial change consists in the form of the matrices corresponding to positions for which $K_{\mathsf{AHF}}=\star$.

$\underline{\mathsf{VHF.TrapGen}([\mathbf{B}],1^\lambda)}$:
$\mathbf{u}\xleftarrow{\$}\mathbb{Z}_p^n$
$K_{\mathsf{AHF}}\xleftarrow{\$}\mathsf{KeyGen}_{\mathsf{AHF}}(1^\lambda)$
set $\left(\mathbf{M}_i,\mathbf{P}_{i,0},\mathbf{P}_{i,1}\right)$ as in Figure 8
set $\left(\mathbf{M}_{k,0},\mathbf{M}_{k,1}\right)$ as in Figure 8
$vk:=\left([\mathbf{u}],\{[\mathbf{M}_i],[\mathbf{P}_{i,b}],[\mathbf{M}_{k,b}]\}_{i\in[k-1],b\in\{0,1\}}\right)$
$td:=\left(\mathbf{u},\{\mathbf{M}_i,\mathbf{P}_{i,b},\mathbf{M}_{k,b}\}_{i\in[k-1],b\in\{0,1\}},K_{\mathsf{AHF}}\right)$

$\underline{\mathsf{VHF.TrapEval}(x,td)}$:
$x\leftarrow AHF_K(x)$
$\beta^\top=\mathbf{u}^\top\cdot\mathbf{M}_x$
$\pi=\left(\{[\mathbf{u}^\top\cdot\mathbf{M}_{\tilde{x}}]:\forall\tilde{x}\prec x\}\right)$
return $([\mathbf{B}\cdot\beta],\pi)$

$\underline{\mathsf{VHF.Vfy}(x,vk,[y],\pi)}$:
check $e([\mathbf{M}_{\tilde{x}}^{(i)}\cdot\mathbf{u}],[\mathbf{1}])\stackrel{?}{=}e([\mathbf{M}_{i,x_i}],[\mathbf{M}_{\tilde{x}}^{(i-1)}\cdot\mathbf{u}]),\forall i\in[k]$

Fig. 7: The adaptive PVHF obtained via AHFs.

The process of sampling the trapdoor keys (td,vk) depends on the key K_{AHF}, and is done as follows:

1. If $K_{\mathsf{AHF},j}\in\{0,1\}$, then the matrices $\left(\mathbf{M}_j,\mathbf{P}_{j,0},\mathbf{P}_{j,1}\right)$ are set as in the selective case (Figure 3).
2. If $K_{\mathsf{AHF},j}=\star$, a more careful instantiation is required. If this is the case, we want that both $\mathbf{M}_j-\mathbf{P}_{j,0}$ and $\mathbf{M}_j-\mathbf{P}_{j,1}$ to map vectors between $\mathscr{S}_{j-1}$ and $\mathscr{S}_j$. For this to happen, an adaptive VHF.TrapGen will sample the matrices as follows:

$$\mathbf{C}_j=\begin{pmatrix}s_{1,1}^{(j-1)}&s_{1,2}^{(j-1)}&\dots&s_{1,n}^{(j-1)}\\ s_{2,1}^{(j-1)}&s_{2,2}^{(j-1)}&\dots&s_{2,n}^{(j-1)}\\ \dots&\dots&\dots&\dots\\ r_{n,1}^{(j-1)}&r_{n,2}^{(j-1)}&\dots&r_{n,n}^{(j-1)}\end{pmatrix}\qquad \mathbf{C}'_j=\begin{pmatrix}s_{1,1}^{(j)}&s_{1,2}^{(j)}&\dots&s_{1,n}^{(j)}\\ s_{2,1}^{(j)}&s_{2,2}^{(j)}&\dots&s_{2,n}^{(j)}\\ \dots&\dots&\dots&\dots\\ r_{n,1}^{(j)}&r_{n,2}^{(j)}&\dots&r_{n,n}^{(j)}\end{pmatrix}$$

$$\mathbf{D}_j=\begin{pmatrix}\mathbf{L}_j&0\\0&1\end{pmatrix}\cdot\begin{pmatrix}s_{1,1}^{(j-1)}&s_{1,2}^{(j-1)}&\dots&s_{1,n}^{(j-1)}\\ s_{2,1}^{(j-1)}&s_{2,2}^{(j-1)}&\dots&s_{2,n}^{(j-1)}\\ \dots&\dots&\dots&\dots\\ 0&0&\dots&\delta\end{pmatrix}\qquad \mathbf{D}'_j=\begin{pmatrix}\mathbf{L}'_j&0\\0&1\end{pmatrix}\cdot\begin{pmatrix}s_{1,1}^{(j)}&s_{1,2}^{(j)}&\dots&s_{1,n}^{(j)}\\ s_{2,1}^{(j)}&s_{2,2}^{(j)}&\dots&s_{2,n}^{(j)}\\ \dots&\dots&\dots&\dots\\ 0&0&\dots&\delta\end{pmatrix}$$

$\mathbf{P}_{j,0}\xleftarrow{\$}\mathscr{D}_0$
$\mathbf{M}_j=\mathbf{C}_j^{-1}\cdot\mathbf{C}'_j+\mathbf{P}_{j,0}$
$\mathbf{P}_{j,0}-\mathbf{P}_{j,1}=\mathbf{D}_j^{-1}\cdot\mathbf{D}'_j$

Fig. 8: Matrices $\mathbf{M}_j,\mathbf{P}_{j,0},\mathbf{P}_{j,1}$ corresponding to $K_{\mathsf{AHF},j}=\star$. For the case where $K_{\mathsf{AHF},j}\neq\star$, the matrices are sampled as in Figure 3. Here, $\mathbf{L}_j$ and $\mathbf{L}'_j$ are uniform $(n-1)\times(n-1)$ matrices and $\delta\xleftarrow{\$}\mathbb{Z}_p$, the first $n-1$ rows in $\mathbf{C}_j$ are sampled from $\mathscr{S}_{j-1}$; $r_{x,y}$ denotes a random element in $\mathbb{Z}_p$.

[1] n/p is the probability that a matrix sampled uniformly at random in $\mathbb{Z}_p^{n\times n}$ is singular.

Lemma 8. *The construction defined in Figure 7 yields an adaptive secure* PVHF.

Proof (Lemma 8). Correctness and *unique provability* are proven as in the selective case. To motivate the *well-distributed outputs*, observe that if $K_{\mathsf{AHF},j} = \star$, then $\mathbf{D}_j^{-1} \cdot \mathbf{D}_j$ matches the pattern $\begin{pmatrix} \mathbf{R} \\ 0\,0\ldots 1 \end{pmatrix}$ where $\mathbf{R}$ stands for a uniformly sampled $(n-1) \times n$ matrix. When $K_{\mathsf{AHF},j} = \star$, then multiplying with $\mathbf{M}_j - \mathbf{P}_{j,x_j}$ will be either $\mathbf{C}_j^{-1} \cdot \mathbf{C}'_j$ or $\mathbf{C}_j^{-1} \cdot \mathbf{C}'_j + \mathbf{D}_j^{-1} \cdot \mathbf{D}'_j$. Both $\mathbf{C}_j^{-1} \cdot \mathbf{C}'_j$ and $\mathbf{D}_j^{-1} \cdot \mathbf{D}'_j$ map $\mathscr{S}_{j-1}$ to $\mathscr{S}_j$, and so does their sum.

To motivate *indistinguishability* for $K_{\mathsf{AHF},j} = \star$, observe that both $\mathbf{C}_j$ and $\mathbf{C}'_j$ are full rank matrices, and the elements in $\mathbf{C}'_j$ are uniformly sampled ($\mathscr{S}_j$ being a uniform subspace), thus $\mathbf{C}'_j$ acting as a randomness extractor. This randomizes $\mathbf{M}_j$. We also have to show that no computational adversary can distinguish $\mathbf{P}_{j,0}$ and $\mathbf{P}_{j,1}$ (thus $\mathbf{D}_j^{-1} \cdot \mathbf{D}'_j$) from a uniform matrix sampled from the same distributions. Both $\mathbf{D}_j$ and $\mathbf{D}'_j$ are pre-multiplied with two upper-corner uniform matrices acting as randomness extractors for $\overline{\mathbf{D}}_j$ and $\overline{\mathbf{D}'}_j$. The top rows of $\mathbf{D}_j$ and $\mathbf{D}'_j$ are therefore uniform. For completeness, we mention that if $K_{\mathsf{AHF},k} = \star$, we instantiate both $\mathbf{M}_{k,0}$ and $\mathbf{M}_{k,1}$ of the form $\mathbf{C}_{k,0/1}^{-1} \cdot \mathbf{C}'_{k,0/1}$.

Finally, we point out that $\mathscr{O}_{\mathsf{check}}$ oracle needed in the adaptive indistinguishability experiment defined in Figure 1 can be implemented using K_{AHF}. □

The adaptive PVHF represents the platform used to build an adaptive secure VRF under a static assumptions via the generic transform given in Appendix A.3(Theorem 3). The final advantage of an adversary $\mathscr{A}$ winning the pseudorandomness game being:

$$\mathbf{Adv}^{\mathsf{vrf}}_{\mathsf{VRF}}(\mathscr{A},\lambda) \leq \mathbf{Adv}^{\mathsf{Indist.}}_{\mathsf{PVHF}}(\mathscr{A},\lambda) + \mathbf{Adv}^{\mathsf{n-LIN}}(\mathscr{A},\lambda) = \ell_{\mathsf{AHF}} \cdot \mathbf{Adv}^{n\text{-}\mathsf{ev}}_{\mathbb{G}}(\mathscr{A},\lambda) + \mathbf{O}\big(\ell_{\mathsf{AHF}} \cdot n/p\big) + \mathbf{Adv}^{\mathsf{n-LIN}}(\mathscr{A},\lambda)\,.$$

Acknowledgements. The author was supported by EU Horizon 2020 research and innovation programme under grant agreement No H2020-MSCA-ITN-2014-643161 ECRYPT-NET.

References

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

AFL12. Michel Abdalla, Dario Fiore, and Vadim Lyubashevsky. From selective to full security: Semi-generic transformations in the standard model. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 316–333. Springer, Heidelberg, May 2012.

BB04. Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 443–459. Springer, Heidelberg, August 2004.

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

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.

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

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

BV15. Zvika Brakerski and Vinod Vaikuntanathan. Constrained key-homomorphic PRFs from standard lattice assumptions - or: How to secretly embed a circuit in your PRF. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 1–30. Springer, Heidelberg, March 2015.

Che06. Jung Hee Cheon. Security analysis of the strong Diffie-Hellman problem. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 1–11. Springer, Heidelberg, May / June 2006.

CHKP10. David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 523–552. Springer, Heidelberg, May 2010.
DY05. Yevgeniy Dodis and Aleksandr Yampolskiy. A verifiable random function with short proofs and keys. In Serge Vaudenay, editor, *PKC 2005*, volume 3386 of *LNCS*, pages 416–431. Springer, Heidelberg, January 2005.
EHK+13. Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. An algebraic framework for Diffie-Hellman assumptions. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 129–147. Springer, Heidelberg, August 2013.
GGM86. Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, October 1986.
GNP+15. Sharon Goldberg, Moni Naor, Dimitrios Papadopoulos, Leonid Reyzin, Sachin Vasant, and Asaf Ziv. NSEC5: Provably preventing DNSSEC zone enumeration. In *NDSS 2015*. The Internet Society, February 2015.
HJ16. Dennis Hofheinz and Tibor Jager. Verifiable random functions from standard assumptions. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part I*, volume 9562 of *LNCS*, pages 336–362. Springer, Heidelberg, January 2016.
HK08. Dennis Hofheinz and Eike Kiltz. Programmable hash functions and their applications. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 21–38. Springer, Heidelberg, August 2008.
HW10. Susan Hohenberger and Brent Waters. Constructing verifiable random functions with large input spaces. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 656–672. Springer, Heidelberg, May 2010.
LMR+05. Matt Larson, Dan Massey, Scott Rose, Roy Arends, and Rob Austein. Resource records for the dns security extensions. *Resource*, 2005.
Lys02. Anna Lysyanskaya. Unique signatures and verifiable random functions from the DH-DDH separation. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 597–612. Springer, Heidelberg, August 2002.
MR02. Silvio Micali and Ronald L. Rivest. Micropayments revisited. In Bart Preneel, editor, *CT-RSA 2002*, volume 2271 of *LNCS*, pages 149–163. Springer, Heidelberg, February 2002.
MR04. Silvio Micali and Leonid Reyzin. Physically observable cryptography (extended abstract). In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 278–296. Springer, Heidelberg, February 2004.
MRV99. Silvio Micali, Michael O. Rabin, and Salil P. Vadhan. Verifiable random functions. In *40th FOCS*, pages 120–130. IEEE Computer Society Press, October 1999.
NR04. Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. *Journal of the ACM*, 51(2):231–262, 2004.
SW14. Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *46th ACM STOC*, pages 475–484. ACM Press, May / June 2014.
Yam17. Shota Yamada. Asymptotically compact adaptively secure lattice ibes and verifiable random functions via generalized partitioning techniques. 2017.

A Standard Definitions

A.1 Certified Bilinear Group Generators

We recall the definition of Bilinear Group Generators [HJ16].

Definition 7 (Certified Bilinear Group Generators). *A* bilinear group generator *is a PPT algorithm* GrpGen *takes as input a security parameter (in unary)* 1^λ*, and outputs a tuple* $\Pi = (p, \mathbb{G}, \mathbb{G}_T, \star, \star_T, e, \phi(1))$ *and such that the following properties hold:*

- p *is prime whose size* $\log(p) \in \Omega(\lambda)$*;*
- $(\mathbb{G}, \star)$ *and* $(\mathbb{G}_T, \star_T)$ *are groups of order* p *described by two efficient isomorphisms* ϕ *and* ϕ_T *from* $(\mathbb{Z}_p, +)$ *to* $(\mathbb{G}, \star)$ *and* $(\mathbb{G}_T, \star_T)$ *respectively, and such that* $\star : \mathbb{G} \times \mathbb{G} \to \mathbb{G}$ *and* $\star_T : \mathbb{G}_T \times \mathbb{G}_T \to \mathbb{G}_T$ *are efficient operations;*
- $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ *is an efficient non-degenerate pairing, so that for any* $x \in \mathbb{Z}_p \setminus \{0\}$*,* $e(\phi(x), \phi(x)) \neq \phi_T(0)$*.*

Furthermore, we say that GrpGen *is certified if there exists a deterministic polynomial-time algorithm* GrpVfy *with the following two properties, termed* parameter validation *and* recognition and unique representation*:*

- *Parameter validation asks that, on input a string* Π*,* GrpVfy *outputs* 1 *if and only if* $\Pi = (p, \mathbb{G}, \mathbb{G}_T, \star, \star_T, e, \phi(1))$ *and satisfies the above properties.*
- *Recognition and unique representation requires that each element in* $\mathbb{G}$ *(defined above) has a* unique *representation, which can be recognized efficiently. Specifically, on inputs* Π, h*,* GrpVfy *outputs* 1 *if and only if* $\mathsf{GrpVfy}(\Pi) = 1$ *and* $h = \phi(x)$ *for some* $x \in \mathbb{Z}_p$*.*

As already mentioned above, we denote by $[x]_g$ the group element g^x, which then corresponds to $\phi(x)$ with $g = \phi(1)$, and adopt similar notation for elements of $\mathbb{G}_T$.

A.2 Admissible Hash Functions

We define admissible hash functions as follows.

Definition 8 (Admissible Hash Functions). *For a function* $\mathsf{AHF} : \{0,1\}^k \to \{0,1\}^{\ell_{\mathsf{AHF}}}$*, and* $K \in \{0,1,\star\}^{\ell_{\mathsf{AHF}}}$*, we define the function* $F_K : \{0,1\}^k \to \{\mathtt{CO},\mathtt{UN}\}$*:*

$$F_K(X) = \mathtt{UN} \iff \forall i : K_i = \mathsf{AHF}(X)_i \vee K_i = \star$$

We say that AHF *is q-admissible if there exist a PPT algorithm* KeyGen *and a polynomial* poly *such that for all* $X^{(0)}, X^{(1)}, \ldots, X^{(q)}$ *with* $X^{(0)} \notin \{X^{(i)}\}_{i=1,\ldots,q}$*, we have:*

$$\Pr\left[F_K(X^{(0)}) = \mathtt{UN} \wedge F_K(X^{(1)}) = \mathtt{CO} \wedge \cdots \wedge F_K(X^{(q)}) = \mathtt{CO}\right] \geq \frac{1}{poly(k)} \ ,$$

where the probability is over $K \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$*. We further say that* AHF *is an* admissible hash function *is* AHF *is q-admissible for any polynomial* $q = q(\lambda)$*.*

A.3 Generic constructions of VRF on top of PVHFs

We state the generic transform of a VRF defined on top of a PVHF. It post-processes the output of the PVHF as by multiplication with a vector acting as a randomness extractor, and adding relevant components to the proof generated by the PVHF.

Theorem 3 (Hofheinz and Jager [HJ16]). *Given an adaptive (selective) secure* PVHF*, the construction depicted in Figure 9 yields an adaptive (selective) secure* VRF *under the* n − LIN *assumption.*

$\underline{\mathsf{VRF.Gen}(1^\lambda)}$:
$\mathbf{v} \xleftarrow{\$} \mathbb{Z}_p^n$
$\Pi \xleftarrow{\$} \mathsf{GrpGen}(1^\lambda)$
$(vk_{\mathsf{PVHF}}, sk_{\mathsf{PVHF}}) \xleftarrow{\$} \mathsf{VHF.Gen}(\Pi, 1^\lambda)$
$vk := \left([\mathbf{v}], vk_{\mathsf{PVHF}}\right)$
$sk := \left(\mathbf{v}, sk_{\mathsf{PVHF}}\right)$

$\underline{\mathsf{VRF.Eval}(x, sk)}$:
$([\mathbf{z}], \pi_{\mathsf{PVHF}}) \xleftarrow{\$} \mathsf{VHF.Eval}(sk, x)$
$[y] := [\mathbf{v}^\top \mathbf{z}] = [\sum_{i=1}^{k+1} \mathbf{v}_i \cdot \mathbf{z}_i]$
$\pi := \left([\mathbf{v}_1 \cdot \mathbf{z}_1], \ldots, [\mathbf{v}_{k+1} \cdot \mathbf{z}_{k+1}], \pi_{\mathsf{PVHF}}\right)$
return $([y], \pi)$

$\underline{\mathsf{VRF.Vfy}(x, vk, [y], \pi)}$:
check $e([\mathbf{v}_i], [\mathbf{z}_i]) \stackrel{?}{=} e([\mathbf{z}_i \cdot \mathbf{v}_i], [1])$
check $[y] \stackrel{?}{=} [\sum_{i=1}^{k+1} \mathbf{v}_i \cdot \mathbf{z}_i]$
check $\mathsf{VHF.Vfy}(vk_{\mathsf{PVHF}}, \pi_{\mathsf{PVHF}}, x, [\mathbf{z}])$

Fig. 9: The generic construction used to obtain verifiable random functions based on programmable verifiable vector hash function, as presented in [HJ16].

B Proof of well-distributed outputs

We prove that $\mathbf{M}_j - \mathbf{P}_{j,X^{(0)}} = \mathbf{L}'_j \cdot \begin{pmatrix} \mathbf{S}_j \\ \mathbf{s}_j^\top \cdot \mathbf{S}_j \end{pmatrix} + (-1)^{X_i^{(0)}} \mathbf{N}$, where the matrices are defined in Figure 3 and $\mathbf{N}$ is the $n \times n$ zero matrix, except the last column being set to $(\alpha_1, \alpha_2, \ldots, \alpha_{n-1}, 1)$.

Proof (Property 1, Lemma 2).

$$\begin{aligned}
\mathbf{M}_j - \mathbf{P}_{j,X^{(0)}} &= \mathbf{L}_j \cdot \begin{pmatrix} \mathbf{S}_j \\ \mathbf{s}^\top \cdot \mathbf{S}_j \end{pmatrix} + \mathbf{P}_{j,1-X_j^{(0)}} - \mathbf{P}_{j,X^{(0)}} \\
&= \mathbf{L}_j \cdot \begin{pmatrix} \mathbf{S}_j \\ \mathbf{s}^\top \cdot \mathbf{S}_j \end{pmatrix} + (-1)^{X_i^{(0)}} \begin{pmatrix} -s_{1,1}^{(j)} & -s_{1,2}^{(j)} & \dots & -s_{1,n}^{(j)} - \alpha_1 \\ s_{2,1}^{(j)} & s_{2,2}^{(j)} & \dots & s_{2,n}^{(j)} - \alpha_2 \\ \dots & \dots & \dots & \dots \\ s_{n-1,1}^{(j)} & s_{n-1,2}^{(j)} & \dots & s_{n-1,n}^{(j)} - \alpha_{n-1} \\ 0 & 0 & \dots & -1 \end{pmatrix} \\
&= \mathbf{L}_j \cdot \begin{pmatrix} \mathbf{S}_j \\ \mathbf{s}^\top \cdot \mathbf{S}_j \end{pmatrix} + (-1)^{X_i^{(0)}} \begin{pmatrix} -1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & \dots & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{S}_j \\ \mathbf{s}^\top \cdot \mathbf{S}_j \end{pmatrix} + (-1)^{X_i^{(0)}} \mathbf{N} \\
&= \mathbf{L}_j' \cdot \begin{pmatrix} \mathbf{S}_j \\ \mathbf{s}_j^\top \cdot \mathbf{S}_j \end{pmatrix} + (-1)^{X_i^{(0)}} \mathbf{N}
\end{aligned} \tag{7}$$

□

C Proofs for Lemmata 3, 4, 5

Lemma 3. Let $\mathbf{S} \xleftarrow{\$} \mathbb{Z}_p^{(n-1)\times n}$ and $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_p^n$. Let $\mathbf{b} \in \mathbb{Z}_p^n$. Then $\exists \mathbf{a} \in \mathbb{Z}_p^n$ such that: $\mathbf{b}^\top \cdot (\mathbf{M}_j - \mathbf{P}_{j,1-X^{(0)}}) = \mathbf{a}^\top \cdot \begin{pmatrix} \mathbf{S}_j \\ \mathbf{s}_j^\top \cdot \mathbf{S}_j \end{pmatrix}$.

Proof (Lemma 3).

$$\begin{aligned}
\mathbf{b}^\top \cdot (\mathbf{M}_j - \mathbf{P}_{j,1-X^{(0)}}) &= \underbrace{\left(\mathbf{b}^\top \cdot \mathbf{L}_j\right)}_{\mathbf{a}^\top} \cdot \begin{pmatrix} \mathbf{S}_j \\ \mathbf{s}_j^\top \cdot \mathbf{S}_j \end{pmatrix} \\
&= \mathbf{a}^\top \cdot \begin{pmatrix} \mathbf{S}_j \\ \mathbf{s}_j^\top \cdot \mathbf{S}_j \end{pmatrix} \\
&= (a_1, a_2, \dots, a_n)^\top \cdot \begin{pmatrix} \mathbf{S}_j \\ \mathbf{s}_j^\top \cdot \mathbf{S}_j \end{pmatrix} \\
&= \sum_{i=1}^{n-1} (a_i + a_n \cdot t_i) \cdot \mathbf{s}_i^\top \in \left\{ \mathbf{z} \middle| \mathbf{z}^\top = \mathbf{a}^\top \cdot \begin{pmatrix} \mathbf{S}_j \\ \mathbf{s}_j^\top \cdot \mathbf{S}_j \end{pmatrix} \right\}
\end{aligned} \tag{8}$$

□

Lemma 4. Let $\mathbf{S} \xleftarrow{\$} \mathbb{Z}_p^{(n-1)\times n}$ and $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_p^n$. Let $\mathbf{v} \in \mathbb{Z}_p^n$ and $\mathbf{b}^\top = \mathbf{v}^\top \cdot \begin{pmatrix} \mathbf{S}_{j-1} \\ \mathbf{s}_{j-1}^\top \cdot \mathbf{S}_{j-1} \end{pmatrix}$. Then $\exists \mathbf{a} \in \mathbb{Z}_p^n$ such that $\mathbf{b}^\top \cdot (\mathbf{M}_j - \mathbf{P}_{j,1-X^{(0)}}) = \mathbf{a}^\top \cdot \begin{pmatrix} \mathbf{S}_j \\ \mathbf{s}_j^\top \cdot \mathbf{S}_j \end{pmatrix}$.

Proof (Lemma 4). Let $\mathbf{b}^\top \in \mathscr{S}_{j-1}$. Then $\mathbf{b}^\top \cdot (\mathbf{M}_j - \mathbf{P}_{j,X^{(0)}}) = \underbrace{\left(\mathbf{b}^\top \cdot \mathbf{L}_j\right)}_{\mathbf{a}^\top} \cdot \begin{pmatrix} \mathbf{S}_j \\ \mathbf{s}_j^\top \cdot \mathbf{S}_j \end{pmatrix} + \underbrace{\mathbf{b}^\top \cdot \mathbf{N}}_{\mathbf{0}} = \mathbf{a}^\top \cdot \begin{pmatrix} \mathbf{S}_j \\ \mathbf{s}_j^\top \cdot \mathbf{S}_j \end{pmatrix}$. □

Lemma 5. Let $\mathbf{S} \xleftarrow{\$} \mathbb{Z}_p^{(n-1)\times n}$ and $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_p^n$. $\forall \mathbf{v} \in \mathbb{Z}_p^n$ let $\mathbf{b}^\top \neq \mathbf{v}^\top \cdot \begin{pmatrix} \mathbf{S}_{j-1} \\ \mathbf{s}_{j-1}^\top \cdot \mathbf{S}_{j-1} \end{pmatrix}$. $\forall \mathbf{a} \in \mathbb{Z}_p^n$ we have $\mathbf{b}^\top \cdot (\mathbf{M}_j - \mathbf{P}_{j,X^{(0)}}) \neq \mathbf{a}^\top \cdot \begin{pmatrix} \mathbf{S}_j \\ \mathbf{s}_j^\top \cdot \mathbf{S}_j \end{pmatrix}$.

Proof (Lemma 5). Let $\mathbf{b}^\top \notin \mathscr{S}_{j-1}$. Then $\mathbf{b}^\top \cdot (\mathbf{M}_j - \mathbf{P}_{j,X^{(0)}}) = \underbrace{\left(\mathbf{b}^\top \cdot \mathbf{L}_j\right)}_{\mathbf{a}^\top} \cdot \begin{pmatrix} \mathbf{S}_j \\ \mathbf{s}_j^\top \cdot \mathbf{S}_j \end{pmatrix} + \underbrace{\mathbf{b}^\top \cdot \mathbf{N}}_{\neq \mathbf{0}} \neq \mathbf{a}^\top \cdot \begin{pmatrix} \mathbf{S}_j \\ \mathbf{s}_j^\top \cdot \mathbf{S}_j \end{pmatrix}$. □

D Procedure for obtaining $(\mathbf{M}_j, \mathbf{P}_{j,0}, \mathbf{P}_{j,1})$ during the hybrids

Input:

$\mathbf{S}$ - (n-1) x n uniform matrix
i - the game index
$X^{(0)}$ - a challenge bitstring

Output:

$(\mathbf{M}_j, \mathbf{P}_{j,0}, \mathbf{P}_{j,1})_{j \in [i-1]}$

$\mathbf{S}_i \leftarrow \mathbf{S}$

for j=i **to** 1 **do**:

$\delta_j \xleftarrow{\$} \mathbb{Z}_p$

$\mathbf{s}_j \xleftarrow{\$} \mathbb{Z}_p^{n-1}$

$\mathbf{L}_j \xleftarrow{\$} \mathbb{Z}_p^{n \times n}$

$\mathbf{S}_{j-1} \xleftarrow{\$} \mathbb{Z}_p^{(n-1)\times n}$

compute $(\alpha_1, \ldots \alpha_{n-1}, 1) \perp \mathbf{S}_{j-1}$

$$\mathbf{P}_{j,0} = \begin{pmatrix} s_{1,1}^{(j)} & s_{1,2}^{(j)} & \ldots & s_{1,n}^{(j)} + \alpha_1 \\ 0 & 0 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & 0 \end{pmatrix}$$

$$\mathbf{P}_{j,1} = \begin{pmatrix} 0 & 0 & \ldots & 0 \\ s_{2,1}^{(j)} & s_{2,2}^{(j)} & \ldots & s_{2,n}^{(j)} - \alpha_2 \\ \ldots & \ldots & \ldots & \ldots \\ s_{n-1,1}^{(j)} & s_{n-1,2}^{(j)} & \ldots & s_{n-1,n}^{(j)} - \alpha_{n-1} \\ 0 & 0 & \ldots & -1 \end{pmatrix}$$

$$\mathbf{M}_j = \mathbf{L}_j \cdot \begin{pmatrix} \mathbf{S}_j \\ \mathbf{s}_j^\top \cdot \mathbf{S}_j \end{pmatrix} + \mathbf{P}_{j,1-X_j^{(0)}}$$

endfor

Fig. 10: A recursive algorithm for sampling trapdoor matrices for $i \in [k-1]$.