Unbiasable Verifiable Random Functions

Emanuele Giunta[1,2,3] , Alistair Stewart[1]

[1] Web3 Foundation, Switzerland.
alistair@web3.foundation
[2] IMDEA Software Institute, Spain.
emanuele.giunta@imdea.org
[3] Universidad Politecnica de Madrid, Spain.

Abstract. Verifiable Random Functions (VRFs) play a pivotal role in Proof of Stake (PoS) blockchain due to their applications in secret leader election protocols. However, the original definition by Micali, Rabin and Vadhan is by itself insufficient for such applications. The primary concern is that adversaries may craft VRF key pairs with skewed output distribution, allowing them to unfairly increase their winning chances.
To address this issue David, Gaži, Kiayias and Russel (2017/573) proposed a stronger definition in the universal composability framework, while Esgin et al. (FC '21) put forward a weaker game-based one. Their proposed notions come with some limitations though. The former appears to be too strong, being seemingly impossible to instantiate without a programmable random oracle. The latter instead is not sufficient to prove security for VRF-based secret leader election schemes.
In this work we close the above gap by proposing a new security property for VRF we call *unbiasability*. On the one hand, our notion suffices to imply fairness in VRF-based leader elections protocols. On the other hand, we provide an efficient compiler in the plain model (with no CRS) transforming any VRF into an unbiasable one under standard assumptions. Moreover, we show folklore VRF constructions in the ROM to achieve our notion without the need to program the random oracle. As a minor contribution, we also provide a generic and efficient construction of certified 1 to 1 VRFs from any VRF.

Table of Contents

1 Introduction

Verifiable Random Functions (VRF), introduced in [MRV99], are the natural extension of pseudorandom functions to the public key setting. A secret key sk is required to perform evaluations. Any users can then verify the result given the public verification key vk and an opening proof. Since their introduction VRFs have found many applications including e-lotteries [MR02, LBM20], secure DNS [GNP+14, PWH+17], and Proof of Stake blockchain [CM16, KRDO17, DGKR17, BASV23b] for secret leader elections.

The VRF definition in [MRV99] however only applies to *honestly generated* keys and offers very limited guarantees against *adversarially generated* ones. This crucially affects VRF-based leader election protocols. In that context, the winner of an election is defined as the user with the lowest VRF output on a random public input. Therefore, being able to choose a key pair with biased outputs directly translates into unfairly higher winning chances.

This issue was first acknowledged in [DGKR17] where a stronger VRF definition was given in the Universal Composability framework [Can01]. One of the requirements in their notion is that the VRF output for any key pair must match the truly random values returned by an ideal functionality. Although this entirely prevents adversaries from skewing the output distribution, their approach comes with some controversial aspects.

First of all, their notion appears to be impossible to instantiate without access to an explicitly programmable random oracle [CDG+18]. This stems from the fact that in the security proof the simulator must program the VRF output, even with maliciously generated keys, to match the truly random output given by the ideal functionality. Note that on the contrary, VRFs are known to exist in the plain model. Secondly, due to the very high level of technical details in the original definitions of UC-VRF, it is not clear what are the exact security guarantees it implies, besides being sufficient for PoS-blockchain applications. The high level of complexity is further highlighted by subsequent revisions [BGK+18, BGQR22] proposing fixes to overlooked corner cases.

An alternative game-based definition was later proposed in [EKS+21]. Although their notion can be achieved in the plain model, it appears to be insufficient for applications for two main reasons. First of all, it does not keep adversaries from biasing some bits of the output, as we show later. Secondly, it does not prevent adversaries to maliciously create several VRF keys returning correlated outputs, also leading to attacks in VRF-based election protocols.

Given the current state of the art, we ask whether it is possible to provide a simple security notion that simultaneously can be obtained in the plain model and suffices to imply security in current applications.

Our Results. In this work we close the aforementioned foundational gap by proposing a game-based security property we call *unbiasability*. Informally, we say VRF is unbiasable if no adversary can find a set of VRF keys whose outputs on random inputs significantly bias an adversarially chosen predicate p. To provide evidence that our notion does indeed fill the above gap we

1. Prove it to imply desirable properties. In particular we show that assuming an adversary to always return correct VRF evaluations, unbiasability implies the output distribution to be computationally close to uniform on random inputs. Our notion also implies the weaker one proposed in [EKS+21].
2. Prove that folklore constructions in the ROM from a verifiable unpredictable function (VUF) satisfy our notion. This notably includes ECVRF [PWH+17] currently part of the standardization effort in [GRPV23].
3. Provide two compilers transforming any VRF into an unbiasable VRF in the plain model. Our most efficient construction only requires one VRF evaluation and is proven secure assuming DDH and a PRF's pseudorandomness hold against adversaries with exponential-time preprocessing[4].
4. Observe it to immediately imply fairness in VRF-based leader election for PoS-blockchain application. In other words, no adversary can win any random election with probability significantly higher than prescribed.

As a minor contribution of independent interest we also construct generically a certified 1 to 1 VRF, which we call *Verifiable Random Bijection* (VRB), from any VRF. This is efficient, requiring only two VRF evaluations, and security just relies on the VRF security and hardness of the discrete logarithm problem.

Technical Overview. We now provide a more detailed overview of our definition, starting from the weaker notion introduced in [EKS+21], and later summarize the constructions we provide and study.

Defining unbiasability. Our starting point to define a strong unbiasability property is the definition provided in [EKS+21] (Section 2.1). There, an adversary is asked initially to generate a verification key vk and guess the final VRF output y^*. Later, on input a random value x, the adversary is asked to provide a tuple (y, π). If π is a valid proof for y and $y = y^*$ it wins the game. According to their notion a VRF is unbiasable if any adversary cannot win with probability significantly higher than randomly guessing y.

Although we see this definition as a step in the right direction, it comes with two limitations. First of all, even if the adversary is unable to guess correctly the output consistently, it may still be able to bias some bits. Secondly, this definition fails to exclude VRFs in which several keys can be crafted to return correlated outputs, although individual values are hard to bias. Such weakness could be exploited in VRF-based leader election by several corrupted users to ensure at least one VRF value is always small enough, allowing them to win VRF-based elections unfairly more than honest users.

To circumvent such limitations, we modify their game as follows: first the adversary chooses several verification keys $\mathsf{vk}_1, \ldots, \mathsf{vk}_n$ and a predicate p with

[4] More precisely we will only assume the preprocessing to be capable of solving the discrete logarithm problem, something that can be also achieved in quantum probabilistic polynomial time [Sho94].

certain properties (which we clarify later). Next, on random inputs $x_1, \ldots, x_m$, the adversary returns a vector $\mathbf{y}$ containing the evaluation of each VRF of its choice on all provided inputs, along with evaluation proofs. Those values in $\mathbf{y}$ whose proof is incorrect are overwritten with $\perp$. Finally the adversary wins if $p(\mathbf{y})$ is true significantly more often than $p(\mathbf{z})$ for a uniformly sampled $\mathbf{z}$.

If we were to allow all predicates p, our game would have a trivial attack: simply choosing p to be 1 if and only if the given vector contains at least one erased value $\perp$ and then, once asked to evaluate the VRF, producing incorrect opening proofs. Note that we sample entries of $\mathbf{z}$ from the output space, meaning they will never be equal to $\perp$. This attack however appears to be orthogonal to the attempt of biasing the output, and rather comes from the fact adversaries can always decide not to evaluate their VRF. We therefore factor this classes of attacks out of our definition by only considering predicates in which erasing any entry of a vector $\mathbf{y}$ can never turn the output from *false* to *true*. We call those predicates *monotone*. Restricting p to be monotone essentially means that an adversary cannot gain any advantage from producing incorrect evaluations.

Constructions in the ROM. In the random oracle model we study two folklore transformation of any VUF into a VRF. The first one is defined by hashing the VUF output salted with the input and the VUF verification key. In other words an evaluation on x is defined as $\mathsf{H}(\mathsf{vk}, x, y)$ where vk is the VUF verification key and y, π the VUF output on input x. We show this to be unbiasable information theoretically when H is a Random Oracle.

In order to capture also ECVRF [PWH$^+$17], we extend our analysis to a weaker VUF to VRF transformation in which the output is simply defined as $\mathsf{H}(y)$ with y being the VUF output. We observe that in general the resulting VRF may not be unbiasable due to corner cases in which for instance, there may be a weak verification key defining a constant function or two distinct keys defining identical functions. Nevertheless we still prove security under mild collision-resistance assumptions, which are satisfied by the VUF underlying ECVRF.

Constructions in the plain model. Next we provide two compilers transforming any VRF into an unbiasable VRF without relying on a CRS nor the ROM. We opt for a modular approach and describe our transformation in two steps: First we transform any VRF into one achieving a weaker notion of pseudorandomness and unbiasability, where the latter holds information theoretically. More in details the above properties are weakened as follows:

- *Pseudorandomness*: Inputs are of the form (x_0, x_1) and security holds if the VRF is never evaluated on different points with the same first entry.
- *Unbiasability*: For each verification key vk_i, the challenger asks for evaluations on a set of freshly sampled points $x_{i,1}, \ldots, x_{i,m}$, as opposed to asking evaluations for all keys on the same set of points.

We then provide a second compiler turning any VRF with the above properties into fully fledged unbiasable VRF.

Padded VRF. Focusing on VRFs which only satisfy our weaker properties greatly simplifies our problem. This is exemplified by our first construction, the *padded VRF*: given a VRF $F_{\mathsf{sk}}(\cdot)$ it is defined as

$$\overline{F}_{\mathsf{sk}}(u, v) \;=\; F_{\mathsf{sk}}(u) \oplus v.$$

Pseudorandomness holds as long as no adversary can query the same u twice. Moreover, on uniformly random inputs (u, v) the output will also be random thanks to v, no matter how skewed the distribution of $F_{\mathsf{sk}}(u)$ ends up being, which we use to show unbiasability. Note that if the VRF returns values in a group $(\mathbb{G}, +)$, the xor operator can be replaced with the group addition up to assuming $v \in \mathbb{G}$.

2 Round Feistel. Another approach to achieve this weaker notion of unbiasability unconditionally is to guarantee the resulting VRF is 1 to 1. We call such VRF a Verifiable Random Bijection and show that they satisfy unbiasability (with independently sampled points for each key) information theoretically, because bijections preserve the input distribution. In order to realize VRF with this property we observe that 2 Feistel Round [LR88] suffices. The construction is schematized in Figure 1 assuming the VRF has output values in a group $(\mathbb{G}, +)$.

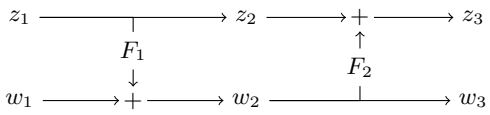

Fig. 1. 2-Feistel Rounds with $F_i(\cdot) = \mathsf{VRF.Eval}(\mathsf{sk}_i, \cdot)$ and output space a group $(\mathbb{G}, +)$.

The reason why two rounds are sufficient as opposed to three or more is that at this stage we only target the weaker pseudorandomness notion, where the VRF is never evaluated on different points with the same first entry z_1.

VRB Compiler. As warm up toward our final compiler, we start with a simpler problem: how can we improve the 2-Feistel Round construction to achieve pseudorandomness? Our approach to this problem is to compose the 2-Feistel Round scheme with a (publicly computable) function ϕ_h, i.e. obtaining $F_{\mathsf{sk}} \circ \phi_h(x)$, and then study what properties should ϕ satisfy.

First of all, we need the composed function to remain 1 to 1 for any key choice. Thus ϕ_h has to be a certified bijection[5]. For pseudorandomness instead, recall that F_{sk} achieves it as long as it is never evaluated on two different inputs with the same first component. A way to enforce this is assuming ϕ_h to be collision resistant in the first component.

[5] In other words, deciding whether ϕ_h is a bijection can be done in polynomial time

The last step is finding a map satisfying the above properties. Our solution is based on a group $\mathbb{G}$ of prime order q where the discrete logarithm problem (DLP) is hard.

$$h = [a_1, a_2] \in \mathbb{G}^2 \qquad \phi_h : \mathbb{F}_q^2 \to \mathbb{G}^2 \ : \ \phi_h(x_1, x_2) \ = \ \begin{bmatrix} a_1 & a_2 \\ 1 & 0 \end{bmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

where we denote $[a] = a \cdot G$, with G being the group generator. Because ϕ_h in the first component is a Pedersen commitment to $x = (x_1, x_2)$, it is collision resistant if a_1, a_2 are sampled uniformly and the DLP is hard. Conversely, even when ϕ_h is created adversarially, it suffice to check $a_2 \neq 0$ to verify that ϕ_h defines a 1 to 1 map.

Final Compiler. Our final problem is to compile any VRF F_{sk} with the weaker pseudorandomness and unbiasability properties defined above to a fully fledged unbiasable VRF.

As before, pseudorandomness can be achieved composing $F_{\mathsf{sk}} \circ \phi_h$ with a slightly more generalized ϕ_h defined as

$$h = \left([\mathbf{a}], \mathbf{b}\right) \qquad \phi_h : \mathbb{F}_q^{\mu} \to \mathbb{G}^2 \ : \ \phi_h(\mathbf{v}) \ = \ \left(\left[\mathbf{a}^\top \mathbf{v}\right], \left[\mathbf{b}^\top \mathbf{v}\right]\right).$$

which can be publicly checked to be surjective and, for uniformly sampled $\mathbf{a}$, is collision resistant in the first component if DLP is hard. The technical challenge is enhancing unbiasability. Our (stronger) definition requires the output to be unbiasable even when multiple verification keys are evaluated on the same (random) input. However F_{sk} is guaranteed to be unbiasable only when we evaluate different verification keys on independently sampled points.

To eventually reduce security to this weaker property we need a mechanism to expand a random input x into several (random-looking) ones $x_1, \ldots, x_m$ to later evaluate the VRF with key vk_i on input x_i. A first attempt to do so is through a PRF f: we interpret the input as a PRF key k and evaluate each VRF on $f_k(\mathsf{vk})$. However the map $k \mapsto f_k(\mathsf{vk})$ may not be collision resistant and finding collisions would allow breaking pseudorandomness. Conversely, the map $k \mapsto (k, f_k(\mathsf{vk}))$ is injective, but since $F_{\mathsf{sk}} \circ \phi_h$ would also receive the PRF key as input in this case, we cannot argue that $f_k(\mathsf{vk})$ is pseudorandom anymore.

Our final solution is to provide $f_k(\mathsf{vk})$ along with a perfectly binding commitment to k (in our case, ElGamal). More specifically we will interpret a VRF input x as a tuple $(\mathbf{r}, k, \mathsf{pp}, \rho)$ and define

$$\mathsf{Encode}(\mathbf{r}, k, \mathsf{pp}, \rho, \mathsf{vk}) \ = \ (\mathbf{r}, f_k(\mathsf{vk}), \mathsf{pp}, \mathsf{Com}_{\mathsf{pp}}(k; \rho))$$

which, using ElGamal, is injective and does not leak any knowledge about k. The elements $\mathbf{r}$ are included for technical reasons, namely because in the proof of unbiasability we need ϕ_h to be a chameleon hash function [KR98] in the first output component. We finally show $F_{\mathsf{sk}} \circ \phi_h \circ \mathsf{Encode}$ to be an unbiasable VRF.

Related Work. The study of VRF, initiated by Goldreich, Goldwasser and Micali [GGM86], is an active area of research in cryptography. Several constructions have been proposed over the years based on (the list is not exhaustive) generic VUF [MRV99], identity-based KEM [ACF09], pairing groups [HW10, BMR10, HJ16, Koh19], lattices [EKS+21, ESLR22] and isogenies [Lai23].

VRF satisfying extra properties have also been introduced and studied. Most notably simulatable VRFs [CL07] where opening proofs can be simulated given a trapdoor key. Constrained VRFs [Fuc14], which allow leaking a constrained secret key to perform evaluation only over a given subset. Ring VRFs [BASV23a], which allow evaluating a VRF anonymously among a ring of users. As the original definition however, they do not imply any form of security for maliciously chosen keys. More related to our work, the notion of UC-VRF [DGKR17, BGK+18, BGQR22] and "unbiasability" in [EKS+21], both address security under maliciously chosen keys, albeit with the aforementioned limitations. Orthogonally, distributed VRF [GLOW21] prevent maliciously generated keys by producing and storing them in a distributed fashion. However, while useful for random beacons [CD17, CD20], distributed VRFs do not appear suited for applications such as leader election, where parties may dynamically join and leave, and need to evaluate the VRF frequently and privately.

Finally, the issue of adversaries generating biased VRF keys was also noted in [BGRV09] while instantiating the hidden-bit model [FLS90] from weak-VRF. The issue was however addressed as in [BY96] for the specific application, without providing a general security notion.

2 Preliminaries

2.1 Notation

λ denotes the security parameter. A function $\varepsilon(\lambda)$ is negligible if it approaches 0 faster than the inverse of any polynomial in λ. $[n] = \{1, \ldots, n\}$. Whenever unspecified, we assume all Turing machines (including security games) in this paper to take as first input 1^λ. Given a set X, $x \leftarrow^{\$} X$ means x is sampled uniformly in X. We denote a random variable x uniformly distributed in X with $x \sim U(X)$. $\Delta(x, y)$ is the statistical distance between x and y. For a Probabilistic Turing Machine $\mathcal{A}$ we write $y \leftarrow^{\$} \mathcal{A}(x)$ to denote that y is the output of $\mathcal{A}$ on input x. $\mathbb{F}_q$ is the field of integers modulo q with q a prime number. Given $\mathbf{a}, \mathbf{b} \in \mathbb{F}_q^n$, then $\mathbf{a}^\top \mathbf{b}$ is their inner product. We write groups $(\mathbb{G}, +)$ in additive notation and, in a prime order group with a canonical generator G we denote $[a] = a \cdot G$.

2.2 Preprocessing Adversaries

Many computationally hard problems are often defined in terms of interactive games, where the adversary is asked to solve a given problem efficiently. These however typically fail to capture real adversaries which may have performed an

inefficient *preprocessing* before approaching the problem. A way to model such attacks is through *non-uniform* adversaries, which initially receive a polynomially bounded advice that is a *function* of the security parameter only[6]. In this section, in order to later state more fine-grained computational assumptions, we give an explicit definition for preprocessing adversary. In particular the following notion allow us to specify the computational class C of the preprocessing algorithm.

Definition 1. *Given a computational class* C*, we say an algorithm* $\mathcal{A}$ *to be* PPT *with* C *preprocessing if there exists* $(\mathcal{A}_0, \mathcal{A}_1)$ *with* $\mathcal{A}_0$ *in* C *returning a polynomially bounded output and* $\mathcal{A}_1$ PPT*, such that*

$$\mathcal{A}(1^\lambda, \mathsf{input}) \;=\; \mathcal{A}_1(1^\lambda, \mathsf{input}, \mathcal{A}_0(1^\lambda)).$$

2.3 Discrete Logarithm Problem and DDH

Given a prime order group $(\mathbb{G}, +)$ with generator G, the discrete logarithm assumption state that any PPT adversary, given xG with $x \sim U(\mathbb{F}_q)$ can retrieve x only with negligible probability. The Decisional Diffie-Hellman assumption instead, state that (G, xG, yG, xyG) is computationally indistinguishable from (G, xG, yG, zG) with $x, y, z \sim U(\mathbb{F}_q)$ for any PPT adversary.

In our constructions we will sometime base security on mildly stronger assumption, requiring that both DLP and DDH are hard even if some preprocessing occurred. More concretely, we will assume this preprocessing to occur in PPT time but with oracle access to a Discrete Logarithm solver. More formally, to make use of the Definition 1, let us call $\mathsf{PPT}^{\mathsf{DL}}$ the class of PPT algorithms $\mathcal{A}^{\mathsf{DL}}$ where for all $H \in \mathbb{G}$, $\mathsf{DL}(H) = x$ with $H = xG$.

Definition 2. *Given a group* $\mathbb{G}$ *with generator* G *we say that* $\mathsf{DLPrep\text{-}DLP}$ *is hard if for any* PPT *adversary with* $\mathsf{PPT}^{\mathsf{DL}}$*-preprocessing, given* $x \sim U(\mathbb{F}_q)$ *there exists a negligible* $\varepsilon(\lambda)$ *such that*

$$\mathsf{Adv}(\mathcal{A}) \;\coloneqq\; \Pr\left[\mathcal{A}(1^\lambda, G, xG) \to x\right] \;\leq\; \varepsilon(\lambda).$$

Definition 3. *Given a group* $\mathcal{G}$ *with generator* g *we say that* $\mathsf{DLPrep\text{-}DDH}$ *is hard if for any* PPT *adversary with* $\mathsf{PPT}^{\mathsf{DL}}$*-preprocessing, given* $x, y, z \sim U(\mathbb{F}_q)$ *there exists a negligible* $\varepsilon(\lambda)$ *such that*

$$\mathsf{Adv}(\mathcal{A}) \coloneqq \left|\Pr\left[\mathcal{A}(1^\lambda, xG, yG, xyG) \to 1\right] - \Pr\left[\mathcal{A}(1^\lambda, xG, yG, zG) \to 1\right]\right| \leq \varepsilon(\lambda).$$

To justify the plausibility of these assumptions we refer to [CK18] where the security of the DLP and DDH is studied in the Generic Group Model with preprocessing. Their results imply that with any polynomially bounded length advice (potentially computed in unbounded time), the DLP only loses $\log(\lambda)$ bit of security with respect to the non-preprocessing DLP. DDH with unbounded

[6] This means that the advice may not even be *computable* by Turing Machines

preprocessing time instead suffers a quadratic loss, or in other words, loses half of its security bits.

We remark that problems proved hard in the GGM are plausibly as hard in groups where generic attacks are also the best known so far. Furthermore, we stress that our assumptions are significantly milder as we only assume the preprocessing capable of breaking DLP (as opposed to being computationally unbounded). Finally, note that in the algebraic group model [FKL18], DLPrep-DLP and DLPrep-DDH reduces both respectively to DLP and DDH without any security loss, as the preprocessing already knows the discrete logarithm of queried group elements.

Certified Groups. As a technical detail we note that all our constructions require the group to be *certified* [HJ16]. This means that even when the group parameters are chosen adversarially, it is possible to verify it has the claimed prime order, and membership test and group additions are guaranteed to be efficient.

2.4 Pseudo Random Functions

We recall the definition of pseudo random function (PRF) [GGM86], that is a couple of algorithm $(\mathsf{PRF.Gen}, f)$, along with a key space $\mathcal{K}$ an input space $\mathcal{X}$ and an output space $\mathcal{Y}$ such that

- $k \leftarrow^{\$} \mathsf{PRF.Gen}(1^\lambda)$ generates a secret key $k \in \mathcal{K}$
- $f : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is a keyed function family.

The main security property of PRFs is pseudorandomness, defined as follows:

Definition 4. *A couple of* PPT *algorithms* $(\mathsf{PRF.Gen}, f)$ *is called a Pseudo Random Function if, for any* PPT *adversary there exists a negligible function* ε *such that*

$$\mathsf{Adv}(\mathcal{A}) \;:=\; \left|\Pr\left[\mathsf{Exp}^{\mathsf{prf}}(\mathcal{A}) = 1\right] - \frac{1}{2}\right| \;\leq\; \varepsilon(\lambda).$$

$\mathsf{Exp}^{\mathsf{prf}}(\mathcal{A})$

1 : $b \leftarrow^{\$} \{0,1\}$
2 : Sample $k \leftarrow^{\$} \mathsf{PRF.Gen}(1^\lambda)$
3 : Sample $f^* \leftarrow^{\$} \{g \;:\; g : \mathcal{X} \to \mathcal{Y}\}$
4 : Run $b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{prf}}}(1^\lambda)$
5 : **Return** $b == b'$

$\mathcal{O}_{\mathsf{prf}}(x)$

1 : **If** $b = 1$:
2 : **Return** $f_k(x)$
3 : **Else**:
4 : **Return** $f^*(x)$

Fig. 2. The pseudorandomness game with adversary $\mathcal{A}$ and PRF $(\mathsf{PRF.Gen}, f)$.

2.5 Verifiable Random Functions

Here we recall the definition of VRF, originally introduced in [MRV99], following the terminology of [Lys02, HJ16].

Definition 5. *A Verifiable Random Function is a triplet of* PPT *algorithms* $(\mathsf{VRF.Gen}, \mathsf{VRF.Eval}, \mathsf{VRF.Vfy})$ *satisfying:*

1. ***Correctness.*** *For any* $(\mathsf{vk}, \mathsf{sk})$ *in the image of* $\mathsf{VRF.Gen}(1^\lambda)$ *and any* $x \in \mathcal{X}$

$$(y, \pi) \leftarrow \mathsf{VRF.Eval}(\mathsf{sk}, x) \quad \Rightarrow \quad \mathsf{VRF.Vfy}(\mathsf{vk}, x, y, \pi) \rightarrow 1.$$

2. ***Unique Provability.*** *For any* vk *(not necessarily in the range of* $\mathsf{VRF.Gen}$*), any input* $x \in \mathcal{X}$ *pair of outputs* $y_0, y_1 \in \mathcal{Y}$ *and pair of proofs* π_0, π_1 *it holds that*

$$\mathsf{VRF.Vfy}(\mathsf{vk}, x, y_0, \pi_0) \rightarrow 1, \quad \mathsf{VRF.Vfy}(\mathsf{vk}, x, y_1, \pi_1) \rightarrow 1 \quad \Rightarrow \quad y_0 = y_1.$$

3. ***Pseudorandomness.*** *For any* PPT *adversary* $\mathcal{A}$ *executed in experiment 3 there exists a negligible function* ε *such that*

$$\mathsf{Adv}(\mathcal{A}) \;=\; \left| \Pr\left[\mathsf{Exp}^{\mathsf{rnd}}(\mathcal{A}) \rightarrow 1\right] - \frac{1}{2} \right| \;\leq\; \varepsilon(\lambda).$$

$\mathsf{Exp}^{\mathsf{rnd}}(\mathcal{A})$

1 : Sample $b \leftarrow^{\$} \{0,1\}$
2 : $\mathsf{vk}, \mathsf{sk} \leftarrow^{\$} \mathsf{VRF.Gen}(1^\lambda)$ and set $x^* = \perp$
3 : $x^* \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{eval}}}(\mathsf{vk})$
4 : **If** x^* was previously queried:
5 : **Return** 0
6 : $y_0 \leftarrow^{\$} \mathcal{Y}$
7 : $(y_1, \pi) \leftarrow \mathsf{VRF.Eval}(\mathsf{sk}, x^*)$
8 : $\mathcal{A}^{\mathcal{O}_{\mathsf{eval}}}(\mathsf{vk}, y_b) \rightarrow b'$
9 : **Return** $b == b'$

$\mathcal{O}_{\mathsf{eval}}(x)$

1 : **If** $x \neq x^*$:
2 : $(y, \pi) \leftarrow \mathsf{VRF.Eval}(\mathsf{sk}, x)$
3 : **Return** (y, π)
4 : **Else**: **Return** $\perp$

Fig. 3. The pseudorandomness security game with adversary $\mathcal{A}$.

Restricted Pseudorandomness. Relaxations of the pseudorandomness properties have been proposed over the years, for instance in [BGRV09] where *weak* pseudorandomness is defined by sampling the points for evaluation queries and for the challenge uniformly in the function's domain. In this work we will consider a notion we call *restricted pseudorandomness* where each message is interpreted as a tuple (x_0, x_1) and two messages $(x_0, x_1), (\bar{x}_0, \bar{x}_1)$ are considered to be the same by the challenger if $x_0 = \bar{x}_0$.

Definition 6. *A triplet* $(\mathsf{VRF.Gen}, \mathsf{VRF.Eval}, \mathsf{VRF.Vfy})$ *with message space* $\mathcal{X} = \mathcal{X}_0 \times \mathcal{X}_1$ *satisfies restricted pseudorandomness with respect to* $\mathcal{X}_0$ *if for any* PPT *adversary* $\mathcal{A}$ *there exists a negligible function* ε *such that*

$$\mathsf{Adv}(\mathcal{A}) \;:=\; \left|\Pr\left[\mathsf{Exp}^{\mathsf{res\text{-}rnd}}_{\mathcal{X}_0}(\mathcal{A}) = 1\right] - \frac{1}{2}\right| \;\leq\; \varepsilon(\lambda).$$

$\mathsf{Exp}^{\mathsf{res\text{-}rnd}}_{\mathcal{X}_0}(\mathcal{A})$

1 : Sample $b \leftarrow^{\$} \{0,1\}$
2 : $\mathsf{vk}, \mathsf{sk} \leftarrow^{\$} \mathsf{VRF.Gen}(1^\lambda)$ and set $x_0^* = \perp$
3 : $(x_0^*, x_1^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{eval}}}(\mathsf{vk})$
4 : **If** $(x_0^*, \cdot)$ was previously queried:
5 : **Return** 0
6 : $y_0 \leftarrow^{\$} \mathcal{Y}$
7 : $(y_1, \pi) \leftarrow \mathsf{VRF.Eval}(\mathsf{sk}, x_0^*, x_1^*)$
8 : $\mathcal{A}^{\mathcal{O}_{\mathsf{eval}}}(\mathsf{vk}, y_b) \rightarrow b'$
9 : **Return** $b == b'$

$\mathcal{O}_{\mathsf{eval}}(x_0, x_1)$

1 : **If** $x_0 \neq x_0^*$:
2 : $(y, \pi) \leftarrow \mathsf{VRF.Eval}(\mathsf{sk}, x_0, x_1)$
3 : **Return** (y, π)
4 : **Else**: **Return** $\perp$

Fig. 4. The restricted pseudorandomness game with adversary $\mathcal{A}$.

Verifiable Unpredictable Functions. Another relaxation of the pseudorandomness property was defined in [MRV99] where Verifiable Unpredictable Functions (VUF) where introduced. A VUF is a tuple of algorithms $(\mathsf{VUF.Gen}, \mathsf{VUF.Eval}, \mathsf{VUF.Vfy})$ satisfying correctness and unique provability as per Definition 5, but instead of pseudorandomness they are only required to satisfy unpredictability, i.e. for any PPT adversary $\mathcal{A}$, there exists a negligible $\varepsilon(\lambda)$ such that

$$\mathsf{Adv}(\mathcal{A}) \;:=\; \Pr\left[\mathsf{Exp}^{\mathsf{unp}}(\mathcal{A}) = 1\right] \;\leq\; \varepsilon(\lambda).$$

With $\mathsf{Exp}^{\mathsf{unp}}(\mathcal{A})$ being as defined in Figure 5.

$\mathsf{Exp}^{\mathsf{unp}}(\mathcal{A})$

1 : $\mathsf{vk}, \mathsf{sk} \leftarrow^{\$} \mathsf{VUF.Gen}(1^\lambda)$
2 : $(x^*, y^*, \pi^*) \leftarrow \mathcal{A}^{\mathsf{VUF.Eval}(\mathsf{sk}, \cdot)}(\mathsf{vk})$
3 : **If** x^* was previously queried: **Return** 0
4 : $(y, \pi) \leftarrow \mathsf{VUF.Eval}(\mathsf{sk}, x^*)$
5 : **Return** $(y^*, \pi^*) == (y, \pi)$

Fig. 5. The unpredictability experiment with adversary $\mathcal{A}$.

3 Unbiasability

3.1 Definition

We define our notion of *unbiasability* through an experiment. Let us begin with an informal overview. Initially an adversary is asked to produce a list of distinct verification keys $\mathsf{vk}_1, \ldots, \mathsf{vk}_n$ along with a predicate p. Next, the challenger sample uniformly some inputs $x_1, \ldots, x_m$. The adversary then evaluates its VRFs on these points and return them along with their proofs. The challenger finally "filters" the values with an incorrect proof, replaces them with $\perp$, and returns this vector $\mathbf{y}$. The adversary's goal is to eventually make the probability of $p(\mathbf{y}) = 1$ significantly greater than that of $p(\mathbf{z}) = 1$ with $\mathbf{z} \sim U(\mathbb{F}_q)$ in the output space.

For technical reasons however we cannot allow p to be *any* predicate. One motivation is that adversaries can always "bias" the output by selectively revealing only certain outputs and not all of them. In order to factor out this class of trivial attacks from our definition, we introduce a class of predicates we call *monotone*.

Definition 7. *Over $(\mathcal{Y} \cup \{\perp\})^n$ we define a partial order*

$$(y_1, \ldots, y_n) \leq (z_1, \ldots, z_n) \quad \Leftrightarrow \quad \forall i \; (y_i = z_i \vee y_i = \perp).$$

A (probabilistic) predicate $p : (\mathcal{Y} \cup \{\perp\})^n \times \{0,1\}^r \to \{0,1\}$ is monotone if, for a uniformly sampled random tape $\rho \sim U(\{0,1\}^r)$

$$\mathbf{y} \leq \mathbf{z} \quad \Rightarrow \quad \Pr\left[p(\mathbf{y}; \rho) = 1\right] \leq \Pr\left[p(\mathbf{z}; \rho) = 1\right].$$

As usually done for probabilistic Turing machines, we will not explicitly write the random tape ρ, and assume that evaluating $p(\mathbf{x})$ really means sampling $\rho \leftarrow^{\$} \{0,1\}^r$ and then computing $p(\mathbf{x}; \rho)$.

Definition 8. *A VRF is unbiasable if for any polynomially bounded integers n, m and PPT adversary $\mathcal{A}$, there exists a negligible function ε such that*

$$\mathsf{Adv}(\mathcal{A}) \coloneqq \Pr\left[\mathsf{Exp}^{\mathsf{bias}}_{0,n,m}(\mathcal{A}) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{bias}}_{1,n,m}(\mathcal{A}) = 1\right] \leq \varepsilon(\lambda)$$

where the experiment $\mathsf{Exp}^{\mathsf{bias}}_{b,n,m}$ is defined in Figure 6.

Note that our notion of unbiasability requires the adversary to evaluate the VRFs linked to the chosen verification keys on the *same* set of points. A relaxation of this notion, which we call *unbiasability on independent points*, instead ask the adversary to evaluate each VRF on an independently sampled set of points. More formally

Definition 9. *A VRF is unbiasable on independent points if for all polynomially bounded integers $n, m \in \mathbb{N}$ and PPT adversary $\mathcal{A}$, there exists a negligible function ε such that*

$$\mathsf{Adv}(\mathcal{A}) \coloneqq \Pr\left[\mathsf{Exp}^{\mathsf{ip-bias}}_{0,n,m}(\mathcal{A})) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{ip-bias}}_{1,n,m}(\mathcal{A})) = 1\right] \leq \varepsilon(\lambda)$$

where the experiment $\mathsf{Exp}^{\mathsf{ip-bias}}_{b,n,m}$ is defined in Figure 7.

$\mathsf{Exp}^{\mathsf{bias}}_{b,n,m}(\mathcal{A})$

1 : $(\mathsf{vk}_1, \ldots, \mathsf{vk}_n, p) \leftarrow \mathcal{A}$ such that:
2 : $\quad \mathsf{vk}_1, \ldots, \mathsf{vk}_n$ are all distinct
3 : $\quad p$ is a PPT computable monotone predicate
4 : $(x_1, \ldots, x_m) \leftarrow^{\$} \mathcal{X}$
5 : $((y_{1,1}, \pi_{1,1}), \ldots, (y_{n,m}, \pi_{n,m})) \leftarrow \mathcal{A}(x_1, \ldots, x_m)$
6 : **For** all $i \in [n],\ j \in [m]$:
7 : $\quad$ **If** $\mathsf{VRF.Vfy}(\mathsf{vk}_i, x_j, y_{i,j}, \pi_{i,j}) \to 0$: Set $y_{i,j} \leftarrow \perp$
8 : Sample $(z_{1,1}, \ldots, z_{n,m}) \leftarrow^{\$} \mathcal{Y}^{n,m}$
9 : **If** $b = 0$: **Return** $p(y_{1,1}, \ldots, y_{n,m})$
10 : **If** $b = 1$: **Return** $p(z_{1,1}, \ldots, z_{n,m})$

Fig. 6. The unbiasability experiment parametrized by $b \in \{0, 1\}$.

$\mathsf{Exp}^{\mathsf{ip-bias}}_{b,n,m}(\mathcal{A})$

1 : $(\mathsf{vk}_1, \ldots, \mathsf{vk}_n, p) \leftarrow \mathcal{A}$ such that:
2 : $\quad \mathsf{vk}_1, \ldots, \mathsf{vk}_n$ are all distinct
3 : $\quad p$ is a PPT computable monotone predicate
4 : $(x_{1,1}, \ldots, x_{n,m}) \leftarrow^{\$} \mathcal{X}$
5 : $((y_{1,1}, \pi_{1,1}), \ldots, (y_{n,m}, \pi_{n,m})) \leftarrow \mathcal{A}(x_{1,1}, \ldots, x_{n,m})$
6 : **For** all $i \in [n]$ and $j \in [m]$:
7 : $\quad$ **If** $\mathsf{VRF.Vfy}(\mathsf{vk}_i, x_{i,j}, y_{i,j}, \pi_{i,j}) \to 0$: Set $y_{i,j} \leftarrow \perp$
8 : Sample $(z_{1,1}, \ldots, z_{n,m}) \leftarrow^{\$} \mathcal{Y}^{n,m}$
9 : **If** $b = 0$: **Return** $p(y_{1,1}, \ldots, y_{n,m})$
10 : **If** $b = 1$: **Return** $p(z_{1,1}, \ldots, z_{n,m})$

Fig. 7. The unbiasability experiment with independently sampled points for each vk.

3.2 Properties

Having provided our unbiasability definition, we now show it to imply some desirable properties.

Pseudorandom Output. Informally if a VRF is unbiasable then its evaluations on random points also looks random (even with adversarially generated keys). This roughly follows as any PPT distinguisher $\mathcal{D}$ trying to distinguish the VRF output from uniformly random values, can be converted into a probabilistic monotone predicate $p^{\mathcal{D}}$ such that

$$p^{\mathcal{D}}(y_1, \ldots, y_m) \ = \ \begin{cases} 0 & \text{If } y_i = \perp \\ \mathcal{D}(y_1, \ldots, y_m) & \text{Otherwise} \end{cases}.$$

If $\mathcal{D}$ were to succeed with significant probability then the algorithm which computed $\mathsf{vk}_1, \ldots, \mathsf{vk}_n$ can be used to bias $p^{\mathcal{D}}$. More in detail, in order to make a meaningful statement about the "VRF output" even when the verification key is maliciously chosen, we will restrict the class of adversaries to those who always returns valid output and proofs on random inputs. For this class of adversaries we can state the following Proposition:

Proposition 1. *Let $\mathcal{A}$ be a* PPT *machine $\mathcal{A}$ initially computing $\mathsf{vk}_1, \ldots, \mathsf{vk}_n$ and then on input $x_1, \ldots, x_m \sim U(\mathcal{X}^m)$ always returning $y_{1,1}, \ldots, y_{n,m}$ along with valid proofs $\pi_{1,1}, \ldots, \pi_{n,m}$, i.e. such that $\mathsf{VRF.Vfy}(\mathsf{vk}_i, x_j, y_{i,j}, \pi_{i,j}) \to 1$.*

If the VRF satisfies unbiasability, sampling $\mathbf{z} \sim U(\mathcal{Y}^{n,m})$, then for any PPT *distinguisher $\mathcal{D}$ there exists a negligible ε such that*

$$\mathsf{Adv}(\mathcal{D}) \;\coloneqq\; \left|\Pr\left[\mathcal{D}(\mathbf{y}) \to 1\right] - \Pr\left[\mathcal{D}(\mathbf{z}) \to 1\right]\right| \;\leq\; \varepsilon(\lambda).$$

Weak Unbiasability. In [EKS+21] another notion of unbiasability was proposed. Their definition informally says that any adversary generating one key vk cannot guess the VRF value on a random point significantly better than randomly guessing. More formally, given a PPT adversary $\mathcal{A}$ their experiment is defined as in Figure 8. They then say that a VRF satisfies their security definition if there exists a negligible function ε such that

$$\mathsf{Adv}(\mathcal{A}) \;=\; \Pr\left[\mathsf{Exp}^{\mathsf{w-bias}}(\mathcal{A}) = 1\right] \;\leq\; \frac{1}{|\mathcal{Y}|} + \varepsilon(\lambda).$$

Our notion immediately implies this one up to observing that the adversary is

$\mathsf{Exp}^{\mathsf{w-bias}}(\mathcal{A})$

1 : $(\mathsf{vk}, y^*) \leftarrow^{\$} \mathcal{A}(1^\lambda)$
2 : $x \leftarrow^{\$} \mathcal{X}$
3 : $(\pi, y) \leftarrow^{\$} \mathcal{A}(x)$
4 : $\beta \leftarrow \mathsf{VRF.Vfy}(\mathsf{vk}, x, y, \pi)$
5 : **Return** 1 iff $\beta = 1$ and $y = y^*$

Fig. 8. The weak-unbiasability experiment [EKS+21].

attempting to bias the predicate $p_{y^*}(y)$ that is 0 if $y^* \neq y$ (which also includes the case $y = \perp$) and 1 otherwise. Note also that $|\mathcal{Y}|^{-1}$ is precisely the probability that $p_{y^*}(z) = 1$ for a randomly chosen z. We thus rename their notion as "weak-unbiasability" and formally state that

Proposition 2. *Every unbiasable VRF if also weakly unbiasable.*

In the Appendix, Section A.1 we further justify the name "weak-unbiasability" by providing examples of weakly unbiasable VRF that are not unbiasable.

Fairness in VRF-based leader election. The goal of secret leader election (SLE) protocols is to anonymously identify one user among many. VRF-based construction [CM16, KRDO17, DGKR17] generally follows the following blueprint: Initially each user P_i publishes a VRF key vk_i. Later a random input is sampled and the user with the lowest VRF output is the winner.

Among the properties such protocol should satisfy, *fairness* requires that no group of corrupted parties can win with probability higher than expected. More formally, a group C of t out of n corrupted parties should only contain a winner with probability $t/n+\varepsilon$. Assuming the VRF to be unbiasable, we can then prove VRF-based leader elections to be fair. Indeed, any adversary corrupting C parties and winning with high probability is also successfully biasing the predicate

$$p(y_1,\ldots,y_n) \ : \ \begin{cases} 0 & \text{If } y_i = \perp,\ i \notin C \\ \min_{i\in C}\{y_i\} \leq \min_{i\notin C}\{y_i\} & \text{Otherwise} \end{cases}$$

where $\perp$ is treated as $+\infty$ in the min operations above. Note p is monotone as replacing any y_i with $\perp$ for $i \notin C$ makes the predicate false, and doing the same for $i \in C$ does not decrease the left hand side of the inequality.

4 Unbiasable VRF in the ROM

4.1 From any VUF

In this section we analyze a folklore compiler that transforms any VUF into a VRF in the ROM, and show the resulting construction to be unbiasable as per Definition 8. The idea is simply to apply the RO on the VUF output, the verification key vk and the VUF input x. A full description is provided in Figure 9

Notice that in order to obtain unbiasability hashing both vk and x along with the VUF output is necessary: If we were to remove x, it may be the case that for some maliciously generated verification key vk^*, the underlying function is constant. This would not contradict unpredictability as vk^* is only chosen with negligible probability. However, for several random inputs the RO applied to the VUF output and verification key would always yield the same value, implying that the resulting construction is not unbiasable.

If we were to remove vk we face similar issues: for some VUF there may exists two distinct verification keys $\mathsf{vk}_1^*, \mathsf{vk}_2^*$ such that the underlying functions are identical. Again this does not contradict unpredictability, but the resulting construction would now give the same output when evaluated for vk_1^* and vk_2^* on the same inputs, which contradicts unbiasability.

Theorem 1. *If $|\mathcal{X}| = \Omega(2^\lambda)$, with $\mathcal{X}$ being the VUF input space, the construction described in Figure 9 is an Unbiasable VRF in the ROM.*

Proof of Theorem 1. Correctness, unique provability and pseudorandomness immediately follows from VUF properties and the usage of the ROM. We thus only focus on unbiasability.

$\mathsf{VRF.Gen}(1^\lambda)$

1 : **Return** $(\mathsf{vk}, \mathsf{sk}) \leftarrow^{\$} \mathsf{VUF.Gen}(1^\lambda)$

$\mathsf{VRF.Eval}(\mathsf{sk}, x)$

1 : $(y^*, \pi^*) \leftarrow \mathsf{VUF.Eval}(\mathsf{sk}, x)$
2 : $y \leftarrow \mathsf{H}(\mathsf{vk}, x, y^*)$
3 : $\pi \leftarrow (\pi^*, y^*)$
4 : **Return** (y, π)

$\mathsf{VRF.Vfy}(\mathsf{vk}, x, y, \pi)$

1 : Parse $\pi = (\pi^*, y^*)$
2 : **Return** 1 if and only if:
3 : $\quad \mathsf{H}(\mathsf{vk}, x, y^*) = y$
4 : $\quad \mathsf{VUF.Vfy}(\mathsf{vk}, x, y^*, \pi^*) = 1$

Fig. 9. Unbiasable VRF based on a VUF using a Random Oracle H.

Unbiasability. Let $\mathcal{A}$ be an adversary playing the game in Figure 6. We will call Q the set of ROM queries it performs before returning $(\mathsf{vk}_1, \ldots, \mathsf{vk}_n, p)$ and define the two events

$$\begin{aligned} \mathsf{Seen} &:= \exists i \in [n],\ j \in [m],\ y \in \{0,1\}^* \ :\ (\mathsf{vk}_i, x_j, y) \in Q \\ \mathsf{Coll} &:= \exists j, j' \in [m] \ :\ x_j = x_{j'} \end{aligned}$$

where Seen means a query with prefix (vk_i, x_j) was made and Coll means no collision occurs among the elements $x_1, \ldots, x_m$. These two events occur with negligible probability

$$\Pr[\mathsf{Seen}] \ \leq\ \frac{m|Q|}{|\mathcal{X}|}, \qquad \Pr[\mathsf{Coll}] \ \leq\ \frac{m^2}{|\mathcal{X}|}.$$

Let now $\mathbf{y}^*$ be the "filtered" output of $\mathcal{A}$ in the unbiasability experiment, so that values with invalid proofs are replaced with $\perp$. For all i and j we define $y_{i,j}$ to be $y^*_{i,j}$ if this value is not $\perp$ and otherwise sample $y_{i,j} \sim U(\mathcal{Y})$. According to Definition 7

$$\mathbf{y}^* \leq \mathbf{y} \quad \Rightarrow \quad \Pr[p(\mathbf{y}^*) = 1] \leq \Pr[p(\mathbf{y}) = 1]$$

with p being the monotone predicate chosen by $\mathcal{A}$. To conclude we observe than when conditioning on $\neg\mathsf{Seen}, \neg\mathsf{Coll}$, all the elements $y^*_{i,j} \neq \perp$ are evaluations of the random oracle on different[7] and not yet queried points, thus are uniformly random. Hence, setting $\mathbf{z} \sim U(\mathcal{Y}^{n,m})$

$$\begin{aligned} \Delta(\mathbf{y}, \mathbf{z}) \ &\leq\ \Delta(\mathbf{y}_{|\neg\mathsf{Seen},\neg\mathsf{Coll}}, \mathbf{z}_{|\neg\mathsf{Seen},\neg\mathsf{Coll}}) + \Pr[\mathsf{Seen} \ \vee\ \mathsf{Coll}] \\ &\leq\ \Pr[\mathsf{Seen}] + \Pr[\mathsf{Coll}] \\ &\leq\ \frac{m|Q| + m^2}{|\mathcal{X}|}. \end{aligned}$$

[7] Because $\mathsf{vk}_i \neq \mathsf{vk}_j$ by construction and $x_i \neq x_j$ by $\neg\mathsf{Coll}$.

where the second inequality follows as $\mathbf{y}$ is uniform upon conditioning on $\neg\mathsf{Seen}$, $\neg\mathsf{Coll}$. We can then bound the advantage of $\mathcal{A}$ as

$$\begin{aligned}\mathsf{Adv}(\mathcal{A}) \;&=\; \Pr\left[p(\mathbf{y}^*)=1\right]-\Pr\left[p(\mathbf{z})=1\right] \;\leq\\ &\leq\; \Pr\left[p(\mathbf{y})=1\right]-\Pr\left[p(\mathbf{z})=1\right] \;\leq\; \Delta(\mathbf{y},\mathbf{z}) \;\leq\; \frac{m|Q|+m^2}{|\mathcal{X}|}.\end{aligned}$$

4.2 From weakly unbiasable VUF

Although the compiler presented in Section 4.1 shows that any VUF can be converted into an unbiasable VRF in the random oracle model, not all schemes used in practice follow that blueprint. This most notably include the case of ECVRF [PWH+17] currently included in the standardization effort put forward in [GRPV23].

In order to capture ECVRF we propose a slightly different compiler, which does not include the verification key vk and input x in the RO query for evaluations. This, as discussed before, may lead to insecure constructions in general. Therefore to prove security we will need the VUF to satisfy two extra properties, both achieved by the VUF used in ECVRF. First of all, we need that no adversary can guess the VUF output on a random input, even when it get to chose the VUF keys. In order words, we need the VUF to be weakly unbiasable (see Section 3.2). Moreover, we further need that for any given key vk, two distinct random inputs x_1, x_2 and their respective outputs y_1, y_2, then $\Pr[y_1 = y_2]$ is negligible. This second property is required to exclude VUFs with two keys defining (almost) the same function. Assuming these properties we can then prove our weaker compiler, presented in Figure 10, to be a secure unbiasable VRF.

$\mathsf{VRF.Gen}(1^\lambda)$

1 : **Return** $(\mathsf{vk},\mathsf{sk}) \leftarrow^{\$} \mathsf{VUF.Gen}(1^\lambda)$

$\mathsf{VRF.Eval}(\mathsf{sk}, x)$

1 : $(y^*, \pi^*) \leftarrow \mathsf{VUF.Eval}(\mathsf{sk}, x)$
2 : $y \leftarrow \mathsf{H}(y^*)$
3 : $\pi \leftarrow (\pi^*, y^*)$
4 : **Return** (y, π)

$\mathsf{VRF.Vfy}(\mathsf{vk}, x, y, \pi)$

1 : Parse $\pi = (\pi^*, y^*)$
2 : **Return** 1 if and only if:
3 : $\mathsf{H}(y^*) = y$
4 : $\mathsf{VUF.Vfy}(\mathsf{vk}, x, y^*, \pi^*) = 1$

Fig. 10. Unbiasable VRF based on a special VUF. Differences from the construction in Figure 9 are highlighted.

Theorem 2. *Let* $(\mathsf{VUF.Gen}, \mathsf{VUF.Eval}, \mathsf{VUF.Vfy})$ *be a weakly unbiasable (see Section 3.2) VUF so that there exists a negligible function* ε *such that for all* vk, $x_1, x_2 \sim U(\mathcal{X})$ *and* y_1, y_2 *for which there exists* π_1, π_2 *with*

$$\mathsf{VUF.Vfy}(\mathsf{vk}_1, x, y_1, \pi_1) = 1, \quad \mathsf{VUF.Vfy}(\mathsf{vk}_2, x, y_2, \pi_2) = 1$$

then $\Pr[y_1 = y_2] \leq \varepsilon(\lambda)$.

Then the construction presented in Figure 10 is an Unbiasable VRF in the random oracle model.

A proof of this Theorem appears in the Appendix, Section B.1.

5 Constructions in the Standard Model

In this section we will first provide two unbiasable VRFs in the standard model achieving weaker notions of both pseudorandomness and unbiasability. More specifically both constructions achieve

1. Restricted pseudorandomness on the first component of their input space (see Definition 6).
2. Unbiasability on independent points (Definition 9) holding unconditionally, i.e. against computationally unbounded adversaries.

We target these properties as at the end of this Section we provide a compiler transforming any such VRF into a fully-fledged unbiasable VRF in the standard model.

5.1 1st Preliminary Construction: Padded VRF

Our first construction critically shows how targeting only restricted pseudorandomness and unbiasability with independent points simplifies the problem. The construction is based on two observations:

First, any (publicly computable) permutation is unbiasable with independent points (although not a VRF), as it preserves the input distribution. The easiest example is the identity function. The second idea is that, given a VRF $F_{\mathsf{sk}}(\cdot) = \mathsf{VRF.Eval}(\mathsf{sk}, \cdot)$ and a publicly computable unbiasable function $f(\cdot)$, we can combine the two, returning on input $x = (u, v)$ the output $F_{\mathsf{sk}}(u) \oplus f(v)$. This informally remains unbiasable because $f(v)$ preserves the entropy in v, and $F_{\mathsf{sk}}(u)$ is independent from v. Moreover, this achieves restricted pseudorandomness on the first component because the adversary is not allowed to evaluate the VRF on two different points $x, \bar{x}$ with the same first component u, $\bar{u}$.

The resulting construction, given for $f = \mathsf{id}$ and $(\mathsf{VRF}^*\mathsf{.Gen}, \mathsf{VRF}^*\mathsf{.Eval}, \mathsf{VRF}^*\mathsf{.Vfy})$ a VRF, is described in Figure 11, and we will refer to it as the *padded VRF* construction. There we assume that the VRF output space $\mathcal{Y}^*$ is a group $(\mathbb{G}, +)$. This generalizes the case $\mathcal{Y}^* = \{0,1\}^\mu$ which is a group with the bit-wise xor. Our compiler in Sections 5.5 however will only work for the special case in which the VRF's input space is a prime order group.

$\mathsf{VRB.Gen}(1^\lambda)$

1 : **Return** $(\mathsf{vk}, \mathsf{sk}) \leftarrow^{\$} \mathsf{VRF}^*.\mathsf{Gen}(1^\lambda)$

$\mathsf{VRB.Eval}(\mathsf{sk}, x)$

1 : Parse $x = (u, v)$
2 : $(w, \pi) \leftarrow \mathsf{VRF}^*.\mathsf{Eval}(\mathsf{sk}, u)$
3 : $y \leftarrow w + v$
4 : **Return** y

$\mathsf{VRB.Vfy}(\mathsf{vk}, x, y, \pi)$

1 : Parse $x = (u, v)$
2 : $w \leftarrow y - v$
3 : **Return** 1 if and only if:
4 : $\quad \mathsf{VRF.Vfy}(\mathsf{vk}, w, u, \pi) \rightarrow 1$

Fig. 11. Padded VRF construction from a VRF $(\mathsf{VRF}^*.\mathsf{Gen}, \mathsf{VRF}^*.\mathsf{Eval}, \mathsf{VRF}^*.\mathsf{Vfy})$ with output space a group $(\mathbb{G}, +)$.

Theorem 3. *If* $(\mathsf{VRF}^*.\mathsf{Gen}, \mathsf{VRF}^*.\mathsf{Eval}, \mathsf{VRF}^*.\mathsf{Vfy})$ *is a secure VRF, then the construction in Figure 11 is a VRF with restricted pseudorandomness. Moreover it satisfies unbiasability on independent points against computationally unbounded adversaries.*

A proof of this Theorem appears in the Appendix, Section B.2.

5.2 Verifiable Random Bijection

Our second approach to construct a VRF satisfying unbiasability on independent points is to guarantee that for each key, including maliciously generated ones, the resulting VRF is 1 to 1. In this section we formally defined this class of VRF, which we call *Verifiable Random Bijection* (VRB). For the sake of generality, we do not require each function to be always 1 to 1, which would exclude VRFs admitting "bad" keys that cannot be evaluated over the full domain. Instead, we more generally ask the function to have input and output space of the same cardinality and to always be injective. Formally

Definition 10. *A Verifiable Random Bijection is a VRF* $(\mathsf{VRB.Gen}, \mathsf{VRB.Eval}, \mathsf{VRB.Vfy})$ *whose input space* $\mathcal{X}$ *and output space* $\mathcal{Y}$ *have the same cardinality* $|\mathcal{X}| = |\mathcal{Y}|$ *and furthermore satisfies*

4. ***Injectivity.*** *For any* vk *(not necessarily in the range of* $\mathsf{VRF.Gen}$*), any output* $y \in \mathcal{Y}$*, pair of inputs* $x_0, x_1 \in \mathcal{X}$ *and proofs* π_0, π_1 *if holds that*

$$\mathsf{VRF.Vfy}(\mathsf{vk}, x_0, y, \pi_0) \rightarrow 1, \quad \mathsf{VRF.Vfy}(\mathsf{vk}, x_1, y, \pi_1) \rightarrow 1 \quad \Rightarrow \quad x_0 = x_1.$$

In the Appendix, Section B.3 we provide a proof of the following theorem, stating that for any VRF (not even necessarily pseudorandom), injectivity suffices to achieve unconditionally unbiasability on independent points.

Theorem 4. *Any tuple* $(\mathsf{VRF.Gen}, \mathsf{VRF.Eval}, \mathsf{VRF.Vfy})$ *with* $|\mathcal{X}| = |\mathcal{Y}|$ *satisfying correctness, unique provability and injectivity, then further satisfies unbiasability on independent points against computationally unbounded adversaries.*

5.3 2nd Preliminary Construction: 2-Feistel Rounds

In this section we observe that 2 Feistel rounds [LR88] suffice to transform any VRF into a VRB, at the cost of only achieving restricted pseudorandomness. Such weaker security notion is the reason why we do not need 3 or more rounds. Indeed, using notation from Figure 1, known attacks to break 2 Feistel rounds require to perform queries with the same component z_1 but different w_1. This is disallowed when considering pseudorandomness restricted to the first component, where two queries with the same first entry are considered the same by the evaluation oracle.

Note that to instantiate this construction we do need the VRF's input space $\mathcal{X}$ to be a group $\mathbb{G}$ with some operation $+$ and that the output space is also $\mathbb{G}$. The first condition is achieved assuming $\mathbb{G} \subseteq \mathcal{X}$ and then restricting the input space. The second one instead can be achieved by amplifying the VRF output length and then applying a universal hash to the group (in prime order groups, this hash can simply be the exponentiation by a generator). A full description of the scheme is presented in Figure 12

$\mathsf{VRB.Gen}(1^\lambda)$

1 : $\mathsf{vk}_1, \mathsf{sk}_1 \leftarrow^{\$} \mathsf{VRF.Gen}(1^\lambda), \quad \mathsf{vk}_2, \mathsf{sk}_2 \leftarrow^{\$} \mathsf{VRF.Gen}(1^\lambda)$
2 : $\mathsf{vk} \leftarrow (\mathsf{vk}_1, \mathsf{vk}_2),\ \mathsf{sk} \leftarrow (\mathsf{sk}_1, \mathsf{sk}_2)$
3 : **Return** $(\mathsf{vk}, \mathsf{sk})$

$\mathsf{VRB.Eval}(\mathsf{sk}, x)$

1 : Parse $x = (z_1, w_1)$
2 : $(u_1, \pi_1) \leftarrow \mathsf{VRF.Eval}(\mathsf{sk}_1, z_1)$
3 : $(z_2, w_2) \leftarrow (z_1, w_1 + u_1)$
4 : $(u_2, \pi_2) \leftarrow \mathsf{VRF.Eval}(\mathsf{sk}_2, w_2)$
5 : $(z_3, w_3) \leftarrow (z_2 + u_2, w_2)$
6 : $y \leftarrow (z_3, w_3)$
7 : $\pi \leftarrow (u_1, u_2, \pi_1, \pi_2)$
8 : **Return** $y \leftarrow (z_3, w_3)$

$\mathsf{VRF.Vfy}(\mathsf{vk}, x, y, \pi)$

1 : Parse $x = (z_1, w_1)$
2 : Parse $\pi = (u_1, u_2, \pi_1, \pi_2)$
3 : $(z_2, w_2) \leftarrow (z_1, w_1 + u_1)$
4 : $(z_3, w_3) \leftarrow (z_2 + u_2, w_2)$
5 : **Return** 1 if and only if:
6 : $\quad \mathsf{VRF.Vfy}(\mathsf{vk}_1, z_1, u_1, \pi_1) \to 1$
7 : $\quad \mathsf{VRF.Vfy}(\mathsf{vk}_2, w_2, u_2, \pi_2) \to 1$
8 : $\quad (z_3, w_3) = y$

Fig. 12. 2-Feistel round construction

Theorem 5. *If* $(\mathsf{VRF.Gen}, \mathsf{VRF.Eval}, \mathsf{VRF.Vfy})$ *is a secure VRF with input space* $\mathbb{G}$*, then the construction in Figure 12 is a VRB with message space* $\mathbb{G} \times \mathbb{G}$ *and restricted pseudorandomness on the first component.*

The Theorem is again proven in the Appendix, Section B.4.

5.4 VRB Compiler

Before presenting in the next section a general compiler lifting our preliminary constructions to fully fledged unbiasable VRF, we discuss how to modify the 2-Feistel Round construction (or more generally, any VRB with constrained pseudorandomness) in order to obtain a VRB satisfying the regular pseudorandomness property. We see this as a useful stepping stone to introduce in a simpler setting some of the techniques used later on. Furthermore, security of this construction will eventually depend on simpler assumptions. Namely the underlying VRF security and the hardness of standard DLP.

The main issue of the construction in Section 5.3 is that pseudorandomness fails when an adversary evaluates the VRF in two different points sharing the same first component. A trivial fix could be to compose the 2-Feistel rounds with an hash functions, so that on input $x \in \mathbb{G}$, we evaluate that VRF on $(h(x), x)$. This however introduces collisions which, although hard to compute, would break injectivity.

Our solution is to rely on a Pedersen hash: given $\mathbf{x} \in \mathbb{F}_q^2$ and $[\mathbf{a}]$ a hash key, the hash of $\mathbf{x}$ is the inner product $\left[\mathbf{a}^\top \mathbf{x}\right]$. With this tool we can build a map from $\mathbb{F}_q^2 \to \mathbb{G}^2$ keyed on $[\mathbf{a}]$ such that

$$\mathbf{x} \mapsto \left(\left[\mathbf{a}^\top \mathbf{x}\right],\ \left[\mathbf{e}_1^\top \mathbf{x}\right]\right).$$

were $\mathbf{e}_1 = (1, 0)$. This map is collision resistant on the first component when $\mathbf{a}$ is sampled by an honest user, which we needed for pseudorandomness. Moreover, even when $\mathbf{a}$ is sampled maliciously, it is possible to verify whether this map is a bijection or not by checking $[a_2] \neq 0$. Indeed in this case the matrix $A = (\mathbf{a}, \mathbf{e}_1)^\top$ has non-zero determinant and thus it is bijection. Our solution, consisting of the composition of this map and the construction of Section 5.3, is schematized in Figure 13 and formally described in Figure 14.

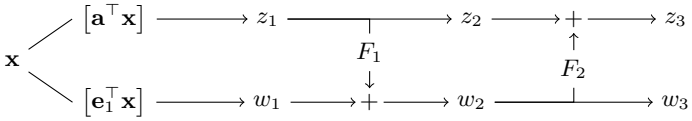

Fig. 13. 2-Feistel rounds with Petersen hash, where $F_i(\cdot) = \mathsf{VRF.Eval}(\mathsf{sk}_i, \cdot)$ and $\mathbf{a}, \mathbf{e}_1, \mathbf{x}$ are vectors in $\mathbb{F}_q^2$ with $\mathbf{e}_1 = (1, 0)$.

Theorem 6. *If* $(\mathsf{VRB}^*.\mathsf{Gen}, \mathsf{VRB}^*.\mathsf{Eval}, \mathsf{VRB}^*.\mathsf{Vfy})$ *is a VRB with domain* $\mathbb{G}^2$ *satisfying restricted pseudorandomness with respect to the first component, and* DLP *is hard, then the construction in Figure 14 is a secure VRB.*

5.5 Unbiasable VRF Compiler

We finally provide our compiler lifting any VRF satisfying restricted pseudorandomness and unbiasability on independent points to fully fledged unbiasable

$\mathsf{VRB.Gen}(1^\lambda)$

1 : Sample $\mathsf{vk}^*, \mathsf{sk}^* \leftarrow^\$ \mathsf{VRB}^*.\mathsf{Gen}(1^\lambda)$
2 : Sample $\mathbf{a} \leftarrow^\$ \mathbb{F}_q^2$ with $a_2 \neq 0$
3 : $\mathsf{vk} \leftarrow (\mathsf{vk}^*, [\mathbf{a}]),\ \mathsf{sk} \leftarrow \mathsf{sk}^*$
4 : **Return** $(\mathsf{vk}, \mathsf{sk})$

$\mathsf{VRB.Eval}(\mathsf{sk}, \mathbf{x})$

1 : // $\mathbf{x} \in \mathbb{F}_q^2$
2 : $z \leftarrow \left[\mathbf{a}^\top \mathbf{x}\right]$
3 : $w \leftarrow \left[\mathbf{e}_1^\top \mathbf{x}\right]$
4 : $(y, \pi) \leftarrow \mathsf{VRB}^*.\mathsf{Eval}(\mathsf{sk}^*, (z, w))$
5 : **Return** (y, π)

$\mathsf{VRB.Vfy}(\mathsf{vk}, x, y, \pi)$

1 : $z \leftarrow \left[\mathbf{a}^\top \mathbf{x}\right]$
2 : $w \leftarrow \left[\mathbf{e}_1^\top \mathbf{x}\right]$
3 : **Return** 1 if and only if:
4 : $\quad \mathsf{VRB}^*.\mathsf{Vfy}(\mathsf{vk}^*, (z, w), y, \pi)$
5 : $\quad [a_2] \neq 0.$

Fig. 14. Compiler returning a secure VRB from one with restricted pseudorandomness.

VRF. Combining this with the constructions in Section 5.1 and 5.3 yields two compilers from VRF to unbiasable VRF. Note the resulting constructions are in the standard model, meaning without random oracles and setup assumptions, i.e. no CRS.

In order to enhance pseudorandomness we use the same technique of Section 5.4: Interpreting the input as a vector $\mathbf{x} \in \mathbb{F}_q^\mu$, we compute the first component as a Petersen Hash $\left[\mathbf{a}^\top \mathbf{x}\right]$ and the second one as a linear function $\mathbf{b}^\top \mathbf{x}$, with $[\mathbf{a}]$ and $\mathbf{b}$ being part of the public key.

We now discuss how to obtain unbiasability. The challenge is that in Definition 8 we require the adversary to evaluate all of the VRFs on the *same* set of inputs, whereas the underlying VRF we start with is only unbiasable if each VRF is evaluated on *independently sampled* ones. Our idea is to derive those independently sampled sets, one for each key vk chosen by the adversary, from a common one using a PRF. In the introduction we discussed as non-working examples interpreting the input as a PRF key k and evaluating the VRF of $f_k(\mathsf{vk})$ or $(k, f_k(\mathsf{vk}))$. These however either break pseudorandomness if $k \mapsto f_k(\mathsf{vk})$ is not injective, or prevent us from using the PRF pseudorandomness in the proof. To fix notation, let Encode be the procedure performing such expansion (i.e. returning a vector $\mathbf{v}$ with entries in $\mathbb{F}_q$ which we pass as input to the Petersen Hash). Our solution improves on the trivial ones by relying on a perfectly binding commitment scheme Com:

– The input x is assumed to be of the form $(\mathbf{r}, k, \mathsf{pp}, \rho)$ with $\mathbf{r} \in \mathbb{F}_q$ and pp the public parameter of a perfectly binding commitment scheme.

– $\mathsf{Encode}(\mathbf{r}, k, \mathsf{pp}, \rho) = (\mathbf{r}, f_k(\mathsf{vk}), \mathsf{pp}, \mathsf{Com}_{\mathsf{pp}}(k; \rho))$ where $\mathbf{r} \in \mathbb{F}_q^2$ is used to turn the Pedersen hash into a chameleon hash, $f_k(\mathsf{vk}) \in \mathbb{F}_q^2$ randomizes the Petersen hash output *independently* for each vk.

Note that one key requirement of Encode is to be injective to preserve pseudorandomness. Hence the above solution only works if the map $(k, \mathsf{pp}, \rho) \mapsto \mathsf{Com}_{\mathsf{pp}}(k; \rho)$ is injective. Furthermore the commitment scheme must admit uniformly random public parameters. This unfortunately excludes a large class of constructions (e.g. lattice based). However all the above properties are true for ElGamal commitments, where, given a public group element $[\alpha]$ a commitment to k is defined as $[\beta], [\alpha\beta + k]$.

We eventually provide a full description of our solution for this specific choice of commitment scheme: first we let $\mathsf{repr} : \mathbb{G} \to \mathbb{F}_q^\eta$ be an efficiently computable and injective map. If $\mathbb{G} \subseteq \{0,1\}^\ell$ it exists with $\eta \leq \ell / \lfloor \log_2 q \rfloor$ by encoding $\lfloor \log_2 q \rfloor$ bits per field element. Given such representation map, we define $\mathsf{Encode} : \mathbb{F}_q^4 \times \mathcal{K} \to \mathbb{F}_q^{4+3\eta}$, with $\mathcal{K}$ being the set of possible verification keys vk, as

$$\mathsf{Encode}(r_1, r_2, \alpha, \beta, k, \mathsf{vk}) \;:=\; (r_1, r_2, f_k(\mathsf{vk}), \mathsf{repr}([\alpha]), \mathsf{repr}([\beta]), \mathsf{repr}([\alpha\beta + k]))\,.$$

Having defined the encoding procedure we can finally describe our compiler, taking as input a VRF $(\mathsf{VRF}^*.\mathsf{Gen}, , \mathsf{VRF}^*.\mathsf{Eval}, \mathsf{VRF}^*.\mathsf{Vfy})$, described in Figure 15. A proof of Theorem 7 is presented in the Appendix, Section B.6.

$\mathsf{VRF.Gen}(1^\lambda)$

1 : Sample $(\mathsf{vk}^*, \mathsf{sk}^*) \leftarrow^{\$} \mathsf{VRF}^*.\mathsf{Gen}(1^\lambda)$
2 : Sample $\mathbf{a} \leftarrow^{\$} \mathbb{F}_q^{4+3\eta}$ with $a_2, a_4 \neq 0$
3 : Set $\mathsf{vk} \leftarrow (\mathsf{vk}^*, [\mathbf{a}])$ and $\mathsf{sk}^* \leftarrow \mathsf{sk}$
4 : **Return** $(\mathsf{vk}, \mathsf{sk})$

$\mathsf{VRF.Eval}(\mathsf{sk}, x)$

1 : // $x \in \mathbb{F}_q^5$
2 : $\mathbf{v} \leftarrow \mathsf{Encode}(x, \mathsf{vk})$
3 : $z \leftarrow [\mathbf{a}^\top \mathbf{v}]$
4 : $w \leftarrow [\mathbf{e}^\top \mathbf{v}]$ // $\mathbf{e} = \mathbf{e}_1 + \mathbf{e}_3$
5 : $(y, \pi) \leftarrow^{\$} \mathsf{VRF}^*.\mathsf{Eval}(\mathsf{sk}^*, (z, w))$
6 : **Return** (y, π)

$\mathsf{VRF.Vfy}(\mathsf{vk}, x, y, \pi)$

1 : $\mathbf{v} \leftarrow \mathsf{Encode}(x, \mathsf{vk})$
2 : $z \leftarrow [\mathbf{a}^\top \mathbf{v}]$
3 : $w \leftarrow [\mathbf{e}^\top \mathbf{v}]$ // $\mathbf{e} = \mathbf{e}_1 + \mathbf{e}_3$
4 : **Return** 1 if and only if:
5 : $[a_2] \neq 0,\ [a_4] \neq 0$
6 : $\mathsf{VRF}^*.\mathsf{Vfy}(\mathsf{vk}^*, (z, w), y, \pi) \to 1$

Fig. 15. Compiler returning an unbiasable VRF.

Theorem 7. *Let* $(\mathsf{VRF}^*.\mathsf{Gen}, \mathsf{VRF}^*.\mathsf{Eval}, \mathsf{VRF}^*.\mathsf{Vfy})$ *be a VRF with domain* $\mathbb{G} \times \mathbb{G}$*, restricted pseudorandomness on the first component, and unconditional unibasability on inpendent points (i.e. holding against unbounded adversaries). If* $\mathsf{DLPrep\text{-}DDH}$ *is hard in* $\mathbb{G}$ *and* f *is a PRF secure against* PPT *adversaries with preprocessing in* $\mathsf{PPT}^{\mathsf{DL}}$ *(see Section 2.3), then the construction described in Figure 15 is an Unbiasable VRF.*

6 Conclusions

In conclusion we initiated the study of unbiasable verifiable random functions and proved them to be useful for applications and realizable in the plain model. Regarding future research directions, we list some problems we leave open whose solution would improve our understanding of these objects.

First of all, it is unclear to us under what hypothesis it is possible into compile any VRF to an unbiasable one. For instance, having a simpler construction based only on symmetric key primitives would narrow down the efficiency gap between plain model and ROM constructions. A simpler problem would be to understand whether unbiasability can be based only on plausibly post-quantum hypothesis. We further leave open understanding whether preprocessing is necessary to achieve unbiasability in the plain model. This was the case in our proof as the reduction uses the adversary as a black-box oracle to evaluate the VRF, and eventually needs to evaluate such VRF on a partially unknown point. This means it must violate the pseudorandomness property, and super polynomial preprocessing is the extra edge to do so. We hence believe that overcoming such limitations would also likely lead to novel proof techniques.

Acknowledgments

The authors would like to thanks Handan Kılınç Alper for the helpful discussions regarding the UC-Security of VRF. This work has been partially supported by PRODIGY Project (TED2021-132464B-I00) funded by MCIN/AEI/10.13039/501100011033/ and the European Union NextGenerationEU/PRTR.

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.

BASV23a. Jeffrey Burdges, Handan Kılınç Alper, Alistair Stewart, and Sergey Vasilyev. Ethical identity, ring VRFs, and zero-knowledge continuations. Cryptology ePrint Archive, Report 2023/002, 2023. `https://eprint.iacr.org/2023/002`.

BASV23b. Jeffrey Burdges, Handan Kılınç Alper, Alistair Stewart, and Sergey Vasilyev. Sassafras and semi-anonymous single leader election. Cryptology ePrint Archive, Report 2023/031, 2023. https://eprint.iacr.org/2023/031.

BGK+18. Christian Badertscher, Peter Gazi, Aggelos Kiayias, Alexander Russell, and Vassilis Zikas. Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 913–930. ACM Press, October 2018.

BGQR22. Christian Badertscher, Peter Gazi, Iñigo Querejeta-Azurmendi, and Alexander Russell. A composable security treatment of ECVRF and batch verifications. In Vijayalakshmi Atluri, Roberto Di Pietro, Christian Damsgaard Jensen, and Weizhi Meng, editors, *ESORICS 2022, Part III*, volume 13556 of *LNCS*, pages 22–41. Springer, Heidelberg, September 2022.

BGRV09. Zvika Brakerski, Shafi Goldwasser, Guy N. Rothblum, and Vinod Vaikuntanathan. Weak verifiable random functions. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 558–576. Springer, Heidelberg, March 2009.

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 2010*, pages 131–140. ACM Press, October 2010.

BY96. Mihir Bellare and Moti Yung. Certifying permutations: Noninteractive zero-knowledge based on any trapdoor permutation. *Journal of Cryptology*, 9(3):149–166, June 1996.

Can01. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001.

CD17. Ignacio Cascudo and Bernardo David. SCRAPE: Scalable randomness attested by public entities. In Dieter Gollmann, Atsuko Miyaji, and Hiroaki Kikuchi, editors, *ACNS 17*, volume 10355 of *LNCS*, pages 537–556. Springer, Heidelberg, July 2017.

CD20. Ignacio Cascudo and Bernardo David. ALBATROSS: Publicly AttestabLe BATched Randomness based On Secret Sharing. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part III*, volume 12493 of *LNCS*, pages 311–341. Springer, Heidelberg, December 2020.

CDG+18. Jan Camenisch, Manu Drijvers, Tommaso Gagliardoni, Anja Lehmann, and Gregory Neven. The wonderful world of global random oracles. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 280–312. Springer, Heidelberg, April / May 2018.

CK18. Henry Corrigan-Gibbs and Dmitry Kogan. The discrete-logarithm problem with preprocessing. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 415–447. Springer, Heidelberg, April / May 2018.

CL07. Melissa Chase and Anna Lysyanskaya. Simulatable VRFs with applications to multi-theorem NIZK. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 303–322. Springer, Heidelberg, August 2007.

CM16. Jing Chen and Silvio Micali. Algorand. *arXiv preprint arXiv:1607.01341*, 2016.

DGKR17. Bernardo David, Peter Gaži, Aggelos Kiayias, and Alexander Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake protocol. Cryptology ePrint Archive, Report 2017/573, 2017. https://eprint.iacr.org/2017/573.

EKS+21. Muhammed F. Esgin, Veronika Kuchta, Amin Sakzad, Ron Steinfeld, Zhenfei Zhang, Shifeng Sun, and Shumo Chu. Practical post-quantum few-time verifiable random function with applications to algorand. In Nikita Borisov and Claudia Díaz, editors, *FC 2021, Part II*, volume 12675 of *LNCS*, pages 560–578. Springer, Heidelberg, March 2021.

ESLR22. Muhammed F. Esgin, Ron Steinfeld, Dongxi Liu, and Sushmita Ruj. Efficient hybrid exact/relaxed lattice proofs and applications to rounding and VRFs. Cryptology ePrint Archive, Report 2022/141, 2022. https://eprint.iacr.org/2022/141.

FKL18. Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Heidelberg, August 2018.

FLS90. Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In *31st FOCS*, pages 308–317. IEEE Computer Society Press, October 1990.

Fuc14. Georg Fuchsbauer. Constrained verifiable random functions. In Michel Abdalla and Roberto De Prisco, editors, *SCN 14*, volume 8642 of *LNCS*, pages 95–114. Springer, Heidelberg, September 2014.

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

GLOW21. David Galindo, Jia Liu, Mihair Ordean, and Jin-Mann Wong. Fully distributed verifiable random functions and their application to decentralised random beacons. pages 88–102, 2021.

GNP+14. Sharon Goldberg, Moni Naor, Dimitrios Papadopoulos, Leonid Reyzin, Sachin Vasant, and Asaf Ziv. NSEC5: Provably preventing DNSSEC zone enumeration. Cryptology ePrint Archive, Report 2014/582, 2014. https://eprint.iacr.org/2014/582.

GRPV23. Sharon Goldberg, Leonid Reyzin, Dimitrios Papadopoulos, and Jan Včelák. Verifiable Random Functions (VRFs). RFC 9381, August 2023.

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.

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 / June 2010.

Koh19. Lisa Kohl. Hunting and gathering - verifiable random functions from standard assumptions with short proofs. In Dongdai Lin and Kazue Sako, editors, *PKC 2019, Part II*, volume 11443 of *LNCS*, pages 408–437. Springer, Heidelberg, April 2019.

KR98. Hugo Krawczyk and Tal Rabin. Chameleon hashing and signatures. Cryptology ePrint Archive, Report 1998/010, 1998. https://eprint.iacr.org/1998/010.

KRDO17. Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 357–388. Springer, Heidelberg, August 2017.

Lai23. Yi-Fu Lai. CAPYBARA and TSUBAKI: Verifiable random functions from group actions and isogenies. Cryptology ePrint Archive, Report 2023/182, 2023. `https://eprint.iacr.org/2023/182`.

LBM20. Bei Liang, Gustavo Banegas, and Aikaterini Mitrokotsa. Statically aggregate verifiable random functions and application to e-lottery. *Cryptography*, 4(4):37, 2020.

LR88. Michael Luby and Charles Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM Journal on Computing*, 17(2):373–386, 1988.

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.

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.

PWH+17. Dimitrios Papadopoulos, Duane Wessels, Shumon Huque, Moni Naor, Jan Včelák, Leonid Reyzin, and Sharon Goldberg. Making NSEC5 practical for DNSSEC. Cryptology ePrint Archive, Report 2017/099, 2017. `https://eprint.iacr.org/2017/099`.

Sho94. Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th FOCS*, pages 124–134. IEEE Computer Society Press, November 1994.

A Examples

A.1 Separating weak unbiasability from unbiasability

In this section we show that weak-unbiasability (see Section 3.2) does not imply unbiasability, even when the latter notion is restricted to the special case $n = 1$ and $m = 1$, that is, the adversary can only chose one verification key and has to evaluate the VRF on only one point.

Weak Unbiasability $\not\Rightarrow$ Unbiasability. Let $(\mathsf{VRF.Gen}, \mathsf{VRF.Eval}, \mathsf{VRF.Eval})$ be a weakly-unbiasable VRF. Then we can produce a new VRF with a pair of weak keys. Let vk in the range $\mathsf{VRF.Gen}$ and vk^* a bit-string not in the range of $\mathsf{VRF.Gen}$. Then we can modify th verification algorithm to treat vk^* as vk. I.e. for any x, y, π input, output and proof, $\mathsf{VRF.Vfy}(\mathsf{vk}^*, x, y, \pi) \coloneqq \mathsf{VRF.Vfy}(\mathsf{vk}, x, y, \pi)$.

This VRF clearly remains a secure VRF, with pseudorandomness being unaffected as vk^* is never sampled. Moreover the new scheme is weakly-unbiasable since if there exists an adversary winning the weak-unbiasability which initially

chooses vk^*, we can compile it into one initially selecting vk and then returning the same output and proof. However the scheme is not unbiasable as an adversary $\mathcal{A}$ can choose initially $\mathsf{vk}, \mathsf{vk}^*$ and the predicate $p(y_1, y_2) : y_1 = y_2$, with p being 0 if any of its entries is $\perp$. On input x it then evaluates the VRF on x for both keys and return the correct evaluations y_1, y_2 and proof π_1, π_2. By construction $y_1 = y_2$, meaning that $p(y_1, y_2) = 1$ with probability 1 whereas with random inputs z_1, z_2 this is the case with probability $|\mathcal{Y}|^{-1}$. We then conclude $\mathcal{A}$ wins with significant advantage $1 - |\mathcal{Y}|^{-1} \geq 1/2$.

Weak Unbiasability $\not\Rightarrow$ Unbiasability with $n = m = 1$. Informally the idea is to observe that two "parallel executions" of a weakly-unbiasable VRF preserve its security properties. More precisely, given a weakly unbiasable VRF ($\mathsf{VRF}^*.\mathsf{Gen}$, $\mathsf{VRF}^*.\mathsf{Eval}, \mathsf{VRF}^*.\mathsf{Vfy}$), we define a new scheme in Figure 16.

$\mathsf{VRF}.\mathsf{Gen}(1^\lambda)$

1 : Sample $\mathsf{vk}_1, \mathsf{sk}_1 \leftarrow^{\$} \mathsf{VRF}^*.\mathsf{Gen}(1^\lambda)$ and $\mathsf{vk}_2, \mathsf{sk}_2 \leftarrow^{\$} \mathsf{VRF}^*.\mathsf{Gen}(1^\lambda)$
2 : Set $\mathsf{vk} \leftarrow (\mathsf{vk}_1, \mathsf{vk}_2)$ and $\mathsf{sk} \leftarrow (\mathsf{sk}_1, \mathsf{sk}_2)$
3 : **Return** $(\mathsf{vk}, \mathsf{sk})$

$\mathsf{VRF}.\mathsf{Eval}(\mathsf{sk}, x)$

1 : $(y_1, \pi_1) \leftarrow \mathsf{VRF}^*.\mathsf{Eval}(\mathsf{sk}_1, x)$
2 : $(y_2, \pi_2) \leftarrow \mathsf{VRF}^*.\mathsf{Eval}(\mathsf{sk}_2, x)$
3 : $y \leftarrow (y_1, y_2)$
4 : $\pi \leftarrow (\pi_1, \pi_2)$
5 : **Return** (y, π)

$\mathsf{VRF}.\mathsf{Vfy}(\mathsf{vk}, x, y, \pi)$

1 : Parse $y = (y_1, y_2)$
2 : Parse $\pi = (\pi_1, \pi_2)$
3 : **Return** 1 iff:
4 : $\quad \mathsf{VRF}^*.\mathsf{Vfy}(\mathsf{vk}_1, x, y_1, \pi_1) \to 1$
5 : $\quad \mathsf{VRF}^*.\mathsf{Vfy}(\mathsf{vk}_2, x, y_2, \pi_2) \to 1$

Fig. 16. Example of weakly-unbiasable but not unbiasable VRF (for $n = 1$, $m = 1$).

The new VRF clearly satisfies correctness and unique provability. Pseudorandomness can be proven through a standard hybrid game. Finally, if the output space the original VRF has exponential size in the security parameter, we can show this to be still weakly unbiasable. Indeed, given an adversary $\mathcal{A}$ against this property we can describe one $\mathcal{B}$ breaking weak-unbiasability for the underlying VRF: If $\mathcal{A}$ initially chooses $(\mathsf{vk}_1, \mathsf{vk}_2)$ and outputs (y_1^*, y_2^*), then $\mathcal{B}$ returns vk_1 and y_1^*. On input x it forwards the value to $\mathcal{A}$, which will compute (y_1, y_2) and (π_1, π_2). $\mathcal{B}$ thus returns y_1, π_1 and halts. Since $\Pr[y_1 = y_1^*,\ y_2 = y_2^*]$ is smaller than $\Pr[y_1 = y_1^*]$, we conclude that

$$\mathsf{Adv}(\mathcal{A}) \ \leq \ \mathsf{Adv}(\mathcal{B}) - \frac{1}{|\mathcal{Y}|} + \frac{1}{|\mathcal{Y}^2|}.$$

However this VRF is not unbiasable for $n = 1, m = 1$. Indeed, choosing $\mathsf{vk} = (\mathsf{vk}_1, \mathsf{vk}_1)$, that is a single key, implies that the predicate $p(y_1||y_2) : y_1 = y_2$ is satisfied by the VRF's output $y = y_1||y_2$ on any point x. However with a random value z this occurs only with negligible probability.

A.2 When weak unbiasability implies unbiasability

We study in this section under which conditions weak unbiasability implies our stronger notion. From Section A.1 we know that for $n = m = 1$, i.e. when our game is executed with only one VRF key and one input, *and the output space has super-polynomial size*, this is not the case. We now show however that the two notions are equivalent when $n = m = 1$ and the output space is polynomially bounded. For the sake of notation, we say (only in this section) that a VRF is (n, m)-unbiasable if for any ppt adversary

$$\mathsf{Adv}(A) \;=\; \Pr\left[\mathsf{Exp}^{\mathsf{unb}}_{0,n,m}(\mathcal{A}) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{unb}}_{1,n,m}(\mathcal{A}) = 1\right]$$

with the experiment being as defined in Section 6.

Proposition 3. *Let* $(\mathsf{VRF.Gen}, \mathsf{VRF.Eval}, \mathsf{VRF.Vfy})$ *be a VRF with weak unbiasability and polynomially bounded output space, i.e.* $|\mathcal{Y}| \leq \mathsf{poly}(\lambda)$*. Then said VRF is also* $(1, 1)$*-unbiasable.*

Proof. Let $\mathcal{A}$ be an adversary for $(1, 1)$-unbiasability. Then we describe an adversary $\mathcal{B}$ for weak unbiasability. Initially $\mathcal{B}$ executes $\mathcal{A}$ and collets its output vk and p, with p being a monotone predicate. $\mathcal{B}$ then computes the set $\mathcal{S}$ of element $y \in \mathcal{Y}$ such that $p(y) = 1$, which can be efficiently done as p is PPT and $\mathcal{Y}$ has polynomially bounded size. Given $\mathcal{S}$, it uniformly sampled $v \leftarrow^{\$} \mathcal{S}$ and returns (vk, v) to its challenger. Upon receiving x, it forwards x to $\mathcal{A}$ which eventually returns (y, π). $\mathcal{B}$ thus conclude returning (y, π).

B Postponed Proof

B.1 VRF from special VUF in the ROM

Proof of Theorem 2. Correctness, Unique Provability and Pseudorandomness of this transform are easily proved and well known to hold in the ROM. We therefore only focus on unbiasability.

Unbiasability. Let $\mathcal{A}$ be a PPT adversary executed in the unbiasability experiment. We will call Q the set of RO queries it performed before returning the keys $\mathsf{vk}_1, \ldots, \mathsf{vk}_n$ and predicate p. Furthermore let $\mathbf{y} = (y_{1,1}, \ldots, y_{n,m})$ be the output of $\mathcal{A}$ filtered by $\mathsf{Exp}^{\mathsf{bias}}_{b,n,m}$, i.e. where values with incorrect proofs are replaced with $\perp$. We will call $y^*_{i,j}$ the VUF that appears in the proof $\pi_{i,j}$ for $y_{i,j}$. In other words, if $y_{i,j} \neq \perp$ then $\pi = (y^*_{i,j}, \pi^*_{i,j})$ and

$$\mathsf{VUF.Vfy}(\mathsf{vk}_i, x_j, y^*_{i,j}, \pi^*_{i,j}) \to 1 \qquad \mathsf{H}(y^*_{i,j}) = y_{i,j}.$$

To proceed we define the event Seen, occurring when some of the non-erased values $y^*_{i,j}$ appears in Q. Coll instead is the event of two non-erased values $y^*_{i,j}$, $y^*_{h,k}$ colliding. Formally

$$\mathsf{Seen}_{i,j} \ : \ y^*_{i,j} \in Q, \ y^*_{i,j} \neq \perp \qquad \mathsf{Coll}_{i,j,h,k} \ : \perp \neq y^*_{i,j} = y^*_{h,k} \neq \perp, \ (i,j) \neq (h,k)$$
$$\mathsf{Seen} \ : \ \exists i,j : \mathsf{Seen}_{i,j} \qquad \mathsf{Coll} \ : \ \exists i,j,h,k : \mathsf{Coll}_{i,j,h,k}.$$

Next we claim that both Seen and Coll only occurs with negligible probability.

Claim 1 *There exists a negligible ε_1 such that* $\Pr[\mathsf{Seen}] \leq \varepsilon_1(\lambda)$.

Claim 2 *There exists a negligible ε_2 such that* $\Pr[\mathsf{Coll}] \leq \varepsilon_2(\lambda)$.

Given the above claims, we conclude as in the proof of Theorem 1. We let $\mathbf{z}$ be such that $z_{i,j} = y_{i,j}$ if $y_{i,j} \neq \perp$ and otherwise set $z_{i,j} \sim U(\mathcal{Y})$. By the monotonicity of p, and using the partial order introduced in Definition 7

$$\mathbf{y} \ \leq \ \mathbf{z} \quad \Rightarrow \quad \Pr[p(\mathbf{y}) = 1] \ \leq \ \Pr[p(\mathbf{z}) = 1].$$

Finally, conditioning on $\neg\mathsf{Seen}$ and $\neg\mathsf{Coll}$, all entries of $\mathbf{z}$ are uniformly sampled, as the value $y^*_{i,j}$ was never queries by the adversary when generating vk, and independent as all non-erased values $y^*_{i,j}$ are distinct. Letting $\mathbf{u}$ be the uniform distribution, we then can bound its statistical distance from $\mathbf{z}$ as

$$\Delta(\mathbf{z},\mathbf{u}) \ \leq \ \Pr[\mathsf{Seen}] + \Pr[\mathsf{Coll}] + \Delta(\mathbf{z}_{|\neg\mathsf{Seen},\neg\mathsf{Coll}}, \mathbf{u}_{|\neg\mathsf{Seen},\neg\mathsf{Coll}}) \ \leq \ \varepsilon_1 + \varepsilon_2.$$

We can then conclude that

$$\begin{aligned} & \Pr[p(\mathbf{y}) = 1] \ \leq \ \Pr[p(\mathbf{z}) = 1] \ \leq \ \Pr[p(\mathbf{u}) = 1] - \Pr[\mathsf{Seen}] - \Pr[\mathsf{Coll}] \\ \Rightarrow \quad & \mathsf{Adv}(\mathcal{A}) \leq \varepsilon_1(\lambda) + \varepsilon_2(\lambda). \end{aligned}$$

Proof of Claim 1. We describe an adversary $\mathcal{B}$ breaking the weak unbiasability of the underlying VUF.

Initially $\mathcal{B}$ executes $\mathcal{A}$, forwarding its queries Q to the RO until it returns $\mathsf{vk}_1, \ldots, \mathsf{vk}_n$ and a predicate p. It then sample a random entry $v^* = Q[k^*]$, with $k^* \in [|Q|]$, two indices i^*, j^* and return to its challenger (vk_{i^*}, v^*). Next, upon receiving a random input x, it samples $x_1, \ldots, x_m$ uniformly from the VUF input space and set $x_{j^*} = x$. It then run $\mathcal{A}$ on input $(x_1, \ldots, x_m)$, collecting its output $\mathbf{y}, \pi$. Finally, it parse $\pi_{i^*,j^*} = (y^*_{i^*,j^*}, \pi^*_{i^*,j^*})$ and return this couple to its challenger.

Note first of all that $\mathcal{B}$ perfectly simulates $\mathsf{Exp}^{\mathsf{bias}}_{b,n,m}$ and that $\mathcal{A}$ has no information on v^* and i^*, j^* since x is uniformly sampled too. Finally, if Seen_{i^*,j^*} occurs, $\mathcal{B}$ wins with probability greater than $|Q|^{-1}$, since at least one of the

entries in Q equals $y^*_{i^*,j^*}$. We then conclude that

$$\begin{aligned}
\mathsf{Adv}(\mathcal{B}) &\geq \frac{1}{|Q|} \cdot \Pr\left[\mathsf{Seen}_{i^*,j^*}\right] \\
&\geq \frac{1}{|Q|} \cdot \sum\nolimits_{i=1}^{n} \sum\nolimits_{j=1}^{m} \Pr\left[i = i^*,\ j = j^*\right] \Pr\left[\mathsf{Seen}_{i,j}\right] \\
&\geq \frac{1}{|Q|nm} \cdot \sum\nolimits_{i=1}^{n} \sum\nolimits_{j=1}^{m} \Pr\left[\mathsf{Seen}_{i,j}\right] \\
&\geq \frac{1}{|Q|nm} \cdot \Pr\left[\mathsf{Seen}\right]
\end{aligned}$$

and in particular $\Pr\left[\mathsf{Seen}\right] \leq |Q|nm \cdot \mathsf{Adv}(\mathcal{B})$ that is negligible.

Proof of Claim 2. We split the event Coll into the disjunction of two events Coll^0 and Coll^1 defined as

$$\mathsf{Coll}^0 \ : \ \exists i,j,k,h : (\mathsf{Coll}_{i,j,k,h},\ j \neq h) \qquad \mathsf{Coll}^1 \ : \ \exists i,j,k : \mathsf{Coll}_{i,j,k,j}$$

In other words, Coll^0 happens when the collision occurs for different inputs x_j, x_h, whereas Coll^1 only when the collision occurs for the same x_j.

The bound on Coll^1 comes directly from our second hypothesis on the VUF: namely that for any two keys $\mathsf{vk}_i, \mathsf{vk}_k$ and a point x_j the probability of $y^*_{i,j} = y^*_{k,j}$ is smaller than ε. Using a union bound

$$\Pr\left[\mathsf{Coll}^1\right] \ \leq \ \sum\nolimits_{i,j,k} \Pr\left[\mathsf{Coll}_{i,j,k,j}\right] \ \leq \ n^2 m \cdot \varepsilon(\lambda).$$

Regarding Coll^0, we provide an adversary $\mathcal{B}$ breaking the weak unbiasability of the underlying VUF if Coll^0 is not negligible.

Initially $\mathcal{B}$ executes $\mathcal{A}$ to get $\mathsf{vk}_1, \ldots, \mathsf{vk}_n$ and the predicate p, which is discarded. Then it randomly chooses $i', k' \leftarrow^{\$} [n]$ and $j', h' \leftarrow^{\$} [m]$ with $j' \neq h'$, and samples a point $x_{h'}$. At this point $\mathcal{B}$ wishes to return as a guess for the output of $\mathsf{vk}_{i'}$ on a random input, the value $y^*_{k',h'}$ obtained evaluating the VRF with key $\mathsf{vk}_{k'}$ on input $x_{h'}$. However, $\mathcal{B}$ does not have access to the VRF secret key. So, to achieve this goal it first continues the execution of $\mathcal{A}$ giving it $x_1, \ldots, x_m$ randomly sampled, and extract in this way the value $y^*_{k',h'}$. It then return to its challenger $\mathsf{vk}_{i'}$ and $y^*_{k',h'}$ and waits for it to respond with a point x. Now, in order to extract the VUF output on x, $\mathcal{B}$ sampled again random point $x_1, \ldots, x_m$ up to setting $x_{j'} = x$ and $x_{h'} = x$. Finally, it rewinds $\mathcal{A}$ to the state in which it was before receiving the input points, and re-executes it with $x_1, \ldots, x_m$, getting the value $y^*_{i',j'}, \pi^*_{i',j'}$. At this point, if $y^*_{k',h'}$ obtained in the first execution has an incorrect proof, but the one obtained in the second one does not, it abort. Otherwise it returns $(y^*_{i',j'}, \pi^*_{i',j'})$.

Let Abort be the event that $\mathcal{B}$ aborts. We immediately observe that this event occurs with probability smaller than $1/2$. Indeed, letting p denote the probability that $y^*_{k',h'}$ has an incorrect proof when $\mathcal{A}$ receives uniformly sampled points $\bar{x}_1, \ldots, \bar{x}_m$ conditioned on $\bar{x}_{h'} = x_{h'}$, then

$$\Pr\left[\mathsf{Aboort}\right] \ = \ p(p-1) \ \leq \ \frac{1}{2}.$$

Finally, conditioning on $\neg\mathsf{Abort}$, we observe that $\mathsf{Coll}_{i',j',k',h'}$ implies that $\mathcal{B}$ correctly guesses its output. This is the case as $y^*_{k',h'}$ is returned with a correct proof in the second execution of $\mathcal{A}$ which implies, having conditioned on $\neg\mathsf{Abort}$ than the same occurs in the first execution. From the VUF unique provability, these values must then be the same, and in particular, $y^*_{k',h'} = y^*_{i',j'}$. We therefore conclude that

$$\mathsf{Adv}(\mathcal{B}) \;\geq\; \Pr\left[\neg\mathsf{Abort}\right] \cdot \Pr\left[\mathsf{Coll}_{i',j',k',h'}\right] \;\geq\; \frac{1}{2} \cdot \frac{1}{n^2 m^2} \cdot \Pr\left[\mathsf{Coll}^1\right].$$

This concludes the proof of this claim as

$$\Pr\left[\mathsf{Coll}\right] \leq n^2 m \varepsilon(\lambda) + 2n^2 m^2 \cdot \mathsf{Adv}(\mathcal{B}).$$

B.2 Padded VRF Construction

Proof of Theorem 3. Completeness and Unique Provability follows immediately from the underlying VRF. We thus focus on restricted pseudorandomness and unconditional unbiasability on independent points, i.e. against computationally unbounded adversaries.

Pseudorandomness. Given an adversary $\mathcal{A}$ breaking restricted pseudorandomness we construct $\mathcal{B}$ attacking the pseudorandomness of the underlying VRF.

On input vk, $\mathcal{B}$ executes $\mathcal{A}(\mathsf{vk})$. Whenever $\mathcal{A}$ queries an evaluation on $x = (u, v)$, $\mathcal{B}$ compute $w \leftarrow \mathsf{VRF.Eval}(u)$ and returns $y = w + v$. When $\mathcal{A}$ sends a challenge point $x^*(u^*, v^*)$, then $\mathcal{B}$ returns u^* to its challenger and waits for a reply w^*. Eventually it replies to $\mathcal{A}$ with $y^* \leftarrow w^* + v^*$. If at any point $\mathcal{A}$ queried $x = (u, v)$ with $u = u^*$, abort the simulation and return 0. When $\mathcal{A}$ halts and returns a bit, $\mathcal{B}$ outputs the same bit.

Due to our definition of restricted pseudorandomness, $\mathcal{A}$ never queries x with a first component equal to u^*. Thus the evaluation queries $\mathcal{B}$ sends to $\mathsf{VRF.Eval}$ are all different than u^* and thus they all receive a correct reply. In particular $\mathcal{B}$ perfectly replies to $\mathcal{A}$'s evaluation queries. Next, we call b the challenge bit chosen by $\mathcal{B}$'s challenger. When $b = 1$, $w^* = \mathsf{VRF}^*.\mathsf{Eval}(\mathsf{sk}, u^*)$ and in particular, y^* is computed with the real VRF algorithm $\mathsf{VRF.Eval}(\mathsf{sk}, (u, v))$. Conversely, if $b = 0$, $w^* \sim U(\mathbb{G})$ with $\mathbb{G} = \mathcal{Y}^*$ being the output space of the inner VRF. Since w^* is independent from v^*, we have that $y^* = w^* + v^* \sim U(\mathbb{G})$. We can then conclude that $\mathsf{Adv}(\mathcal{B}) = \mathsf{Adv}(\mathcal{A})$ where the former is negligible assuming the inner VRF is pseudorandom.

Unbiasability on Independent Points. We show the argument information-theoretically. For any vk let $F_{\mathsf{vk}} : \mathcal{X}^* \to \mathcal{Y}^* \cup \{\perp\}$ be a function such that

$$F_{\mathsf{vk}}(x) = \begin{cases} \perp & \text{If } \nexists y, \pi \,:\, \mathsf{VRF}^*.\mathsf{Vfy}(\mathsf{vk}, x, y, \pi) \to 1 \\ y & \text{If } \exists \pi \,:\, \mathsf{VRF}^*.\mathsf{Vfy}(\mathsf{vk}, x, y, \pi) \to 1 \end{cases}$$

This is a function due to unique provability of the inner VRF. Next define $G_{\mathsf{vk}}(x)$ so that if $F_{\mathsf{vk}}(x) \neq \perp$, then $G_{\mathsf{vk}}(x) = F_{\mathsf{vk}}(x)$, and otherwise $G_{\mathsf{vk}}(x) = 0$. Finally, to emulate the construction in Figure 11, we let $H_{\mathsf{vk}}(u, v) = G_{\mathsf{vk}}(u) + v$.

We now prove the property. Let $\mathcal{A}$ be an unbounded adversary against unbiasability on independent points, $\mathsf{vk}_1, \ldots, \mathsf{vk}_n$ the verification keys initially produced and p the monotone predicate it chooses. We call $x_{i,j} = (u_{i,j}, v_{i,j})$ the points sampled by the challenger and $\mathbf{y}$ the *filtered output* of $\mathcal{A}$ computed at the end of the experiment in Figure 7. For all i, j if $y_{i,j} \neq \perp$ it means that $\mathcal{A}$ produces a proof $\pi_{i,j}$ for $y_{i,j}$ valid for vk_i. In particular $\pi_{i,j}$ is by construction a proof for the inner VRF with verification key vk_i, input $u_{i,j}$ and output $w_{i,j} = y_{i,j} - v_{i,j}$. Thus by our definitions

$$\begin{aligned} w_{i,j} = F_{\mathsf{vk}_i}(u_{i,j}) = G_{\mathsf{vk}_i}(u_{i,j}) \;\Rightarrow \\ \Rightarrow \;\; y_{i,j} = w_{i,j} + v_{i,j} = G_{\mathsf{vk}_i}(u_{i,j}) + v_{i,j} = H_{\mathsf{vk}_i}(u_{i,j}, v_{i,j}). \end{aligned}$$

Hence $y_{i,j} \leq H_{\mathsf{vk}_i}(x_{i,j})$ according to the partial order in Definition 7.

Finally we observe that the evaluations of H follow the uniform distribution. Indeed the elements $v_{i,j}$ are then uniformly random, meaning that also $G_{\mathsf{vk}_i}(u_{i,j}) + v_{i,j}$ are, as $u_{i,j}$ and $v_{i,j}$ are independent. Calling $\mathbf{h}$ the vector of values $H_{\mathsf{vk}_i}(x_{i,j})$ for $i \in [n]$ and $j \in [m]$ we conclude that

$$\mathbf{y} \;\leq\; \mathbf{h} \quad \Rightarrow \quad \Pr[p(\mathbf{y}) = 1] \;\leq\; \Pr[p(\mathbf{h}) = 1] \quad \Rightarrow \quad \mathsf{Adv}(\mathcal{A}) \;\leq\; 0.$$

B.3 Unbiasability of VRB

Proof of Theorem 4. For any vk let F_{vk} be a function mapping x to y if a valid proof π for y exists, i.e. such that $\mathsf{VRF.Vfy}(\mathsf{vk}, x, y, \pi) \to 1$, or else mapping x to $\perp$ if no such y exists. Due to unique provability and injectivity this is an injective functions restricted in the set $\mathcal{S}_{\mathsf{vk}} = \{x \in \mathcal{X} \,:\, F_{\mathsf{vk}}(x) \neq \perp\}$. As $|\mathcal{X}| = |\mathcal{Y}|$ we can extend each F_{vk} to a bijection $G_{\mathsf{vk}} : \mathcal{X} \to \mathcal{Y}$ which agrees on F_{vk} over $\mathcal{S}_{\mathsf{vk}}$.

We now prove the Theorem. For any $\mathcal{A}$ computationally unbounded and monotone predicate p of its choice let $\mathsf{vk}_1, \ldots, \mathsf{vk}_n$ be the verification keys it initially returns, $x_{1,1}, \ldots, x_{n,m}$ the points chosen by the challenger and $y_{1,1}, \ldots, y_{n,m}$ be the filtered output computed by $\mathsf{Exp}^{\mathsf{ip-bias}}_{b,n,m}(\mathcal{A})$. By the way the experiment is defined, $y_{i,j} \neq \perp$ only if $\mathcal{A}$ provides a valid proof for this values, which implies $y_{i,j} = F_{\mathsf{vk}_i}(x_{i,j}) = G_{\mathsf{vk}_i}(x_{i,j})$. Hence

$$(y_{1,1}, \ldots, y_{n,m}) \leq (G_{\mathsf{vk}_1}(x_{1,1}), \ldots, G_{\mathsf{vk}_n}(x_{n,m})) \coloneqq (z_{1,1}, \ldots, z_{n,m}).$$

We finally observe that $\mathbf{z} \sim U(\mathcal{Y}^{n,m})$ because G_{vk_i} are all permutations and $x_{i,1}, \ldots, x_{i,m} \sim U(\mathcal{X}^{n,m})$. In conclusion, using the monotonicity of p

$$\mathbf{y} \leq \mathbf{z} \quad \Rightarrow \quad \Pr[p(\mathbf{y}) = 1] \;\leq\; \Pr[p(\mathbf{z}) = 1] \quad \Rightarrow \quad \mathsf{Adv}(\mathcal{A}) \leq 0$$

B.4 2-Feistel Rounds Construction

Proof of Theorem 5. Correctness follows immediately from the VRF's correctness. We then proceed to prove unique provability, injectivity and restricted pseudorandomness.

Unique Provability. Let $\pi, \bar{\pi}$ be two valid proofs respectively for y and $\bar{y}$ on input x and verification key vk. We call (u_i, z_i, w_i) and $(\bar{u}_i, \bar{z}_i, \bar{w}_i)$ the intermediate values computed while verifying $\pi, \bar{\pi}$ respectively as described in Figure 12. As the input is the same in both schemes $z_1 = \bar{z}_1$ and $w_1 = \bar{w}_1$. By unique provability of the underlying VRF, $z_1 = \bar{z}_1$ implies $u_1 = \bar{u}_1$. By construction we then have $z_2 = \bar{z}_2$ and $w_2 = \bar{w}_2$. Again using the unique provability of the underlying scheme, $w_2 = \bar{w}_2$ implies $u_2 = \bar{u}_2$ and in particular $z_3 = \bar{z}_3$ and $w_3 = \bar{w}_3$. Finally, the last check in the verification procedure ensures that $y = (z_3, w_3) = (\bar{z}_3, \bar{w}_3) = \bar{y}$ concluding the proof.

Injectivity. Let $\pi, \bar{\pi}$ be two valid proofs for y on input $x, \bar{x}$ respectively and verification key vk. As before let (u_i, z_i, w_i) and $(\bar{u}_i, \bar{z}_i, \bar{w}_i)$ the intermediate values computed in the verification procedure as described in Figure 12. Then, by the check in line 8, as y is the same in both verifications, $z_3 = \bar{z}_3$ and $w_3 = \bar{w}_3$. By construction this further implies $w_2 = \bar{w}_2$ which in turns, by unique probability, that $u_2 = \bar{u}_2$, and in particular $z_2 = \bar{z}_2$. Applying the same argument for the first Feistel round we get $z_1 = \bar{z}_1$ and $w_1 = \bar{w}_1$ which implies that $x = \bar{x}$.

Restricted Pseudorandomness. We prove restricted pseudorandomness on the first component through a sequence of hybrid games

- G_1: The restricted pseudorandomness game with challenger's bit $b = 1$.
- G_2: As G_1 but on challenge input x^*, reply with $y^* \leftarrow \mathsf{VRB.Eval}_2(\mathsf{sk}, x^*)$ described in Figure 17.
- G_3: As G_2 but if at any point, an evaluation query produces $w_2 = w_2^*$, return 0.
- G_4: As G_3 but on challenge input x^*, reply with $y^* \leftarrow \mathsf{VRB.Eval}_4(\mathsf{sk}, x^*)$ described in Figure 17.
- G_5: As G_4 but no abort occurs if $w_2 = w_2^*$.
- G_6: The restricted pseudorandomness game with challenger's bit $b = 0$.

At a high level we will reduce $\mathsf{G}_1 \approx \mathsf{G}_2$ the pseudorandomness of the VRF with the first key, $\mathsf{G}_2 \approx \mathsf{G}_3$ by reducing the bad event $w_2 = w_2^*$ to the VRF's pseudorandomness with the first key. $\mathsf{G}_3 \approx \mathsf{G}_4$ follow by pseudorandomness of the VRF in the second key and $\mathsf{G}_4 \approx \mathsf{G}_5$ is proved as $\mathsf{G}_2 \approx \mathsf{G}_3$, while $\mathsf{G}_5 \equiv \mathsf{G}_6$ is proven information theoretically.

Proof of $\mathsf{G}_1 \approx \mathsf{G}_2$. Given a distinguisher $\mathcal{D}$, we describe an adversary $\mathcal{B}$ breaking the pseudorandomness of the VRF scheme. On input vk^*, $\mathcal{B}$ sets $\mathsf{vk}_1 = \mathsf{vk}^*$, generates the second key $\mathsf{vk}_2, \mathsf{sk}_2 \leftarrow^{\$} \mathsf{VRF.Gen}(1^\lambda)$. Then it sets $z_1^* = \perp$ and run $\mathcal{D}(\mathsf{vk})$ with $\mathsf{vk} = (\mathsf{vk}_1, \mathsf{vk}_2)$. When $\mathcal{D}$ queries an evaluation in $x = (z_1, w_1)$, $\mathcal{B}$ first checks that $z_1 \neq z_1^*$. If this is the case computes (y, π) as prescribed, replacing the VRF evaluation in line 2 with an oracle query $u_1 \leftarrow \mathcal{O}_{\mathsf{eval}}(z_1)$. Analogously, when $\mathcal{D}$ return a challenge point $x^* = (z_1^*, w_1^*)$, first check that z_1^* was never queried before. If this is the case compute the challenge response y^* as

$\mathsf{VRB.Eval}_2(\mathsf{sk}, x)$

1 : Parse $x = (z_1, w_1)$
2 : Sample $u_1 \leftarrow^{\$} \mathbb{G}$
3 : $(z_2, w_2) \leftarrow (z_1, w_1 + u_1)$
4 : $(u_2, \pi_2) \leftarrow \mathsf{VRF.Eval}(\mathsf{sk}_2, w_2)$
5 : $(z_3, w_3) \leftarrow (z_2 + u_2, w_2)$
6 : $y \leftarrow (z_3, w_3)$
7 : **Return** $(y, \perp)$

$\mathsf{VRB.Eval}_4(\mathsf{sk}, x)$

1 : Parse $x = (z_1, w_1)$
2 : Sample $u_1 \leftarrow^{\$} \mathbb{G}$
3 : $(z_2, w_2) \leftarrow (z_1, w_1 + u_1)$
4 : Sample $u_2 \leftarrow^{\$} \mathbb{G}$
5 : $(z_3, w_3) \leftarrow (z_2 + u_2, w_2)$
6 : $y \leftarrow (z_3, w_3)$
7 : **Return** $(y, \perp)$

Fig. 17. Hybrid $\mathsf{VRB.Eval}$ for G_2, G_4. Changes are highlighted.

prescribed replacing again the VRF evaluation in line 2 by returning a challenge point z_1^* and using the challenger's reply u_1^*. Finally, when $\mathcal{D}$ returns a bit b', return the same bit.

We begin showing that $\mathcal{B}$ correctly replies to each query: Indeed if $\mathcal{D}$ returns a challenge x^* whose first component matches a previously queried x, the game fails, and if any subsequent queries (z_1, w_1) is such that $z_1 = z_1^*$, $\mathcal{B}$ returns $\perp$. Conversely, if $z_1 \neq z_1^*$ then $\mathcal{B}$ query to the oracle will return the $(u_1, \pi_1) = \mathsf{VRF.Eval}(\mathsf{sk}^*, z_1)$, meaning that $\mathcal{B}$ eventually returns a correct evaluation to $\mathcal{D}$.

Finally, calling b the bit chosen by $\mathcal{B}$'s challenger, if $b = 1$, then the challenge response y^* is computed as in G_1 with the correct evaluation algorithm. Conversely if $b = 0$, y^* is computed as in G_2. We thus conclude that $\mathsf{Adv}(\mathcal{B}) = \mathsf{Adv}(\mathcal{D})$.

Proof of $\mathsf{G}_2 \approx \mathsf{G}_3$. Given an adversary $\mathcal{A}$ executed in G_2 we denote $z_1^{(i)}, w_1^{(i)}, \ldots$ the intermediate values computed during the i-th evaluation query. Then let us define the event $\mathsf{Coll}_i : w_2^{(i)} = w_2^*$. Let t be a polynomial upper bound on the number of queries performed by A. Our goal will be to prove that there exists a negligible function bounding the probability of all these events for $i \in \{1, \ldots, t\}$. To this aim we define $\mathcal{C}_i$ which uses $\mathcal{A}$ to break the inner VRF pseudorandomness if Coll_i occurs with significant probability. Informally $\mathcal{C}_i$ uses the verification keys it receive from its challenger as vk_1 and generates $\mathsf{vk}_2, \mathsf{sk}_2$. Queries until the i-th are generate with $\mathcal{O}_{\mathsf{eval}}$ and the challenge's response y is computes sampling $u_1 \leftarrow^{\$} \mathbb{G}$ as prescribed in G_2. Finally, for the i-th query, it computes $u_1^{(i)}$ sending $z_1^{(i)}$ as a challenge point. If this value cases $w_2^{(i)}$ to collide with w_2^*, it guesses $u_1^{(i)}$ was not random. A full description of $\mathcal{B}$ is provided in Figure 18

To analyze $\mathcal{C}_i$ we call Before_i the event in which the i-th query from $\mathcal{A}$ in G_2 occurs before sending the challenge point, and After_i its complement. We further call b the challenge bit chosen by $\mathcal{C}_i$'s challenger. As $\mathcal{C}_i$ perfectly simulates G_2

Reduction $\mathcal{C}_i^{\mathcal{O}_{\mathsf{eval}}}(\mathsf{vk}^*)$

1 : // Public parameters generation
2 : Sample $i \leftarrow^{\$} \{1, \ldots, t\}$ and generate $\mathsf{vk}_2, \mathsf{sk}_2 \leftarrow^{\$} \mathsf{VRF.Gen}(1^\lambda)$
3 : Set $\mathsf{vk} \leftarrow (\mathsf{vk}^*, \mathsf{vk}_2)$, $z_1^* \leftarrow \perp$ and run $\mathcal{A}(\mathsf{vk})$
4 : // Answering the first $i-1$ evaluation queries
5 : **When** $\mathcal{A}$ queries the j-th time $x = (z_1, w_1)$ with $j < i$ and $z_1 \neq z_1^*$:
6 : $(u_1, \pi_1) \leftarrow \mathcal{O}_{\mathsf{eval}}(z_1)$
7 : $(z_2, w_2) \leftarrow (z_1, w_1 + u_1)$
8 : $(u_2, \pi_2) \leftarrow \mathsf{VRF.Eval}(\mathsf{sk}_2, w_2)$
9 : $(z_3, w_3) \leftarrow (z_2 + u_2, w_2)$
10 : Set $y \leftarrow (z_3, w_3)$ and $\pi \leftarrow (u_1, u_2, \pi_1, \pi_2)$
11 : Send $\mathcal{A} \leftarrow (y, \pi)$
12 : // Answering challenge request
13 : **When** $\mathcal{A}$ sends its challenge input $x^* = (z_1^*, w_1^*)$ with z_1^* not yet queried:
14 : Sample $u_1^* \leftarrow^{\$} \mathbb{G}$
15 : $(z_2^*, w_2^*) \leftarrow (z_1^*, w_1^* + u_1^*)$
16 : $(u_2^*, \pi_2^*) \leftarrow \mathsf{VRF.Eval}(\mathsf{sk}_2, w_2^*)$
17 : $(z_3^*, w_3^*) \leftarrow (z_2^* + u_2^*, w_2^*)$ and set $y \leftarrow (z_3^*, w_3^*)$
18 : Send $\mathcal{A} \leftarrow y$
19 : // Answering the i-th evaluation query
20 : **When** $\mathcal{A}$ queries the i-th time $x^{(i)} = (z_1^{(i)}, w_1^{(i)})$ with $z_1^{(i)} \neq z_1^*$:
21 : **If** x^* was not queried yet: **Return** 0
22 : Send the challenge $z_1^{(i)}$ and wait for the reply $u_1^{(i)}$
23 : $w_2^{(i)} \leftarrow w_1^{(i)} + u_1^{(i)}$
24 : **If** $w_2^{(i)} = w_2^*$: **Return** 1
25 : **Else**: **Return** 0

Fig. 18. $\mathcal{C}_i$ breaking VRF pseudorandomness if $\mathcal{A}$ finds a collision $w_2^{(i)} = w_2^*$.

until the i-th query is performed, we have that

$$\begin{aligned}\mathsf{Adv}(\mathcal{C}_i) &= |\Pr[\mathcal{C}_i \to 1 \mid b = 1] - \Pr[\mathcal{C}_i \to 1 \mid b = 0]| \\ &\geq \Pr[\mathcal{C}_i \to 1 \mid b = 1] - \Pr[\mathcal{C}_i \to 1 \mid b = 0] \\ &\geq \Pr[\mathsf{Coll}_i \wedge \mathsf{After}_i] - \frac{1}{|\mathbb{G}|}\end{aligned}$$

where the final inequality follows as, when $b = 0$, then $u_1^{(i)}$ is uniformly random and independent from $w_1^{(i)}$, w_2^*. Thus a collision in that case occurs only if $u_1^{(i)} = w_2^* - w_1^{(1)}$ which happens with probability $|\mathbb{G}|^{-1}$. By the security of the VRF, there exists a negligible ε_i such that $\mathsf{Adv}(\mathcal{C}_i) = \varepsilon_i$. We can then derive a bound on the probability of Coll_i:

$$\Pr[\mathsf{Coll}_i] = \Pr[\mathsf{Coll}_i \wedge \mathsf{After}_i] + \Pr[\mathsf{Coll}_i \wedge \mathsf{Before}_i] \leq \varepsilon_i + \frac{1}{|\mathbb{G}|} + \frac{1}{|\mathbb{G}|} = \varepsilon_i + \frac{2}{|\mathbb{G}|}$$

where the inequality also follows as $\Pr[\mathsf{Coll}_i \mid \mathsf{Before}_i] \leq |\mathbb{G}|^{-1}$, which is true because if $x^{(i)}$ is queried before x^*, then u_1^* is sampled uniformly and independently from w_1^* and $w_2^{(i)}$. As a collision only occurs if $u_1^* = w_2^{(i)} - w_1^*$, this only happens with probability $|\mathbb{G}|^{-1}$.

Finally we need to argue that the disjunction of all these events has negligible probability of occurring. To this aim, we observe that the the maximum running time of all $\mathcal{C}_i$ a polynomial factor away from the runtime of $\mathcal{A}$ (roughly coming from evaluating the queries). Thus we can define an adversary $\mathcal{C}(\mathsf{vk}^*)$ sampling a random $i \leftarrow^{\$} \{1, \ldots, t\}$ and the executing $\mathcal{C}_i(\mathsf{vk}^*)$. This adversary has advantage

$$\begin{aligned}\varepsilon(\lambda) \geq \mathsf{Adv}(\mathcal{C}) = \sum_{i=1}^{t} \frac{1}{t} \cdot \mathsf{Adv}(\mathcal{C}_i) = \frac{1}{t} \cdot \sum_{i=1}^{t} \varepsilon_i(\lambda) \quad \Rightarrow \\ \Rightarrow \quad \sum_{i=1}^{t} \varepsilon_i(\lambda) \leq t\varepsilon(\lambda).\end{aligned}$$

for a negligible ε. We finally conclude that with a union bound that calling $\mathsf{Coll} = \mathsf{Coll}_1 \wedge \ldots \wedge \mathsf{Coll}_t$, the probability $\Pr[\mathsf{Coll}] \leq \varepsilon(\lambda) + 2t|\mathbb{G}|^{-1}$. As $\mathsf{G}_2, \mathsf{G}_3$ are identical conditioning on $\neg\mathsf{Coll}$, the proof is complete.

Proof of $\mathsf{G}_3 \approx \mathsf{G}_4$. The proof is similar to the one for $\mathsf{G}_1 \approx \mathsf{G}_2$ with the exception that now the VRF instantiated by the challenger is used for the second key. More precisely, let $\mathcal{D}$ be a distinguisher for $\mathsf{G}_3, \mathsf{G}_4$. We build $\mathcal{B}$ breaking pseudorandomness for the underlying VRF. Initially $\mathcal{B}(\mathsf{vk}^*)$ set $\mathsf{vk}_2 \leftarrow \mathsf{vk}^*$ and samples $\mathsf{vk}_1, \mathsf{sk}_1 \leftarrow^{\$} \mathsf{VRF.Gen}(1^\lambda)$. Then it sets $z_1^* = \perp$, $\mathsf{vk} \leftarrow (\mathsf{vk}_1, \mathsf{vk}_2)$ and runs $\mathcal{D}(\mathsf{vk})$. Evaluation queries are computed as in the original scheme, computing the second VRF evaluation in line 4 with the evaluation oracle $\mathcal{O}_{\mathsf{eval}}(w_2)$. Similarly, the challenge's response is evaluated as in G_3, with $u_1^* \sim U(\mathbb{G})$, but with u_2^* is computed as the response to the challenge point w_2^*. Finally, if at any point w_2^* is equal to w_2 during an evaluation query, $\mathcal{B}$ aborts the computation. A full description of $\mathcal{B}$ is provided in Figure 19.

Reduction $\mathcal{B}^{\mathcal{O}_{\mathsf{eval}}}(\mathsf{vk}^*)$

// Public parameters generation
Sample $\mathsf{vk}_1, \mathsf{sk}_1 \leftarrow^{\$} \mathsf{VRF.Gen}(1^\lambda)$, set $\mathsf{vk} \leftarrow (\mathsf{vk}_1, \mathsf{vk}^*)$ and $z_1^* \leftarrow \perp$
Run $\mathcal{D}(\mathsf{vk})$
// Answering evaluation queries
When $\mathcal{D}$ queries an evaluation in $x = (z_1, w_1)$ with $z_1 \neq z_1^*$:
  $(u_1, \pi_1) \leftarrow \mathsf{VRF.Eval}(\mathsf{sk}_1, z_1)$
  $(z_2, w_2) \leftarrow (z_1, w_1 + u_1)$
  $(u_2, \pi_2) \leftarrow \mathcal{O}_{\mathsf{eval}}(w_2)$
  $(z_3, w_3) \leftarrow (z_2 + u_2, w_2)$
  Set $y \leftarrow (z_3, w_3)$ and $\pi \leftarrow (u_1, u_2, \pi_1, \pi_2)$
  Send $\mathcal{D} \leftarrow (y, \pi)$
// Answering challenge request
When $\mathcal{D}$ sends its challenge input x^*:
  $(z_1^*, w_1^*) \leftarrow [Ax^*]$
  Sample $u_1^* \leftarrow^{\$} \mathbb{G}$
  $(z_2^*, w_2^*) \leftarrow (z_1^*, w_2^* + u_2^*)$
  Send the challenge w_2^* and wait for the reply u_2^*
  $(z_3^*, w_3^*) \leftarrow (z_2^* + u_2^*, w_2^*)$ and call $y \leftarrow (z_3^*, w_3^*)$
  Send $\mathcal{D} \leftarrow y$
// Final Output
When $\mathcal{D} \to b'$: **Return** b'

Fig. 19. $\mathcal{B}$ reducing the indistinguishability of $\mathsf{G}_3, \mathsf{G}_4$ to the VRF pseudorandomness

Note that if any points $w_2 = w_2^*$, then at least one query of $\mathcal{B}$ is rejected by the challenger, and we implicitly assumes it to halt the simulation and return 0 in that case. Next we observe that as long as $w_2 \neq w_2^*$ for all w_2 obtained in the execution, then $\mathcal{B}$ responds correctly to all evaluation queries as $\mathsf{VRF.Eval}(w_2) = \mathsf{VRF.Eval}(\mathsf{sk}^*, w_2)$.

Next, call b the challenge bit tossed by $\mathcal{B}$'s challenger. If $b = 1$ then $\mathcal{B}$ perfectly simulates G_3 as to compute the response, u_2^* given by the challenger is the output of $\mathsf{VRF.Eval}(\mathsf{sk}^*, w_2^*)$. Conversely if $b = 0$ then $\mathcal{B}$ simulates G_4 perfectly as u_2^* is uniformly sampled. Thus $\mathsf{Adv}(\mathcal{B}) = \mathsf{Adv}(\mathcal{D})$ which concludes the proof.

Proof of $\mathsf{G}_4 \approx \mathsf{G}_5$. The proof is identical to $\mathsf{G}_2 \approx \mathsf{G}_3$ with the exception that u_2^* is now sampled randomly, which does not affect the argument.

Proof of $\mathsf{G}_5 \equiv \mathsf{G}_6$. We show this observing that in G_5 when the adversary return the challenge point $x^* = (z_1^*, z_2^*)$, the challenger computes y as

$$y \;=\; (z_3, w_3) \;=\; (z_2 + u_2, w_2) \;=\; (z_1 + u_2, w_1 + u_1)$$

for uniformly sampled $u_2, u_1 \sim U(\mathbb{G})$. The output is thus uniformly distributed as in G_6. This concludes the proof of restricted pseudorandomness.

B.5 VRB Compiler

Proof of Theorem 6. Completeness and unique provability follows trivially from the underlying VRB, whose only limitation is a weaker pseudorandomness notion.

Injectivity. Let $\pi, \bar{\pi}$ be two proof for inputs $\mathbf{x}, \bar{\mathbf{x}}$, output y on verification key $\mathsf{vk} = (\mathsf{vk}^*, [\mathbf{a}])$. Let us further denote z, w and $\bar{z}, \bar{w}$ the intermediate values computed by $\mathsf{VRB.Vfy}$ as described in Figure 14. By injectivity of the underlying weaker VRB, as y is the same in both proofs, $z = \bar{z}$ and $w = \bar{w}$. To conclude, because of the check on line 5, $[a_2] \neq 0$, which implies $a_2 \neq 0$. As a consequence the matrix $A = (\mathbf{a}, \mathbf{e}_1)^\top$ is invertible (its determinant equals a_2) and in particular

$$(z, w) \;=\; (\bar{z}, \bar{w}) \quad \Rightarrow \quad A\mathbf{x} = A\bar{\mathbf{x}} \quad \Rightarrow \quad \mathbf{x} = \bar{\mathbf{x}}.$$

Thus the outer VRB is injective.

Pseudorandomness. We proceed in two steps. First we show that for any adversary executed in the pseudorandomness experiment, querying two different inputs $\mathbf{x}, \bar{\mathbf{x}}$ which yields the same $z = \bar{z}$ can be reduced to break discrete logarithm. Next, using the fact that with high probability collisions in the first component do not occur, we reduce pseudorandomness to the restricted pseudorandomness of the inner VRB.

More formally, given a PPT adversary $\mathcal{A}$ against the pseudorandomness of the outer VRB, we will let Coll be the event

$$\mathsf{Coll} \;:\; \mathbf{x} \neq \mathbf{x}^* \;\wedge\; \left[\mathbf{a}^\top \mathbf{x}\right] = \left[\mathbf{a}^\top \mathbf{x}^*\right] \text{ for some queried } \mathbf{x}$$

where $\mathbf{x}^*$ is the point $\mathcal{A}$ returns to the challenger.

Claim 3 *If the DLP is hard, then* $\Pr[\mathsf{Coll}]$ *is negligible.*

Assuming the claim, we can then describe a reduction $\mathcal{C}$ to the restricted pseudorandomness for the inner VRB. A full description of $\mathcal{C}$ appears in Figure 20.

We being observing that, assuming $\neg\mathsf{Coll}$, $\mathcal{C}$ correctly replies all the evaluation queries as $z \neq z^*$ implies that the oracle $\mathsf{VRF.Eval}$ replies with $y = \mathsf{VRB}^*.\mathsf{Eval}(\mathsf{sk}^*, (z, w))$. Next, again assuming $\neg\mathsf{Coll}$, when $\mathcal{C}$ send (z^*, w^*) to the challenger, its game is not aborted as no previous queries as first component equal to z^*.

Finally, let us call b the bit chosen by $\mathcal{C}$'s challenger. If $b = 1$ then $y^* = \mathsf{VRB}^*.\mathsf{Eval}(\mathsf{sk}^*, (z^*, w^*))$ and in particular $\mathcal{C}$ perfectly simulates the pseudorandomness game for $\mathcal{A}$ with challenge bit 1. Analogously, when $b = 1$ then $y^* \sim$

Reduction $\mathcal{C}(\mathsf{vk}^*)$

1 : // Public parameters generation
2 : Sample $[\mathbf{a}] \leftarrow^{\$} \mathbb{G}^2$ with $a_2 \neq 0$ and set $\mathsf{vk} \leftarrow (\mathsf{vk}^*, [\mathbf{a}])$
3 : Initialize $\mathbf{x}^* \leftarrow \perp$ and run $\mathcal{A}(\mathsf{vk})$
4 : // Answering evaluation queries
5 : **When** $\mathcal{A}$ queries an evaluation in $\mathbf{x} \neq \mathbf{x}^*$:
6 : $\quad z \leftarrow \left[\mathbf{a}^\top \mathbf{x}\right]$, $w \leftarrow \left[\mathbf{e}_1^\top \mathbf{x}\right]$, $(y, \pi) \leftarrow \mathcal{O}_{\mathsf{eval}}((z, w))$
7 : $\quad$ Send $\mathcal{A} \leftarrow (y, \pi)$
8 : // Answering challenge request
9 : **When** $\mathcal{A}$ send challenge input $\mathbf{x}^*$:
10 : $\quad$ **If** $\mathbf{x}^*$ was previously queried: **Return** $\perp$
11 : $\quad z^* \leftarrow \left[\mathbf{a}^\top \mathbf{x}^*\right]$, $w \leftarrow \left[\mathbf{e}_1^\top \mathbf{x}^*\right]$
12 : $\quad$ Send (z^*, w^*) to the challenger and wait for y^*
13 : $\quad$ Send $\mathcal{A} \leftarrow y^*$
14 : // Final Output
15 : **When** $\mathcal{A} \to b'$: **Return** b'

Fig. 20. $\mathcal{C}$ using $\mathcal{A}$ to break restricted pseudorandomness for the inner VRB.

$U(\mathcal{Y})$ and in particular perfectly simulates the pseudorandomness game for $\mathcal{A}$ with challenge bit 0. In conclusion

$$\begin{aligned}
\mathsf{Adv}(\mathcal{C}) &= |\Pr[\mathcal{C} \to 1 \,|\, b = 0] - \Pr[\mathcal{C} \to 1 \,|\, b = 1]| \\
&\geq \Pr[\neg\mathsf{Coll}] \cdot |\Pr[\mathcal{C} \to 1 \,|\, \neg\mathsf{Coll},\ b = 0] - \Pr[\mathcal{C} \to 1 \,|\, \neg\mathsf{Coll},\ b = 1]| \\
&= \Pr[\neg\mathsf{Coll}] \cdot |\Pr[\mathcal{A} \to 1 \,|\, \neg\mathsf{Coll},\ b = 0] - \Pr[\mathcal{A} \to 1 \,|\, \neg\mathsf{Coll},\ b = 1]| \\
&\geq \mathsf{Adv}(\mathcal{A}) - |\Pr[\mathcal{A} \to 1,\ \mathsf{Coll} \,|\, b = 0] - \Pr[\mathcal{A} \to 1,\ \mathsf{Coll} \,|\, b = 1]| \\
&\geq \mathsf{Adv}(\mathcal{A}) - \Pr[\mathsf{Coll}] .
\end{aligned}$$

Proof of Claim 3 We prove this describing $\mathcal{B}$ which uses $\mathcal{A}$ to find a linear relation among two random group elements H, K, a problem equivalent to DLP. A full description of $\mathcal{B}$ appears in Figure 21.

We immediately observe that $\mathcal{B}$ perfectly simulates $\mathcal{A}$'s challenger as it can sample the inner VRB's keys $\mathsf{vk}^*, \mathsf{sk}^*$ and all the computation in $[\mathbf{a}]$ can be performed without knowing $\mathbf{a}$. Moreover, $[a_2] = K \neq 0$, or else $\mathcal{A}$ immediately finds a correct non-trivial linear relation among H and K.

Finally, if Coll occurs then either $\mathcal{B}$ wins because $K = 0$, or the condition in line 19 is satisfied and $\mathcal{B}$ returns $\mathbf{x} - \mathbf{x}^* \neq \mathbf{0}$. These is a non trivial linear relation among H, K since

$$z = z^* \ \Rightarrow\ \left[\mathbf{a}^\top \mathbf{x}\right] \ =\ \left[\mathbf{a}^\top \mathbf{x}^*\right] \ \Rightarrow\ \left[\mathbf{a}^\top (\mathbf{x} - \mathbf{x}^*)\right] = \mathbf{0}.$$

Reduction $\mathcal{B}(H, K)$

1 : // Public parameters generation
2 : Sample $\mathsf{vk}^*, \mathsf{sk}^* \leftarrow^{\$} \mathsf{VRB}^*.\mathsf{Gen}(1^\lambda)$
3 : **If** $K = 0$: **Return** $(0, 1)$ // $0 \cdot H + 1 \cdot K = 0$
4 : Set $[\mathbf{a}] \leftarrow (H, K)$ and $\mathsf{vk} \leftarrow (\mathsf{vk}^*, [\mathbf{a}])$
5 : Initialize $\mathbf{x}^* \leftarrow \perp$ and run $\mathcal{A}(\mathsf{vk})$
6 : // Answering evaluation queries
7 : **When** $\mathcal{A}$ queries an evaluation in $\mathbf{x} \neq \mathbf{x}^*$:
8 : $z \leftarrow \left[\mathbf{a}^\top \mathbf{x}\right]$, $w \leftarrow \left[\mathbf{e}_1^\top \mathbf{x}\right]$, $(y, \pi) \leftarrow \mathsf{VRB}^*.\mathsf{Eval}(\mathsf{sk}^*, (z, w))$
9 : Send $\mathcal{A} \leftarrow (y, \pi)$
10 : // Answering challenge request
11 : **When** $\mathcal{A}$ send challenge input $\mathbf{x}^*$:
12 : **If** $\mathbf{x}^*$ was previously queried: **Return** $\perp$
13 : Sample $b \leftarrow^{\$} \{0, 1\}$ // simulate challenger's bit
14 : $z^* \leftarrow \left[\mathbf{a}^\top \mathbf{x}^*\right]$, $w^* \leftarrow \left[\mathbf{e}_1^\top \mathbf{x}^*\right]$, $(y_1^*, \pi_1^*) \leftarrow \mathsf{VRB}^*.\mathsf{Eval}(\mathsf{sk}^*, (z^*, w^*))$
15 : $y_0^* \leftarrow^{\$} \mathcal{Y}$
16 : Send $\mathcal{A} \leftarrow y_b^*$
17 : // Final Output
18 : **When** $\mathcal{A}$ halts:
19 : **If** $\mathcal{A}$ queried $\mathbf{x} \neq \mathbf{x}^*$ such that $z = z^*$: **Return** $\mathbf{x} - \mathbf{x}^*$
20 : **Else**: **Return** $\perp$

Fig. 21. $\mathcal{B}$ breaking discrete logarithm if Coll occurs.

Conversely, if $\neg\mathsf{Coll}$, then $\mathcal{B}$ return $\perp$. Therefore $\Pr[\mathsf{Coll}] \leq \mathsf{Adv}(\mathcal{B})$ that is negligible.

B.6 Unbiasable VRF Compiler

Proof of Theorem 7. We recall for our definition of Encode:

$$\mathsf{Encode}(r, s, \alpha, \beta, k, \mathsf{vk}) \;=\; (r, s, f_k(\mathsf{vk}), \mathsf{repr}([\alpha]), \mathsf{repr}([\beta]), \mathsf{repr}([\alpha\beta + k])).$$

To simply notation we identify in the proof $\mathbb{G}$ with $\mathsf{repr}(\mathbb{G})$ and refrain from explicitly using repr in the encoding function. As in previous proof, the VRF used by our compiler will be referred to as the *inner* VRF, whereas the compiler output is denoted as the *outer* VRF. Correctness and Unique Provability immediately follow from the underlying VRF. We thus focus on pseudorandomness and unbiasability, the latter being the most challenging to show.

Pseudorandomness. Let $\mathcal{A}$ a PPT adversary breaking pseudorandomness. Before providing a reduction, we show that, if the DLP is hard in $\mathbb{G}$, then it is hard for

$\mathcal{A}$ to query two point x, x^* (with x^* being the challenge point) whose encoding $\mathbf{v}$, $\mathbf{v}^*$ are such that $\left[\mathbf{a}^\top \mathbf{v}\right] = \left[\mathbf{a}^\top \mathbf{v}^*\right]$. We will define Coll such event.

Claim 4 *If the DLP problem is hard in $\mathbb{G}$, then $\Pr[\mathsf{Coll}]$ is negligible.*

Given the claim we can provide a reduction to the *inner* VRF's restricted pseudorandomness with respect to the first component. To do we describe a PPT adversary $\mathcal{B}$. On input vk^*, $\mathcal{B}$ samples $[\mathbf{a}] \leftarrow^{\$} \mathbb{G}^{4+3\eta}$ with $[a_2] \neq 0$, sets $\mathsf{vk} \leftarrow (\mathsf{vk}^*, [\mathbf{a}])$ and executes $\mathcal{A}$ on input vk. When $\mathcal{A}$ queries an evaluation on x, $\mathcal{B}$ compute $\mathbf{v}$ the encoding of x, vk, the two inner products $z \leftarrow \left[\mathbf{a}^\top \mathbf{v}\right]$ and $w \leftarrow \left[\mathbf{e}^\top \mathbf{v}\right]$ and queries its own VRF oracle on $\mathcal{O}_{\mathsf{eval}}((z, w))$. It then forwards the oracle reply (y, π) to $\mathcal{A}$. When $\mathcal{A}$ returns a challenge point x^*, $\mathcal{B}$ computes again its encoding $\mathbf{v}^*$, the two inner products $z^* = \left[\mathbf{a}^\top \mathbf{v}^*\right]$, $w^* = \left[\mathbf{e}^\top \mathbf{v}^*\right]$ and sends (z^*, w^*) to its challenger, eventually forwarding its response y^* to $\mathcal{A}$. Finally, when $\mathcal{A}$ returns a bit, $\mathcal{B}$ outputs the same bit.

We begin observing that if $\neg\mathsf{Coll}$, then $\mathcal{B}$ perfectly simulates $\mathcal{A}$'s challenger, as all queries $x \neq x^*$ it makes eventually yield $z \neq z^*$, which can be answered by $\mathcal{O}_{\mathsf{eval}}$ used by $\mathcal{B}$. Proceeding as in the proof of Theorem 6 we can finally show that

$$\mathsf{Adv}(\mathcal{B}) \geq \mathsf{Adv}(\mathcal{A}) - \Pr[\mathsf{Coll}].$$

Unbiasability. In order to prove unbiasability, we first define an intermediate game. Here an adversary $\mathcal{B}$ has to bias the VRF described in Figure 15, but instead of receiving inputs x it receives its encoding $\mathbf{v} = \mathsf{Encode}(x, \mathsf{vk})$. A full description of the Experiment appears in Figure 22. As with the regular unbiasability game the advantage of $\mathcal{B}$ is defined as

$$\mathsf{Adv}(\mathcal{B}) \coloneqq \Pr\left[\mathsf{Exp}^{\mathsf{enc-bias}}_{0,m,n}(\mathcal{B}) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{enc-bias}}_{1,m,n}(\mathcal{B}) = 1\right].$$

To prove the Theorem we first show that if the intermediate experiment is hard to bias for PPT adversaries with $\mathsf{PPT}^{\mathsf{DL}}$ preprocessing, then our VRF construction is unbiasable. Then, to prove hardness of the intermediate experiment, we transform it through a sequence of $2m$ hybrids $\mathsf{G}^0_1, \mathsf{G}^1_1, \mathsf{G}^2_1, \mathsf{G}^3_1, \mathsf{G}^0_2, \ldots, \mathsf{G}^3_m$ such that

- G^0_1: The Intermediate Unbiasability game as in Figure 22.
- G^1_ℓ: As G^0_ℓ but samples $\gamma \sim U(\mathbb{F}_q)$ and sets $\mathbf{v}_{i,\ell} \leftarrow \mathsf{HybEncode}_1(x_\ell, \mathsf{vk}_i, \gamma)$ for all $i \in [n]$.
- G^2_ℓ: As G^1_ℓ but $\mathbf{v}_{i,\ell}$ is computed as $\mathsf{HybEncode}_2(x_\ell, \mathsf{vk}_i)$ for all $i \in [n]$.
- G^3_ℓ: As G^2_ℓ but $\mathbf{v}_{i,\ell}$ is computed as $\mathsf{HybEncode}_3(x_\ell, \mathsf{vk}_i)$ for all $i \in [n]$.
- G^0_ℓ: Identical to $\mathsf{G}^3_{\ell-1}$.

Finally, we show that any PPT adversary with $\mathsf{PPT}^{\mathsf{DL}}$ preprocessing $\mathcal{B}$ executed in game G^3_m can be turned into an unbounded adversary $\mathcal{C}$ breaking unbiasability on independent points for the inner VRF, which completes the proof.

$\mathsf{Exp}^{\mathsf{enc-bias}}_{n,m}(\mathcal{B})$

1 : $(\mathsf{vk}_1, \ldots, \mathsf{vk}_n, p) \leftarrow \mathcal{B}$ different keys, p monotone
2 : $x_1, \ldots, x_m \leftarrow^{\$} \mathcal{X}$
3 : **For** all $i \in [n],\ j \in [m]$: $\mathbf{v}_{i,j} \leftarrow \mathsf{Encode}(x_j, \mathsf{vk}_i)$
4 : $((y_{1,1}, \pi_{1,1}), \ldots, (y_{n,m}, \pi_{n,m})) \leftarrow \mathcal{B}(\mathbf{v}_{1,1}, \ldots, \mathbf{v}_{n,m})$
5 : **For** all $i \in [n],\ j \in [m]$:
6 : **If** $\mathsf{VRF.Vfy}(\mathsf{vk}_i, x_j, y_{i,j}, \pi_{i,j}) = 0$: Set $y_{i,j} \leftarrow \perp$
7 : Sample $z_{1,1}, \ldots, z_{n,m} \leftarrow^{\$} \mathcal{Y}^{n,m}$
8 : **If** $b = 0$: **Return** $p(y_{1,1}, \ldots, y_{n,m})$
9 : **If** $b = 1$: **Return** $p(z_{1,1}, \ldots, z_{n,m})$

$\mathsf{HybEncode}_1(x, \mathsf{vk}, \gamma)$

1 : Parse $x = (r, s, \alpha, \beta, k)$
2 : **Return** $(r, s, f_k(\mathsf{vk}), \mathsf{repr}([\alpha]), \mathsf{repr}([\beta]), \mathsf{repr}([\gamma]))$

$\mathsf{HybEncode}_2(x, \mathsf{vk})$

1 : Parse $x = (r, s, \alpha, \beta, k)$
2 : **Return** $(r, s, f_k(\mathsf{vk}), \mathsf{repr}([\alpha]), \mathsf{repr}([\beta]), \mathsf{repr}([\alpha\beta]))$

$\mathsf{HybEncode}_3(x, \mathsf{vk})$

1 : Parse $x = (r, s, \alpha, \beta, k)$
2 : Sample $\mathbf{t} \leftarrow^{\$} \mathbb{F}_q^2$
3 : **Return** $(r, s, \mathbf{t}, \mathsf{repr}([\alpha]), \mathsf{repr}([\beta]), \mathsf{repr}([\alpha\beta]))$

Fig. 22. Intermediate Unbiasability and Hybrid Encoding. Differences are highlighted.

Intermediate Game ⇒ Unbiasability. Assume the game in Figure 22 to be hard against PPT adversaries with $\mathsf{PPT}^{\mathsf{DL}}$ preprocessing. Given $\mathcal{A}$ PPT playing against the unbiasability experiment we show that for any constant $c > 0$, asymptotically $\mathsf{Adv}(\mathcal{A}) \leq \lambda^{-c}$.

To do so we describe $\mathcal{B}$ depending on a parameter $\vartheta = \mathsf{poly}(\lambda)$ we specify later. Initially $\mathcal{B}$ executes $\mathcal{A}(1^\lambda)$ to get $\mathsf{vk}_1, \ldots, \mathsf{vk}_n$. Parsing each key $\mathsf{vk}_i = (\mathsf{vk}_i^*, [\mathbf{a}_i])$ it uses the DL oracle, accessible only during the preprocessing phase, to extract $\mathbf{a}_i$. This ends the preprocessing phase.

On input $\mathbf{v}_{i,j}$ it evaluates $z_{i,j} \leftarrow \left[\mathbf{a}_i^\top \mathbf{v}_{i,j}\right]$ and $w_{i,j} \leftarrow \left[\mathbf{e}_1^\top \mathbf{v}_{i,j}\right]$ and tries to evaluate the inner VRF with verification key vk^*. This would be trivial if it could somehow extract sk^*, however under our assumptions this does not appear to be possible. Instead $\mathcal{B}$ exploits the fact that, due to the redundant random elements r, s in the encoding of x, the Petersen hash is in fact a Chameleon Hash [KR98]. This means that for any α, β, k it can find r, s such the corresponding x has encoding $\mathbf{v}$ with $\left[\mathbf{a}_i^\top \mathbf{v}\right] = z_{i,j}$ and $\left[\mathbf{a}_i^\top \mathbf{v}\right] = w_{i,j}$. The strategy is then to sample ϑ-times these inputs colliding with $\mathbf{v}_{i,j}$ and run $\mathcal{A}$ on them. If $\mathcal{A}$ provides a valid

evaluation of the outer VRF during any of these re-executions, $\mathcal{B}$ can extract a value and a proof for the *inner* VRF on input $(z_{i,j}, w_{i,j})$. Conversely, for ϑ large enough, we conclude that $\mathcal{A}$ opens a random input whose encoding's hash collides with $\mathbf{v}_{i,j}$ only with probability $1/\mathsf{poly}(\lambda)$. In this case we let $\mathcal{B}$ give up and set $y_{i,j} \leftarrow \perp$. Finally, as $\mathcal{B}$ obtains all the values $y_{i,j}$ and proofs $\pi_{i,j}$, it returns them. A full description of $\mathcal{B}$ appears in Figure 23.

Reduction $\mathcal{B}$:

1 : Run $\mathcal{A}(1^\lambda) \to ((\mathsf{vk}_i^*, [\mathbf{a}_i])_{i=1}^n, p)$ and store its state st
2 : Use the DL oracle to compute $\mathbf{a}_i$ // $n(4+3\eta)$ queries
3 : Parse $\mathbf{a}_i = (a_{i,1}, a_{i,2}, \bar{\mathbf{a}}_i) \in \mathbb{F}_q \times \mathbb{F}_q \times \mathbb{F}_q^{2+3\eta}$
4 : Initialize an empty table T
5 : Send $(\mathsf{vk}_1^*, \ldots, \mathsf{vk}_n^*, p)$ and wait for inputs $\mathbf{v}_{i,j}$
6 : **For** all $i \in [n],\ j \in [m]$ with $a_{i,2} \neq 0$:
7 : Compute $z_{i,j} \leftarrow \left[\mathbf{a}_i^\top \mathbf{v}_j\right]$ and $w_{i,j} \leftarrow \left[\mathbf{e}^\top \mathbf{v}_j\right]$
8 : **For** $\nu \in \{1, \ldots, \vartheta\}$: // Try ϑ times to extract $y_{i,j}$
9 : Rewind $\mathcal{A}$ to state st.
10 : Sample points $x_1, \ldots, x_m \leftarrow^{\$} \mathcal{X}$
11 : Sample $k, \alpha, \beta \leftarrow^{\$} \mathbb{F}_q^3$
12 : Set $\mathbf{u} \leftarrow (f_k(\mathsf{vk}_i), [\alpha], [\beta], [\alpha\beta + k])$ // $\mathbf{u} \in \mathbb{F}_q^{2+3\eta}$
13 : // Chose s, r causing a collision with $\mathbf{v}_{i,j}$. u_1 is the 1^{st} entry of $\mathbf{u}$.
14 : $r \leftarrow \mathbf{e}^\top \mathbf{v}_{i,j} - u_1$ and $s \leftarrow a_{i,2}^{-1} \cdot \left(\mathbf{a}_i^\top \mathbf{v}_{i,j}^\top - a_1 r - \bar{\mathbf{a}}_i^\top \mathbf{u}\right)$
15 : $x_j \leftarrow (r, s, k, \alpha, \beta)$
16 : Run $\mathcal{A}(x_1, \ldots, x_m) \to (\mathbf{y}, \pi)$
17 : **If** $\mathsf{VRF}^*.\mathsf{Vfy}(\mathsf{vk}^*, (z_{i,j}, w_{i,j}), y_{i,j}, \pi_{i,j}) \to 1$:
18 : Store $T[i,j] \leftarrow (y_{i,j}, \pi_{i,j})$
19 : **End**
20 : **Return** $(T[1,1], \ldots, T[n,m])$

Fig. 23. $\mathcal{B}$ in $\mathsf{PPT}^{\mathsf{DL}}$ breaking the game in Figure 22 given $\mathcal{A}$.

We now analyze $\mathcal{B}$. First let us introduce some notation.

$\widetilde{x}_j$: The initial points chosen by the challenger, so that $\mathbf{v}_{i,j} = \mathsf{Encode}(\widetilde{x}_j, \mathsf{vk}_i)$.
$\mathsf{Valid}_{i,j}(\mathbf{x})$: The event $\mathcal{A}(\mathbf{x}) \to (\mathbf{y}, \pi)$ with $1 \leftarrow \mathsf{VRF}.\mathsf{Vfy}(\mathsf{vk}_i, x_j, y_{i,j}, \pi_{i,j})$.
$\mathsf{Fail}_{i,j}$: The event $T[i,j] = \perp \wedge \mathsf{Valid}_{i,j}(\widetilde{\mathbf{x}})$.
Fail : The event $\exists i, j : \mathsf{Fail}_{i,j}$.

With this notation we state a sequence of claims

Claim 5 *The values r, s computed in line 14, calling $\mathbf{v}^* = \mathsf{Encode}(r, s, k, \alpha, \beta)$, satisfy*

$$\left[\mathbf{a}_i^\top \mathbf{v}^*\right] \;=\; z_{i,j} \;\coloneqq\; \left[\mathbf{a}_i^\top \mathbf{v}_{i,j}\right] \qquad \left[\mathbf{e}^\top \mathbf{v}^*\right] \;=\; w_{i,j} \;\coloneqq\; \left[\mathbf{e}^\top \mathbf{v}_{i,j}\right].$$

Moreover $x_j = (r, s, k, \alpha, \beta) \sim U(\mathbb{F}_q^5)$.

Claim 6 *The check in line 17 passes iff $\mathsf{VRF.Vfy}(\mathsf{vk}_i, x_j, y_{i,j}, \pi_{i,j}) \to 1$.*

Claim 7 *If $\neg\mathsf{Fail}$, calling $\widetilde{\mathbf{y}}$ the output of $\mathcal{A}$ in the unbiasability experiment with challenger points $\widetilde{x}_1, \ldots, \widetilde{x}_n$ and $T[i,j] = (t_{i,j}, \pi_{i,j})$ then $\widetilde{y}_{i,j} \leq t_{i,j}$.*

Claim 8 $\Pr\left[\mathsf{Fail}_{i,j}\right] \leq 1/\vartheta$ *and in particular* $\Pr\left[\mathsf{Fail}\right] \leq nm/\vartheta$.

Given these claims we show a relation between the advantage of $\mathcal{A}$ and $\mathcal{B}$. Indeed, with $\vartheta = mn \cdot \lambda^c$, Fail occurs with probability smaller than λ^{-c}. Thus, given $\mathbf{z} \sim U(\mathcal{Y}^{n,m})$

$$\begin{aligned}
\mathsf{Adv}(\mathcal{B}) \;&=\; \Pr\left[\mathsf{Exp}^{\mathsf{enc-bias}}_{0,n,m}(\mathcal{B}) = 1\right] - \Pr\left[p(\mathbf{z}) = 1\right] \\
&\geq\; \Pr\left[\neg\mathsf{Fail}\right] \cdot \Pr\left[\mathsf{Exp}^{\mathsf{enc-bias}}_{0,n,m}(\mathcal{B}) = 1 \,\middle|\, \neg\mathsf{Fail}\right] - \Pr\left[p(\mathbf{z}) = 1\right] \\
&\geq\; \left(1 - \frac{1}{\lambda^c}\right) \cdot \Pr\left[\mathsf{Exp}^{\mathsf{bias}}_{0,n,m}(\mathcal{A}) = 1\right] - \Pr\left[p(\mathbf{z}) = 1\right] \\
&\geq\; \left(1 - \frac{1}{\lambda^c}\right) \cdot \left(\Pr\left[\mathsf{Exp}^{\mathsf{bias}}_{0,n,m}(\mathcal{A}) = 1\right] - \Pr\left[p(\mathbf{z}) = 1\right]\right) - \frac{1}{\lambda^c} \\
&=\; \left(1 - \frac{1}{\lambda^c}\right) \cdot \mathsf{Adv}(\mathcal{A}) - \frac{1}{\lambda^c} \\
&\geq \frac{1}{2} \cdot \mathsf{Adv}(\mathcal{A}) - \frac{1}{\lambda^c}
\end{aligned}$$

where the second inequality follows from Claim 8, Claim 7 and the monotonicity of p, while the forth is asymptotically true. We then conclude that, as we assumed the intermediate game to be hard, there exists a negligible function $\varepsilon_c(\lambda)$ bounding the advantage of $\mathcal{B}$. For sufficiently large λ, $\varepsilon_c(\lambda) \leq \lambda^{-c}$, which asymptotically implies

$$\mathsf{Adv}(\mathcal{A}) \leq 4 \cdot \lambda^{-c}$$

concluding the proof.

Proof of $\mathsf{G}^0_\ell \approx \mathsf{G}^1_\ell$. For any distinguisher $\mathcal{D}$ with preprocessing $\mathsf{PPT}^{\mathsf{DL}}$ we provide an adversary $\mathcal{R}_1$ in the same class breaking $\mathsf{DLPrep\text{-}DDH}$.

The reduction begin executing $\mathcal{D}$ and using its preprocessing DL oracle to answer DL queries from $\mathcal{D}$. Once the preprocessing phase ends, let H, U, V be the input DDH tuple, so that $H = [\alpha]$, $U = [\beta]$ and V is either $[\alpha\beta]$ or $V = [\gamma]$ Next $\mathcal{R}_1$ waits for $\mathcal{D}$ to return $\mathsf{vk}_1, \ldots \mathsf{vk}_m$. Then it samples $x_1, \ldots x_{j-1}, x_{j+1}, x_m$ from $\mathbb{F}_q^5$ and computes

– If $j < \ell$: $\mathbf{v}_{i,j} \leftarrow^{\$} \mathsf{HybEncode}_3(x_j, \mathsf{vk}_i)$

– If $j > \ell$: $\mathbf{v}_{i,j} \leftarrow \mathsf{Encode}(x_i, \mathsf{vk}_i)$

Finally, to compute $\mathbf{v}_{i,\ell}$ it samples $r, s, k \leftarrow^{\$} \mathbb{F}_q$ and sets

$$\mathbf{v}_{i,\ell} \leftarrow (r, s, f_k(\mathsf{vk}_i), H, U, V + [k]) .$$

Finally, waits for $\mathcal{D}$ to output a bit and returns the same bit.

By construction, $\mathcal{R}_1$ correctly simulates the initial DL queries, an the encoding $\mathbf{v}_{i,j}$ with $j \neq \ell$. Finally, if receives as input a DDH tuple $[\alpha]$, $[\beta]$, $[\alpha\beta]$, then $\mathbf{v}_{i,\ell}$ is computed as

$$\mathbf{v}_{i,\ell} = (r, s, f_k(\mathsf{vk}_i), \mathsf{repr}([\alpha]), \mathsf{repr}([\beta]), \mathsf{repr}([\alpha\beta + k]))$$

that is, as defined in G^0_ℓ. Conversely, if $\mathcal{R}_1$ receives a random tuple $[\alpha], [\beta], [\gamma]$, then

$$\mathbf{v}_{i,\ell} = (r, s, f_k(\mathsf{vk}_i), [\alpha], [\beta], [\gamma + k]).$$

Since $\gamma \sim U(\mathbb{F}_q)$ and k is the same for all $i \in [n]$, this distribution is identical to the one in G^1_ℓ, observed through the map $\gamma \mapsto \gamma + k$. In conclusion $\mathsf{Adv}(\mathcal{D}) = \mathsf{Adv}(\mathcal{R}_1)$ which is negligible assuming $\mathsf{DLPrep\text{-}DDH}$.

Proof of $\mathsf{G}^1_\ell \approx \mathsf{G}^2_\ell$. As in the previous proof, given a distinguisher $\mathcal{D}$ with preprocessing in $\mathsf{PPT}^{\mathsf{DL}}$, we define a reduction $\mathcal{R}_2$ in the same computational class to $\mathsf{DLPrep\text{-}DDH}$.

$\mathcal{R}_2$ begins by running $\mathcal{D}$ and answering DL queries with its DL oracle in the preprocessing phase. As the end of the preprocessing phase it gets H, U, V either a DDH tuple or a random one. $\mathcal{R}_2$ continues by running $\mathcal{D}$ to get $\mathsf{vk}_1, \ldots, \mathsf{vk}_m$. In order to computes the inputs encoding $\mathbf{v}_{i,j}$ it samples $x_1, \ldots, x_m \leftarrow^{\$} \mathcal{X}$ (although x_ℓ will not be used) and sets

– If $j < \ell$: $\mathbf{v}_{i,j} \leftarrow \mathsf{HybEncode}_3(x_j, \mathsf{vk}_i)$ for all $i \in [n]$.
– If $j > \ell$: $\mathbf{v}_{i,j} \leftarrow \mathsf{Encode}(x_j, \mathsf{vk}_i)$ for all $i \in [n]$.

Finally, it samples r, s, k and sets for all $i \in [n]$

$$\mathbf{v}_{i,\ell} \leftarrow (r, s, f_k(\mathsf{vk}_i), H, U, V) .$$

Eventually, when $\mathcal{D}$ outputs a bit, $\mathcal{R}_2$ does the same.

As observed in the previous proof, $\mathbf{v}_{i,j}$ with $j \neq \ell$ follows the right distribution. We thus only focus on $\mathbf{v}_{i,\ell}$. If $\mathcal{R}_2$ receives a DDH tuple, then $(H, V, U) = ([\alpha], [\beta], [\alpha\beta])$ meaning that

$$\mathbf{v}_{i,\ell} = (r, s, f_k(\mathsf{vk}_i), [\alpha], [\beta], [\alpha\beta])$$

which is as defined in G^2_ℓ. Conversely, if H, U, V there exists $\gamma \sim U(\mathbb{F}_q)$ and independent from r, s, k, α, β, which $H = [\alpha]$ and $U = [\beta]$ such that $V = [\gamma]$. In particular it is immediate to observe that $\mathbf{v}_{i,\ell}$ are computed as prescribed in G^1_ℓ. Therefore $\mathsf{Adv}(\mathcal{D}) \leq \mathsf{Adv}(\mathcal{R}_2)$ which is negligible assuming $\mathsf{DLPrep\text{-}DDH}$ to be hard.

Proof of $\mathsf{G}^2_\ell \approx \mathsf{G}^3_\ell$. Given a distinguisher $\mathcal{D}$ with $\mathsf{PPT}^{\mathsf{DL}}$ preprocessing we provide a reduction $\mathcal{R}_3$ in the same computational class to the pseudorandomness of the PRF f.

Initially $\mathcal{R}_3$ runs $\mathcal{D}$ forwarding DL queries to its own DL oracle during the preprocessing phase. Once the preprocessing is over, it receives access to $\mathcal{O}_{\mathsf{prf}}$ to evaluate the PRF, and runs $\mathcal{D}$ to get $\mathsf{vk}_1, \ldots, \mathsf{vk}_n$. Next, it samples $x_1, \ldots, x_m \leftarrow^{\$} \mathcal{X}$ (although x_ℓ will not be used) and computes $\mathbf{v}_{i,j}$ as

- If $j < \ell$: $\mathbf{v}_{i,j} \leftarrow^{\$} \mathsf{HybEncode}_3(x_j, \mathsf{vk}_i)$ for all $i \in [n]$.
- If $j > \ell$: $\mathbf{v}_{i,j} \leftarrow \mathsf{Encode}(x_i, \mathsf{vk}_i)$ for all $i \in [n]$.

Finally, to compute $\mathbf{v}_{i,\ell}$ it samples r, s, α, β from $\mathbb{F}_q$ and sets

$$\mathbf{v}_{i,\ell} \geq \left(r, s, \mathcal{O}_{\mathsf{prf}}(\mathsf{vk}_i), [\alpha], [\beta], [\alpha\beta]\,.\right)$$

Eventually, when $\mathcal{D}$ returns a bit, it outputs the same bit.

As previously we only focus on the distribution of $\mathbf{v}_{i,\ell}$ in the two experiments. When $\mathcal{R}_3$ interact with a real PRF, let k be the uniformly random key used. Then $\mathcal{O}_{\mathsf{prf}}(\mathsf{vk}_i) = f_k(\mathsf{vk}_i)$, meaning that $\mathbf{v}_{i,\ell}$ is computed as in G^2_ℓ. Conversely, if $\mathcal{R}_3$ has oracle access to a real random function, since by construction all keys returned by $\mathcal{D}$ must be different, the values $\mathbf{s}_i = \mathcal{O}_{\mathsf{prf}}(\mathsf{vk}_i)$ are all uniformly distributed and independent. Thus $\mathbf{v}_{i,\ell}$ for $i \in [n]$ are computed as prescribed in G^3_ℓ. We thus conclude that $\mathsf{Adv}(\mathcal{D}) = \mathsf{Adv}(\mathcal{R}_3)$ that is negligible if the PRF is secure against $\mathsf{PPT}^{\mathsf{DL}}$ adversaries.

G^3_m *is hard unconditionally.* We finally show that given an adversary $\mathcal{B}$ with $\mathsf{PPT}^{\mathsf{DL}}$ preprocessing for G^3_m we can describe an unbounded $\mathcal{C}$ for the unbiasability on independent points of the inner VRF.

Initially $\mathcal{C}$ executes $\mathcal{B}$ and replies to its discrete logarithm queries by actually computing it. Once the preprocessing phase is over, it waits for $\mathcal{B}$ to output its list of verification keys $\mathsf{vk}_1, \ldots, \mathsf{vk}_n$ and predicate. It then parse each key as $\mathsf{vk}_i = (\mathsf{vk}^*_i, [\mathbf{a_i}])$, compute $\mathbf{a}_i$ and sends to its challenger $\mathsf{vk}^*_1, \ldots, \mathsf{vk}^*_n$, which are keys for the *inner* VRF, along with p. When the challenger replies with $(z_{i,j}, w_{i,j})$ group elements it computes in exponential time their discrete logarithm $\zeta_{i,j}, \omega_{i,j}$, i.e. such that $z_{i,j} = [\zeta_{i,j}]$ and $w_{i,j} = [\omega_{i,j}]$. At this point $\mathcal{C}$ computes $\mathbf{v}_{i,j}$ such that $\left[\mathbf{a}_i^\top \mathbf{v}_{i,j}\right] = z_{i,j}$ and $\left[\mathbf{e}^\top \mathbf{v}_{i,j}\right] = w_{i,j}$ (we later explain how). Given these vectors, it sends them to $\mathcal{B}$ which eventually replies with $(y_{i,j}, \pi_{i,j})$. $\mathcal{C}$ then returns these values and proofs $(y_{i,j}, \pi_{i,j})$ and halts.

Assuming $\mathbf{v}_{i,j}$ were correctly sampled, $\mathcal{C}$ simulates perfectly G^3_m. Moreover, $y_{i,j}, \pi_{i,j}$ is accepted only if $\mathsf{VRF}^*.\mathsf{Eval}(\mathsf{vk}^*_i, (z_{i,j}, w_{i,j}), y_{i,j}, \pi_{i,j}) \to 1$. Calling then $\widetilde{\mathbf{y}}$ the filtered output of $\mathcal{B}$ in G^3_m and $\widehat{\mathbf{y}}$ the filtered output of $\mathcal{C}$ in $\mathsf{Exp}^{\mathsf{ip-bias}}_{b,n,m}$, then according to the partial order introduced in Definition 7, $\widetilde{\mathbf{y}} \leq \widehat{\mathbf{y}}$. Thus by monotonicity of p

$$\Pr\left[p(\widetilde{\mathbf{y}}) = 1\right] \;\leq\; \Pr\left[p(\widehat{\mathbf{y}}) = 1\right] \quad \Rightarrow \quad \mathsf{Adv}(\mathcal{B}) \;\leq\; \mathsf{Adv}(\mathcal{C}).$$

Finally we have to show how the vectors $\mathbf{v}_{i,j}$ are computed. Unfortunately we cannot find these preimages as in previous proofs using the first two component of $\mathbf{v}_{i,j}$ as these have to be equal for all $i \in [n]$. Instead we use the fact that third and fourth components $\mathbf{t} \in \mathbb{F}_q^2$ are uniformly random and independent in each vector. To see how we call $\mathbf{v}_{i,j} = (r_i, s_i, \mathbf{t}_{i,j}, \mathbf{u}_i)$. The equation that needs to be satisfied is

$$\begin{pmatrix}\zeta_{i,j}\\ \omega_{i,j}\end{pmatrix} = \begin{pmatrix}a_{i,1} & a_{i,2}\\ 1 & 0\end{pmatrix} \cdot \begin{pmatrix}r_j\\ s_j\end{pmatrix} + \begin{pmatrix}a_{i,3} & a_{i,4}\\ 1 & 0\end{pmatrix} \cdot \mathbf{t}_{i,j} + \begin{pmatrix}a_{i,5} & \dots & a_{i,4+4\eta}\\ 0 & \dots & 0\end{pmatrix} \cdot \mathbf{u}_j.$$

If vk_i satisfies $a_{i,4} \neq 0$ (if it does not, no proof would be accepted for vk_i), the second square matrix is invertible and in particular there always exists one $\mathbf{t}_{i,j}$ satisfying the above, given $r_j, s_j, \mathbf{u}_j$ sampled according to G_m^3. Moreover, as $\zeta_{i,j}, \omega_{i,j} \sim U(\mathbb{F}_q^2)$ then also $\mathbf{t}_{i,j} \sim U(\mathbb{F}_q^2)$.

Proof of Claim 5. By definition of r, s as per line 14 and by the definition of encoding $\mathbf{v}^* = (r, s, \mathbf{u})$ we have that

$$\begin{aligned} r = \mathbf{e}^\top \mathbf{v}_{i,j} - u_1 \quad &\Rightarrow \quad \mathbf{e}^\top \mathbf{v}^* = r + u_1 = \mathbf{e}^\top \mathbf{v}_{i,j} \\ s = a_{i,2}^{-1}\left(\mathbf{a}_i^\top \mathbf{v}_{i,j} - a_{i,1} r - \bar{\mathbf{a}}_i^\top \mathbf{u}\right) \quad &\Rightarrow \quad \mathbf{a}_i^\top \mathbf{v}^* = a_{i,1} r + a_{i,2} s + \bar{\mathbf{a}}_i^\top \mathbf{u} = \mathbf{a}_i^\top \mathbf{v}_{i,j}. \end{aligned}$$

where the second equality follows as $\mathbf{e} = \mathbf{e}_1 + \mathbf{e}_3$, and in particular $\mathbf{e}_3^\top \mathbf{v}^*$ returns the first entry of $\mathbf{u}$. Regarding the distribution of x_j instead, observe that k, α, β already follows the right distribution. Moreover $\mathbf{e}_1^\top \mathbf{v}_{i,j}$ is the first component of $\widetilde{x}_j$, the input used to compute $\mathbf{v}_{i,j}$. Thus this value is uniform and independent from k, α, β and so is r. Finally, as $a_{i,2} \neq 0$ (or else the loop starting at line 8 is never executed), we have that $\mathbf{a}_i^\top \mathbf{v}_{i,j}$ is uniformly random even conditioning on $r = r_0$, as this inner product is the sum of $a_{i,2}\widetilde{s}$ and another term independent of $\widetilde{s}$, with $\widetilde{s}$ being the second component of $\widetilde{x}$. Thus, conditioning on r, k, α, β, the variable s is still uniform over $\mathbb{F}_q$, as $a_{i,2}^{-1} \cdot \mathbf{a}_i^\top \mathbf{v}_{i,j}$ is.

Proof of Claim 6. By Claim 5, $z_{i,j}, w_{i,j}$ are respectively equal to $\left[\mathbf{a}_i^\top \mathbf{v}^*\right]$ and $\left[\mathbf{e}^\top \mathbf{v}^*\right]$ with $\mathbf{v}^*$ the encoding of x_j. Thus, if $\mathsf{VRF.Vfy}(\mathsf{vk}_i, x_j, y_{i,j}, \pi_{i,j}) = 1$, then this implies that the check in line 17 passes. Conversely if the check passes, as the inner loop is executed only if $a_{i,2} \neq 0$, then also $\mathsf{VRF.Vfy}(\mathsf{vk}_i, x_j, y_{i,j}, \pi_{i,j}) = 1$.

Proof of Claim 7. If $T[i,j] = \perp$, then $\neg\mathsf{Fail}$ implies that $\mathcal{A}$ executed with input $\widetilde{x}_1, \dots, \widetilde{x}_m$ does not return a valid proof for $\widetilde{y}_{i,j}$. Thus $\widetilde{y}_{i,j} = \perp \leq t_{i,j}$ regardless of the value of $t_{i,j}$. Conversely, if $T[i,j] = (t_{i,j}, \pi_{i,j})$, this is the output of the *inner* VRF on input $(z_{i,j}, w_{i,j})$ by construction. Since $\mathsf{Encode}(\widetilde{x}_j, \mathsf{vk}_i) = \mathbf{v}_{i,j}$ by the way we defined $z_{i,j}, w_{i,j}$, we have that $t_{i,j}$ is the unique output of the outer VRF. Thus either $\widetilde{y}_{i,j} = \perp$, which implies $\widetilde{y}_{i,j} \leq t_{i,j}$, or $\widetilde{y}_{i,j} = t_{i,j}$, which again yields $\widetilde{y}_{i,j} \leq t_{i,j}$.

Proof of Claim 8. Let P be the probability that on random inputs $x_1, \dots, x_m$, with x_j being such that

$$\mathbf{v}^* \coloneqq \mathsf{Encode}(x_j, \mathsf{vk}_i) \;\Rightarrow\; z_{i,j} = \left[\mathbf{a}_i^\top \mathbf{v}^*\right],\ w_{i,j} = \left[\mathbf{e}^\top \mathbf{v}^*\right].$$

then $\mathcal{A}$ returns a valid proof for $y_{i,j}$. Then $\mathsf{Fail}_{i,j}$ occurs if and only ϑ repetitions of this experiment fails, and only the last one succeeds. As, given the above conditions, all executions are statistically independent, we have that $\Pr[\mathsf{Fail}_{i,j}] = (1-P)^{\vartheta}P$. For $0 \leq P \leq 1$ the right hand side has maximum in $1/(1+\vartheta)$, which implies that

$$\Pr[\mathsf{Fail}_{i,j}] \;\leq\; (1-P)^{\vartheta}P \;\leq\; \left(1-\frac{1}{\vartheta+1}\right)^{\vartheta} \cdot \frac{1}{\vartheta+1} \;\leq\; \frac{1}{\vartheta+1} \;\leq\; \frac{1}{\vartheta}$$

where the third inequality follows as the first factor is smaller than 1.