Verifiable Random Functions from Standard Assumptions

Dennis Hofheinz*and Tibor Jager†

January 30, 2016

Abstract

The question whether there exist verifiable random functions with exponential-sized input space and full adaptive security based on a non-interactive, *constant-size* assumption is a long-standing open problem. We construct the first verifiable random functions which achieve all these properties simultaneously.

Our construction can securely be instantiated in groups with symmetric bilinear map, based on any member of the $(n-1)$-*linear* assumption family with $n \geq 3$. This includes, for example, the 2-linear assumption, which is also known as the *decision linear* (DLIN) assumption.

1 Introduction

A verifiable random function (VRF) V_{sk} is essentially a pseudorandom function, but with the additional feature that it is possible to create a *non-interactive* and *publicly verifiable* proof π that a given function value Y was computed correctly as $Y = V_{sk}(X)$. VRFs are useful ingredients for applications as various as resettable zero-knowledge proofs [37], lottery systems [38], transaction escrow schemes [31], updatable zero-knowledge databases [34], or e-cash [4, 5].

Desired properties of VRFs. The standard security properties required from VRFs are *pseudorandomness* (when no proof is given, of course) and *unique provability*. The latter means that for each X there is only one *unique* value Y such that a proof for the statement "$Y = V_{sk}(X)$" *exists*. Unique provability is a very strong requirement, because *not even the party that creates sk* (possibly maliciously) may be able to create fake proofs. For example, the natural attempt of constructing a VRF by combining a pseudorandom function with a non-interactive zero-knowledge proof system fails, because zero-knowledge proofs are *simulatable*, which contradicts uniqueness.

Most known constructions of verifiable random functions allow an only *polynomially bounded* input space, or do not achieve *full adaptive security*, or are based on an *interactive* complexity assumption. In the sequel, we will say that a VRF has *all desired properties*, if is has an *exponential-sized input space* and a proof of *full adaptive security* under a *non-interactive* complexity assumption.

VRFs with all desired properties. All known examples of VRFs that possess all desired properties are based on so-called *Q-type* complexity assumptions. For example, the VRF of Hohenberger and Waters [28] relies on the assumption that, given a list of group elements

$$(g, h, g^x, \ldots, g^{x^{Q-1}}, g^{x^{Q+1}}, \ldots, g^{x^{2Q}}, t) \in \mathbb{G}^{2Q+1} \times \mathbb{G}_T$$

*Karlsruhe Institute of Technology, dennis.hofheinz@kit.edu. Supported by DFG grants HO 4534/2-2 and HO 4534/4-1

†Ruhr-University Bochum, tibor.jager@rub.de

and a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, it is computationally infeasible to distinguish $t = e(g, h)^{x^Q}$ from a random element of $\mathbb{G}_T$ with probability significantly better than $1/2$. Note that the assumption is parametrized by an integer Q, which determines the number of group elements in a given problem instance.

The main issue with Q-type assumptions is that they get stronger with increasing Q, as demonstrated by Cheon [18]. For example, the VRF described in [28] is based on a Q-type assumption with $Q = \Theta(q \cdot k)$, where k is the security parameter and q is the number of function evaluations queried by the attacker in the security experiment. Constructions from weaker Q-type assumptions were described by Boneh *et al.* [11] and Abdalla *et al.* [2], both require $Q = \Theta(k)$. A VRF-security proof for the classical verifiable *unpredictable* function of Lysyanskaya [35], which requires a Q-type assumption with only $Q = O(\log k)$, was recently given in [29]. Even though this is complexity assumption is relatively weak, it is still Q-type.

In summary, the construction of a VRF with all desired security properties, which is based on a standard, *constant-size* assumption (like the *decision-linear* assumption, for example) is a long-standing open problem, posed for example in [28, 29]. Some authors even asked if it is possible to prove that a Q-type assumption is *inherently necessary* to construct such VRFs [28]. Indeed, by adopting the techniques of [30] to the setting of VRFs, one can prove [33] that some known VRF-constructions are *equivalent* to certain Q-type assumptions, which means that a security proof under a strictly weaker assumption is impossible. This includes the VRFs of Dodis-Yampolskiy [20] and Boneh *et al.* [11]. It is also known that it is impossible to construct verifiable random functions from one-way permutations [13] or even trapdoor permutations in a black-box manner [23].

Our contribution. We construct the first verifiable random functions with exponential-sized input space, and give a proof of full adaptive security under any member of the $(n-1)$-*linear* assumption family with $n \geq 3$ in symmetric bilinear groups. The $(n-1)$-linear assumption is a family of non-interactive, *constant-size* complexity assumptions, which get progressively weaker with larger n [42]. A widely-used special case is the 2-linear assumption, which is also known as the *decision-linear* (DLIN) assumption [10].

Recently, a lot of progress has been made in proving the security of cryptosystems which previously required a Q-type assumption, see [43, 26, 16], for example. Verifiable random functions with all desired properties were one of the last cryptographic applications that required Q-type assumptions. Our work eliminates VRFs from this list.

The new construction and proof idea. The starting point for our construction is the VRF of Lysyanskaya [35]. Her function is in fact the Naor-Reingold pseudorandom function [40] with

$$V_{sk}(X) \;=\; g^{\prod_{i=1}^{k} a_{i,x_i}},$$

where $X = (x_1, \ldots, x_k)$, and the $a_{i,b}$ are randomly chosen exponents. However, unlike [40], Lysyanskaya considers this function in a "Diffie-Hellman gap group".[1] The corresponding verification key consists of all $g^{a_{i,x_i}}$. Relative to this verification key, an image y can be proven to be of the form $g^{\prod_{i=1}^{k} a_{i,x_i}}$ by publishing all "partial products in the exponent", that is, all values $\pi_\ell := g^{\prod_{i=1}^{\ell} a_{i,x_i}}$ for $\ell \in \{2, \ldots, k-1\}$. (Since the Decisional Diffie-Hellman problem is assumed to be easy, these partial products can be checked for consistency with the $g^{a_{i,b}}$ one after the other.)

Note that pseudorandomness of this construction is not obvious. Indeed, Lysyanskaya's analysis requires a computational assumption that offers k group elements in a computational

[1] In a Diffie-Hellman gap group, the Decisional Diffie-Hellman problem is easy, but the Computational Diffie-Hellman is hard. A prominent candidate of such groups are pairing-friendly groups.

challenge. (This "size-k" assumption could be reduced to a "size-$(\log(k))$" assumption recently [29].) One reason for this apparent difficulty lies in the verifiability property of a VRF. For instance, the original Naor-Reingold analysis of [40] (that shows that this V_{sk} is a *PRF*) can afford to gradually substitute images given to the adversary by random images, using a hybrid argument. Such a proof is not possible in a setting in which the adversary can ask for "validity proofs" for some of these images. (Note that by the uniqueness property of a VRF, we cannot expect to be able to simulate such validity proofs for non-images.) As a result, so far security proofs for VRFs have used "one-shot reductions" to suitable computational assumptions (which then turned out to be rather complex).

We circumvent this problem by a more complex function (with more complex public parameters) that can be modified *gradually*, using simpler computational assumptions. Following [22], in the sequel we will write $g^{\mathbf{u}}$, where $\mathbf{u} = (u_1, \ldots, u_n)^\top \in \mathbb{Z}_p^n$ is a vector, to denote the vector $g^{\mathbf{u}} := (g^{u_1}, \ldots, g^{u_n})$. We will also extend this notation to matrices in the obvious way. To explain our approach, consider the function

$$G_{sk}(X) \;=\; g^{\mathbf{u}^\top \cdot \prod_{i=1}^{k} \mathbf{M}_{i,x_i}}$$

for random (quadratic) *matrices* $\mathbf{M}_{i,x_i}$ and a random vector $\mathbf{u}$. The function G_{sk} will not be the VRF V_{sk} we seek, but it will form the basis for it. (In fact, V_{sk} will only postprocess G_{sk}'s output, in a way we will explain below.) V_{sk}'s verification key will include $g^{\mathbf{u}}$ and the $g^{\mathbf{M}_{i,b}}$. As in the VRF described above, validity proofs of images contain all partial products $g^{\mathbf{u}^\top \cdot \prod_{i=1}^{\ell} \mathbf{M}_{i,x_i}}$. (However, note that to *check* proofs, we now need a bilinear map, and not only an efficient DDH-solver, as with Lysyanskaya's VRF.)

To show pseudorandomness, let us first consider the case of *selective security* in which the adversary $\mathcal{A}$ first commits to a challenge preimage X^*. Then, $\mathcal{A}$ receives the verification key and may ask for arbitrary images $V_{sk}(X)$ *and proofs* for $X \neq X^*$. Additionally, $\mathcal{A}$ gets either $V_{sk}(X^*)$ (without proof), or a random image, and has to decide which it is.

In this setting, we can gradually adapt the $g^{\mathbf{M}_{i,b}}$ given to $\mathcal{A}$ such that $\prod_{i=1}^{k} \mathbf{M}_{i,x_i}$ has full rank if and only if $X = X^*$. To this end, we choose $\mathbf{M}_{i,b}$ as a full-rank matrix exactly for $b = x_i^*$. (This change can be split up in a number of local changes, each of which changes only one $\mathbf{M}_{i,b}$ and can be justified with the $(n-1)$-linear assumption, where n is the dimension of $\mathbf{M}_{i,b}$.) Even more: we show that if we perform these changes carefully, and in a "coordinated" way, we can achieve that $\mathbf{v}^\top := \mathbf{u}^\top \prod_{i=1}^{k} \mathbf{M}_{i,x_i}$ lies in a fixed subspace $\mathfrak{U}^\top$ if and only if $X \neq X^*$. In other words, if we write $\mathbf{v} = \sum_{i=1}^{n} \beta_i \mathbf{b}_i$ for a basis $\{\mathbf{b}_i\}_{i=1}^{n}$ such that $\{\mathbf{b}_i\}_{i=1}^{n-1}$ is a basis of $\mathfrak{U}$, then we have that $\beta_n = 0$ if and only if $X \neq X^*$. Put differently: $\mathbf{v}$ has a $\mathbf{b}_n$-component if and only if $X = X^*$.

Hence, we could hope to embed (part of) a challenge from a computational hardness assumption into $\mathbf{b}_n$. For instance, to obtain a VRF secure under the Bilinear Decisional Diffie-Hellman (BDDH) assumption, one could set $V_{sk}(X) = e(G_{sk}(X), g^{\alpha})^{\beta}$ for a pairing e and random α, β. A BDDH challenge can then be embedded into $\mathbf{b}_n$, α, and β. (Of course, also validity proofs need to be adapted suitably.)

In the main part of the paper, we show how to generalize this idea simultaneously to adaptive security (with a semi-generic approach that employs admissible hash functions), and based on the $(n-1)$-linear assumption for arbitrary $n \geq 3$ (instead of the BDDH assumption).

We note that we pay a price for a reduction to a standard assumption: since our construction relies on *matrix* multiplication (instead of multiplication of exponents), it is less efficient than previous constructions. For instance, compared to Lysyanskaya's VRF, our VRF has less compact proofs (by a factor of about n, when building on the $(n-1)$-linear assumption), and requires more pairing operations (by a factor of about n^2) for verification.

Programmable vector hash functions. The proof strategy sketched above is implemented by a new tool that we call *programmable vector hash functions* (PVHFs). Essentially, PVHFs can be seen as a variant of programmable hash functions of Hofheinz and Kiltz [27], which captures the "coordinated" setup of G_{sk} described above in a modular building block. We hope that this building block will be useful for other cryptographic constructions.

More related work. VRFs were introduced by Micali, Rabin, and Vadhan [36]. Number-theoretic constructions of VRFs were described in [36, 35, 19, 20, 1, 28, 11, 2, 29]. Abdalla *et al.* [1, 2] also gave a generic construction from a special type of identity-based key encapsulation mechanisms. Most of these either do not achieve full adaptive security for large input spaces, or are based on *interactive* complexity assumptions, the exceptions [28, 11, 2, 29] were mentioned above. We wish to avoid interactive assumptions to prevent circular arguments, as explained by Naor [39].

The notion of *weak* VRFs was proposed by Brakerski *et al.* [13], along with simple and efficient constructions, and proofs that neither VRFs, nor weak VRFs can be constructed (in a black-box way) from one-way permutations. Several works introduced related primitives, like simulatable VRFs [15] and constrained VRFs [25].

Other approaches to avoid Q-type assumptions. One may ask whether the techniques presented by Chase and Meiklejohn [16], which in certain applications allow to replace Q-type assumption with *constant-size* subgroup hiding assumptions, give rise to alternative constructions of VRFs from constant-size assumptions. This technique is based on the idea of using the *dual-systems* approach of Waters [43], and requires to add *randomization* to group elements. This randomization makes it difficult to construct VRFs that meet the unique provability requirement. Consequently, Chase and Meiklejohn were able to prove that the VRF of Dodis and Yampolski [20] forms a secure pseudorandom function under a static assumption, but not that it is a secure VRF.

Open problems. The verifiable random functions constructed in this paper are relatively inefficient, when compared to the q-type-based constructions of [28, 11, 2, 29], for example. An interesting open problem is therefore the construction of more efficient VRFs from standard assumptions. In particular, it is not clear whether the constructions in this paper can also be instantiated from the SXDH assumption in asymmetric bilinear groups. This would potentially yield a construction with smaller matrices, and thus shorter proofs.

2 Certified Bilinear Group Generators

In order to be able to prove formally that a given verifiable random function satisfies *uniqueness* in the sense of Definition 5.1, we extend the notion of *certified trapdoor permutations* [7, 8, 32] to *certified bilinear group generators*. Previous works on verifiable random functions were more informal in this aspect, e.g., by requiring that group membership can be tested efficiently and that each group element has a unique representation.

Definition 2.1. *A* bilinear group generator *is a probabilistic polynomial-time algorithm* GrpGen *that takes as input a security parameter k (in unary) and outputs $\Pi = (p, \mathbb{G}, \mathbb{G}_T, \circ, \circ_T, e, \phi(1)) \xleftarrow{\$} \mathsf{GrpGen}(1^k)$ such that the following requirements are satisfied.*

1. *p is prime and $\log(p) \in \Omega(k)$.*
2. *$\mathbb{G}$ and $\mathbb{G}_T$ are subsets of $\{0,1\}^*$, defined by algorithmic descriptions of maps $\phi : \mathbb{Z}_p \to \mathbb{G}$ and $\phi_T : \mathbb{Z}_p \to \mathbb{G}_T$.*
3. *$\circ$ and $\circ_T$ are algorithmic descriptions of efficiently computable (in the security parameter) maps $\circ : \mathbb{G} \times \mathbb{G} \to \mathbb{G}$ and $\circ_T : \mathbb{G}_T \times \mathbb{G}_T \to \mathbb{G}_T$, such that*
   (a) $(\mathbb{G}, \circ)$ and $(\mathbb{G}_T, \circ_T)$ form algebraic groups and

(b) ϕ is a group isomorphism from $(\mathbb{Z}_p, +)$ to $(\mathbb{G}, \circ)$ and
(c) ϕ_T is a group isomorphism from $(\mathbb{Z}_p, +)$ to $(\mathbb{G}_T, \circ_T)$.

4. e is an algorithmic description of an efficiently computable (in the security parameter) bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. We require that e is non-degenerate*, that is,*

$$x \neq 0 \implies e(\phi(x), \phi(x)) \neq \phi_T(0)$$

Definition 2.2. *We say that group generator* GrpGen *is* certified*, if there exists a deterministic polynomial-time algorithm* GrpVfy *with the following properties.*

Parameter validation. *Given a string Π (which is not necessarily generated by GrpGen), algorithm $\mathsf{GrpVfy}(\Pi)$ outputs* 1 if and only if Π *has the form*

$$\Pi = (p, \mathbb{G}, \mathbb{G}_T, \circ, \circ_T, e, \phi(1))$$

and all requirements from Definition 2.1 are satsified.

Recognition and unique representation of elements of $\mathbb{G}$. *Furthermore, we require that each element in $\mathbb{G}$ has a* unique *representation, which can be efficiently recognized. That is, on input two strings Π and s, $\mathsf{GrpVfy}(\Pi, s)$ outputs* 1 if and only if $\mathsf{GrpVfy}(\Pi) = 1$ *and it holds that $s = \phi(x)$ for some $x \in \mathbb{Z}_p$. Here $\phi : \mathbb{Z}_p \to \mathbb{G}$ denotes the fixed group isomorphism contained in Π to specify the representation of elements of $\mathbb{G}$ (see Definition 2.1).*

3 Programmable Vector Hash Functions

Notation. As explained in the introduction, for a vector $\mathbf{u} = (u_1, \ldots, u_n)^\top \in \mathbb{Z}_p^n$ we will write $g^{\mathbf{u}}$ to denote the vector $g^{\mathbf{u}} := (g^{u_1}, \ldots, g^{u_n})$, and we will generalize this notation to matrices in the obvious way. Moreover, whenever the reference to a group generator $g \in \mathbb{G}$ is clear (note that a generator $g = \phi(1)$ is always contained in the group parameters Π generated by GrpGen), we will henceforth follow [22] and simplify our notation by writing $[x] := g^x \in \mathbb{G}$ for an integer $x \in \mathbb{Z}_p$, $[\mathbf{u}] := g^{\mathbf{u}} \in \mathbb{G}^n$ for a vector $\mathbf{u} \in \mathbb{Z}_p^n$, and $[\mathbf{M}] := g^{\mathbf{M}} \in \mathbb{G}^{n \times n}$ for a matrix $\mathbf{M} \in \mathbb{Z}_p^{n \times n}$. We also extend our notation for bilinear maps: we write $e([\mathbf{A}], [\mathbf{B}])$ (for matrices $\mathbf{A} = (a_{i,j})_{i,j} \in \mathbb{Z}_p^{n_1 \times n_2}$ and $\mathbf{B} = (b_{i,j})_{i,j} \in \mathbb{Z}_p^{n_2 \times n_3}$) for the matrix whose (i,j)-th entry is $\prod_{\ell=1}^{n_2} e([a_{i,\ell}], [b_{\ell,j}])$. In other words, we have $e([\mathbf{A}], [\mathbf{B}]) = e(g,g)^{\mathbf{AB}}$.

For a vector space $\mathfrak{U} \subseteq \mathbb{Z}_p^{n \times n}$ of column vectors, we write $\mathfrak{U}^\top := \{\mathbf{u}^\top \mid \mathbf{u} \in \mathfrak{U}\}$ for the respective set of row vectors. Furthermore, we write $\mathfrak{U}^\top \cdot \mathbf{M} := \{\mathbf{u}^\top \cdot \mathbf{M} \mid \mathbf{u}^\top \in \mathfrak{U}^\top\}$ for an element-wise vector-matrix multiplication. Finally, we denote with $\mathrm{GL}_n(\mathbb{Z}_p) \subset \mathbb{Z}_p^{n \times n}$ the set of invertible n-by-n matrices over $\mathbb{Z}_p$. Recall that a uniformly random $\mathbf{M} \in \mathbb{Z}_p$ is invertible except with probability at most n/p. (Hence, the uniform distributions on $\mathrm{GL}_n(\mathbb{Z}_p)$ and $\mathbb{Z}_p^{n \times n}$ are statistically close.)

3.1 Vector Hash Functions

Definition 3.1. *Let* GrpGen *be group generator algorithm and let $n \in \mathbb{N}$ be a positive integer. A* verifiable vector hash function (VHF) *for* GrpGen *with domain $\{0,1\}^k$ and range $\mathbb{G}^n$ consists of algorithms $(\mathsf{Gen}_{\mathsf{VHF}}, \mathsf{Eval}_{\mathsf{VHF}}, \mathsf{Vfy}_{\mathsf{VHF}})$ with the following properties.*

- $\mathsf{Gen}_{\mathsf{VHF}}$ *takes as input parameters $\Pi \xleftarrow{\$} \mathsf{GrpGen}(1^k)$ and outputs a verification key vk and an evaluation key ek as $(vk, ek) \xleftarrow{\$} \mathsf{Gen}_{\mathsf{VHF}}(\Pi)$.*
- $\mathsf{Eval}_{\mathsf{VHF}}$ *takes as input an evaluation key ek and a string $X \in \{0,1\}^k$. It outputs $([\mathbf{v}], \pi) \leftarrow \mathsf{Eval}_{\mathsf{VHF}}(ek, X)$, where $[\mathbf{v}] = ([v_1], \ldots, [v_n])^\top \in \mathbb{G}^n$ is the function value and $\pi \in \{0,1\}^*$ is a corresponding proof of correctness.*

- $\mathsf{Vfy}_{\mathsf{VHF}}$ *takes as input a verification key* vk*, vector* $[\mathbf{v}] \in \mathbb{G}^n$*, proof* $\pi \in \{0,1\}^*$*, and* $X \in \{0,1\}^k$*, and outputs a bit:* $\mathsf{Vfy}_{\mathsf{VHF}}(vk, [\mathbf{v}], \pi, X) \in \{0,1\}$.

We require correctness *and* unique provability *in the following sense.*

Correctness. *We say that* $(\mathsf{Gen}_{\mathsf{VHF}}, \mathsf{Eval}_{\mathsf{VHF}}, \mathsf{Vfy}_{\mathsf{VHF}})$ *is* correct*, if for all* $\Pi \xleftarrow{\$} \mathsf{GrpGen}(1^k)$*, all* $(vk, ek) \xleftarrow{\$} \mathsf{Gen}_{\mathsf{VHF}}(\Pi)$*, and all* $X \in \{0,1\}^k$ *holds that*

$$\Pr\left[\mathsf{Vfy}_{\mathsf{VHF}}(vk, [\mathbf{v}], \pi, X) = 1 : \begin{array}{l}(vk, ek) \xleftarrow{\$} \mathsf{Gen}_{\mathsf{VHF}}(\Pi), \\ ([\mathbf{v}], \pi) \leftarrow \mathsf{Eval}_{\mathsf{VHF}}(ek, X)\end{array}\right] = 1$$

Unique provability. *We say that a VHF has* unique provability*, if for all strings* $vk \in \{0,1\}^*$ *(not necessarily generated by* $\mathsf{Gen}_{\mathsf{VHF}}$*) and all* $X \in \{0,1\}^k$ *there does not exist any tuple* $([\mathbf{v}_0], \pi_0, [\mathbf{v}_1], \pi_1)$ *with* $[\mathbf{v}_0] \neq [\mathbf{v}_1]$ *and* $[\mathbf{v}_0], [\mathbf{v}_1] \in \mathbb{G}^n$ *such that*

$$\mathsf{Vfy}_{\mathsf{VHF}}(vk, [\mathbf{v}_0], \pi_0, X) = \mathsf{Vfy}_{\mathsf{VHF}}(vk, [\mathbf{v}_1], \pi_1, X) = 1$$

3.2 Selective Programmability

Definition 3.2. *We say that VHF* $(\mathsf{Gen}_{\mathsf{VHF}}, \mathsf{Eval}_{\mathsf{VHF}}, \mathsf{Vfy}_{\mathsf{VHF}})$ *is* selectively programmable*, if additional algorithms* $\mathsf{Trap}_{\mathsf{VHF}} = (\mathsf{TrapGen}_{\mathsf{VHF}}, \mathsf{TrapEval}_{\mathsf{VHF}})$ *exist, with the following properties.*

- $\mathsf{TrapGen}_{\mathsf{VHF}}$ *takes group parameters* $\Pi \xleftarrow{\$} \mathsf{GrpGen}(1^k)$*, matrix* $[\mathbf{B}] \in \mathbb{G}^{n \times n}$*, and* $X^{(0)} \in \{0,1\}^k$*. It computes* $(vk, td) \xleftarrow{\$} \mathsf{TrapGen}_{\mathsf{VHF}}(\Pi, [\mathbf{B}], X^{(0)})$*, where* vk *is a verification key with corresponding trapdoor evaluation key* td*.*
- $\mathsf{TrapEval}_{\mathsf{VHF}}$ *takes as input a trapdoor evaluation key* td *and a string* $X \in \{0,1\}^k$*. It outputs a vector* $\boldsymbol{\beta} \leftarrow \mathsf{TrapEval}_{\mathsf{VHF}}(td, X)$ *with* $\boldsymbol{\beta} \in \mathbb{Z}_p^n$ *and a proof* $\pi \in \{0,1\}^k$*.*

We furthermore have the following requirements.

Correctness. *For all* $\Pi \xleftarrow{\$} \mathsf{GrpGen}(1^k)$*, all* $[\mathbf{B}] \in \mathbb{G}^{n \times n}$*, and all* $X, X^{(0)} \in \{0,1\}^k$ *we have*

$$\Pr\left[\mathsf{Vfy}_{\mathsf{VHF}}(vk, [\mathbf{v}], X) = 1 : \begin{array}{l}(vk, td) \xleftarrow{\$} \mathsf{TrapGen}_{\mathsf{VHF}}(\Pi, [\mathbf{B}], X^{(0)}) \\ (\boldsymbol{\beta}, \pi) \leftarrow \mathsf{TrapEval}_{\mathsf{VHF}}(td, X) \\ [\mathbf{v}] := [\mathbf{B}] \cdot \boldsymbol{\beta}\end{array}\right] = 1$$

Indistinguishability. *Verification keys generated by* $\mathsf{TrapGen}_{\mathsf{VHF}}$ *are computationally indistinguishable from keys generated by* $\mathsf{Gen}_{\mathsf{VHF}}$*. More precisely, we require that for all PPT algorithms* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ *holds that*

$$\mathsf{Adv}^{\mathsf{vhf\text{-}sel\text{-}ind}}_{\mathsf{VHF}, \mathsf{Trap}_{\mathsf{VHF}}}(k) := 2 \cdot \Pr\left[\begin{array}{l}\Pi \xleftarrow{\$} \mathsf{GrpGen}(1^k);\ (X^{(0)}, st) \xleftarrow{\$} \mathcal{A}_0(1^k) \\ (vk_0, ek) \xleftarrow{\$} \mathsf{Gen}_{\mathsf{VHF}}(\Pi);\ \mathbf{B} \xleftarrow{\$} \mathrm{GL}_n(\mathbb{Z}_p) \\ (vk_1, td) \xleftarrow{\$} \mathsf{TrapGen}_{\mathsf{VHF}}(\Pi, [\mathbf{B}], X^{(0)}) \\ \overline{b} \xleftarrow{\$} \{0,1\};\ \mathcal{A}_1^{\mathcal{O}_{\overline{b}}}(st, vk_{\overline{b}}) = \overline{b}\end{array}\right] - 1$$

is negligible, where oracles $\mathcal{O}_0$ *and* $\mathcal{O}_1$ *are defined in Figure 1.*

Well-distributed outputs. *Let* $q = q(k) \in \mathbb{N}$ *be a polynomial, and let* $\beta_n^{(i)}$ *denote the* n*-th coordinate of vector* $\boldsymbol{\beta}^{(i)} \in \mathbb{Z}_p^n$*. There exists a polynomial* poly *such that for all* $(X^{(0)}, \ldots, X^{(q)}) \in (\{0,1\}^k)^{q+1}$ *with* $X^{(0)} \neq X^{(i)}$ *for* $i \geq 1$ *holds that*

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

$\mathcal{O}_0(X)$:	$\mathcal{O}_1(X)$:	$\mathcal{O}_{\mathsf{check}}(X)$:
$(\mathbf{v}, \pi) \leftarrow \mathsf{Eval}_{\mathsf{VHF}}(ek, X)$	$(\boldsymbol{\beta}, \pi) \leftarrow \mathsf{TrapEval}_{\mathsf{VHF}}(td, X)$	$(\boldsymbol{\beta}, \pi) \leftarrow \mathsf{TrapEval}_{\mathsf{VHF}}(td, X)$
Return $([\mathbf{v}], \pi)$	$[\mathbf{v}] := [\mathbf{B}] \cdot \boldsymbol{\beta}$	$(\beta_1, \ldots, \beta_n) := \boldsymbol{\beta}$
	Return $([\mathbf{v}], \pi)$	**If** $\beta_n \neq 0$ **then Return** 1
		Else Return 0

Figure 1: Definition of oracles $\mathcal{O}_0$, $\mathcal{O}_1$, and $\mathcal{O}_{\mathsf{check}}$.

We note that in our security definitions, $\mathbf{B}$ is always a random *invertible* matrix, although $\mathsf{TrapGen}_{\mathsf{VHF}}$ would also work on arbitrary $\mathbf{B}$.

Furthermore, note that we only require a noticeable "success probability" in our "well-distributed outputs" requirement above. This is sufficient for our application; however, our (selectively secure) PVHF construction achieves a success probability of 1. (On the other hand, our adaptively secure construction only achieves well-distributedness in the sense above, with a significantly lower – but of course still noticeable – success probability.)

3.3 Adaptive Programmability

Definition 3.3. *We say that VHF* $(\mathsf{Gen}_{\mathsf{VHF}}, \mathsf{Eval}_{\mathsf{VHF}}, \mathsf{Vfy}_{\mathsf{VHF}})$ *is (adaptively)* programmable*, if algorithms* $\mathsf{Trap}_{\mathsf{VHF}} = (\mathsf{TrapGen}_{\mathsf{VHF}}, \mathsf{TrapEval}_{\mathsf{VHF}})$ *exist, which have exactly the same syntax and requirements on* correctness*,* indistinguishability*, and* well-formedness *as in Definition 3.2, with the following differences:*

- $\mathsf{TrapGen}_{\mathsf{VHF}}(\Pi, [\mathbf{B}])$ *does not take an additional string* $X^{(0)}$ *as input.*
- *In the indistinguishability experiment,* $\mathcal{A}_0$ *is the trivial algorithm, which outputs the empty string* $\emptyset$*, while* $\mathcal{A}_1$ *additionally gets access to oracle* $\mathcal{O}_{\mathsf{check}}$ *(see Fig. 1). We stress that this oracle always uses td to compute its output, independently of* $\overline{b}$*. We denote with* $\mathsf{Adv}^{\mathsf{vhf\text{-}ad\text{-}ind}}_{\mathsf{VHF},\mathsf{Trap}_{\mathsf{VHF}}}(k)$ *the corresponding advantage function.*

4 A PVHF based on the Matrix-DDH Assumption

Overview. In this section, we present a programmable vector hash function, whose security is based upon the "Matrix-DDH" assumption introduced in [22] (which generalizes the matrix-DDH assumption of Boneh *et al.* [12] and the matrix d-linear assumption of Naor and Segev [41]). This assumption can be viewed as a relaxation of the $(n-1)$-linear assumption, so that in particular our construction will be secure under the $(n-1)$-linear assumption with $n \geq 3$.

Assumption 4.1. *The* n-rank assumption *states that* $[\mathbf{M}_{n-1}] \overset{\mathrm{c}}{\approx} [\mathbf{M}_n]$*, where* $\mathbf{M}_i \in \mathbb{Z}_p^{n \times n}$ *is a uniformly distributed rank-i matrix, i.e., that*

$$\mathsf{Adv}^{n\text{-}\mathsf{rank}}_{\mathcal{A}}(k) := \Pr\left[\mathcal{A}([\mathbf{M}_{n-1}]) = 1\right] - \Pr\left[\mathcal{A}([\mathbf{M}_n]) = 1\right]$$

is negligible for every PPT adversary $\mathcal{A}$*.*

4.1 The construction

Assume a bilinear group generator GrpGen and an integer $n \in \mathbb{N}$ as above. Consider the following vector hash function VHF:

- $\mathsf{Gen}_{\mathsf{VHF}}(\mathsf{GrpGen})$ uniformly chooses $2k$ invertible matrices $\mathbf{M}_{i,b} \in \mathbb{Z}_p^{n\times n}$ (for $1 \le i \le k$ and $b \in \{0,1\}$) and a nonzero vector $\mathbf{u}^\top \in \mathbb{Z}_p^n \setminus \{0\}$. The output is (vk, ek) with

$$vk = \big(([\mathbf{M}_{i,b}])_{1\le i\le k, b\in\{0,1\}}, [\mathbf{u}]\big) \qquad ek = \big((\mathbf{M}_{i,b})_{1\le i\le k, b\in\{0,1\}}, \mathbf{u}\big).$$

- $\mathsf{Eval}_{\mathsf{VHF}}(ek, X)$ (for $X = (x_1, \ldots, x_k)$) computes and outputs an image $[\mathbf{v}] = [\mathbf{v}_k] \in \mathbb{G}^n$ and a proof $\pi = ([\mathbf{v}_1], \ldots, [\mathbf{v}_{k-1}]) \in (\mathbb{G}^n)^{k-1}$, where

$$\mathbf{v}_i^\top = \mathbf{u}^\top \cdot \prod_{j=1}^{i} \mathbf{M}_{j,x_j}. \tag{1}$$

- $\mathsf{Vfy}_{\mathsf{VHF}}(vk, [\mathbf{v}], \pi, X)$ outputs 1 if and only if

$$e([\mathbf{v}_i^\top], [1]) = e([\mathbf{v}_{i-1}^\top], [\mathbf{M}_{i,x_i}]), \tag{2}$$

  holds for all i with $1 \le i \le k$, where we set $[\mathbf{v}_0] := [\mathbf{u}]$ and $[\mathbf{v}_k] := [\mathbf{v}]$.

Theorem 4.2 (Correctness and uniqueness of VHF)**.** VHF *is a vector hash function. In particular,* VHF *satisfies the correctness and uniqueness conditions from Definition 3.1.*

Proof. First, note that (2) is equivalent to $\mathbf{v}_i^\top = \mathbf{v}_{i-1}^\top \cdot \mathbf{M}_{i,x_i}$. By induction, it follows that (2) holds for all i if and only if $\mathbf{v}_i^\top = \mathbf{u}^\top \cdot \prod_{j=1}^{i} \mathbf{M}_{j,x_j}$ for all i. By definition of $\mathsf{Eval}_{\mathsf{VHF}}$, this yields correctness. Furthermore, we get that $\mathsf{Vfy}_{\mathsf{VHF}}$ outputs 1 for precisely one value $[\mathbf{v}] = [\mathbf{v}_k]$ (even if the $\mathbf{M}_{i,b}$ are not invertible). In fact, the proof π is uniquely determined by vk and X. □

4.2 Selective Security

We proceed to show the selective security of VHF:

Theorem 4.3 (Selectively programmabability of VHF)**.** VHF *is selectively programmable in the sense of Definition 3.2.*

The trapdoor algorithms. We split up the proof of Theorem 4.3 into three lemmas (that show correctness, well-distributed outputs, and indistinguishability of VHF). But first, we define the corresponding algorithms $\mathsf{TrapGen}_{\mathsf{VHF}}$ and $\mathsf{TrapEval}_{\mathsf{VHF}}$.

- $\mathsf{TrapGen}_{\mathsf{VHF}}(\Pi, [\mathbf{B}], X^{(0)})$ first chooses $k+1$ subspaces $\mathfrak{U}_i$ of $\mathbb{Z}_p^n$ (for $0 \le i \le k$), each of dimension $n-1$. Specifically,
  - the first k subspaces $\mathfrak{U}_i$ (for $0 \le i \le k-1$) are chosen independently and uniformly,
  - the last subspace $\mathfrak{U}_k$ is the subspace spanned by the first $n-1$ unit vectors. (That is, $\mathfrak{U}_k$ contains all vectors whose last component is 0.)

  Next, $\mathsf{TrapGen}_{\mathsf{VHF}}$ uniformly chooses $\mathbf{u} \in \mathbb{Z}_p^n \setminus \mathfrak{U}_0$ and $2k$ matrices $\mathbf{R}_{i,b}$ (for $1 \le i \le k$ and $b \in \{0,1\}$), as follows:

$$\begin{aligned} \mathbf{R}_{i,1-x_i^{(0)}} &\text{ uniformly of rank } n-1 \text{ subject to} & \mathfrak{U}_{i-1}^\top \cdot \mathbf{R}_{i,1-x_i^{(0)}} &= \mathfrak{U}_i^\top \\ \mathbf{R}_{i,x_i^{(0)}} &\text{ uniformly of rank } n \text{ subject to} & \mathfrak{U}_{i-1}^\top \cdot \mathbf{R}_{i,x_i^{(0)}} &= \mathfrak{U}_i^\top. \end{aligned} \tag{3}$$

  Finally, $\mathsf{TrapGen}_{\mathsf{VHF}}$ sets

$$\begin{aligned} \mathbf{M}_{i,b} &= \mathbf{R}_{i,b} \quad \text{for } 1 \le i \le k-1 \\ [\mathbf{M}_{k,0}] = [\mathbf{R}_{k,0} \cdot \mathbf{B}^\top] & \qquad [\mathbf{M}_{k,1}] = [\mathbf{R}_{k,1} \cdot \mathbf{B}^\top], \end{aligned} \tag{4}$$

  and outputs

$$td = \big((\mathbf{R}_{i,b})_{i\in[k-1], b\in\{0,1\}}, \mathbf{u}, [\mathbf{B}]\big) \qquad vk = \big(([\mathbf{M}_{i,b}])_{i\in[k], b\in\{0,1\}}, [\mathbf{u}]\big).$$

- $\mathsf{TrapEval}_{\mathsf{VHF}}(td, X)$ first computes an image $[\mathbf{v}] = [\mathbf{v}_k]$, along with a corresponding proof $\pi = [\mathbf{v}_1, \ldots, \mathbf{v}_{k-1}]$ exactly like $\mathsf{Eval}_{\mathsf{VHF}}$, using (1). (Note that $\mathsf{TrapEval}_{\mathsf{VHF}}$ can compute all $[\mathbf{v}_i]$ from its knowledge of $\mathbf{u}$, all $\mathbf{R}_{i,b}$, and $[\mathbf{B}]$.) Next, observe that the image $[\mathbf{v}]$ satisfies

$$\mathbf{v}^\top = \mathbf{v}_k^\top = \mathbf{u}^\top \cdot \prod_{j=1}^{k} \mathbf{M}_{j,x_j} = \underbrace{\Big(\mathbf{u}^\top \cdot \prod_{j=1}^{k} \mathbf{R}_{j,x_j} \Big)}_{=:\boldsymbol{\beta}^\top} \cdot \mathbf{B}^\top. \tag{5}$$

  Hence, $\mathsf{TrapEval}_{\mathsf{VHF}}$ outputs $(\boldsymbol{\beta}, \pi)$.

Lemma 4.4 (Correctness of $\mathsf{Trap}_{\mathsf{VHF}}$)**.** *The trapdoor algorithms* $\mathsf{TrapGen}_{\mathsf{VHF}}$ *and* $\mathsf{TrapEval}_{\mathsf{VHF}}$ *above satisfy correctness in the sense of Definition 3.2.*

Proof. This follows directly from (5). □

Lemma 4.5 (Well-distributedness of $\mathsf{Trap}_{\mathsf{VHF}}$)**.** *The above trapdoor algorithms* $\mathsf{TrapGen}_{\mathsf{VHF}}$ *and* $\mathsf{TrapEval}_{\mathsf{VHF}}$ *enjoy well-distributed outputs in the sense of Definition 3.2.*

Proof. Fix any preimage $X^{(0)} = (x_i^{(0)})_{i=1}^k \in \{0,1\}^k$, matrix $[\mathbf{B}] \in \mathbb{G}^{n \times n}$, and corresponding keypair $(td, vk) \xleftarrow{\$} \mathsf{TrapGen}_{\mathsf{VHF}}(\Pi, [\mathbf{B}], X^{(0)})$. We will show first that for all $X = (x_i)_{i=1}^k \in \{0,1\}^k$, the corresponding vectors $\mathbf{v}_i^\top$ computed during evaluation satisfy

$$\mathbf{u}^\top \cdot \prod_{j=1}^{i} \mathbf{R}_{i,x_i} \in \mathfrak{U}_i^\top \quad \Longleftrightarrow \quad x_j \neq x_j^{(0)} \text{ for some } j \leq i. \tag{6}$$

(6) can be proven by induction over i. The case $i = 0$ follows from the setup of $\mathbf{u} \notin \mathfrak{U}_0$. For the induction step, assume (6) holds for $i-1$. To show (6) for i, we distinguish two cases:

- If $x_i = x_i^{(0)}$, then $\mathbf{R}_{i,x_i}$ has full rank, and maps $\mathfrak{U}_{i-1}^\top$ to $\mathfrak{U}_i^\top$. Thus, $\mathbf{u}^\top \cdot \prod_{j=1}^{i} \mathbf{R}_{j,x_j} \in \mathfrak{U}_i^\top$ if and only if $\mathbf{u}^\top \cdot \prod_{j=1}^{i-1} \mathbf{R}_{j,x_j} \in \mathfrak{U}_{i-1}^\top$.[2] By the induction hypothesis, and using $x_i = x_i^{(0)}$, hence, $\mathbf{u}^\top \cdot \prod_{j=1}^{i} \mathbf{R}_{j,x_j} \in \mathfrak{U}_i^\top$ if and only if $x_j \neq x_j^{(0)}$ for some $j \leq i$. This shows (6).
- If $x_i \neq x_i^{(0)}$, then $\mathbf{R}_{i,x_i}$ has rank $n-1$. Together with $\mathfrak{U}_{i-1}^\top \cdot \mathbf{R}_{i,x_i} = \mathfrak{U}_i^\top$, this implies that in fact $(\mathbb{Z}_p^n)^\top \cdot \mathbf{R}_{i,x_i} = \mathfrak{U}_i^\top$. Hence, both directions of (6) hold.

This shows that (6) holds for all i. In particular, if we write

$$\boldsymbol{\beta}^\top = (\beta_1, \ldots, \beta_n) = \mathbf{u}^\top \cdot \prod_{j=1}^{k} \mathbf{R}_{i,x_i}$$

(as in (5)), then $\boldsymbol{\beta} \in \mathfrak{U}_k$ if and only if $X \neq X^{(0)}$. By definition of $\mathfrak{U}_k$, this means that $\beta_n = 0 \Leftrightarrow X \neq X^{(0)}$. Well-distributedness as in Definition 3.2 follows. □

Lemma 4.6 (Indistinguishability of $\mathsf{Trap}_{\mathsf{VHF}}$)**.** *If the n-rank assumption holds relative to* GrpGen*, then the above algorithms* $\mathsf{TrapGen}_{\mathsf{VHF}}$ *and* $\mathsf{TrapEval}_{\mathsf{VHF}}$ *satisfy the indistinguishability property from Definition 3.2. Specifically, for every adversary $\mathcal{A}$, there exists an adversary $\mathcal{B}$ (of roughly the same complexity) with*

$$\mathsf{Adv}^{\mathsf{vhf\text{-}sel\text{-}ind}}_{\mathsf{VHF},\mathsf{TrapGen}_{\mathsf{VHF}},\mathsf{TrapEval}_{\mathsf{VHF}},\mathcal{A}}(k) \;=\; k \cdot \mathsf{Adv}^{n\text{-}\mathsf{rank}}_{\mathbb{G},n,\mathcal{B}}(k) + \mathbf{O}(kn/p).$$

[2] Recall our notation from Section 3.

Proof. Fix an adversary $\mathcal{A}$. We proceed in games.

Game 0. Game 0 is identical to the indistinguishability game with $\overline{b} = 0$. In this game, $\mathcal{A}$ first selects a "target preimage" $X^{(0)}$, and then gets a verification key vk generated by $\mathsf{Gen}_{\mathsf{VHF}}$, and oracle access to an evaluation oracle $\mathcal{O}$. Let G_0 denote $\mathcal{A}$'s output in this game. (More generally, let G_i denote $\mathcal{A}$'s output in Game i.) Our goal will be to gradually change this setting such that finally, vk is generated by $\mathsf{TrapGen}_{\mathsf{VHF}}(\mathsf{GrpGen}, [\mathbf{B}], X^{(0)})$ (for an independently uniform invertible $\mathbf{B}$), and $\mathcal{O}$ uses the corresponding trapdoor to generate images and proofs. Of course, $\mathcal{A}$'s output must remain the same (or change only negligibly) during these transitions.

Game 1.ℓ (for $0 \le \ell \le k$). In Game 1.ℓ (for $0 \le \ell \le k$), vk is generated in part as by $\mathsf{TrapGen}_{\mathsf{VHF}}$, and in part as by $\mathsf{Gen}_{\mathsf{VHF}}$. ($\mathcal{O}$ is adapted accordingly.) Specifically, Game 1.ℓ proceeds like Game 0, except for the following changes:

- Initially, the game chooses $\ell+1$ subspaces $\mathfrak{U}_i$ (for $0 \le i \le \ell$) of dimension $n-1$ independently and uniformly, and picks $\mathbf{u} \in \mathbb{Z}_p^n \setminus \mathfrak{U}_0$. (Note that unlike in an execution of $\mathsf{TrapGen}_{\mathsf{VHF}}$, also $\mathfrak{U}_k$ is chosen uniformly when $\ell = k$.)
- Next, the game chooses $2k$ matrices $\mathbf{R}_{i,b}$ (for $1 \le i \le k$ and $b \in \{0,1\}$), as follows. For $i \le \ell$, the $\mathbf{R}_{i,b}$ are chosen as by $\mathsf{TrapGen}_{\mathsf{VHF}}$, and thus conform to (3). For $i > \ell$, the $\mathbf{R}_{i,b}$ are chosen uniformly and independently (but invertible).
- Finally, the game sets up $\mathbf{M}_{i,b} := \mathbf{R}_{i,b}$ for all i, b. (Again, note the slight difference to $\mathsf{TrapGen}_{\mathsf{VHF}}$, which follows (4).)

The game hands the resulting verification key vk to $\mathcal{A}$; since all $\mathbf{M}_{i,b}$ are known over $\mathbb{Z}_p$, oracle $\mathcal{O}$ can be implemented as $\mathsf{Eval}_{\mathsf{VHF}}$.

Now let us take a closer look at the individual games Game 1.ℓ. First, observe that Game 1.0 is essentially Game 0: all $\mathbf{M}_{i,b}$ are chosen independently and uniformly, and $\mathcal{O}$ calls are answered in the only possible way (given vk). The only difference is that $\mathbf{u}$ is chosen independently uniformly from $\mathbb{Z}_p^n \setminus \{0\}$ in Game 0, and from $\mathbb{Z}_p^n \setminus \mathfrak{U}_0$ (for a uniform dimension-$(n-1)$ subspace $\mathfrak{U}_0$) in Game 1.0. However, both choices lead to the same distribution of $\mathbf{u}$, so we obtain

$$\Pr[G_0 = 1] \;=\; \Pr[G_{1.0} = 1]\,. \tag{7}$$

Next, we investigate the change from Game 1.$(\ell-1)$ to Game 1.ℓ. We claim the following:

Lemma 4.7. *There is an adversary $\mathcal{B}$ on the n-rank problem with*

$$\sum_{\ell=1}^{k} \Pr[G_{1.\ell} = 1] - \Pr\left[G_{1.(\ell-1)} = 1\right] \;=\; k \cdot \mathsf{Adv}^{n\text{-}\mathsf{rank}}_{\mathbb{G},n,\mathcal{B}}(k) + \mathbf{O}(kn/p). \tag{8}$$

We postpone a proof of Lemma 4.7 until after the main proof.

Game 2. Finally, in Game 2, we slightly change the way $\mathfrak{U}_k$ and the matrices $\mathbf{M}_{k,b}$ (for $b \in \{0,1\}$) are set up:

- Instead of setting up $\mathfrak{U}_k$ uniformly (like all other $\mathfrak{U}_i$), we set up $\mathfrak{U}_k$ like $\mathsf{TrapGen}_{\mathsf{VHF}}$ would (i.e., as the subspace spanned by the first $n-1$ unit vectors).
- Instead of setting up $\mathbf{M}_{k,b} = \mathbf{R}_{k,b}$, we set $\mathbf{M}_{k,b} = \mathbf{R}_{k,b} \cdot \mathbf{B}^\top$ for an independently and uniformly chosen invertible $\mathbf{B}$, exactly like $\mathsf{TrapGen}_{\mathsf{VHF}}$ would.

Observe that since $\mathbf{B}$ is invertible, these modifications do not alter the distribution of the matrices $\mathbf{M}_{k,b}$ (compared to Game 1.ℓ). Indeed, in both cases, both $\mathbf{M}_{k,b}$ map $\mathfrak{U}_{k-1}^\top$ to the same uniformly chosen $(n-1)$-dimension subspace. In Game 1.ℓ, this subspace is $\mathfrak{U}_k$, while in Game 2, this subspace is the subspace spanned by the first $n-1$ columns of $\mathbf{B}$. We obtain:

$$\Pr[G_{1.\ell} = 1] \;=\; \Pr[G_2 = 1]\,. \tag{9}$$

Finally, it is left to observe that Game 2 is identical to the indistinguishability experiment with $\bar{b} = 1$: vk is prepared exactly as with $\mathsf{TrapGen}_{\mathsf{VHF}}(\Pi, [\mathbf{B}], X^{(0)})$ for a random $\mathbf{B}$, and $\mathcal{O}$ outputs the images and proofs uniquely determined by vk. Hence,

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{vhf\text{-}sel\text{-}ind}}_{\mathsf{VHF},\mathsf{Trap}_{\mathsf{VHF}},\mathcal{A}}(k) &= \Pr[G_2 = 1] - \Pr[G_0 = 1] \\
&\overset{(7),(9)}{=} \Pr[G_{1.k} = 1] - \Pr[G_{1.0} = 1] \\
&= \sum_{\ell=1}^{k} \Pr[G_{1.\ell} = 1] - \Pr\left[G_{1.(\ell-1)} = 1\right] \\
&\overset{(8)}{=} k \cdot \mathsf{Adv}^{n\text{-}\mathsf{rank}}_{\mathbb{G},n,\mathcal{A}}(k) + \mathbf{O}(kn/p)
\end{aligned}$$

as desired. □

It remains to prove Lemma 4.7.

Proof of Lemma 4.7. We describe an adversary $\mathcal{B}$ on the n-rank problem. $\mathcal{B}$ gets as input a matrix $[\mathbf{A}]$ "in the exponent," such that $\mathbf{A}$ is either of rank n, or of rank $n-1$. Initially, $\mathcal{B}$ uniformly picks $\ell \in \{1, \ldots, k\}$. Our goal is construct $\mathcal{B}$ such that it internally simulates Game 1.$(\ell-1)$ or Game 1.ℓ, depending on $\mathbf{A}$'s rank. To this end, $\mathcal{B}$ sets up vk as follows:

- Like Game 1.$(\ell-1)$, $\mathcal{B}$ chooses ℓ subspaces $\mathfrak{U}_0, \ldots, \mathfrak{U}_{\ell-1}$, and $\mathbf{u} \in \mathbb{Z}_p^n \setminus \mathfrak{U}_0$ uniformly.
- For $i < \ell$, $\mathcal{B}$ chooses matrices $\mathbf{R}_{i,b}$ like $\mathsf{TrapGen}_{\mathsf{VHF}}$ does, ensuring (3). For $i > \ell$, all $\mathbf{R}_{i,b}$ are chosen independently and uniformly but invertible. The case $i = \ell$ is more complicated and will be described next.
- To set up $\mathbf{M}_{\ell,0}$ and $\mathbf{M}_{\ell,1}$, $\mathcal{B}$ first asks $\mathcal{A}$ for its challenge input $X^{(0)} = (x_i^{(0)})_{i=1}^k$. Next, $\mathcal{B}$ embeds its own challenge $[\mathbf{A}]$ as $[\mathbf{R}_{\ell,1-x_\ell^{(0)}}] := [\mathbf{A}]$. To construct an $[\mathbf{R}_{\ell,x_\ell^{(0)}}]$ that achieves (3) (for $i = \ell$), $\mathcal{B}$ first uniformly chooses a basis $\{\mathbf{c}_1, \ldots, \mathbf{c}_n\}$ of $\mathbb{Z}_p^n$, such that $\{\mathbf{c}_1, \ldots, \mathbf{c}_{n-1}\}$ forms a basis of $\mathfrak{U}_{\ell-1}$. (Note that $\mathcal{B}$ chooses the subspace $\mathfrak{U}_{\ell-1}$ on its own and over $\mathbb{Z}_p$, so this is possible efficiently for $\mathcal{B}$.) In the sequel, let $\mathbf{C}$ be the matrix whose i-th row is $\mathbf{c}_i^\top$, and let $\mathbf{C}^{-1}$ be the inverse of $\mathbf{C}$. Jumping ahead, the purpose of $\mathbf{C}^{-1}$ is to help translate vectors from $\mathfrak{U}_{\ell-1}$ (as obtained through a partial product $\mathbf{u}^\top \prod_{j=1}^{\ell-1} \mathbf{M}_{j,x_j}$) to a "more accessible" form.
  Next, $\mathcal{B}$ samples $n-1$ random vectors $[\mathbf{c}'_i]$ (for $1 \le i \le n-1$) in the image of $[\mathbf{A}]$ (e.g., by choosing random $\mathbf{r}_i^\top$ and setting $[\mathbf{c}'_i] = \mathbf{r}_i^\top \cdot [\mathbf{A}]$). Furthermore, $\mathcal{B}$ samples $\mathbf{c}'_n \in \mathbb{Z}_p^n$ randomly. Let $[\mathbf{C}']$ be the matrix whose i-th row is $[\mathbf{c}_i'^\top]$. The purpose of $\mathbf{C}'$ is to define the image of $\mathbf{R}_{\ell,x_\ell^{(0)}}$. Specifically, $\mathcal{B}$ computes

  $$[\mathbf{R}_{\ell,x_\ell^{(0)}}] \;:=\; \mathbf{C}^{-1} \cdot [\mathbf{C}'].$$

  (Note that $\mathcal{B}$ can compute $[\mathbf{R}_{\ell,x_\ell^{(0)}}]$ efficiently, since $\mathbf{C}^{-1}$ is known "in the clear.") We will show below that, depending on the rank of $\mathbf{A}$, either $\mathfrak{U}_{\ell-1}^\top \cdot \mathbf{R}_{\ell,x_\ell^{(0)}} = \mathfrak{U}_{\ell-1}^\top \cdot \mathbf{A}$, or $\mathfrak{U}_{\ell-1}^\top \cdot \mathbf{R}_{\ell,x_\ell^{(0)}}$ is an independently random subspace of dimension $n-1$.
- Finally, $\mathcal{B}$ sets $[\mathbf{M}_{i,b}] = [\mathbf{R}_{i,b}]$ for all i, b, and hands $\mathcal{A}$ the resulting verification key vk.

Furthermore, $\mathcal{B}$ implements oracle $\mathcal{O}$ as follows: if $\mathcal{A}$ queries $\mathcal{O}$ with some $X = (x_i)_{i=1}^k \in \{0,1\}^k$, then $\mathcal{B}$ can produce the (uniquely determined) image and proof from the values

$$[\mathbf{v}_i^\top] \;=\; [\mathbf{u}^\top \cdot \prod_{j=1}^{i} \mathbf{M}_{j,x_j}]. \tag{10}$$

On the other hand, $\mathcal{B}$ can compute all $[\mathbf{v}_i^\top]$ efficiently, since it knows all factors in (10) over $\mathbb{Z}_p$, except for (at most) one factor $[\mathbf{M}_{\ell,x_\ell}]$.

Finally, $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs.

We now analyze this simulation. First, note that vk and $\mathcal{O}$ are simulated exactly as in both Game 1.$(\ell-1)$ and Game 1.ℓ, except for the definition of $[\mathbf{R}_{\ell,x_i^{(0)}}]$ and $[\mathbf{R}_{\ell,1-x_i^{(0)}}]$. Now consider how these matrices are set up depending on the rank of $\mathcal{B}$'s challenge $\mathbf{A}$.

- If $\mathbf{A}$ is of rank n, then $\mathbf{R}_{\ell,x_\ell^{(0)}}$ and $\mathbf{R}_{\ell,1-x_\ell^{(0)}}$ are (statistically close to) independently and uniformly random invertible matrices. Indeed, then each row ${\mathbf{c}'_i}^\top$ of $\mathbf{C}'$ is independently and uniformly random: $\mathbf{c}'_1,\ldots,\mathbf{c}'_{n-1}$ are independently random elements of the image of $\mathbf{A}$ (which is $\mathbb{Z}_p^n$), and $\mathbf{c}'_n$ is independently random by construction. Hence, $\mathbf{C}'$ is independently and uniformly random (and thus invertible, except with probability n/p). On the other hand, $\mathbf{R}_{\ell,x_\ell^{(0)}}=\mathbf{A}$ is uniformly random and invertible by assumption.
- If $\mathbf{A}$ is of rank $n-1$, then $\mathbf{R}_{\ell,x_\ell^{(0)}}$ and $\mathbf{R}_{\ell,1-x_\ell^{(0)}}$ are (statistically close to) distributed as in (3). Indeed, then the rank of $\mathbf{R}_{\ell,1-x_\ell^{(0)}}=\mathbf{A}$ is $n-1$, and the rank of $\mathbf{R}_{\ell,x_\ell^{(0)}}=\mathbf{C}^{-1}\cdot\mathbf{C}'$ is n, except with probability at most n/p.[3] Moreover, if we write $\mathfrak{U}_\ell^\top := (\mathbb{Z}_p^n)^\top\cdot\mathbf{A}$, then by construction $(\mathbb{Z}_p^n)^\top\cdot\mathbf{R}_{\ell,1-x_\ell^{(0)}}=\mathfrak{U}_\ell^\top$, but also
$$(\mathbb{Z}_p^n)^\top\cdot\mathbf{R}_{\ell,x_\ell^{(0)}}=\mathfrak{W}^\top\cdot\mathbf{C}'=\mathfrak{U}_\ell^\top,$$
where $\mathfrak{W}$ is the vector space spanned by the first $n-1$ unit vectors. Furthermore, $\mathbf{R}_{\ell,x_\ell^{(0)}}$ and $\mathbf{R}_{\ell,1-x_\ell^{(0)}}$ are distributed uniformly with these properties.

Hence, summarizing, up to a statistical defect of at most $2/p$, $\mathcal{B}$ simulates Game 1.$(\ell-1)$ if $\mathbf{A}$ is of rank n, and Game 1.ℓ if $\mathbf{A}$ is of rank $n-1$. This shows (8). □

4.3 Adaptive Security

The idea behind the adaptively-secure construction is very similar to the selective case. Both the construction and the security proof are essentially identical, except that we apply an *admissible hash function* (AHF) $\mathsf{AHF}:\{0,1\}^k\to\{0,1\}^{\ell_{\mathsf{AHF}}}$ (cf. Definition 4.8) to the inputs X of $\mathsf{Eval}_{\mathsf{VHF}}$ and $\mathsf{TrapEval}_{\mathsf{VHF}}$ before computing the matrix products. (We mention that suitable AHFs with $\ell_h=\mathbf{O}(k)$ exist [35, 24].) Correctness and unique provability follow immediately. In order to prove well-distributedness, we rely on the properties of the admissible hash function. By a slightly more careful, AHF-dependent embedding of low-rank matrices in the verification key, these properties ensure that, for any sequence of queries issued by the adversary, it holds with non-negligible probability that the vector $\left[\mathbf{v^{(0)}}\right]$ assigned to input $X^{(0)}$ does not lie in the subspace generated by $(\mathbf{b_1},\ldots,\mathbf{b_{n-1}})$, while all vectors $\left[\mathbf{v^{(i)}}\right]$ assigned to input $X^{(i)}$ do, which then yields the required well-distributedness property.

Admissible hash functions. To obtain adaptive security, we rely on a semi-blackbox technique based on admissible hash functions (AHFs, [35, 9, 14, 3, 24]). In the following, we use the formalization of AHFs from [24]:

Definition 4.8 (AHF)**.** *For a function* $\mathsf{AHF}:\{0,1\}^k\to\{0,1\}^{\ell_{\mathsf{AHF}}}$ *(for a polynomial* $\ell_{\mathsf{AHF}}=\ell_{\mathsf{AHF}}(k)$*) and* $K\in(\{0,1,\bot\})^{\ell_{\mathsf{AHF}}}$*, define the function* $F_K:\{0,1\}^k\to\{\mathtt{CO},\mathtt{UN}\}$ *through*
$$F_K(X)=\mathtt{UN}\iff\forall i: K_i=\mathsf{AHF}(X)_i\vee K_i=\bot,$$

[3] To see this, observe that except with probability $(n-1)/p$, the first $n-1$ columns of $\mathbf{C}'$ are linearly independent (as they are random elements in the image of the rank-$(n-1)$ matrix $\mathbf{A}$). Further, the last row (which is independently and uniformly random) does not lie in the span of the first $n-1$ rows except with probability at most $1/p$.

where $\mathsf{AHF}(X)_i$ *denotes the* i*-th component of* $\mathsf{AHF}(X)$*. We say that* AHF *is* q*-admissible if there exists a PPT algorithm* KGen *and a polynomial* $\mathsf{poly}(k)$*, such that for all* $X^{(0)}, \ldots, X^{(q)} \in \{0,1\}^k$ *with* $X^{(0)} \notin \{X^{(i)}\}$*,*

$$\Pr\left[F_K(X^{(0)}) = \mathtt{UN} \wedge F_K(X^{(1)}) = \cdots = F_K(X^{(q)}) = \mathtt{CO}\right] \geq 1/\mathsf{poly}(k), \tag{11}$$

where the probability is over $K \xleftarrow{\$} \mathsf{KGen}(1^k)$*. We say that* AHF *is an admissible hash function (AHF) if* AHF *is* q*-admissible for all polynomials* $q = q(k)$*.*

There are efficient constructions of admissible hash functions [35, 24] with $\ell_{\mathsf{AHF}} = \mathbf{O}(k)$ from error-correcting codes.

A hashed variant of VHF**.** Fix an AHF $\mathsf{AHF} : \{0,1\}^k \to \{0,1\}^{\ell_{\mathsf{AHF}}}$ and a corresponding KGen algorithm. Essentially, we will hash preimages (using AHF) before feeding them into VHF to obtain a slight variant VHF' of VHF that we can prove adaptively secure. More specifically, let VHF' be the verifiable hash function that is defined like VHF, except for the following differences:

- $\mathsf{Gen}'_{\mathsf{VHF}}(\mathsf{GrpGen})$ proceeds like $\mathsf{Gen}_{\mathsf{VHF}}(\mathsf{GrpGen})$, but samples $2\ell_{\mathsf{AHF}}$ (not $2k$) matrices $\mathbf{M}_{i,b}$.
- $\mathsf{Eval}'_{\mathsf{VHF}}(ek, X)$ (for $X \in \{0,1\}^k$), first computes $X' = (x'_i)_{i=1}^{\ell_{\mathsf{AHF}}} = \mathsf{AHF}(X) \in \{0,1\}^{\ell_{\mathsf{AHF}}}$, and then outputs an image $[\mathbf{v}] = [\mathbf{v}_{\ell_{\mathsf{AHF}}}]$ and a proof

$$\pi = ([\mathbf{v}_1], \ldots, [\mathbf{v}_{\ell_{\mathsf{AHF}}-1}])$$

  where $\mathbf{v}_i^\top = \mathbf{u}^\top \cdot \prod_{j=1}^{i} \mathbf{M}_{j,x'_j}$.
- $\mathsf{Vfy}'_{\mathsf{VHF}}(vk, [\mathbf{v}], \pi, X)$ computes $X' = (x'_i)_{i=1}^{\ell_{\mathsf{AHF}}} = \mathsf{AHF}(X) \in \{0,1\}^{\ell_{\mathsf{AHF}}}$ and outputs 1 if and only if $e([\mathbf{v}_i^\top], [1]) = e([\mathbf{v}_{i-1}^\top], [\mathbf{M}_{i,x'_i}])$ holds for all i with $1 \leq i \leq \ell_{\mathsf{AHF}}$, where $[\mathbf{v}_0] := [\mathbf{u}]$ and $[\mathbf{v}_{\ell_{\mathsf{AHF}}}] := [\mathbf{v}]$.

Theorem 4.9 (Adaptive programmabability of VHF'). VHF' *is adaptively programmable in the sense of Definition 3.3.*

The trapdoor algorithms. We proceed similarly to the selective case and start with a description of the algorithms $\mathsf{TrapGen}'_{\mathsf{VHF}}$ and $\mathsf{TrapEval}'_{\mathsf{VHF}}$.

- $\mathsf{TrapGen}'_{\mathsf{VHF}}(\Pi, [\mathbf{B}])$ proceeds like algorithm $\mathsf{TrapGen}_{\mathsf{VHF}}$ from Section 4.2, except that
  - $\mathsf{TrapGen}'_{\mathsf{VHF}}$ initializes $K \xleftarrow{\$} \mathsf{KGen}(1^k)$ and includes K in td.
  - $\mathsf{TrapGen}'_{\mathsf{VHF}}$ chooses $\ell_{\mathsf{AHF}} + 1$ (and not $k+1$) subspaces $\mathfrak{U}_i$ (for $0 \leq i \leq \ell_{\mathsf{AHF}}$). (The last subspace $\mathfrak{U}_{\ell_{\mathsf{AHF}}}$ is chosen in a special way, exactly like $\mathfrak{U}_k$ is chosen by $\mathsf{TrapGen}_{\mathsf{VHF}}$.)
  - $\mathsf{TrapGen}'_{\mathsf{VHF}}$ chooses $2\ell_{\mathsf{AHF}}$ (and not $2k$) matrices $\mathbf{R}_{i,b}$ (for $1 \leq i \leq \ell_{\mathsf{AHF}}$ and $b \in \{0,1\}$), as follows:
    - If $K_i = b$, then $\mathbf{R}_{i,b}$ is chosen uniformly of rank $n-1$, subject to

$$\mathfrak{U}_{i-1}^\top \cdot \mathbf{R}_{i,1-x_i^{(0)}} = \mathfrak{U}_i^\top$$

    - If $K_i \neq b$, then $\mathbf{R}_{i,b}$ is chosen uniformly of rank n, subject to

$$\mathfrak{U}_{i-1}^\top \cdot \mathbf{R}_{i,x_i^{(0)}} = \mathfrak{U}_i^\top$$

- $\mathsf{TrapEval}'_{\mathsf{VHF}}(td, X)$ proceeds like algorithm $\mathsf{TrapEval}_{\mathsf{VHF}}$ on input a preimage $\mathsf{AHF}(X) \in \{0,1\}^{\ell_{\mathsf{AHF}}}$. Specifically, $\mathsf{TrapEval}'_{\mathsf{VHF}}$ computes $[\mathbf{v}] = [\mathbf{v}_k]$, along with a corresponding proof $\pi = [\mathbf{v}_1, \ldots, \mathbf{v}_{k-1}]$ exactly like $\mathsf{Eval}'_{\mathsf{VHF}}$. Finally, and analogously to $\mathsf{TrapEval}_{\mathsf{VHF}}$, $\mathsf{TrapEval}'_{\mathsf{VHF}}$ outputs $(\boldsymbol{\beta}, \pi)$ for $\boldsymbol{\beta}^\top := \mathbf{u}^\top \cdot \prod_{j=1}^{k} \mathbf{R}_{j,x_j}$

Correctness and indistinguishability follow as for $\mathsf{TrapGen}_{\mathsf{VHF}}$ and $\mathsf{TrapEval}_{\mathsf{VHF}}$, so we state without proof:

Lemma 4.10 (Correctness of $\mathsf{Trap}'_{\mathsf{VHF}}$)**.** *The trapdoor algorithms* $\mathsf{TrapGen}'_{\mathsf{VHF}}$ *and* $\mathsf{TrapEval}'_{\mathsf{VHF}}$ *above satisfy correctness in the sense of Definition 3.3.*

Lemma 4.11 (Indistinguishability of $\mathsf{Trap}'_{\mathsf{VHF}}$)**.** *If the* n*-rank assumption holds relative to* GrpGen*, then the above algorithms* $\mathsf{TrapGen}'_{\mathsf{VHF}}$ *and* $\mathsf{TrapEval}'_{\mathsf{VHF}}$ *satisfy the indistinguishability property from Definition 3.3. Specifically, for every adversary* $\mathcal{A}$*, there exists an adversary* $\mathcal{B}$ *(of roughly the same complexity) with*

$$\mathsf{Adv}^{\mathsf{vhf\text{-}sel\text{-}ind}}_{\mathsf{VHF}',\mathsf{Trap}'_{\mathsf{VHF}},\mathcal{A}}(k) \;=\; \ell_{\mathsf{AHF}} \cdot \mathsf{Adv}^{n\text{-}\mathsf{rank}}_{\mathbb{G},n,\mathcal{B}}(k) + \mathbf{O}(\ell_{\mathsf{AHF}} n/p).$$

The (omitted) proof of Lemma 4.11 proceeds exactly like that Lemma 4.6, only adapted to AHF-hashed inputs. Note that the additional oracle $\mathcal{O}_{\mathsf{check}}$ an adversary gets in the adaptive indistinguishability game can be readily implemented with the key K generated by $\mathsf{TrapGen}'_{\mathsf{VHF}}$. (The argument from the proof of Lemma 4.6 does not rely on a secret $X^{(0)}$, and so its straightforward adaptation could even expose the full key K to an adversary.)

Lemma 4.12 (Well-distributedness of $\mathsf{Trap}'_{\mathsf{VHF}}$)**.** *The above trapdoor algorithms* $\mathsf{TrapGen}'_{\mathsf{VHF}}$ *and* $\mathsf{TrapEval}'_{\mathsf{VHF}}$ *have well-distributed outputs in the sense of Definition 3.3.*

Proof. First, we make an observation. Fix a matrix $[\mathbf{B}]$, and a corresponding keypair $(td, vk) \xleftarrow{\$} \mathsf{TrapGen}_{\mathsf{VHF}}(\Pi, [\mathbf{B}])$. Like (6), we can show that for all $X' = (x'_i)_{i=1}^{\ell_{\mathsf{AHF}}}$, the corresponding vectors $\mathbf{v}_i^\top$ computed during evaluation satisfy

$$\mathbf{u}^\top \cdot \prod_{j=1}^{i} \mathbf{R}_{i,x_i} \in \mathfrak{U}_i^\top \quad\iff\quad x_j = K_j \text{ for some } j \le i.$$

Hence, $\boldsymbol{\beta} \in \mathfrak{U}_{\ell_{\mathsf{AHF}}}$ (and thus $\beta_n = 0$) for the value $\boldsymbol{\beta}$ that is computed by $\mathsf{TrapEval}'_{\mathsf{VHF}}(td, X)$ if and only if $F_K(X) = \mathtt{C0}$. By property (11) of AHF, the lemma follows. □

5 VRFs from Verifiable PVHFs

Let $(\mathsf{Gen}_{\mathsf{VRF}}, \mathsf{Eval}_{\mathsf{VRF}}, \mathsf{Vfy}_{\mathsf{VRF}})$ be the following algorithms.

- Algorithm $(vk, sk) \xleftarrow{\$} \mathsf{Gen}_{\mathsf{VRF}}(1^k)$ takes as input a security parameter k and outputs a key pair (vk, sk). We say that sk is the *secret key* and vk is the *verification key.*
- Algorithm $(Y, \pi) \xleftarrow{\$} \mathsf{Eval}_{\mathsf{VRF}}(sk, X)$ takes as input secret key sk and $X \in \{0,1\}^k$, and outputs a function value $Y \in \mathcal{Y}$, where $\mathcal{Y}$ is a finite set, and a proof π. We write $V_{sk}(X)$ to denote the function value Y computed by $\mathsf{Eval}_{\mathsf{VRF}}$ on input (sk, X).
- Algorithm $\mathsf{Vfy}_{\mathsf{VRF}}(vk, X, Y, \pi) \in \{0,1\}$ takes as input verification key vk, $X \in \{0,1\}^k$, $Y \in \mathcal{Y}$, and proof π, and outputs a bit.

Definition 5.1. *We say that a tuple of algorithms* $(\mathsf{Gen}_{\mathsf{VRF}}, \mathsf{Eval}_{\mathsf{VRF}}, \mathsf{Vfy}_{\mathsf{VRF}})$ *is a* verifiable random function *(VRF), if all the following properties hold.*

Correctness. *Algorithms* $\mathsf{Gen}_{\mathsf{VRF}}$*,* $\mathsf{Eval}_{\mathsf{VRF}}$*,* $\mathsf{Vfy}_{\mathsf{VRF}}$ *are polynomial-time algorithms, and for all* $(vk, sk) \xleftarrow{\$} \mathsf{Gen}_{\mathsf{VRF}}(1^k)$ *and all* $X \in \{0,1\}^k$ *holds: if* $(Y, \pi) \xleftarrow{\$} \mathsf{Eval}_{\mathsf{VRF}}(sk, X)$*, then we have* $\mathsf{Vfy}_{\mathsf{VRF}}(vk, X, Y, \pi) = 1$ *.*

Unique provability. *For all strings* (vk, sk) *(not necessarily generated by* $\mathsf{Gen}_{\mathsf{VRF}}$*) and all* $X \in \{0,1\}^k$*, there does not* exist *any* (Y_0, π_0, Y_1, π_1) *such that* $Y_0 \neq Y_1$ *and* $\mathsf{Vfy}_{\mathsf{VRF}}(vk, X, Y_0, \pi_0) = \mathsf{Vfy}_{\mathsf{VRF}}(vk, X, Y_1, \pi_1) = 1$*.*

Pseudorandomness. *Let* $\mathsf{Exp}^{\mathsf{VRF}}_{\mathcal{B}}$ *be the security experiment defined in Figure 2, played with adversary* $\mathcal{B}$*. We require that the advantage function*

$$\mathsf{Adv}^{\mathsf{VRF}}_{\mathcal{B}}(k) := 2 \cdot \Pr\left[\mathsf{Exp}^{\mathsf{VRF}}_{\mathcal{B}}(1^k) = 1\right] - 1$$

is negligible for all PPT $\mathcal{B}$ *that never query* **Evaluate** *on input* X^**.*

Initialize(1^k) :	**Challenge**(X^*) :	$\mathsf{Exp}^{\mathsf{VRF}}_{\mathcal{A}}(1^k)$:
$b \xleftarrow{\$} \{0,1\}$	$(Y_0, \pi) \xleftarrow{\$} \mathsf{Eval}_{\mathsf{VRF}}(sk, X^*)$	$vk \xleftarrow{\$}$ **Initialize**(1^k)
$(vk, sk) \xleftarrow{\$} \mathsf{Gen}_{\mathsf{VRF}}(1^k)$	$Y_1 \xleftarrow{\$} \mathcal{Y}$	$X^* \xleftarrow{\$} \mathcal{A}^{\mathbf{Evaluate}}$
Return vk	**Return** Y_b	$Y_b \xleftarrow{\$}$ **Challenge**(X^*)
		$B \xleftarrow{\$} \mathcal{A}^{\mathbf{Evaluate}}$
Evaluate(X) :		**Return** $(B = b)$
$(Y, \pi) \xleftarrow{\$} \mathsf{Eval}_{\mathsf{VRF}}(sk, X)$		
Return (Y, π)		

Figure 2: The VRF security experiment.

5.1 A Generic Construction from Verifiable PVHFs

Let $(\mathsf{Gen}_{\mathsf{VHF}}, \mathsf{Eval}_{\mathsf{VHF}}, \mathsf{Vfy}_{\mathsf{VHF}})$ be a vector hash function according to Definition 3.1, and let $(\mathsf{GrpGen}, \mathsf{GrpVfy})$ be a certified bilinear group generator according to Definitions 2.1 and 2.2. Let $(\mathsf{Gen}_{\mathsf{VRF}}, \mathsf{Eval}_{\mathsf{VRF}}, \mathsf{Vfy}_{\mathsf{VRF}})$ be the following algorithms.

Key generation. $\mathsf{Gen}_{\mathsf{VRF}}(1^k)$ runs $\Pi \xleftarrow{\$} \mathsf{GrpGen}(1^k)$ to generate bilinear group parameters, and then $(ek, vk') \xleftarrow{\$} \mathsf{Gen}_{\mathsf{VHF}}(\Pi)$. Then it chooses a random vector $\mathbf{w} \xleftarrow{\$} (\mathbb{Z}_p^*)^n$, defines $sk := (\Pi, ek, \mathbf{w})$ and $vk := (\Pi, vk', [\mathbf{w}])$, and outputs (vk, sk).

Function evaluation. On input $sk := (\Pi, ek, \mathbf{w})$ with $\mathbf{w} = (w_1, \ldots, w_n)^\top \in (\mathbb{Z}_p^*)^n$ and $X \in \{0,1\}^k$, algorithm $\mathsf{Eval}_{\mathsf{VRF}}(sk, X)$ first runs

$$([\mathbf{v}], \pi') \leftarrow \mathsf{Eval}_{\mathsf{VHF}}(ek, X).$$

Then it computes the function value Y and an additional proof $[\mathbf{z}] \in \mathbb{G}^n$ as

$$Y := \prod_{i=1}^{n} \left[\frac{v_i}{w_i}\right] \qquad \text{and} \qquad [\mathbf{z}] := \left[(z_1, \ldots, z_n)^\top\right] := \left[\left(\frac{v_1}{w_1}, \ldots, \frac{v_n}{w_n}\right)^\top\right]$$

Finally, it sets $\pi := ([\mathbf{v}], \pi', [\mathbf{z}])$ and outputs (Y, π).

Proof verification. On input (vk, X, Y, π), $\mathsf{Vfy}_{\mathsf{VRF}}$ outputs 0 if any of the following properties is not satisfied.

1. vk has the form $vk = (\Pi, vk', [\mathbf{w}])$, such that $[\mathbf{w}] = ([w_1], \ldots, [w_n])$ and the bilinear group parameters and group elements contained in vk are valid. That is, it holds that $\mathsf{GrpVfy}(\Pi) = 1$ and $\mathsf{GrpVfy}(\Pi, [w_i]) = 1$ for all $i \in \{1, \ldots, n\}$.
2. $X \in \{0,1\}^k$.
3. π has the form $\pi = ([\mathbf{v}], \pi', [\mathbf{z}])$ with $\mathsf{Vfy}_{\mathsf{VHF}}(vk', [\mathbf{v}], \pi', X) = 1$ and both vectors $[\mathbf{v}]$ and $[\mathbf{z}]$ contain only validly-encoded group elements, which can be checked by running GrpVfy.
4. It holds that and $[z_i] = [v_i/w_i]$ for all $i \in \{1, \ldots, n\}$ and $Y = [\sum_{i=1}^n v_i/w_i]$. This can be checked by testing

$$e([z_i], [w_i]) \stackrel{?}{=} e([v_i], [1]) \quad \forall i \in \{1, \ldots, n\} \qquad \text{and} \qquad Y \stackrel{?}{=} \prod_{i=1}^{n} [z_i]$$

If all the above checks are passed, then $\mathsf{Vfy}_{\mathsf{VRF}}$ outputs 1.

5.2 Correctness, Unique Provability, and Pseudorandomness

Theorem 5.2 (Correctness and unique provability)**.** *The triplet of algorithms* $(\mathsf{Gen}_{\mathsf{VRF}}, \mathsf{Eval}_{\mathsf{VRF}}, \mathsf{Vfy}_{\mathsf{VRF}})$ *forms a correct verifiable random function, and it satisfies the* unique provability *requirement in the sense of Definition 5.1.*

Proof. Correctness is straightforward to verify, therefore we turn directly to unique provability. We have to show that there does not *exist* any (Y_0, π_0, Y_1, π_1) such that $Y_0 \neq Y_1$ and $\mathsf{Vfy}_{\mathsf{VRF}}(vk, X, Y_0, \pi_0) = \mathsf{Vfy}_{\mathsf{VRF}}(vk, X, Y_1, \pi_1) = 1$. Let us first make the following observations.

- First of all, note that $\mathsf{Vfy}_{\mathsf{VRF}}$ on input $((\Pi, vk', [\mathbf{w}]), X, Y, ([\mathbf{v}], \pi', (\mathbf{z}))$ checks whether Π contains valid certified bilinear group parameters by running $\mathsf{GrpVfy}(\Pi)$. Moreover, it checks whether all group elements contained in $[\mathbf{w}]$, $[\mathbf{v}]$, and $[\mathbf{z}]$ are valid group elements with respect to Π. Thus, we may assume in the sequel that all these group elements are valid and have a unique encoding. In particular, $[\mathbf{w}]$ is uniquely determined by vk.
- Furthermore, it is checked that $X \in \{0,1\}^k$. The unique provability property of the vector hash function $(\mathsf{Gen}_{\mathsf{VHF}}, \mathsf{Eval}_{\mathsf{VHF}}, \mathsf{Vfy}_{\mathsf{VHF}})$ guarantees that for all strings $vk' \in \{0,1\}^*$ and all $X \in \{0,1\}^k$ there does not exist any tuple $([\mathbf{v}_0], \pi_0, [\mathbf{v}_1], \pi_1)$ with $[\mathbf{v}_0] \neq [\mathbf{v}_1]$ and $[\mathbf{v}_0], [\mathbf{v}_1] \in \mathbb{G}^n$ such that
$$\mathsf{Vfy}_{\mathsf{VHF}}(vk', [\mathbf{v}_0], \pi_0, X) = \mathsf{Vfy}_{\mathsf{VHF}}(vk', [\mathbf{v}_1], \pi_1, X) = 1$$
Thus, we may henceforth use that there is only one unique vector of group elements $[\mathbf{v}]$ which passes the test $\mathsf{Vfy}_{\mathsf{VHF}}(vk', [\mathbf{v}], \pi', X) = 1$ performed by $\mathsf{Vfy}_{\mathsf{VRF}}$. Thus, $[\mathbf{v}]$ is uniquely determined by X and the values Π and vk' contained in vk.
- Finally, note that $\mathsf{Vfy}_{\mathsf{VRF}}$ tests whether $[z_i] = [v_i / w_i]$ holds. Due to the fact that the bilinear group is certified, which guarantees that each group element has a unique encoding and that the bilinear map is non-degenerate, for each $i \in \{1, \ldots, n\}$ there exists only one unique group element encoding $[z_i]$ such that the equality $[z_i] = [v_i / w_i]$ holds.

Therefore the value $Y = \prod_{i=1}^{n} [z_i]$ is uniquely determined by $[\mathbf{v}]$ and $[\mathbf{w}]$, which in turn are uniquely determined by X and the verification key vk. □

Assumption 5.3. *The* $(n-1)$-linear assumption *states that* $[\mathbf{c}, \mathbf{d}, \sum_{i=1}^{n} d_i/c_i] \overset{\mathrm{c}}{\approx} [\mathbf{c}, \mathbf{d}, r]$*, where* $\mathbf{c} = (c_1, \ldots, c_n)^\top \in (\mathbb{Z}_p^*)^n$*,* $\mathbf{d} = (d_1, \ldots, d_n)^\top \in \mathbb{Z}_p^n$*, and* $r \in \mathbb{Z}_p$ *are uniformly random. That is, we require that*

$$\mathsf{Adv}_{\mathcal{A}}^{(n-1)\text{-}\mathsf{lin}}(k) := \Pr\left[\mathcal{A}([\mathbf{c}, \mathbf{d}, \sum_{i=1}^{n} d_i/c_i]) = 1\right] - \Pr\left[\mathcal{A}([\mathbf{c}, \mathbf{d}, r]) = 1\right]$$

is negligible for every PPT adversary $\mathcal{A}$*.*

We remark that the above formulation is an equivalent formulation of the standard $(n-1)$-linear assumption, cf. [21, Page 9], for instance.

Theorem 5.4 (Pseudorandomness)**.** *If* $(\mathsf{Gen}_{\mathsf{VHF}}, \mathsf{Eval}_{\mathsf{VHF}}, \mathsf{Vfy}_{\mathsf{VHF}})$ *is an adaptivly programmable VHF in the sense of Definition 3.1 and the* $(n-1)$*-linear assumption holds relative to* GrpGen*, then algorithms* $(\mathsf{Gen}_{\mathsf{VRF}}, \mathsf{Eval}_{\mathsf{VRF}}, \mathsf{Vfy}_{\mathsf{VRF}})$ *form a verifiable random function which satisfies the* pseudorandomness *requirement in the sense of Definition 5.1.*

Proof sketch. The proof is based on a reduction to the indistinguishability and well-distributedness of the programmable vector hash function. The well-distributedness yields a leverage to embed the given instance of the $(n-1)$-linear assumption in the view of the adversary, following the approach sketched already in the introduction. Given the PVHF as a powerful building block, the remaining main difficulty of the proof lies in dealing with the fact that the

"partitioning" proof technique provided by PVHFs is incompatible with "decisional" complexity assumptions. This is a well-known difficulty, which appeared in many previous works. It stems from the fact that different sequences of queries of the VRF-adversary may lead to different abort probabilities in the security proof. We can overcome this issue by employing the standard *artificial abort* technique [44], which has also been used to prove security of Waters' IBE scheme [44] and the VRF of Hohenberger and Waters [28], for example.

Proof. Let $\mathcal{A}$ be an adversary in the VRF security experiment from Definition 5.1. We will construct an adversary $\mathcal{B}$ on the $(n-1)$-linear assumption, which simulates the VRF pseudorandomness security experiment for $\mathcal{A}$. However, before we can construct this adversary, we have to make some changes to the security experiment. Consider the following sequence of games, where we let $\mathsf{Exp}^i_{\mathcal{A}}(1^k)$ denote the experiment executed in Game i and we write $\mathsf{Adv}^i_{\mathcal{A}}(k) := \Pr\left[\mathsf{Exp}^i_{\mathcal{A}}(1^k) = 1\right]$ to denote the advantage of $\mathcal{A}$ in Game i.

Game 0. This is the original VRF security experiment, executed with algorithms ($\mathsf{Gen}_{\mathsf{VRF}}$, $\mathsf{Eval}_{\mathsf{VRF}}, \mathsf{Vfy}_{\mathsf{VRF}}$) as constructed above. Clearly, we have

$$\mathsf{Adv}^0_{\mathcal{A}}(k) = \mathsf{Adv}^{\mathsf{VRF}}_{\mathcal{A}}(k)$$

In the sequel we write vk'_0 to denote the VHF-key generated by $(vk'_0, ek) \xleftarrow{\$} \mathsf{Gen}_{\mathsf{VHF}}(\Pi)$ in the experiment.

Game 1. This game proceeds exactly as before, except that it additionally samples a uniformly random invertible matrix $\mathbf{B} \xleftarrow{\$} \mathrm{GL}_n(\mathbb{Z}_p)$ and generates an *additional* key for the vector hash function as $(vk'_1, td) \xleftarrow{\$} \mathsf{TrapGen}_{\mathsf{VHF}}(\Pi, [\mathbf{B}])$, which is *not* given to the adversary. That is, the adversary in Game 1 receives as input a VRF verification key $vk = (\Pi, vk'_0, [\mathbf{w}])$, where vk'_0 is generated by $\mathsf{Gen}_{\mathsf{VHF}}$, exactly as in Game 0.

Whenever $\mathcal{A}$ issues an $\mathbf{Evaluate}(X^{(i)})$-query on some input $X^{(i)}$, the experiment proceeds as in Game 0, and additionally computes $((\beta_1, \ldots, \beta_n), \pi) \xleftarrow{\$} \mathsf{TrapEval}_{\mathsf{VHF}}(td, X^{(i)})$. If $\beta_n \neq 0$, then the experiment aborts and outputs a random bit. Moreover, when $\mathcal{A}$ issues a $\mathbf{Challenge}(X^{(0)})$-query, then the experiment computes $((\beta_1, \ldots, \beta_n), \pi) \xleftarrow{\$} \mathsf{TrapEval}_{\mathsf{VHF}}(td, X^{(0)})$. If $\beta_n = 0$, then the experiment aborts and outputs a random bit.

The well-distributedness of the PVHF guarantees that there is a polynomial poly such that for all possible queries $X^{(0)}, X^{(1)}, \ldots, X^{(q)}$ the probability that the experiment is not aborted is at least

$$\Pr\left[\beta^{(0)}_n = 0 \wedge \beta^{(i)}_n \neq 0 \ \forall i \in \{1, \ldots, q\}\right] \geq 1/\mathsf{poly}(k) \geq \lambda$$

where λ is a non-negligible lower bound on the probability of not aborting.

Artificial abort. Note that the probability that the experiment aborts depends on the particular sequence of queries issued by $\mathcal{A}$. This is problematic, because different sequences of queries may have different abort probabilities (cf. Appendix A). Therefore the experiment in Game 1 performs an additional *artificial abort* step, which ensures that the experiment is aborted with always the (almost) same probability $1 - \lambda$, independent of the particular sequence of queries issued by $\mathcal{A}$. To this end, the experiment proceeds as follows.

After $\mathcal{A}$ terminates and outputs a bit B, the experiment estimates the concrete abort probability $\eta(\mathbf{X})$ for the sequence of queries $\mathbf{X} := (X^{(0)}, \ldots, X^{(q)})$ issued by $\mathcal{A}$. To this end, the experiment:

1. Computes an estimate η' of $\eta(\mathbf{X})$, by R-times repeatedly sampling trapdoors $(vk'_j, td_j) \xleftarrow{\$} \mathsf{TrapGen}_{\mathsf{VHF}}(\Pi, [\mathbf{B}])$ and checking whether $\beta^{(0)}_n = 0$ or $\beta^{(i)}_n \neq 0$, where

$$((\beta^{(i)}_1, \ldots, \beta^{(i)}_n), \pi) \leftarrow \mathsf{TrapEval}_{\mathsf{VHF}}(td_j, X^{(i)}) \quad \text{for} \quad i \in \{0, \ldots, q\}$$

for sufficiently large R. Here ϵ is defined such that $2 \cdot \epsilon$ is a lower bound on the advantage of $\mathcal{A}$ in the original security experiment.

2. If $\eta' \geq \lambda$, then the experiment aborts artificially with probability $(\eta' - \lambda)/\eta'$, and outputs a random bit.

Note that if η' was *exact*, that is, $\eta' = \eta(\mathbf{X})$, then the total probability of *not* aborting would *always* be $\eta(\mathbf{X}) \cdot (1 - (\eta' - \lambda)/\eta') = \lambda$, independent of the particular sequence of queries issued by $\mathcal{A}$. In this case we would have $\mathsf{Adv}^1_{\mathcal{A}}(k) = \lambda \cdot \mathsf{Adv}^0_{\mathcal{A}}(k)$. However, the estimate η' of $\eta(\mathbf{X})$ is not necessarily exact. By applying the standard analysis technique from [44] (see also [17, 28]), one can show that setting $R := \mathbf{O}(\epsilon^{-2} \ln(1/\epsilon)\lambda^{-1} \ln(1/\lambda))$ is sufficient to obtain

$$\mathsf{Adv}^1_{\mathcal{A}}(k) \geq \mathbf{O}(\epsilon \cdot \lambda) \cdot \mathsf{Adv}^0_{\mathcal{A}}(k)$$

Game 2. The experiment now provides the adversary with the *trapdoor* VHF verification key vk'_1, by including it in the VRF verification key $vk = (\Pi, vk'_1, [\mathbf{z}])$ in place of vk'_0. Moreover, the experiment now evaluates the VHF on inputs X by running $(\boldsymbol{\beta}, \pi) \xleftarrow{\$} \mathsf{TrapEval}_{\mathsf{VHF}}(td, X)$ and then computing $[\mathbf{v}] := [\mathbf{B}] \cdot \boldsymbol{\beta}$. The rest of the experiment proceeds exactly as before.

We claim that any adversary $\mathcal{A}$ distinguishing Game 2 from Game 1 implies an adversary $\mathcal{B}$ breaking the indistinguishability of the VHF according to Definition 3.3. Adversary $\mathcal{B}^{\mathcal{O}_b, \mathcal{O}_{\mathsf{check}}}(vk'_b)$ receives as input a verification key vk'_b, which is either generated by $\mathsf{Gen}_{\mathsf{VHF}}(\Pi)$ or $\mathsf{TrapGen}_{\mathsf{VHF}}(\Pi, [\mathbf{B}])$ for a uniformly invertible random matrix $\mathbf{B}$. It simulates the security experiment from Game 2 for $\mathcal{A}$ as follows.

- The given VHF verification key vk'_b is embedded in the VRF verification key $vk = (\Pi, vk'_b, [\mathbf{z}])$, where b is the random bit chosen by the indistinguishability security experiment played by $\mathcal{B}$. All other values are computed exactly as before.
- In order to evaluate the VHF on input X, $\mathcal{B}$ is able to query its oracle $\mathcal{O}_b$, which either computes and returns $([\mathbf{v}], \pi) \leftarrow \mathsf{Eval}_{\mathsf{VHF}}(ek, X)$ (in case $b = 0$), or it computes $(\boldsymbol{\beta}, \pi) \leftarrow \mathsf{TrapEval}_{\mathsf{VHF}}(td, X)$ and returns $[\mathbf{v}] := [\mathbf{B}] \cdot \boldsymbol{\beta}$ (in case $b = 1$).
- To test whether a given value X requires a (non-artificial) abort, $\mathcal{B}$ queries $\mathcal{O}_{\mathsf{check}}(X)$, which returns 1 if and only if $((\beta_1, \ldots, \beta_n), \pi) \leftarrow \mathsf{TrapEval}_{\mathsf{VHF}}(td)$ with $\beta_n \neq 0$.
- The artificial abort step is performed by $\mathcal{B}$ exactly as in Game 1.

Note that if $b = 0$, then the view of $\mathcal{A}$ is identical to Game 1, while if $b = 1$ then it is identical to Game 2. Thus, by the adaptive indistinguishability of the VHF, we have

$$\mathsf{Adv}^2_{\mathcal{A}}(k) \geq \mathsf{Adv}^1_{\mathcal{A}}(k) - \mathsf{negl}(k)$$

for some negligible function $\mathsf{negl}(k)$.

Game 3. Finally, we have to make one last technical modification before we are able to describe our reduction to the $(n-1)$-linear assumption. Game 3 proceeds exactly as before, except that matrix $[\mathbf{B}]$ has a slightly different distribution. In Game 2, $\mathbf{B} \xleftarrow{\$} \mathrm{GL}_n(\mathbb{Z}_p)$ is chosen uniformly random (and invertible). In Game 3, we instead choose matrix $\mathbf{B}$ by sampling $\mathbf{b_1}, \ldots, \mathbf{b_{n-1}} \xleftarrow{\$} (\mathbb{Z}_p^*)^n$ and $\mathbf{b_n} \xleftarrow{\$} \mathbb{Z}_p^n$, defining $\mathbf{B} := (\mathbf{b_1}, \ldots, \mathbf{b_n})$, and then computing $[\mathbf{B}]$. Thus, we ensure that the first $n-1$ vectors do not have any component which equals the identity element. This is done to adjust the distribution of $[\mathbf{B}]$ to the distribution chosen by our reduction algorithm.

By applying the union bound, we have $\left|\mathsf{Adv}^3_{\mathcal{A}}(k) - \mathsf{Adv}^2_{\mathcal{A}}(k)\right| \leq n^2/p$. Since n is polynomially bounded and $\log(p) \in \Omega(k)$, we have

$$\mathsf{Adv}^3_{\mathcal{A}}(k) \geq \mathsf{Adv}^2_{\mathcal{A}}(k) - \mathsf{negl}(k)$$

for some negligible function $\mathsf{negl}(k)$.

The reduction to the $(n-1)$-linear assumption. In this game, we describe our actual reduction algorithm $\mathcal{B}$. Adversary $\mathcal{B}$ receives as input a $(n-1)$-linear challenge $[\mathbf{c}, \mathbf{d}, t]$, where $\mathbf{c} = (c_1, \ldots, c_n)^\top \xleftarrow{\$} (\mathbb{Z}_p^*)^n$, $\mathbf{d} \xleftarrow{\$} \mathbb{Z}_p^n$, and either $t = \sum_{i=1}^n d_i/c_i$ or $t \xleftarrow{\$} \mathbb{Z}_p$. It simulates the VRF security experiment exactly as in Game 3, with the following differences.

Initialization and set-up of parameters. Matrix $[\mathbf{B}]$ is computed as follows. First, $\mathcal{B}$ chooses $n(n-1)$ random integers $\alpha_{i,j} \xleftarrow{\$} \mathbb{Z}_p^*$ for $i \in \{1, \ldots, n-1\}$ and $j \in \{1, \ldots, n\}$. Then it sets $[\mathbf{b_i}] := (\alpha_{i,1}c_1, \ldots, \alpha_{i,n}, c_n)^\top$ and $[\mathbf{b_n}] := [\mathbf{d}]$, and finally $[\mathbf{B}] := [\mathbf{b_1}, \ldots, \mathbf{b_n}]$. Vector $[\mathbf{w}]$ is set to $[\mathbf{w}] := [\mathbf{c}]$.

Note that matrix $[\mathbf{B}]$ and vector $[\mathbf{w}]$ are distributed exactly as in Game 3. Observe also that the first $n-1$ column vectors of $[\mathbf{B}]$ depend on $\mathbf{c}$, while the last vector is equal to $\mathbf{d}$.

Answering Evaluate-queries. Whenever $\mathcal{A}$ issues an **Evaluate**-query on input $X^{(j)}$, then $\mathcal{B}$ computes $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_n)^\top \leftarrow \mathsf{TrapEval}_{\mathsf{VHF}}(td, X^{(j)})$. If $\beta_n \neq 0$, then $\mathcal{B}$ aborts and outputs a random bit. Otherwise it computes

$$[\mathbf{v}] := [\mathbf{B} \cdot \beta] = \left[(\mathbf{b_1}, \ldots, \mathbf{b_{n-1}}) \cdot (\beta_1, \ldots, \beta_{n-1})^\top\right] = \left[(\gamma_1 c_1, \ldots, \gamma_n c_n)^\top\right]$$

for integers $\gamma_1, \ldots, \gamma_n$, which are efficiently computable from $\boldsymbol{\beta}$ and the $\alpha_{i,j}$-values chosen by $\mathcal{B}$ above. Here we use that $\beta_n = 0$ holds for all **Evaluate**-queries that do not cause an abort.

Next, $\mathcal{B}$ computes the proof elements in $[\mathbf{z}]$ by setting $[z_i] := [\gamma_i]$ for all $i \in \{1, \ldots, n\}$. Note that, due to our setup of $[\mathbf{w}]$, it holds that

$$[\gamma_i] = \left[\frac{\gamma_i c_i}{c_i}\right] = \left[\frac{v_i}{w_i}\right]$$

thus all proof elements can be computed correctly by $\mathcal{B}$. Finally, $\mathcal{B}$ sets

$$Y := \prod_{i=1}^n [z_i] = \left[\sum_{i=1}^n z_i\right]$$

which yields the correct function value. Thus, all **Evaluate**-queries can be answered by $\mathcal{B}$ exactly as in Game 3.

Answering the Challenge-query. When $\mathcal{A}$ issues a **Challenge**-query on input $X^{(0)}$, then $\mathcal{B}$ computes $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_n)^\top \leftarrow \mathsf{TrapEval}_{\mathsf{VHF}}(td, X^{(0)})$. If $\beta_n = 0$, then $\mathcal{B}$ aborts and outputs a random bit. Otherwise again it computes the γ_i-values in

$$\begin{aligned}[\mathbf{v}] := [\mathbf{B} \cdot \beta] &= \left[(\mathbf{b_1}, \ldots, \mathbf{b_{n-1}}) \cdot (\beta_1, \ldots, \beta_{n-1})^\top + \mathbf{b_n} \cdot \beta_n\right] \\ &= \left[(\gamma_1 c_1, \ldots, \gamma_n c_n)^\top + \mathbf{d} \cdot \beta_n\right]\end{aligned}$$

Writing v_i and d_i to denote the i-th component of $\mathbf{v}$ and $\mathbf{d}$, respectively, it thus holds that $v_i = \gamma_i c_i + d_i \beta_n$. Observe that then the function value is

$$Y = \left[\sum_{i=1}^n \frac{v_i}{c_i}\right] = \left[\sum_{i=1}^n \frac{\gamma_i c_i + d_i \beta_n}{c_i}\right]$$

$\mathcal{B}$ computes and outputs $[t \cdot \beta_n] \cdot [\sum_{i=1}^n \gamma_i] = [t \cdot \beta_n + \sum_{i=1}^n \gamma_i]$. Observe that if $[t] = [\sum_{i=1}^n d_i/c_i]$, then it holds that

$$\left[t \cdot \beta_n + \sum_{i=1}^n \gamma_i\right] = \left[\beta_n \cdot \sum_{i=1}^n \frac{d_i}{c_i} + \sum_{i=1}^n \frac{\gamma_i c_i}{c_i}\right] = \left[\sum_{i=1}^n \frac{\gamma_i c_i + d_i \beta_n}{c_i}\right] = Y$$

Thus, if $[t] = [\sum_{i=1}^{n} d_i/c_i]$, then $\mathcal{B}$ outputs the correct function value Y. However, if $[t]$ is uniformly random, then $\mathcal{B}$ outputs a uniformly random group element.

Finally, $\mathcal{B}$ performs an artificial abort step exactly as in Game 2. Note that $\mathcal{B}$ provides a perfect simulation of the experiment in Game 3, which implies that

$$\mathsf{Adv}_{\mathcal{B}}^{(n-1)\text{-}\mathsf{lin}}(k) = \mathsf{Adv}_{\mathcal{A}}^{3}(k)$$

which is non-negligible, if $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{VRF}}(k)$ is. □

References

[1] Michel Abdalla, Dario Catalano, and Dario Fiore. "Verifiable Random Functions from Identity-Based Key Encapsulation". In: *Proc. EUROCRYPT 2009*. Vol. 5479. Lecture Notes in Computer Science. Springer, 2009, pp. 554–571.

[2] Michel Abdalla, Dario Catalano, and Dario Fiore. "Verifiable Random Functions: Relations to Identity-Based Key Encapsulation and New Constructions". In: *J. Cryptology* 27.3 (2014), pp. 544–593.

[3] Michel Abdalla, Dario Fiore, and Vadim Lyubashevsky. "From Selective to Full Security: Semi-generic Transformations in the Standard Model". In: *Proc. Public Key Cryptography 2012*. Vol. 7293. Lecture Notes in Computer Science. Springer, 2012, pp. 316–333.

[4] Man Ho Au, Willy Susilo, and Yi Mu. "Practical Compact E-Cash". In: *Information Security and Privacy, 12th Australasian Conference, ACISP 2007, Townsville, Australia, July 2-4, 2007, Proceedings*. 2007, pp. 431–445.

[5] Mira Belenkiy, Melissa Chase, Markulf Kohlweiss, and Anna Lysyanskaya. "Compact E-Cash and Simulatable VRFs Revisited". In: *Proc. Pairing 2009*. Vol. 5671. Lecture Notes in Computer Science. Springer, 2009, pp. 114–131.

[6] Mihir Bellare and Thomas Ristenpart. "Simulation without the Artificial Abort: Simplified Proof and Improved Concrete Security for Waters' IBE Scheme". In: *Proc. EUROCRYPT 2009*. Vol. 5479. Lecture Notes in Computer Science. Springer, 2009, pp. 407–424.

[7] Mihir Bellare and Moti Yung. "Certifying Cryptographic Tools: The Case of Trapdoor Permutations". In: *Proc. CRYPTO 1992*. Vol. 740. Lecture Notes in Computer Science. Springer, 1993, pp. 442–460.

[8] Mihir Bellare and Moti Yung. "Certifying Permutations: Noninteractive Zero-Knowledge Based on Any Trapdoor Permutation". In: *J. Cryptology* 9.3 (1996), pp. 149–166.

[9] Dan Boneh and Xavier Boyen. "Secure Identity Based Encryption Without Random Oracles". In: *Proc. CRYPTO 2004*. Vol. 3152. Lecture Notes in Computer Science. Springer, 2004, pp. 443–459.

[10] Dan Boneh, Xavier Boyen, and Hovav Shacham. "Short Group Signatures". In: *Proc. CRYPTO 2004*. Vol. 3152. Lecture Notes in Computer Science. Springer, 2004, pp. 41–55.

[11] Dan Boneh, Hart William Montgomery, and Ananth Raghunathan. "Algebraic pseudorandom functions with improved efficiency from the augmented cascade". In: *Proc. ACM Conference on Computer and Communications Security 2010*. ACM, 2010, pp. 131–140.

[12] Dan Boneh, Shai Halevi, Michael Hamburg, and Rafail Ostrovsky. "Circular-Secure Encryption from Decision Diffie-Hellman". In: *Proc. CRYPTO 2008*. Vol. 5157. Lecture Notes in Computer Science. Springer, 2008, pp. 108–125.

[13] Zvika Brakerski, Shafi Goldwasser, Guy N. Rothblum, and Vinod Vaikuntanathan. "Weak Verifiable Random Functions". In: *Proc. TCC 2009*. Vol. 5444. Lecture Notes in Computer Science. Springer, 2009, pp. 558–576.

[14] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. "Bonsai Trees, or How to Delegate a Lattice Basis". In: *Proc. EUROCRYPT 2010*. Vol. 6110. Lecture Notes in Computer Science. Springer, 2010, pp. 523–552.

[15] Melissa Chase and Anna Lysyanskaya. "Simulatable VRFs with Applications to Multi-theorem NIZK". in: *Proc. CRYPTO 2007*. Vol. 4622. Lecture Notes in Computer Science. Springer, 2007, pp. 303–322.

[16] Melissa Chase and Sarah Meiklejohn. "Déjà Q: Using Dual Systems to Revisit q-Type Assumptions". In: *Proc. EUROCRYPT 2014*. Vol. 8441. Lecture Notes in Computer Science. Springer, 2014, pp. 622–639.

[17] Sanjit Chatterjee and Palash Sarkar. *On (Hierarchical) Identity Based Encryption Protocols with Short Public Parameters*
(With an Exposition of Waters' Artificial Abort Technique). Cryptology ePrint Archive, Report 2006/279. `http://eprint.iacr.org/`. 2006.

[18] Jung Hee Cheon. "Security Analysis of the Strong Diffie-Hellman Problem". In: *Proc. EUROCRYPT 2006*. Vol. 4004. Lecture Notes in Computer Science. Springer, 2006, pp. 1–11.

[19] Yevgeniy Dodis. "Efficient Construction of (Distributed) Verifiable Random Functions". In: *Proc. Public Key Cryptography 2003*. Vol. 2567. Lecture Notes in Computer Science. Springer, 2002, pp. 1–17.

[20] Yevgeniy Dodis and Aleksandr Yampolskiy. "A Verifiable Random Function with Short Proofs and Keys". In: *Proc. Public Key Cryptography 2005*. Vol. 3386. Lecture Notes in Computer Science. Springer, 2005, pp. 416–431.

[21] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. *An Algebraic Framework for Diffie-Hellman Assumptions*. Cryptology ePrint Archive, Report 2013/377. `http://eprint.iacr.org/`. 2013.

[22] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge L. Villar. "An Algebraic Framework for Diffie-Hellman Assumptions". In: *Proc. CRYPTO (2) 2013*. Vol. 8043. Lecture Notes in Computer Science. Springer, 2013, pp. 129–147.

[23] Dario Fiore and Dominique Schröder. "Uniqueness Is a Different Story: Impossibility of Verifiable Random Functions from Trapdoor Permutations". In: *Proc. TCC 2012*. Vol. 7194. Lecture Notes in Computer Science. Springer, 2012, pp. 636–653.

[24] Eduarda S. V. Freire, Dennis Hofheinz, Kenneth G. Paterson, and Christoph Striecks. "Programmable Hash Functions in the Multilinear Setting". In: *Proc. CRYPTO (1) 2013*. Vol. 8042. Lecture Notes in Computer Science. Springer, 2013, pp. 513–530.

[25] Georg Fuchsbauer. "Constrained Verifiable Random Functions". In: *Proc. SCN 2014*. Vol. 8642. Lecture Notes in Computer Science. Springer, 2014, pp. 95–114.

[26] Michael Gerbush, Allison B. Lewko, Adam O'Neill, and Brent Waters. "Dual Form Signatures: An Approach for Proving Security from Static Assumptions". In: *Proc.*

ASIACRYPT 2012. Vol. 7658. Lecture Notes in Computer Science. Springer, 2012, pp. 25–42.

[27] Dennis Hofheinz and Eike Kiltz. "Programmable Hash Functions and Their Applications". In: *Proc. CRYPTO 2008*. Vol. 5157. Lecture Notes in Computer Science. Springer, 2008, pp. 21–38.

[28] Susan Hohenberger and Brent Waters. "Constructing Verifiable Random Functions with Large Input Spaces". In: *Proc. EUROCRYPT 2010*. Vol. 6110. Lecture Notes in Computer Science. Springer, 2010, pp. 656–672.

[29] Tibor Jager. "Verifiable Random Functions from Weaker Assumptions". In: *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015*. Vol. 9015. Lecture Notes in Computer Science. Springer, 2015, pp. 121–143.

[30] David Jao and Kayo Yoshida. "Boneh-Boyen Signatures and the Strong Diffie-Hellman Problem". In: *Proc. Pairing 2009*. Vol. 5671. Lecture Notes in Computer Science. Springer, 2009, pp. 1–16.

[31] Stanislaw Jarecki and Vitaly Shmatikov. "Handcuffing Big Brother: an Abuse-Resilient Transaction Escrow Scheme". In: *Proc. EUROCRYPT 2004*. Vol. 3027. Lecture Notes in Computer Science. Springer, 2004, pp. 590–608.

[32] Saqib A. Kakvi, Eike Kiltz, and Alexander May. "Certifying RSA". in: *Proc. ASIACRYPT 2012*. Vol. 7658. Lecture Notes in Computer Science. Springer, 2012, pp. 404–414.

[33] Sebastian Lauer. *Verifiable Random Functions*. Master Thesis, Ruhr-University Bochum. 2015.

[34] Moses Liskov. "Updatable Zero-Knowledge Databases". In: *Proc. ASIACRYPT 2005*. Vol. 3788. Lecture Notes in Computer Science. Springer, 2005, pp. 174–198.

[35] Anna Lysyanskaya. "Unique Signatures and Verifiable Random Functions from the DH-DDH Separation". In: *Proc. CRYPTO 2002*. Vol. 2442. Lecture Notes in Computer Science. Springer, 2002, pp. 597–612.

[36] Silvio Micali, Michael O. Rabin, and Salil P. Vadhan. "Verifiable Random Functions". In: *Proc. FOCS 1999*. IEEE Computer Society, 1999, pp. 120–130.

[37] Silvio Micali and Leonid Reyzin. "Soundness in the Public-Key Model". In: *Proc. CRYPTO 2001*. Vol. 2139. Lecture Notes in Computer Science. Springer, 2001, pp. 542–565.

[38] Silvio Micali and Ronald L. Rivest. "Micropayments Revisited". In: *Proc. CT-RSA 2002*. Vol. 2271. Lecture Notes in Computer Science. Springer, 2002, pp. 149–163.

[39] Moni Naor. "On Cryptographic Assumptions and Challenges". In: *Proc. CRYPTO 2003*. Vol. 2729. Lecture Notes in Computer Science. Springer, 2003, pp. 96–109.

[40] Moni Naor and Omer Reingold. "Number-theoretic constructions of efficient pseudo-random functions". In: *J. ACM* 51.2 (2004), pp. 231–262.

[41] Moni Naor and Gil Segev. "Public-Key Cryptosystems Resilient to Key Leakage". In: *Proc. CRYPTO 2009*. Vol. 5677. Lecture Notes in Computer Science. Springer, 2009, pp. 18–35.

[42] Hovav Shacham. *A Cramer-Shoup Encryption Scheme from the Linear Assumption and from Progressively Weaker Linear Variants*. Cryptology ePrint Archive, Report 2007/074. `http://eprint.iacr.org/`. 2007.

[43] Brent Waters. "Dual System Encryption: Realizing Fully Secure IBE and HIBE under Simple Assumptions". In: *Proc. CRYPTO 2009*. Vol. 5677. Lecture Notes in Computer Science. Springer, 2009, pp. 619–636.

[44] Brent Waters. "Efficient Identity-Based Encryption Without Random Oracles". In: *Proc. EUROCRYPT 2005*. Vol. 3494. Lecture Notes in Computer Science. Springer, 2005, pp. 114–127.

A The need for an artificial abort

The "artificial abort" technique of Waters [44] has become standard for security proofs that combine a "partitioning" proof technique with a "decisional" complexity assumption. For example, it is also used to analyze Waters' IBE scheme [44], the verifiable random function of Hohenberger and Waters [28], and many other works.

Unfortunately, the artificial abort is necessary, because our $(n-1)$-linear reduction algorithm $\mathcal{B}$ is not able to use the output of $\mathcal{A}$ directly in case the experiment it not aborted. This is because the abort probability may depend on the particular sequence of queries issued by $\mathcal{A}$. For example, it may hold that $\Pr[B = b] = 1/2 + \epsilon$ for some non-negligible ϵ, which means that $\mathcal{A}$ has a non-trivial advantage in breaking the VRF-security, while $\Pr[B = b \mid \neg\mathsf{abort}] = 1/2$, which means that $\mathcal{B}$ does not have any non-trivial advantage in breaking the $(n-1)$-linear assumption. Essentially, the artificial abort ensures that $\mathcal{B}$ aborts for all sequences of queries made by $\mathcal{A}$ with approximately the same probability.

Alternatively, we could avoid the artificial abort by following the approach of Bellare and Ristenpart [6], which yields a tighter (but more complex) reduction. To this end, we would have to define and construct a PVHF which guarantees sufficiently close upper and lower bounds on the abort probability. This is possible by adopting the idea of balanced admissible hash functions (AHFs) from [29] to "balanced PVHFs". Indeed, instantiating our adaptively-secure PVHF with the balanced AHF from [29] yields such a balanced PVHF. However, this would have made the definition of PVHFs much more complex. We preferred to keep this novel definition as simple as possible, thus used the artificial abort approach of Waters [44].