Verifiable Random Permutations

Yevgeniy Dodis* Prashant Puniya†

February 27, 2006

Abstract

Pseudorandom Functions (PRFs), introduced by Goldreich, Goldwasser and Micali [9], allow one to efficiently simulate the computation of a function which is indistinguishable from a truly random function. A seemingly stronger primitive is that of a (strong) pseudorandom permutation (PRP) [13], which allows one to efficiently simulate a truly random permutation (and its inverse). The celebrated result of Luby and Rackoff [13] shows that these primitives are, in fact, equivalent: four rounds of the Feistel transform are necessary and sufficient to turn a PRF into a (strong) PRP.

In this paper we study a similar conversion for the *verifiable* analogs of PRFs and PRPs, called Verifiable Random Functions (VRFs) and Verifiable Random Permutations (VRPs). VRFs, introduced by Micali, Rabin and Vadhan [16], extend the notion of a PRF to allow the owner of the secret key for the VRF to prove to the outside parties that a given VRF value was correctly (and uniquely!) computed. Yet, such proofs do not violate the pseudorandomness of the remaining, yet "unopened" values. VRPs, introduced in this paper, similarly extend the notion of PRPs. We notice that the result of Luby and Rackoff no longer applies to converting VRFs into VRPs, since the VRP proofs must reveal the VRF outputs (and proofs) of the intermediate rounds. Indeed, we show that even logarithmic (in the security parameter) number of rounds is not enough for this conversion. Our main result, however, shows that *super-logarithmic* number of rounds of the Feistel transform suffice to build a VRP out of an arbitrary VRF.

As an application, we give a construction of non-interactive zero-knowledge (NIZK) proofs with efficient provers for any NP language from any VRF. The result is obtained from our VRF→VRP conversion, by noticing that VRPs easily yield "invariant signatures" of Goldwasser and Ostrovsky [10], which are known to imply NIZK. (We also notice that the detour through VRPs seems necessary for this implication, since using VRFs in place of invariant signatures is provably *insufficient* for the NIZK construction of [10] to go through.)

*Department of Computer Science, New York University, 251 Mercer Street, New York, NY 10012, USA. Email: dodis@cs.nyu.edu

†Department of Computer Science New York University, 251 Mercer Street, New York, NY 10012, USA. Email: puniya@cs.nyu.edu

1 Introduction

PRFs and PRPs. Pseudorandom Functions (PRFs), introduced by Goldreich, Goldwasser and Micali [9], form one of the most fundamental primitives of symmetric-key cryptography. They allow one to sample an efficient function from a small family of functions, such that this function is indistinguishable from a truly random (exponential size) function, as long as the distinguisher does not know the secret key of the PRF. However, in some applications, such as CBC-mode encryption, one actually needs to have a random *permutation* (with its inverse) and not just a random function. The corresponding efficient primitive, which allows one to compute both the permutation and its inverse, is called a (strong) pseudorandom permutation (PRP) [13], or a block cipher, and is also extremely important in symmetric-key cryptography. Upon an artificial look, a PRP seems to be a much more powerful primitive than a PRF. Indeed, it is trivial to see that a PRP (on any "non-trivial domain") is always a PRF, but the converse is clearly false. However, the celebrated result of Luby and Rackoff [13] shows that this intuition is wrong and that these primitives are, in fact, equivalent!

The construction proposed by Luby and Rackoff [13] is based on the design of the popular block cipher *DES*. *DES* is basically an iterated construction that involves multiple applications of the *Feistel permutation* (aka "Feistel Transform"). The *Feistel permutation* is a simple construction that when applied to a function f from n to n bits, gives a permutation on $2n$ bits: $\Psi_f(x_L \| x_R) = x_R \| (x_L \oplus f(x_R))$. Luby and Rackoff [13] then showed that 4 rounds of the Feistel permutation, with independent PRFs used in each round, is always a (strong) PRP, while 3 rounds or fewer are not enough. Following this, there has been a lot of work improving on the result of Luby and Rackoff, see [17, 15, 18]. We remark that a crucial reason for the validity of all these results is that the *intermediate Feistel values arising after each round are never given to the attacker.*

VRFs and VRPs. In this paper we study the verifiable analogs of PRFs and PRPs, called Verifiable Random Functions (VRFs) and Verifiable Random Permutations (VRPs). For concreteness, we concentrate of PRFs/VRFs first, returning later to PRPs/VRPs. Intuitively, PRFs have a limitation that one must trust the owner of the secret key that a given PRF value is correctly computed. And even when done so, a party receiving a correct PRF value cannot later convince some other party that the value is indeed correct (i.e., PRFs values are "non-transferable"). In fact, since the function values are supposed to be (pseudo)random, it seems that such verifiability of outputs of a PRP would contradict its pseudorandomness. The way out of this apparent contradiction was provided by Micali, Rabin and Vadhan [16], who introduced the notion of a VRF.[1] As with PRFs, the output of a VRF should look *indistinguishable* from a truly random function. However, it should also be possible to provide a short proof that this output is computed honestly. More concretely, a VRF f has a public key PK_f and a private key SK_f such that,

- Given the private key SK_f, it should be possible to evaluate f at any input x and get output $f(x)$. At the same time, it should also be possible to give a proof $proof(x)$ that $f(x)$ is indeed the output of evaluating f on x honestly.
- Given only the public key PK_f, an input/output pair $(x, f(x))$ and the corresponding proof $proof(x)$, it should be possible to tell if $f(x)$ is the output of honestly evaluating f on x.

In terms of security, the public key PK_f should commit the "owner" of the VRF to all its values in a unique way, even if the owner tries to select an "improper" public key. On the other hand, every "unopened" VRF value (i.e., the one for which no proof was given yet) should look indistinguishable from random, even if

[1] A similar notion of *invariant signatures* was suggested earlier by Goldwasser and Ostrovsky [10] in the context of constructing non-interactive zero-knowledge proofs. However, there is a crucial difference which we will address later.

many other values were "opened" (by giving the proof). Micali et al. [16] also gave a secure construction of a VRF based on the RSA assumption. Since then, several more efficient constructions of VRFs have been proposed based on various cryptographic assumptions; see [14, 5, 7].

The notion of a VRP, which we introduce in this paper, adds verifiability to PRPs, in exactly the same natural way as VRFs do to PRFs. We will describe an application of VRPs later, but now let us see the relation between VRFs and VRPs. On the one hand, it is again very easy to see that a VRP (on a "non-trivial domain") is also a VRF, just like in the PRP/PRF case. On a first look, we might hope that the converse implication holds as well, by simply applying the Luby-Rackoff result to VRFs in place of PRFs. However, a moment of reflection shows that this is not the case. Indeed, the proof for the iterated Feistel construction must include all the VRF values for the intermediate rounds, together with their proofs. Thus, the attacker can legally obtain all the intermediate round values, for every input/output except for the one on which he is being challenged. And the technique of Luby-Rackoff [13] (as well as all of the subsequent papers in the PRF→PRP domain; e.g. [15, 18]) crucially relies on the secrecy of such values. In fact, we will show that even up to a *logarithmic* number (in the security parameter) of rounds is *never enough* to get a VRP via an iterated Feistel construction, which is very different from the PRF→PRP situation! Therefore, a fundamentally new proof technique is needed to build VRPs out of VRFs.

Our Main Result. We show that, although logarithmic number of Feistel rounds is not enough to build VRP from a VRF, *any super-logarithmic number of rounds works*. More precisely,

Theorem 1.1 *A k-round Feistel Transform applied to $k = \omega(\log(\lambda))$ (for security parameter λ) independent VRFs yields a secure VRP. In particular, against any VRP adversary that makes at most $q = \mathcal{O}\left(\mathsf{Fibonacci}\left(\frac{k}{4}\right)\right)$ queries, the exact security of the k-round Feistel Transform with independent VRFs in each round is*

$$\mathcal{O}\left(qk^2 \cdot \max\left\{\epsilon_f, \frac{(q \cdot k)^4}{2^n}\right\}\right).$$

Here ϵ_f denotes the exact security of the VRFs used in the construction.

The main idea behind the proof is to argue that, by making at most $q = \mathcal{O}\left(\mathsf{Fibonacci}\left(\frac{k}{4}\right)\right)$ forward or inverse VRP queries and observing the intermediate results, any adversary has only a negligible chance of causing a collision on the intermediate "right" values appearing in round $k/2$. Once argued, all the values queried by the adversary, including the challenge value, will likely have distinct values $R_{k/2}$,[2] from which point the corresponding VRP output (or input in case of the inverse query) value should be random by the pseudorandomness of VRFs (which, at level $k/2$, is applied to a "fresh" value). The actual proof that $R_{k/2}$ is likely to be always distinct is then done using a delicate inductive argument, which forms the main technical part of our result. Part of the difficulty also comes from the fact that, unlike PRFs, we cannot "in one shot" substitute a VRF by a truly random function (since we would not be able to provide proofs for any of the points). In fact, if a VRF proof is given for some point, which *is the case for all the intermediate results*, we cannot replace VRF output by random *at all*. The way we solve this technical problem is to settle for a weaker combinatorial condition satisfied by VRFs: it is hard to find a constant (in our case, 4 is enough) number of inputs, whose outputs are *XOR-resistant* (see section 3). In particular, our main technical lemma 3.1 shows that this purely combinatorial condition on the round function is enough to prevent the adversary from causing collisions at level $k/2$.

Application: Building NIZK proofs. Non-interactive zero-knowledge (NIZK) proofs [2] allow the prover, who has a witness w for some NP statement x, to prove that x is true by sending a single message to

[2]This is somewhat akin the Luby-Rackoff argument that all the "right" values of the first and next-to-last rounds are distinct in the PRF/PRP case.

a polynomially bounded verifier. This is achieved in the common reference string model, which is assumed to be honestly and randomly generated, and is available to the prover and the verifier. Feige et al [8] showed that NIZK proofs exist for all NP statements in the *hidden bit string* (HBS) model, and then showed how to implement the HBS model using trapdoor permutations. Goldwasser and Ostrovsky [10] also implemented the HBS model using what they call *invariant signatures* (at the time, they left open the question of constructing such signatures). In modern terminology, invariant signatures are essentially VRFs but with one subtle but crucial addition. Namely, the security of VRFs against the prover only assumes that the VRF must induce a (pseudo)random output distribution, when applied to a truly random input, *even if the secret key owner tries to select an improper public key.* Let us call this extra property *"balancedness"*.

While seemingly a minor property, we show that balancedness is crucial in the implementation of HBS model in [10]. Namely

(a) Plain VRFs do not have to satisfy this property (and, as far as we can see, there is no trivial way to enforce it in VRFs; although, our results will imply a non-trivial easy to enforce it).

(b) More severely, there exist secure (and, of course, unbalanced) VRFs for which the transformation of [10] is completely insecure.

To briefly see point (a), imagine adding a new special public key PK^* to any secure VRF, for which the VRF is defined to be identically zero. It is clear that this again yields a VRF, since the prover is still committed to a unique function, even for the key PK^*. (And pseudorandomness holds, since the chances PK^* will be selected are negligible.) Yet, the new VRF is obviously unbalanced. In fact, if we use this new VRF in place of the invariant signature in the construction of [10], we will get a completely insecure HBS system (thus, showing (b)). Briefly, in the construction of [10] a VRF *selected by the prover* is applied to a bunch or random points to define the "hidden random string" (for which the prover can selectively open some part later). If the prover chooses PK^* as his public key, then the hidden random string is all zero as well, and it is easy to see that NIZK construction of [8] will completely fail with such non-random HRS (if it didn't, we could do NIZK without CRS).

On the positive side it is easy to see that balanced VRFs, and thus VRPs, trivially work in the HBS model implementation of [10]. Thus by our main result, this also means that

Theorem 1.2 *The existence of VRFs implies the existence of NIZK proofs in the common random string model.*

As one final note, we recall that Micali, Rabin and Vadhan also showed how to construct VRFs from an even weaker primitive of Verifiable *Unpredictable* Functions (VUFs). This means that one can construct VRPs and NIZK proofs from VUFs as well.

Related Work. We have already surveyed most of the related work with respect to VRFs and the Feistel transform. In terms of NIZK we mention that Canetti et al. [4] improved upon the result of Feige et al. [8] and showed that a somewhat weaker primitive than trapdoor permutation, namely *publicly verifiable trapdoor predicate*, suffices to implement the HBS model and thus give NIZK proofs for any NP statement. It is easy to see that such predicates can be constructed from *balanced* VRFs (see above), and thus from VRPs. However, a straightforward construction from *general* VRFs is not known.

Finally, we mention a recent work by the authors [6] about showing the equivalence of the ideal cipher (IC) and the random oracle (RO) models against "honest-but-curious" adversaries. The implication from RO to the IC model also used super-logarithmic number of Feistel rounds. However, the actual settings and proof techniques are quite different. For example, in the IC/RO setting one has to build a simulator, while here we need to do a reductionist proof. Also, in the former setting we conjecture that even 6 Feistel

rounds suffice to get the implication, but in this case super-logarithmic number of rounds is shown to be optimal. Next, in the IC/RO setting we are dealing with truly random functions, that simplifies lots of details of the proof, while in our setting we already mentioned that we cannot replace a VRF by a truly random function, and must rely entirely on combinatorics. Finally, in the IC/RO setting all values are public and there is no secret key, while here only the revealed values are public. To summarize, while some similarities exist (after all, we are applying the Feistel transform in both cases), the actual details are quite different.

2 Definitions and Preliminaries

VERIFIABLE RANDOM FUNCTIONS Let us start by briefly recalling the notion of VRFs. A VRF $f_{(\cdot)} : \{0,1\}^{\ell(\lambda)} \to \{0,1\}^{m(\lambda)}$, with security parameter λ, consists of three algorithms

- A probabilistic *function generator* G that receives as input 1^λ, and outputs a public/secret key pair (PK, SK).
- A deterministic *function evaluator* $F = (F_1, F_2)$, that receives the secret key SK and VRF input x and outputs two binary strings (the function value $F_1(SK, x) = f_{SK}(x)$ and the proof $F_2(SK, x) = proof_{SK}(x)$.
- A deterministic *function verifier* V that takes as input four bit strings $(PK, x, v, proof)$ and outputs YES/NO.

The input and output length functions (resp. $\ell(\cdot)$ and $m(\cdot)$) are both polynomial time computable functions. We will represent by $negl(\lambda)$ any negligible function of λ. The algorithms (G, F, V) should satisfy three main properties:

- **Complete Provability** For all $(PK, SK) \leftarrow G(1^\lambda)$, $x \in \{0,1\}^{\ell(\lambda)}$ and $(y, proof) \leftarrow F(SK, x)$, it holds that $V(PK, x, y, proof) = YES$
- **Unique Provability** For any $PK, x, y_1, y_2, proof_1$ and $proof_2$ such that $y_1 \neq y_2$, for either $i = 1$ *or* 2, it holds that $V(PK, x, y_i, proof_i) = NO$
- **Pseudorandomness** For any probabilistic polynomial-time oracle machine $A = (A_1, A_2)$ that does not query its oracle on x,

$$Pr\left[b = b' \;\middle|\; \begin{array}{c}(PK, SK) \leftarrow G(1^\lambda); (x, \tau) \leftarrow A_1^{F(SK,\cdot)}(PK); y_0 \leftarrow F_1(SK, x); \\ y_1 = \{0,1\}^{m(\lambda)}; b \leftarrow \{0,1\}; b' \leftarrow A_2^{F(SK,\cdot)}(y_b, \tau)\end{array}\right] < \frac{1}{2} + negl(\lambda)$$

Note that, the *unique provability* property requires that an incorrect proof should not verify correctly for any public key PK, and not just the one corresponding to the secret key SK that is used to generate the correct proof.

VERIFIABLE RANDOM PERMUTATIONS Along the same lines as VRFs, we can also define the notion of *verifiable random permutations*. Similar to VRFs, a VRP π also has a public key PK and a secret key SK. Given the secret key SK it is easy to evaluate the forward/inverse permutation (π/π^{-1}), as well as give proofs for either direction. However, given only the public key PK one can verify the proof of correctness for an input/output pair. Formally speaking, a VRP $\pi_{(\cdot)} : \{0,1\}^{n(\lambda)} \to \{0,1\}^{n(\lambda)}$ consists of three algorithms (below λ is the security parameter),

- A probabilistic *permutation generator* G_π that on receiving 1^λ outputs a public/secret key pair (PK, SK).
- A deterministic *permutation evaluator* $\Pi = (\Pi_1, \Pi_2)$ that takes as input a bit inv and two binary strings, the secret key SK and the VRP input x and outputs the forward/inverse permutation value $\Pi_1(inv, SK, x)$ and a proof $\Pi_2(inv, SK, x)$. If $inv = 0$ then $\Pi_1(inv, SK, x) = \pi_{SK}(x)$, else $\Pi_1(inv, SK, x) = \pi^{-1}_{SK}(x)$.
- A deterministic *permutation verifier* V_π that takes as inputs PK, x, y and $proof$, and outputs YES/NO.

The input/output length $n(\lambda)$ is a polynomial-time computable function of the security parameter λ. And as before, the algorithms (G_π, Π, V_π) should satisfy three properties:

- **Complete Provability** For all $(PK, SK) \leftarrow G_\pi(1^\lambda)$ and $x \in \{0,1\}^{n(\lambda)}$, if $(y, proof) \leftarrow \Pi(0, SK, x)$ or $(x, proof) \leftarrow \Pi(1, SK, y)$, then $V_\pi(PK, x, y, proof) = YES$
- **Unique Provability** For any $PK, x, y_1, y_2, proof_1$ and $proof_2$ such that $y_1 \neq y_2$, for either $i = 1$ *or* 2, it holds that $V_\pi(PK, x, y_i, proof_i) = NO$. And for the inverse VRP, we have that for any $PK, x_1, x_2, y, proof_1$ and $proof_2$ such that $x_1 \neq x_2$, for either $i = 1$ *or* 2, it holds that $V_\pi(PK, x_i, y, proof_i) = NO$.
- **Pseudorandomness** For any probabilistic polynomial-time oracle machine $A = (A_1, A_2)$ that does not make a forward/inverse query on the challenge input (i.e. $\pi(x)$ or $\pi^{-1}(y_b)$),

$$Pr\left[b = b' \;\middle|\; \begin{array}{c} (PK, SK) \leftarrow G_\pi(1^\lambda); (inv', x, \tau) \leftarrow A_1^{\Pi(\cdot,SK,\cdot)}(PK); \\ y_0 \leftarrow \Pi_1(inv', SK, x); y_1 = \{0,1\}^{n(\lambda)}; b \leftarrow \{0,1\}; b' \leftarrow A_2^{\Pi(\cdot,SK,\cdot)}(y_b, \tau) \end{array}\right] < \frac{1}{2} + negl(\lambda)$$

2.1 Luby Rackoff Construction

Let $f : \{0,1\}^n \rightarrow \{0,1\}^n$ be a function from n bits to n bits. The *Feistel permutation* using f is a permutation Ψ_f on $2n$ bits defined as, $\Psi_f(x) \stackrel{def}{=} x_R \parallel x_L \oplus f(x_R)$. The symbols x_L and x_R denote the left and right halves of the $2n$ bit string x, while $\parallel$ and $\oplus$ denote the concatenation and bit-by-bit XOR of two binary strings, respectively. Note that the Feistel permutation Ψ_f is easily invertible (indeed, $\Psi_f^{-1}(y) = y_R \oplus f(y_L) \parallel y_L$). We will call a construction with k iterated applications of the Feistel permutation, a k round LR construction.

A k round LR construction $\Psi_{f_1 \ldots f_k}$ takes inputs of the form (b, x), where b is a bit and x is a binary string. The bit b indicates if the query is a forward ($b = 0$) or an inverse ($b = 1$) query

3 Matrix Representation of LR construction

In this section, we will prove that if the round functions $f_1 \ldots f_k$ ($f_i : \{0,1\}^n \rightarrow \{0,1\}^n$) used in the k round LR construction $\Psi_{f_1 \ldots f_k}$ are *"XOR resistant"* (in a sense which we will define below) then $\Psi_{f_1 \ldots f_k}$ satisfies a nice combinatorial property which will prove useful to us in finding secure constructions of VRPs from VRFs. We will often call the k round LR construction Ψ_k and the round functions will be understood from the context.

Consider a sequence of m inputs provided to Ψ_k, $(b_1, x_1), \ldots, (b_m, x_m)$ (each of which may be adaptively chosen). As described above, each of these inputs will be divided into n bit halves and then k rounds of the

Feistel permutation are applied to it. Let us denote the round values computed for the i^{th} input, (b_i, x_i), as $R_0^i, R_1^i \ldots R_j^i \ldots R_k^i, R_{k+1}^i$. If $b = 0$, then $R_0^i \parallel R_1^i = x_i$ otherwise $R_k^i \parallel R_{k+1}^i = x_i$. We store the round values generated in the computation of Ψ_k in an $m \times (k+2)$ matrix, Φ, where the i^{th} row corresponds to the i^{th} input, (b_i, x_i), and the columns represent the round numbers. We index the columns as $0 \ldots k+1$, corresponding to the convention we use for round values. Thus, round value R_j^i is stored in $\Phi[i, j]$. This matrix Φ is illustrated below,

$$\Phi = \begin{bmatrix} R_0^1 & R_1^1 & \ldots & R_k^1 & R_{k+1}^1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ R_0^m & R_1^m & \ldots & R_k^m & R_{k+1}^m \end{bmatrix} \begin{matrix} (b_1, x_1) \\ \vdots \\ (b_m, x_m) \end{matrix}$$

We also maintain a m-vector B in which we store each of the bits b_i that indicates whether the i^{th} input is a forward or an inverse one, that is $B[i] = b_i$. Together, the matrix Φ and the vector B represent the entire computation performed by the LR construction Ψ_k on the m inputs it is provided. Indeed, the entry $B[i]$ indicates whether the i^{th} query was a forward query or an inverse query. If $B[i] = 0$ then the round values $\Phi[i, 0],\ \Phi[i, 1],\ \Phi[i, 2] \ldots \Phi[i, k+1]$ were computed in this order, otherwise these values were computed in the order $\Phi[i, k+1],\ \Phi[i, k], \ldots, \Phi[i, 0]$. For the remainder of this section, we will use the $m \times (k+2)$ matrix Φ and m vector B to represent the computation of the LR construction Ψ_k.

c-XOR RESISTANT ROUND FUNCTIONS We call a round function f_j used in the LR construction Ψ_k a **c**-*XOR resistant round function* if for any sequence of m queries made to Ψ_k and any new round value $\Phi[i, j]$ generated while responding to these queries, the round function value $f_j(\Phi[i, j])$ is not an XOR of upto **c** previously defined round values. A round value $\Phi[i', j']$ is said to be defined before $\Phi[i, j]$ if,

- $\underline{(\mathbf{i'} < \mathbf{i})}$:- Input number i' was processed before the i^{th} input.
- $\underline{(\mathbf{B[i]} = \mathbf{0}) \wedge (\mathbf{i'} = \mathbf{i}) \wedge (\mathbf{j'} < \mathbf{j})}$:- $\Phi[i', j']$ and $\Phi[i, j]$ are both part of the same forward query i, but $\Phi[i', j']$ comes before $\Phi[i, j]$ (i.e. $j' < j$).
- $\underline{(\mathbf{B[i]} = \mathbf{1}) \wedge (\mathbf{i'} = \mathbf{i}) \wedge (\mathbf{j'} > \mathbf{j})}$:- $\Phi[i', j']$ and $\Phi[i, j]$ are both part of the same inverse query i, but $\Phi[i', j']$ comes before $\Phi[i, j]$ (i.e. $j' > j$).

If f_j is a **c**-XOR resistant round function, then the round function value assigned to a new round value $\Phi[i, j]$ is not equal to an XOR of upto **c** round values of the above types. As an example, if f_j is a truly random function then it is **c**-XOR resistant for any constant **c** with high probability, if the number of inputs m is polynomial.

THE COMBINATORIAL LEMMA Now we will go on to prove that if all the round functions used in the k round LR construction Ψ_k are 4-XOR resistant, then for a polynomial (in k) number of inputs (even adaptively chosen) there can be no two inputs such that their $\left(\frac{k}{2}\right)^{th}$ round values collide. We formally state this lemma below:

Lemma 3.1 *Let Ψ_k be a k round LR construction, that uses round functions $f_1 \ldots f_k : \{0,1\}^n \to \{0,1\}^n$ each of which is 4-XOR resistant. And let the $m \times (k+2)$ matrix Φ and m vector B denote the computation of Ψ_k on a sequence of m different inputs provided to it. Then,*

$$\left(\exists i_1, i_2 \in \{1 \ldots m\} \ : \ (i_1 \neq i_2) \wedge \left(\Phi\left[i_1, \frac{k}{2}\right] = \Phi\left[i_2, \frac{k}{2}\right]\right)\right) \Longrightarrow \left(m \geq \mathsf{Fibonacci}\left(\frac{k}{4}\right)\right)$$

Here n, k are both polynomial in the security parameter λ. And $\mathsf{Fibonacci}(i)$ denotes the i^{th} Fibonacci number, which for large i can be approximated as $\mathsf{Fibonacci}(i) \cong \Theta((\sqrt{5}+1)/2)^i)$

Note that lemma 3.1 is deterministic and implies *impossibility* of a collision between two $\left(\frac{k}{2}\right)^{th}$ round values. We give an informal intuition for the result here, and formal proof in appendix A.

Proof Intuition: As we mentioned above, the lemma under consideration is purely combinatorial and deterministic in nature. The proof of this lemma involves a complicated inductive argument. The combinatorial structure upon which the induction is carried out is the matrix representation Φ of the inputs given to the k round LR construction Ψ. The statement of the lemma says that if two different rows of such a matrix representation Φ collide in the $\left(\frac{k}{2}\right)^{th}$ round value, then either the round functions used in the underlying LR construction are not 4-XOR resistant, or the number of rows of Φ, i.e. m, is exponential.

The proof considers any such matrix representation Φ, which consists of two rows i and i' that are different but collide in the $\left(\frac{k}{2}\right)^{th}$ round value. And since we cannot say anything about inputs i' to m, in the worst case, we assume that $i' = m$, i.e. the collision involves the last row of Φ. Next we assume that Φ does not consist of two identical rows. Note that if two such rows $i_1, i_2 (> i_1)$ do exist, then we can ignore row number i_2 without loss of generality. This is because all useful computation for the input corresponding to these rows is performed in row number i_1 (which comes before i_2). Thus the matrix Φ is assumed to be such that for some row number i, $\Phi\left[i, \frac{k}{2}\right] = \Phi\left[m, \frac{k}{2}\right]$, and no two rows of Φ are identical.

Note that every input in Φ is either computed in either the forward or the inverse direction, i.e. starting with $\Phi[i,0], \Phi[i,1]$ and going to $\Phi[i,k], \Phi[i,k+1]$ or the other way round. Our argument counts the number of inputs that were necessary in order to get the collision mentioned above. Thus it will change depending on whether a particular input was computed in a forward or inverse fashion. However, for simplicity of illustration, we assume at each step of the induction the input under consideration to be a forward input. As it turns out this will also be the worst case for our argument, i.e. the case where we get the least number of inputs in Φ.

Corresponding to each round value $\Phi[i,j]$, we define a "first occurrence" input number which we represent as a function $\mathsf{p}(i,j)$. This input number is such that $\Phi[i,j] = \Phi[\mathsf{p}(i,j), j]$, and $\Phi[\mathsf{p}(i,j), j] \neq \Phi[i', j]$ for any $i' < \mathsf{p}(i,j)$. That is, the round value $\Phi[i,j]$ *occurs for the first time* in input number $\mathsf{p}(i,j)$.

We first show that in order for a collision to have occurred between $\Phi[m,k]$ and $\Phi[i,k]$ it must have been the case that all the round values $\Phi[m,1] \ldots \Phi[m, \frac{k}{2}-1]$ were defined already in some input before the m^{th} input, that is $\mathsf{p}(m,j) < m$ for all $j = 1 \ldots \frac{k}{2}$. This fact is formally proven in claim A.1.

Next we analyze each of these "first occurrence" inputs, i.e. the inputs $\mathsf{p}(m,j)$ for $j = 1 \ldots \frac{k}{2}$, and try to find an order in which these inputs could have occurred. We show that these inputs could have occurred only in one of very few special orders. In each of these special orders, there is a $j \in \left\{1, \frac{k}{4}\right\}$ such that,

$$\mathsf{p}(m,1) > \ldots > \mathsf{p}(m,(j-1)) > \mathsf{p}(\mathbf{m},\mathbf{j}) < \ldots < \mathsf{p}\left(m, \frac{k}{2}-1\right) < \mathsf{p}\left(m, \frac{k}{2}\right)$$

That is, the sequence of inputs $\mathsf{p}(m,1), \ldots, \mathsf{p}\left(m, \frac{k}{2}\right)$ can be divided into exactly two parts such that the left part of this sequence of inputs is in strictly decreasing temporal order (i.e. input $\mathsf{p}(m,1)$ comes after $\mathsf{p}(m,2)$ and so on until $\mathsf{p}(m,j)$), while the right part is in strictly increasing temporal order. This result is stated in claim A.2.

The next step can be thought of as the inductive step of our argument. We consider an arbitrary input (say input number i) and consider three consecutive round values of this input which were all defined prior to this input. That is, we consider round values $\Phi[i,j]$, $\Phi[i,j+1]$ and $\Phi[i,j+2]$, where it is the case that $\mathsf{p}(i,\ell)_{\ell=j,\ j+1,\ j+2} < i$. For simplicity, we assume that inputs i, $\mathsf{p}(i,\ell)_{\ell=j,\ j+1,\ j+2}$ are all forward inputs. In fact, a similar argument can be carried out irrespective of whether these are forward or inverse inputs.

For such a triple of "first occurrence" inputs, we show that if $\mathsf{p}(i,j) > \mathsf{p}(i,j+1) > \mathsf{p}(i,j+2)$ then there are at least $j-2$ inputs that lie strictly between inputs $\mathsf{p}(i,j)$ and $\mathsf{p}(i,j+1)$. These are the "first occurrence" inputs for the round values $\Phi[\mathsf{p}(i,j), \ell]_{\ell=1 \ldots j-2}$. In fact, we also show that these inputs are

also in strictly decreasing temporal order. Thus setting $i_j = \mathsf{p}(i, j)$ and $i_{j+1} = \mathsf{p}(i, j+1)$, we show that $i_j > \mathsf{p}(i_j, 1) > \ldots > \mathsf{p}(i_j, j-2) > i_{j+1}$.

Hence, we show that the inputs $\mathsf{p}(i_j, 1), \ldots, \mathsf{p}(i_j, j-2)$ not only occur in a strictly decreasing temporal order, but these inputs are also "sandwiched" between the inputs i_j and i_{j+1}. This result is stated and proven formally as claim A.3. We also note that claim A.3 works even for a strictly increasing ordering of the inputs $\mathsf{p}(i, \ell)_{\ell=j,\ j+1,\ j+2}$, except that we count the "first occurrence" inputs for the round values of $\mathsf{p}(i, j+2)$ instead of $\mathsf{p}(i, j)$.

The next and last step in the argument is to apply claim A.3 to the inputs $\mathsf{p}(m, \ell)_{\ell=1,\ldots,\frac{k}{2}}$. However, we only consider one part of this input sequence, i.e. either the left (descending temporally) or the right (ascending temporally). Note that since there are only two such parts, one of them must necessarily contain half of the sequence, or $\frac{k}{4}$ inputs. We take this part and apply claim A.3 to this part. Without loss of generality, say it is the left part which contains $i \geq \frac{k}{4}$ inputs. Clearly, we can divide the inputs $\mathsf{p}(m, \ell)_{\ell=1,\ldots,i}$ into $(i-2)$ such consecutive triples $(\mathsf{p}(m,1), \mathsf{p}(m,2), \mathsf{p}(m,3)), \ldots, (\mathsf{p}(m,(i-2)), \mathsf{p}(m,(i-1)), \mathsf{p}(m,i))$. After having done this, we can apply claim A.3 to each of these triples, and then recursively to each of the further sequence of inputs we get from claim A.3 (which are also in strict temporal order as well as "sandwiched" between the sequence of inputs $\mathsf{p}(m, \ell)_{\ell=1,\ldots,i}$.

Continuing in this fashion, we can effectively count a Fibonacci tree of distinct (because of the "sandwiching" property) inputs in the matrix representation Φ. And thus we can deduce that the matrix Φ consists of at least $\mathcal{O}\left(\mathsf{Fibonacci}\left(\frac{k}{4}\right)\right)$ rows, and lemma 3.1 follows. □

Lemma 3.1 states a purely combinatorial result about the iterated Feistel construction with $\omega(\log(n))$ rounds. Note that we do not impose any requirements on the round functions employed in the construction apart from the fact that they should avoid all XOR dependencies (as described in the statement of lemma 3.1). The proof strategy employed here is similar to the one used in the proof of a similar result in [6]. The main difference between the two is that in [6] the round functions used in the Feistel transform are ideal random functions, while here they only need to be 4-XOR resistant. It is not hard to see that an ideal random function is c-XOR resistant for any constant c (and in particular, for $c = 4$). Thus the result presented here is strictly stronger than that in [6]. And as we already argued, we crucially need this strengthening to get our final result.

4 A secure VRP construction

The problem of constructing VRPs using VRFs is analogous to that of constructing *pseudorandom permutations* using *pseudorandom functions*. However, as mentioned in the introduction, this problem is much harder than the PRF to PRP problem. In particular, we will show in section 5 that the 4 round LR construction used by [13] does not work in constructing VRPs from VRFs. We will show in this section that if we use $\omega(\log \lambda)$ (for security parameter λ) rounds in the LR construction then we can overcome this problem and the resulting construction (using VRFs as round functions) is a secure VRP construction.
Thus, assume that we are given k independent *verifiable random functions* $\{f_i = (G_i, F_i, V_i)\}_{1\ldots k}$ [3]. Our construction of a *verifiable random permutation* is the k round LR construction $\Psi_{f_1 \ldots f_k}$, which we will refer to as $\pi^{LR}_{(.)} : \{0,1\}^{2n} \rightarrow \{0,1\}^{2n}$. A formal specification of the construction is given below.

Definition 1 (The VRP construction π^{LR}) *Let $\{f_i = (G_i, F_i, V_i)\}_{1\ldots k}$ be k independent* verifiable random functions *defined on $\{0,1\}^n \rightarrow \{0,1\}^n$. The VRP construction $\pi^{LR} = (G_\pi, \Pi, V_\pi)$ using $f_1 \ldots f_k$ is a*

[3] In fact, for small k, it is possible to get k round functions from a single VRF by dividing the key space appropriately

permutation on $\{0,1\}^{2n}$ *defined as follows:*

- *The* **permutation generator** $G_\pi(1^\lambda)$ *runs the* VRF generators *to get* $(PK_i, SK_i) \leftarrow G_i(1^\lambda)$*, and outputs*

$$(PK_\pi, SK_\pi) = ((PK_1, \ldots, PK_k), (SK_1, \ldots, SK_k))$$

- *The* **permutation evaluator** $\Pi = (\Pi_1, \Pi_2)$ *takes as input a bit* inv*, secret key* $SK_\pi = (SK_1, \ldots, SK_k)$ *and the VRP input* x *and computes,*

$$\Pi_1(inv, SK_\pi, x) = \begin{cases} \Psi_{F_1(SK_1,\cdot),\ldots,F_k(SK_k,\cdot)}(x) & \text{if } inv = 0 \\ \Psi^{-1}_{F_1(SK_1,\cdot),\ldots,F_k(SK_k,\cdot)}(x) & \text{if } inv = 1 \end{cases}$$

Let $R_0, \ldots, R_{k+1}$ *denote the round values in the LR construction* $\Psi_{F_1(SK_1,\cdot),\ldots,F_k(SK_k,\cdot)}$ *on the input* x*. Then* $\Pi_2(inv, SK_\pi, x)$ *essentially outputs all the pairs* $(R_i, proof_{SK_i}(R_i))$*, where* $proof_{SK_i}(R_i)$ *is the proof provided by the VRF* f_i *for the input* R_i*. Hence*

$$\Pi_2(inv, SK_\pi, x) = \left(R_1, proof_{SK_1}(R_1)\right), \ldots, \left(R_k, proof_{SK_k}(R_k)\right)$$

- *The* **permutation verifier** V_π *gets as input the VRP public key* $PK_\pi = (PK_1, \ldots, PK_k)$*, input* x*, output* y *and* $proof_\pi(x)$*. It first checks whether* $proof_\pi(x)$ *is of the form* $(R_1, proof_1), \ldots, (R_k, proof_k)$*. If the proof is of this form then* V_π *checks if* $R_1 = x_{_R}$ *and* $R_k = y_{_L}$*. If so, then it performs the following verifications:*

$$\forall i \in \{1, k\} \ : \ (V_{f_i}(PK_i, R_i, R_{i+1} \oplus R_{i-1}, proof_i) = YES)$$

If everything goes well, then it outputs YES *otherwise it outputs* NO*. Here the round values* R_0 *and* R_{k+1} *are taken to be* $x_{_L}$ *and* $y_{_R}$*, respectively.*

The input length n *is a polynomial function of the security parameter* λ*.*

4.1 Proof of Security

Recall, a VRP construction needs to satisfy three security properties : *Completeness*, *Soundness* (or unique proofs) and *Pseudorandomness*. Completeness of the construction π^{LR} is a direct consequence of completeness of each of the VRFs used as round functions in the LR construction.

SOUNDNESS In order to prove the soundness of this construction, consider any two "claimed" outputs y and y' for an input $x \in \{0,1\}^{2n}$ and the corresponding proofs $proof_\pi$ and $proof'_\pi$. And say there exists a public key $PK_\pi = (PK_1, \ldots, PK_k)$ such that

$$(V_\pi(PK_\pi, x, y, proof_\pi) = YES) \wedge \left(V_\pi(PK_\pi, x, y', proof'_\pi) = YES\right)$$

Thus the proofs are of the form $proof_\pi = [(R_i, proof_i)]_{i\in\{1,k\}}$ and $proof_\pi = [(R'_i, proof'_i)]_{i\in\{1,k\}}$. We also know that the initial round values in these proofs match (i.e. $(R_0 \parallel R_1) = (R'_0 \parallel R'_1) = x$), while the final round values differ (since $y \neq y'$). We can deduce that there exists $i \in 1, \ldots, k$, such that

$$(R_i = R'_i) \wedge (R_{i-1} \oplus R_{i+1} = R'_{i-1} \oplus R'_{i+1})$$

And since both $proof_\pi$ and $proof'_\pi$ verify correctly using public key $PK_\pi = (PK_1 \ldots PK_k)$, we can also deduce that,

$$\left(V_{f_i}\left(PK_i, R_i, R_{i-1} \oplus R_{i+1}, proof_i\right) = YES\right) \wedge \left(V_{f_i}\left(PK_i, R'_i, R'_{i-1} \oplus R'_{i+1}, proof'_i\right) = YES\right)$$

This is a contradiction, since all the VRFs f_i are sound. Similar argument works for the inverse query case as well. Thus, π^{LR} satisfies the soundness condition for VRPs.

PSEUDORANDOMNESS Now we will prove that when π^{LR} is constructed using $k = \omega(\log \lambda)$ round LR construction, then the construction π^{LR} satisfies the pseudorandomness property for VRPs (see section 2).

Theorem 4.1 *Let $\pi^{LR} = (G_\pi, \Pi, V_\pi)$ be the VRP construction using k Feistel rounds. Then for any probabilistic polynomial time oracle machine $A = (A_1, A_2)$ that does not query its oracle on x or try to invert the response to the challenge query and makes at most $q = \mathcal{O}\left(\mathsf{Fibonacci}\left(\frac{k}{4}\right)\right)$ queries,*

$$\left| Pr\left[b = b' \,\middle|\, (PK_\pi, SK_\pi) \leftarrow G_\pi(1^\lambda); \ldots; b \xleftarrow{\$} \{0,1\}; b' \leftarrow A_2^{\Pi}(\ldots)\right] - \frac{1}{2}\right| < \mathcal{O}\left(qk^2 \cdot \max\left\{\epsilon_f, \frac{(q \cdot k)^4}{2^n}\right\}\right)$$

Here ϵ_f denotes the exact security of the VRFs used.

Note that theorem 4.1 says that the above mentioned exact security holds only if the number of queries made by the VRP adversary is $\mathcal{O}(\mathsf{Fibonacci}(k/4))$. And this makes sense only when $k = \omega(\log \lambda)$, since otherwise $\mathsf{Fibonacci}(k/4)$ will only be polynomial in the security parameter. In fact, as we show in the next section, this bound is "almost" tight in the sense that there exists an adversary that succeeds while making only a $\mathsf{Fibonacci}(k)$ number of queries.

We have divided the proof of theorem 4.1 in three parts:

1. We will first show that, for a polynomial number of queries, a VRF $f = (G_f, F, V_f)$ defined on $\{0,1\}^n$ is c-XOR resistant (for any constant c) with high probability when used in the Feistel-based VRP construction π^{LR}.

2. Then, we will go on to show that if a VRP adversary A_π succeeds in the pseudorandomness game (theorem 4.1) with the construction π^{LR} with noticeable probability, then we can construct another VRP adversary A'_π that succeeds also with noticeable probability but does so only if all the VRFs used in π^{LR} remain 4-XOR resistant.

3. Next we will prove that if the VRP adversary A'_π constructed above succeeds with noticeable probability, then we can construct a VRF adversary A_f that succeeds in the VRF pseudorandomness game (see section 2), involving one of the round functions of π^{LR}, with noticeable probability as well.

VRFS ARE XOR RESISTANT Consider the VRP construction π^{LR}, based on the k round LR construction with independent VRFs $\{f_j = (G^j, F^j, V^j)\}_{j \in \{1,k\}}$ as round functions. Recall that we refer to the intermediate values $\{R_j\}_{j \in \{0,k+1\}}$ as round values, and the corresponding VRF outputs $\{F_1^j(SK_j, R_j)\}_{j \in \{1,k\}}$ are called the round function values. Now suppose there exists a probabilistic oracle Turing machine A_{xor} with oracle access to to π^{LR}, that in the process of making queries to π^{LR} forces a new round function (VRF) value to collide with an XOR of c previously defined round values (R_j) with non-negligible probability (which we also call the advantage of the adversary A_{xor} in the c XOR attack game). Then using A_{xor} as a subroutine, we can design another adversary A_f that has a non-negligible advantage in the pseudorandomness game with one of the VRFs used in π^{LR}.

The reason why this holds is that for any new input provided to a verifiable random function, the VRF output should look completely unpredictable to any adversary that does not hold the secret key for the VRF. In the context of the construction π^{LR}, the output of a VRF used in π^{LR} on a new input is a new round function value. Now say this round function value can be represented as an XOR of previously existing round values, which is what A_{xor} does with non-negligible probability. Then A_{xor} can easily make a pretty good prediction for this new round function value, since there are only a polynomial number of

such XORs of constant round values when the number of queries made by A_{xor} is polynomial. However, this contradicts the security of the VRF for which such an XOR representation exists. A formal description of this reduction is given in appendix B. We simply state the result here.

Claim 4.2 *If there is an adversary A_{xor} that has a non-negligible advantage ϵ_{xor} in the c-XOR attack game described above, then there also exists a VRF adversary A_f that has a non-negligible advantage ϵ_f in differentiating a VRF output from a random binary string, in the VRF pseudorandomness game. In particular, $\epsilon_f \geq \left(\frac{\epsilon_{xor}}{2qk} - \frac{(qk)^c}{2^{n+1}}\right)$.*

We can interpret claim 4.2 in reverse and deduce that if the maximum advantage of a VRF adversary is ϵ_f, then the advantage of any adversary A_{xor} in the c-XOR attack game is at most $qk \cdot (2\epsilon_f + ((qk)^c)/(2^n))$.

MAKING THE VRP ADVERSARY "XOR-FREE" Now we will show that if a VRP adversary A_π has a non-negligible advantage in the pseudorandomness game with the VRP construction π^{LR}, then we can construct another VRP adversary A'_π that has a non-negligible advantage in the pseudorandomness game with π^{LR} and which attacks π^{LR} only if all the round functions of π^{LR} remain 4-XOR free.

It is intuitively clear that this should be the case, since by claim 4.2 all the VRFs used in the construction π^{LR} remain 4-XOR free with all but a negligible probability. And if a non-negligible portion of the advantage of adversary A_π depends on finding XOR representations of some new round function values then we develop an XOR adversary that violates claim 4.2. This fact is stated more formally below, while a proof for the same is in appendix C.

Claim 4.3 *If there is an adversary A_π that has a non-negligible advantage ϵ_π in the pseudorandomness game with π^{LR}, then there also exists an adversary A'_π that also has a non-negligible advantage in the pseudorandomness game with π^{LR} but that only wins when all round functions used in π^{LR} remain 4-XOR free. In particular, we have $Adv(A'_\pi) \geq \epsilon_\pi - \left(1 - (1 - \epsilon_{xor})^k\right)$, where ϵ_{xor} is the maximum advantage of an adversary in the 4-XOR attack game against any of the VRFs used in the LR construction of π^{LR}.*

"XOR-FREE" ADVERSARIES CANNOT WIN Let π^{LR} be the VRP construction that uses a k round LR construction with independent VRFs in each round. And let the number of rounds k be super-logarithmic in the security parameter λ, i.e. $k = \omega(\log(\lambda))$. Now we will prove that if there exists an "4-XOR free" VRP adversary $\mathbf{A}_\pi$ that has non-negligible advantage in the VRP pseudorandomness game against π^{LR}, then we can construct a VRF adversary $\mathbf{A_f}$ that has non-negligible advantage in the pseudorandomness game against a secure VRF f. An 4-XOR free VRP adversary is essentially one that does not force the construction π^{LR} to produce a round function value that can be represented as an XOR of 4 pre-existing round values. We state this result more formally below,

Claim 4.4 *Let A_π be an "XOR-free" VRP adversary that has advantage ϵ_π in the pseudorandomness game with the VRP construction π^{LR}, and makes at most $\mathcal{O}\left(\mathsf{Fibonacci}\left(\frac{k}{4}\right)\right)$ queries to π^{LR} (k is the number of rounds used in π^{LR}). Then there exists a VRF adversary A_f that has advantage $\epsilon_f = \epsilon_\pi$ in the pseudorandomness game with a secure VRF f.*

Understanding why this is true involves five main steps:

- The VRF adversary A_f plugs the VRF f into a "simulated" construction π^{LR} that the VRP adversary A_π is allowed to attack. But plugging f just anywhere in the LR construction does not work. We get the best results if f is used as the middle round function $f_{\frac{k}{2}}$ in π^{LR}.
- A_f enters the "challenge phase" of its attack, exactly when A_π sends its challenge query. And the challenge query of A_f is the $\left(\frac{k}{2}\right)^{th}$ round value in the challenge query made by A_π to π^{LR}.

- We need to ensure that the $\left(\frac{k}{2}\right)^{th}$ round value in the challenge query of A_π is a new one. If not then the VRF attacker A_f attack fails. We get over this problem by making crucial use of the *combinatorial lemma* 3.1. Note that we can do this since the VRP adversary makes at most $\mathcal{O}\left(\mathsf{Fibonacci}\left(\frac{k}{4}\right)\right)$ number of queries to π^{LR}.

- Another important thing to ensure is that if the challenge query for A_f is chosen to be random (instead of the VRF value), then the challenge query that A_f gives to A_π (by computing the "simulated" construction π^{LR}) is also random. But we essentially get from the properties of the iterated Feistel construction.

- We also need to show that once the challenge query response is known to A_π, it cannot force the VRF adversary A_f to provide a proof on the $\left(\frac{k}{2}\right)^{th}$ round value of this query. This is because A_f itself does not know a proof for this round value. This part again makes crucial use of the *combinatorial lemma* 3.1.

Now we are ready to prove theorem 4.1. The basic idea is to combine the above claims. Let ϵ_f denote the maximum advantage of a VRF adversary against any of the VRFs used in π^{LR}. And since we assume all the VRFs used to be secure, the ϵ_f below are negligible. We sketch the details of the proof of theorem 4.1 below.

- Say there exists a VRP adversary that has advantage ϵ_π in the pseudorandomness game against construction π^{LR}. Then we can construct an adversary A'_π that has advantage ϵ'_π in the same game and which is not able to force 4-XOR representations of any round function values in π^{LR}. This can be done using claim 4.3, and we get $\epsilon'_\pi \geq \epsilon_\pi - \left(1 - (1 - \epsilon_{xor})^k\right)$.

- Let ϵ_{xor} is the maximum advantage of an adversary in the XOR attack game against one of the VRFs used in π^{LR}. And from claim 4.2, we know this to be $\epsilon_{xor} \leq qk \cdot \left(2\epsilon_f + \frac{(qk)^4}{2^n}\right)$

- From claim 4.4, we find that using the "XOR-free" adversary A'_π can be used to construct a VRF adversary A_f that has advantage,

$$Adv(A_f) \asymp \epsilon_\pi + qk^2 \cdot \left(2\epsilon_f + \frac{(qk)^4}{2^n}\right)$$

  Thus $Adv(A_f)$ is non-negligible if ϵ_π is non-negligible, ϵ_f is negligible and q, n are polynomial in the security parameter λ. This gives us a contradiction since $Adv(A_f) \leq \epsilon_f$.

5 The construction is "tight"

We proved above that if the construction π^{LR} uses an LR construction with super-logarithmic rounds, and with independent VRFs in each round then it is a secure construction of a VRP. Can we hope to improve this result? We show that the answer is no. Namely, if the π^{LR} uses an LR construction with $\mathcal{O}(\log \lambda)$ rounds then there is a VRP adversary A_π that succeeds in the pseudorandomness game with π^{LR} with overwhelming probability.

Informally speaking, we design a VRP adversary A_π that only makes $\mathcal{O}(\mathsf{Fibonacci}(k))$ forward queries to the LR construction π^{LR} with k rounds, and generates all the round values corresponding to a new input (i.e. an input that is not queried upon). This enables A_π to find the output of π^{LR} on a particular input without querying upon it. Now it is easy to see that pseudorandomness of the construction π^{LR} is violated,

since clearly it should not be possible to invert a random permutation on a new input while making only a polynomial number (which is the case when $k = \mathcal{O}(\log(\lambda))$) of forward queries!

The basic strategy of A_π is to make *random forward queries*, and recover the input X corresponding to the desired output (say 0^{2n}). Since all the queries made by A_π are random, the probability of one of them having output 0^{2n} is exponentially small. Thus, with overwhelming probability, the challenge query X will be a new one and A_π can recognize the construction π^{LR} by comparing the output to 0^{2n}.

Theorem 5.1 *For the construction π^{LR} that uses $k = \mathcal{O}(\log \lambda)$ round LR construction, there exists a PPT oracle Turing machine A_π with a non-negligible advantage in the VRP pseudorandomness game with π^{LR}. That is, π^{LR} with $\mathcal{O}(\log \lambda)$ rounds cannot be a VRP, even with secure and independent VRFs in each round.*

Proof: The VRP adversary A_π gets oracle access to the construction π^{LR}, and on any query $x \in \{0,1\}^{2n}$ it gets all the round values generated in the computation of the LR construction used in π^{LR} (as part of the VRP proof). The attack we describe is based on the simple idea of constructing the response of π^{LR} on an input, without explicitly querying the π^{LR} oracle on this input.

We will first describe a recursive subroutine that the attacker A_π makes use of. Let us refer to the round functions (VRFs) of the construction π^{LR} as $f_1 \dots f_k$ (instead of $F_1(SK_1, x) \dots F_1(SK_k, x)$). The recursive subroutine we describe is a round function value extraction algorithm $E(j, X)$, where $j \in \{1, k\}$ and $X \in \{0,1\}^n$. The task of $E(j, X)$ is to simply find out the value $f_j(X)$ by making only forward queries to the permutation evaluator Π (for π^{LR}). We will describe this procedure recursively below:

- $\mathbf{E(1, X)}$: Choose a random $R'_0 \leftarrow \{0,1\}^n$. And make the query $(0, R'_0 \parallel X)$ to Π. This will give the value $f_1(X)$ as part of the VRP proof.
- $\mathbf{E(j, X)}$, $\mathbf{j > 1}$: Perform the following steps:
  - Choose a random $R_{j-1} \leftarrow \{0,1\}^n$, and call $E(j-1, R_{j-1})$ to get $f_{j-1}(R_{j-1})$.
  - Compute $R_{j-2} = X \oplus f_{j-1}(R_{j-1})$ and recursively call $E(j-2, R_{j-2})$ to get $f_{j-2}(R_{j-2})$.
  - Continue this recursively to get $(R_{j-3}, f_{j-3}(R_{j-3})) \dots (R_1, f_1(R_1))$.
  - Now compute $R_0 = f_1(R_1) \oplus R_2$ and query $(0, R_0 \parallel R_1)$ to the Π oracle. This will give us $f_j(X)$.

The VRP adversary A_π makes use of the above recursive procedure to find a VRP input x such that $\Pi_1(0, SK_\pi, x) = 0^{2n}$. This is done as follows:

- Call $E(k, 0^n)$ to get $f_k(0^n)$.
- Let $R_{k-1} = f_k(0^n) \oplus 0^n$ and run $E(k-1, R_{k-1})$ to get $f_{k-1}(R_{k-1})$.
- Continue in this fashion to get $(R_{k-2}, f_{k-2}(R_{k-2}), \dots, (R_1, f_1(R_1))$.
- At the end, we know that $\Pi_1((R_2 \oplus f_1(R_1)) \parallel R_1) = 0^{2n}$. Send this as challenge query and return 1 if and only if the response is response is 0^{2n}.

Note that the adversary A_π can succeed only if it never queries the Π oracle on the challenge query, i.e. $(0, R_0 \parallel R_1)$, during the experimentation phase. We will argue that this happens only with negligible probability. Indeed, this is easily seen to be true since every query made by the procedure $E(\cdot, \cdot)$ is a random query to the VRP π^{LR} (at least one half of every query is random). Thus the probability that the output of any forward query is 0^{2n} is negligible $\left(\leq \frac{1}{2^n}\right)$. If no such query (i.e. one with output 0^{2n}) is made by A_π, then it has an overwhelming advantage in the VRP attack game $\left(\mathcal{O}\left(1 - \frac{1}{2^n}\right)\right)$.

We note here that the adversary A_π is not the most optimal one. Indeed, it only makes use of forward queries to Π. Thus, it needs to make a $\mathsf{Fibonacci}(k)$ number of queries. It is possible to optimize this attack by using inverse queries to Π as well. While the adversary A_π constructed round values for the challenge query in the order $R_{k-1}, R_{k-2}, \ldots, R_1, R_0$. The more optimized adversary constructs $R_{\frac{k}{2}}, R_{\frac{k}{2}-1}, \ldots, R_0$ using forward queries and $R_{\frac{k}{2}+1}, \ldots R_k$ using inverse queries. This way it only makes $\mathcal{O}\left(\mathsf{Fibonacci}\left(\frac{k}{2}\right)\right)$ number of queries, and still is able to construct a challenge query without querying upon it.

□

We note that it is possible to make our VRP attacker much more efficient, so that it makes only $\mathcal{O}(\mathsf{Fibonacci}(k/2))$ queries. This is done by recovering the first $k/2$ round values of the challenge query using forward queries, and the last $k/2$ round values using inverse queries. We describe the purely forward query based attacker because of its simplicity.

6 Implications for NIZK

NIZK proofs were introduced by Blum et al. [1]. The usual notion of NIZK proofs is in the *Common Reference String* model. Feige et al [8] introduced a more relaxed definition of NIZK proofs, i.e. NIZK proofs in the *Hidden Bit String* (HBS) model. A formal treatment of both these notions can be found in [12]. For completeness, we give an informal exposition of these two models below, and give the formal definitions in appendix F.

An NIZK protocol in the common reference string model assumes that the prover P and the verifier V have access to the common input x and a uniformly distributed common reference string selected by a trusted third party. And the NIZK protocol for showing membership in a language L consists of a single message sent from the prover to the verifier, or the NIZK proof that $x \in L$. The usual definitions of completeness, soundness and zero-knowledge apply. In the *Hidden Bits model*, only the prover is given access to a uniformly distributed random string $\mathcal{R}$. However, when the prover sends its proof to the verifier, it can choose to disclose a subset of the hidden random string ($\mathcal{R}_\mathcal{I}$) to the verifier. Feige et al [8] gave an unconditionally secure NIZK proof in the HBS model for any NP language. Moreover, they [8] also gave an implementation of the HBS model in the *common reference string* model using *trapdoor permutations* (TDPs).

Later, Goldwasser et al [10] showed that one can construct NIZK proofs using *invariant (or unique) signatures*. In particular, they gave an implementation of the *hidden bits model* using invariant signatures. Invariant signatures are quite similar to VRFs, but with one crucial difference. They should induce a (pseudo)random distribution on the output, when applied to a random input. Thus, we can think of *invariant signatures* as "balanced" VRFs. On the other hand, an obvious way to get such signatures from general VRFs in not known. In particular, VRFs cannot be used directly in the NIZK proofs of [10] or [8].

Claim 6.1 *If there exists a VRF (G, F, V), then there also exists a VRF (G', F', V') that is not an invariant signature. In particular, the VRF (G', F', V') does not work in the* hidden bit string *implementation of [10] or [8].*

The definition of the VRF (G', F', V') is clear from the discussion in introduction. We give a formal justification for this claim in appendix G.

From a similar angle, Canetti et al. [4] generalized the result of Feige et al. [8], and gave an implementation of the hidden bits model using a generalization of TDPs, called *publicly verifiable trapdoor predicates*. Informally, a publicly verifiable trapdoor predicate should be efficiently computable for any given pair

(x, y). And given y and a trapdoor td, one can efficiently compute x such that (x, y) satisfies the predicate (of course, the same task is hard without the trapdoor td). Moreover, it should also be possible to sample valid pairs (x, y) (that satisfy the predicate) in such a way that y is uniformly distributed. It is again easy to see that *balanced VRFs* (and thus VRPs) imply such predicates. But an obvious construction of such predicates from general VRFs is not known.

On the positive side, one can easily see that *verifiable random permutations* easily give such proofs by constructing either the invariant signatures of [10] or the publicly verifiable trapdoor predicates of [4]. For completeness, we show how to implement NIZK proofs for NP languages from the secure HBS model NIZK proofs of [8] using VRPs.

Claim 6.2 *Let (P, V) be a hidden-bit proof system for L, and let $\pi = (G_\pi, \Pi, V_\pi)$ be a verifiable random permutation on $\{0,1\}^n$. If (P, V) is zero-knowledge then one can construct an NIZK proof system (P', V') for L using (P, V) and π.*

A proof for the above claim can be found in appendix H. Thus using our construction of a VRP from any VRF, one sees that it is also possible to construct NIZK proofs for NP languages using VRFs.

Acknowledgments

We would like to thank Rafail Ostrovsky for several helpful discussions. In particular, for clarifying the results of [10].

References

[1] M. Blum, A. De Santis, S. Micali, and G. Persiano, *NonInteractive Zero-Knowledge*, in *SIAM Journal on Computing*, 20(6):1084-1118, 1991.

[2] Manuel Blum, Paul Feldman and Silvio Micali, *Non-Interactive Zero-Knowledge and Its Applications (Extended Abstract)* in *STOC 1988*, 103-112.

[3] Mihir Bellare and Moti Yung, *Certifying Permutations: Noninteractive Zero-Knowledge Based on Any Trapdoor Permutation*, in *Journal of Cryptology* 9(3): 149-166 (1996).

[4] Ran Canetti, Shai Halevi and Jonathan Katz, *A forward-secure public-key encryption scheme*, in *Advances in Cryptology - EUROCRYPT'03*.

[5] Y. Dodis, *Efficient construction of (distributed) verifiable random functions*, In *Proceedings of 6th International Workshop on Theory and Practice in Public Key Cryptography*, pp 1 -17, 2003.

[6] Y. Dodis and P. Puniya, *On the relation between Ideal Cipher and Random Oracle Models*, to appear in *Theory of Cryptography Conference 2006*.

[7] Y. Dodis and A. Yampolskiy, *A Verifiable Random Function With Short Proofs and Keys*, In *Workshop on Public Key Cryptography (PKC)*, January 2005.

[8] Uriel Feige, Dror Lapidot and Adi Shamir, *Multiple NonInteractive Zero Knowledge Proofs Under General Assumptions*, in *SIAM Journal of Computing* 29(1): 1-28 (1999).

[9] O. Goldreich, S. Goldwasser and S. Micali, *How to Construct Random Functions* in *Journal of the ACM*, Vol. 33, No. 4, October 1986.

[10] Shafi Goldwasser and Rafail Ostrovsky, *Invariant Signatures and Non-Interactive Zero-Knowledge Proofs are Equivalent (Extended Abstract)*, in *CRYPTO 1992*: 228-245.

[11] Jens Groth, Rafail Ostrovsky and Amit Sahai, *Perfect Non-Interactive Zero Knowledge for NP*, in *Electronic Colloquium on Computational Complexity (097)*, 2005.

[12] Goldreich, O., *Foundations of Cryptography Basic Tools*, published by *Cambridge University Press* (2001).

[13] M. Luby and C. Rackoff, *How to construct pseudo-random permutations from pseudo-random functions*, in *SIAM Journal on Computing*, Vol. 17, No. 2, April 1988.

[14] A. Lysyanskaya, *Unique Signatures and verifiable random functions from DH-DDH assumption*, in *Proceedings of the 22nd Annual International Conference on Advances in Cryptography (CRYPTO)*, pp. 597 612, 2002.

[15] Ueli M. Maurer and Krzysztof Pietrzak, *The Security of Many-Round Luby-Rackoff Pseudo-Random Permutations*, in *EUROCRYPT 2003*, 544-561.

[16] S. Micali, M. Rabin and S. Vadhan, *Verifiable Random functions*, In *Proceedings of the 40th IEEE Symposium on Foundations of Computer Science*, pp. 120 -130, 1999.

[17] Moni Naor and Omer Reingold, *On the construction of pseudo-random permutations: Luby-Rackoff revisited*, in *Journal of Cryptology*, vol 12, 1999, pp. 29-66.

[18] Jacques Patarin, *Security of Random Feistel Schemes with 5 or More Rounds*, in *CRYPTO 2004*, 106-122.

A Proof of Lemma 3.1

We begin by noting that one of the inputs involved in the $\left(\frac{k}{2}\right)^{th}$ round value collision can be assumed to be the last (m^{th}) input. If none of the two colliding inputs are the last input, then we can ignore all the inputs that are processed after the collision and proceed with the remaining matrix representation. Hence let us assume that, $\Phi\left[i, \frac{k}{2}\right] = \Phi\left[m, \frac{k}{2}\right]$. (with $i < m$). We can also assume that for any $i' < i$, $\Phi\left[i, \frac{k}{2}\right] \neq \Phi\left[i', \frac{k}{2}\right]$. That is, we consider the first such collision. We also assume that the matrix representation Φ, storing the inputs provided to the construction Ψ_k, does not contain any duplicate rows. In case such rows exist, we ignore all but the first such row and remove the others from Φ. This is because all useful computation is performed in this row itself.

We define a "first occurrence" function for each round value, $\mathsf{p} : \{1, m\} \times \{0, k+1\} \rightarrow \{1, m\}$. For any round value $\Phi[i, j]$, $\mathsf{p}(i, j)$ is the *least input number* such that $\Phi[\mathsf{p}(i, j), j] = \Phi[i, j]$. Thus $\mathsf{p}(i, j)$ essentially denotes the input where $f_j\left(\Phi[i, j]\right)$ was defined for the first time.

Assume for now, that the m^{th} input provided to Ψ_k is a forward input, i.e. $B[m] = 0$. In case it is an inverse input, then a symmetric argument can be carried out. As a first step, we prove that all of the inputs $\mathsf{p}(m, 1), \ldots, \mathsf{p}\left(m, \frac{k}{2}\right)$ were provided to Ψ_k strictly before input number m. That is all the round values $\Phi[m, 1] \ldots \Phi\left[m, \frac{k}{2}\right]$ were defined strictly before input number m.

Claim A.1 *If $\Phi\left[m, \frac{k}{2}\right] = \Phi\left[i, \frac{k}{2}\right]$ then each of the round values $\Phi[m, 1] \ldots \Phi\left[m, \frac{k}{2} - 1\right]$ were defined before the m^{th} input was processed. That is,*

$$\forall j \in \left\{1, \frac{k}{2}\right\} \ : \ \mathsf{p}(m, j) < m$$

proof of claim A.1: We will use induction on the round number j to show that $\mathsf{p}(m,j) < m$. However, we will start the induction with $j = \frac{k}{2}$ and go down to $j = 1$.

For $j = \frac{k}{2}$, we already know that $\mathsf{p}(m,j) = i$ from the statement of the claim. Now say the same holds for all $j = \frac{k}{2} \ldots c$ (for $c \leq k/2$), then we will show that the same holds for $j = c-1$. Say, for the sake of contradiction, that $\Phi[m, c-1]$ is a new round value in input number m (i.e. $\mathsf{p}(m, c-1) = m$). Then we claim that the round values $\Phi[\mathsf{p}(m,c), c]$ and $\Phi[m,(c-2)]$ were defined before $\Phi[m,(c-1)]$. The former is true from the induction hypothesis. The latter is true because the m^{th} input is a forward input, and so even if $\mathsf{p}(m,(c-2))$ is a new round value it is defined before $\Phi[m,(c-1)]$. Also from the LR construction property, we know that

$$\begin{aligned} f_{c-1}\left(\Phi[m,(c-1)]\right) &= \Phi[m,(c-2)] \oplus \Phi[m,c] \\ &= \Phi[m,(c-2)] \oplus \Phi[\mathsf{p}(m,c),c] \end{aligned}$$

But from the statement of lemma 3.1, f_j is a 4-XOR resistant function. Hence $\mathsf{p}(m,(c-1)) < m$. □

Thus, we know that all the round values $\Phi[m,1] \ldots \Phi\left[m, \frac{k}{2}\right]$ were defined strictly before input number m. Our next step would be to find exactly the orders in which the inputs $\mathsf{p}(m,1) \ldots \mathsf{p}\left(m, \frac{k}{2}\right)$ could occur in Φ. As it turns out, there are only a few special orders in which these inputs could have occurred.

Claim A.2 *For some $j \in \left\{1, \frac{k}{2}\right\}$, it holds that,*

$$\begin{array}{ccccccccc} \mathsf{p}(m,1) & > & \ldots & > & \mathsf{p}(m,(j-1)) & > & \mathsf{p}(m,j) & & \\ & & & & & & \mathsf{p}(m,j) & < \ldots < & \mathsf{p}\left(m,\frac{k}{2}-1\right) < \mathsf{p}\left(m,\frac{k}{2}\right) \end{array}$$

That is the round value $\Phi[m,j]$ was the first one to be defined amongst the first $\frac{k}{2}$ round values. All round values $\Phi[m,j']$ for $j' = 1 \ldots j$ were defined in strictly decreasing temporal order, $\Phi[m,1]$ being the last one to be defined among these j round values. And all round values $\Phi[m,j']$ for $j' = j \ldots \frac{k}{2}$ were defined in strictly increasing temporal order, $\Phi\left[m, \frac{k}{2}\right]$ being the last one to be defined among these $\frac{k}{2} - j + 1$ round values.

proof of claim A.2: We will first prove that for any three consecutive round values $\Phi[m,(i-1)], \Phi[m,i]$ and $\Phi[m,(i+1)]$ (where $i \in \left\{2, \frac{k}{2}-1\right\}$), it holds that,

$$[\mathsf{p}(m,(i-1)) > \mathsf{p}(m,i)] \vee [\mathsf{p}(m,i) < \mathsf{p}(m,(i+1))]$$

The claim will follow as a straightforward consequence of this.

Assume to the contrary that $\mathsf{p}(m,(i-1)) \leq \mathsf{p}(m,i)$ and $\mathsf{p}(m,i) \geq \mathsf{p}(m,(i+1))$ for some $i \in \left\{2, \frac{k}{2}-1\right\}$. If $\mathsf{p}(m,(i-1)) = \mathsf{p}(m,i)$ (or $\mathsf{p}(m,i) = \mathsf{p}(m,(i+1))$ resp.) then it is easy to verify that rows $\mathsf{p}(m,i)$ and m in Φ are identical, which is not possible since we started with a matrix Φ with distinct rows.

Thus, we have the case that $\mathsf{p}(m,(i-1)) < \mathsf{p}(m,i)$ and $\mathsf{p}(m,i) > \mathsf{p}(m,(i+1))$. That is, both the round values $\Phi[m,(i-1)]$ and $\Phi[m,(i+1)]$ were defined prior to the definition of $\Phi[m,i]$. But we also know that,

$$\begin{aligned} & f_i\left(\Phi[m,i]\right) &=& \Phi[m,(i-1)] \oplus \Phi[m,(i+1)] \\ \Rightarrow\ & f_i\left(\Phi[\mathsf{p}(m,i),i]\right) &=& \Phi[m,(i-1)] \oplus \Phi[m,(i+1)] \\ \Rightarrow\ & f_i\left(\Phi[\mathsf{p}(m,i),i]\right) &=& \Phi[\mathsf{p}(m,(i-1)),(i-1)] \oplus \Phi[\mathsf{p}(m,(i+1)),(i+1)] \end{aligned}$$

But the function f_i is 4-XOR resistant, and both round values $\Phi[\mathsf{p}(m,(i-1)),(i-1)]$ and $\Phi[\mathsf{p}(m,(i+1)),(i+1)]$ were defined prior to $\Phi[\mathsf{p}(m,i),i]$. We get a contradiction, and deduce that

$$\forall i \in \left\{2, \frac{k}{2}-1\right\} \ :\ [\mathsf{p}(m,(i-1)) > \mathsf{p}(m,i)] \vee [\mathsf{p}(m,i) < \mathsf{p}(m,(i+1))]$$

Now it is a straightforward task to verify that the only temporal orders of inputs $\mathsf{p}(m,1),\ldots,\mathsf{p}\left(m,\frac{k}{2}\right)$ that are consistent with this constraint are the ones in the statement of claim A.2. □

From claim A.2, we can deduce that there exist at least $\frac{k}{4}$ consecutive round values in the m^{th} row of Φ, whose "first occurrences" are in strictly increasing/decreasing temporal order. Without loss of generality, we will assume that $\mathsf{p}(m,1) > \ldots > \mathsf{p}\left(m,\frac{k}{4}\right)$. Our argument, henceforth, can be easily modified to fit any other strict ordering of "first occurrences" as well. As it turns out, the case we are considering gives the "lowest lower bound" on m and is the worst case.

Thus we have the case that $m > \mathsf{p}(m,1) > \ldots > \mathsf{p}\left(m,\frac{k}{4}\right)$. Our next step will be to prove a general property of such monotonic sequences of fresh round values, which we will apply recursively later. Thus consider any three "first occurrence inputs" $i_j = \mathsf{p}(i',j)$, $i_{j+1} = \mathsf{p}(i',(j+1))$ and $i_{j+2}\mathsf{p}(i',(j+2))$ that are ordered as $i_j > i_{j+1} > i_{j+2}$. Next say that input number i_j is a forward input (a symmetric argument can be applied if it is an inverse input). Then we show that all the round values $\Phi[i_j,1]\ldots\Phi[i_j,(j-2)]$ have their first occurrence strictly in between i_j and i_{j+1}, and in strictly decreasing order (i.e. $\Phi[i_j,1]$ the last to be defined). This fact is stated formally in the following claim.

Claim A.3 *Consider the inputs numbered $\mathsf{p}(i',j),\mathsf{p}(i',j+1)$ and $\mathsf{p}(i',j+2)$ in the matrix Φ, which are the "first occurrence" inputs corresponding to three consecutive round values $\Phi[i',j],\Phi[i',j+1]$ and $\Phi[i',j+2]$ in the i'^{th} input. Moreover, say that $\mathsf{p}(i',j) > \mathsf{p}(i',j+1) > \mathsf{p}(i',j+2)$. Denoting $\mathsf{p}(i',j)$ by i_j and $\mathsf{p}(i',(j+1))$ by i_{j+1}, we claim that if input number i_j is a forward input then,*

$$i_j > \mathsf{p}(i_j,1) > \mathsf{p}(i_j,2) > \ldots > \mathsf{p}(i_j,j-2) > i_{j+1}$$

That is all the inputs $\mathsf{p}(i_j,1),\mathsf{p}(i_j,2),\ldots,\mathsf{p}(i_j,j-2)$ are in strictly decreasing temporal order. Moreover, all these inputs are "sandwiched" between the inputs $i_j = \mathsf{p}(i',j)$ and $\mathsf{p}(i',j+1)$.

proof of claim A.3: We will first define a little notation before moving along. Let $i_j = \mathsf{p}(i',j), i_{j+1} = \mathsf{p}(i',(j+1))$ and $i_{j+2} = \mathsf{p}(i',(j+2))$. Let us now analyze the round values $\Phi[i_j,1]\ldots\Phi[i_j,(j-1)]$ since these round values come before $\Phi[i_j,j]$ in the ${i_j}^{th}$ input, which we have assumed to be a forward input.

To start with, we know that $f_{j+1}\left(\Phi[i',(j+1)]\right) = \Phi[i',j] \oplus \Phi[i',(j+2)]$. Thus we can deduce that,

$$f_{j+1}\left(\Phi[i_{j+1},(j+1)]\right) = \Phi[i_j,j] \oplus \Phi[i_{j+2},(j+2)] \tag{1}$$

Looking at the ${i_j}^{th}$ input, we know that

$$f_{j-1}\left(\Phi[i_j,(j-1)]\right) = \Phi[i_j,(j-2)] \oplus \Phi[i_j,j] \tag{2}$$

From equations (1) and (2), we can deduce that $\mathsf{p}(i_j,(j-1)) < i_j$. Since otherwise, combining these two equations will give,

$$\begin{aligned} f_{j-1}\left(\Phi[i_j,(j-1)]\right) &= \Phi[i_j,(j-2)] \oplus \Phi[i_{j+2},(j+2)] \oplus f_{j+1}\left(\Phi[i_{j+1},(j+1)]\right) \\ &= \Phi[i_j,(j-2)] \oplus \Phi[i_{j+2},(j+2)] \oplus \Phi[i_{j+1},j] \oplus \Phi[i_{j+1},(j+2)] \end{aligned}$$

If $\mathsf{p}(i_j,(j-1)) = i_j$ then it would mean that $\Phi[i_j,(j-1)]$ is a new round value in input i_j. But this is impossible since f_{j-1} is 4-XOR resistant, and all the 4 round values on RHS of the above equation were defined before $\Phi[i_j,(j-1)]$ if it is a new round value. Thus, we can deduce that $\mathsf{p}(i_j,(j-1)) < i_j$. Similarly, we can also deduce that $\mathsf{p}(i_j,(j-2)) < i_j$ because f_{j-2} is also 4-XOR resistant and

$$\begin{aligned} f_{j-2}(\Phi[i_j,(j-2)]) &= \Phi[i_j,(j-3)] \oplus \Phi[i_j,(j-1)] \\ &= \Phi[i_j,(j-3)] \oplus \Phi[\mathsf{p}(i_j,(j-1)),(j-1)] \end{aligned}$$

conclusion 1: Applying the same method repeatedly, we can conclude this $\forall \ell \in \{1, j-1\} \ : \ \mathsf{p}(i_j, \ell) < i_j$.

For now we will concentrate on finding possible orders of the input numbers $\mathsf{p}(i_j, (j-2)), \mathsf{p}(i_j, (j-1))$ and i_{j+1}, all of which come before input number i_j. We start by noting that if input number i_{j+1} is a forward input then,

$$[\mathsf{p}(i_j, (j-2)) > \mathsf{p}(i_j, (j-1))] \quad \vee \quad [i_{j+1} \geq \mathsf{p}(i_j, (j-1))] \tag{3}$$

We know that $\mathsf{p}(i_j, (j-2)) \neq \mathsf{p}(i_j, (j-1))$, since this would imply two identical inputs in the matrix Φ. Thus if relation 3 is false, it must be the case that $[\mathsf{p}(i_j, (j-2)) < \mathsf{p}(i_j, (j-1))] \wedge [i_{j+1} < \mathsf{p}(i_j, (j-1))]$. That is, both inputs $\mathsf{p}(i_j, (j-2))$ and i_{j+1} were computed before $\mathsf{p}(i_j, (j-1))$. And we know from above that

$$f_{j-1}\left(\Phi[i_j, (j-1)]\right) = \Phi[\mathsf{p}(i_j, (j-2)), (j-2)] \oplus \Phi[i_{j+2}, (j+2)] \oplus \Phi[i_{j+1}, j] \oplus \Phi[i_{j+1}, (j+2)]$$

This is impossible since f_{j-1} is a 4-XOR resistant round function, and hence relation (3) holds. Next we note that if input number i_{j+1} is a forward input then,

$$[\mathsf{p}(i_j, (j-2)) > i_{j+1}] \quad \vee \quad [\mathsf{p}(i_j, (j-1)) > i_{j+1}] \tag{4}$$

Say to the contrary, $[\mathsf{p}(i_j, (j-2)) \leq i_{j+1}] \wedge [\mathsf{p}(i_j, (j-1)) \leq i_{j+1}]$. Then it would mean that both $\Phi[i_j, (j-2)]$ and $\Phi[i_j, (j-1)]$ were defined prior to the round value $\Phi[i_{j+1}, j+1]$. But we know that

$$\begin{aligned} f_{j+1}\left(\Phi[i_{j+1}, (j+1)]\right) &= \Phi[i_j, (j-2)] \oplus \Phi[i_{j+2}, (j+2)] \oplus f_{j-1}\left(\Phi[i_j, j-1]\right) \\ &= \Phi[i_j, (j-2)] \oplus \Phi[i_{j+2}, (j+2)] \oplus \Phi[\mathsf{p}(i_j, (j-1)), (j-2)] \oplus \Phi[\mathsf{p}(i_j, (j-1)), j] \end{aligned}$$

And since f_{j+1} is also a 4-XOR resistant round function, we get a contradiction. Hence relation (4) holds as well.

conclusion 2: Since both relations (3) and (4) to hold, we can deduce that

$$[\Phi(i_j, (j-2)) > i_{j+1}] \quad \wedge \quad [\Phi(i_j, (j-2)) > \Phi(i_j, (j-1))] \tag{5}$$

That is, the first occurrence of round value $\Phi[i_j, (j-2)]$ is before that of both $\Phi[i_j, (j-1)]$ and $\Phi[i_{j+1}, (j+1)]$. Even though we assumed above that input i_{j+1} is a forward input, we will come to the same conclusion even if input number i_{j+1} was an inverse input, by slightly modifying the relations (3) and (4).

Now we will look at the order of all the inputs $\mathsf{p}(i_j, \ell)$ for $\ell \in \{1, j-1\}$. From *conclusion 1* above, we know that

$$\forall \ell \in \{1, j-1\} \ : \ \mathsf{p}(i_j, \ell) < i_j$$

And from *conclusion 2*, we know that

$$\mathsf{p}(i_j, (j-1)) < \mathsf{p}(i_j, (j-2))$$

Now we will show that if for any $\ell \in \{1, j-3\}$ if $\mathsf{p}(i_j, (\ell+1)) > \mathsf{p}(i_j, (\ell+2))$ then $\mathsf{p}(i_j, \ell) > \mathsf{p}(i_j, (\ell+1))$. This is not hard to see because,

$$\begin{aligned} & f_{\ell+1}\left(\Phi[i_j, \ell+1]\right) &&= \Phi[i_j, \ell] \oplus \Phi[i_j, \ell+2] \\ \Rightarrow \quad & f_{\ell+1}\left(\Phi\left[\mathsf{p}(i_j, (\ell+1)), (\ell+1)\right]\right) &&= \Phi\left[\mathsf{p}(i_j, \ell), \ell\right] \oplus \Phi\left[\mathsf{p}(i_j, (\ell+2)), (\ell+2)\right] \end{aligned}$$

And since $f_{\ell+1}$ is 4-XOR resistant and $\mathsf{p}(i_j, \ell) \neq \mathsf{p}(i_j, (\ell+1))$ (o/w the inputs i_j and $\mathsf{p}(i_j, \ell)$ are same), we can deduce that,

$$\mathsf{p}(i_j, \ell) > \mathsf{p}(i_j, (\ell+1)) > \mathsf{p}(i_j, (\ell+2))$$

Now we can apply this inductively to $\ell = (j-3)\ldots 1$. Then we can use *conclusions 1 and 2* and deduce that,

$$i_j > \mathsf{p}(i_j, 1) > \mathsf{p}(i_j, 2) > \ldots > \mathsf{p}(i_j, (j-2)) > i_{j+1}$$

□

Let us briefly recall what we have achieved so far. In claim A.1, we saw that all the inputs $\Phi[m,1]\ldots\Phi[m,\frac{k}{2}]$ have their "first occurrences" before input number m. Then in claim A.2, we established the temporal orderings of these "first occurrence" inputs as being one of very few special orders. In addition, we also deduced that each of these orderings gives us a sequence of $\frac{k}{4}$ first occurrence inputs that are monotonically ordered. Then we went onto claim A.3, where we proved a general property of any triple of strictly ordered first occurrence inputs. Now the next step would be to combine these steps to deduce the underlying structure of inputs in Φ that we wish to make explicit.

Without loss of generality, say that $\mathsf{p}(m,1) > \mathsf{p}(m,2) > \ldots > \mathsf{p}\left(m, \frac{k}{4}\right)$. In this sequence of inputs, we can find $\frac{k}{4} - 2$ strictly ordered triples of first occurrence inputs, namely

$$(\mathsf{p}(m,1), \mathsf{p}(m,2), \mathsf{p}(m,3)), \ldots, \left(\mathsf{p}(m, (\frac{k}{4}-2)), \mathsf{p}(m, (\frac{k}{4}-1)), \mathsf{p}(m, \frac{k}{4})\right)$$

The next step will be to apply claim A.3 to each of these triples. Having done so, we find another set of "first occurrence" input sequences that are in strict decreasing temporal order (for a forward input) and are *sandwiched* between the inputs at this level.

Since each of these next level sequences of inputs are also in strictly decreasing temporal order, we can apply claim A.3 to each of these as well. And because of the sandwiching property that claim A.3 gives, none of these sequences of inputs overlap. Hence all inputs in different sequences are distinct. We will now the number of inputs we get by applying claim A.3 to $\mathsf{p}(m,1)\ldots\mathsf{p}\left(m,\frac{k}{4}\right)$. We will denote by $Q(j)$ the number of inputs we get by applying claim A.3 to a sequence of "first occurrence" inputs $\mathsf{p}(i,1)\ldots\mathsf{p}(i,j)$ that are in strictly decreasing temporal order. There is no ambiguity of notation since the number $Q(j)$ is independent of the query number i that the sequence $\mathsf{p}(i,1)\ldots\mathsf{p}(i,j)$ corresponds to. Thus we get,

$$\begin{aligned} Q\left(\frac{k}{4}\right) &= \sum_{j=1}^{\frac{k}{4}-2} Q(i) \\ &= Q\left(\frac{k}{4}-1\right) + Q\left(\frac{k}{4}-2\right) \end{aligned}$$

Now this expression is the familiar recursion corresponding to the expression for the $\left(\frac{k}{4}\right)^{th}$ Fibonacci number. Hence we get that

$$\begin{aligned} & Q\left(\tfrac{k}{4}\right) = \mathsf{Fibonacci}\left(\tfrac{k}{4}\right) \\ \Longrightarrow\ & m \geq \mathsf{Fibonacci}\left(\tfrac{k}{4}\right) \\ \Longrightarrow\ & m = \Omega\left(\left(\tfrac{\sqrt{5}+1}{2}\right)^{\frac{k}{4}}\right) \end{aligned}$$

Thus if the round functions used in a k-round LR construction are all 4-XOR resistant, and the number of rounds k is super-logarithmic in the security parameter, then the number of rows m in the matrix representation Φ (for this construction) is super-polynomial in the security parameter if there is a collision in the $\left(\frac{k}{4}\right)^{th}$ round of two inputs.

B XOR-freeness of VRFs

Let us denote the k VRFs used in the construction $\pi^{LR} = (G_\pi, \Pi, V_\pi)$ by $\{f_j = (G^j, F^j, V^j\}_{j=1\ldots k}$. And let $\{inv_i, X_i\}_{i=1\ldots q}$ be q queries made by the adversary A_{xor} to the VRP evaluator Π, then we will denote by $\{R_0^i, R_1^i, \ldots, R_k^i, R_{k+1}^i\}_{i=1\ldots q}$ the round values computed for these queries. We also know that for a round value R_j^i, the round values from any of the previous queries $(1 \ldots i-1)$ are defined before R_j^i. In addition, all round values $R_0^i \ldots R_{j-1}^i$ (resp. $R_{j+1}^i \ldots R_{k+1}^i$) are defined before R_j^i if the i^{th} query is a forward (resp. inverse) query. For an adversary A_{xor} that makes q queries to the Π oracle, its advantage in the c-XOR attack game is defined to be the probability with which at least one new round function value (in its queries) can be represented as an XOR of c round values defined before it. Thus,

$$Adv_c(A_{xor}) = Pr\left[F_1^j(SK_j, R_j^i) = \bigoplus_{\ell=1\ldots c} Z_\ell \;\middle|\; \begin{array}{c} (PK_\pi, SK_\pi) \leftarrow G_\pi(1^\lambda); (inv_1, X_1) \ldots \\ (inv_q, X_q) \text{ are queries made by } A_{xor}^{\Pi}; i \in \{1, q\}, \\ j \in \{1, k\}; \{Z_\ell\}_{\ell=1\ldots c} \text{ are round values defined before } R_j^i \end{array} \right]$$

The target round value R_j^i above, should be a new round value. We restate claim 4.2 here, which essentially states that secure VRFs resist attacks from all such adversaries A_{xor}.

Claim B.1 *If there is an adversary A_{xor} that has a non-negligible advantage ϵ_{xor} in the c-XOR attack game described above, then there also exists a VRF adversary A_f that has a non-negligible advantage ϵ_f in differentiating a VRF output from a random binary string, in the VRF pseudorandomness game. In particular,*

$$\epsilon_f \geq \left(\frac{\epsilon_{xor}}{2qk} - \frac{(qk)^c}{2^{n+1}} \right)$$

proof of claim B.1: The VRF adversary A_f is given access to the function evaluator $F(SK, \cdot)$ for the VRF f, and also gets the corresponding public key PK. It simply uses the XOR attack adversary A_{xor} as a subroutine by simulating the construction π^{LR} for A_{xor}. To do this, A_f first chooses a random round number $j \leftarrow \{1, k\}$ and plugs in the VRF f as the j^{th} round function. Then A_f generates $k-1$ independent round function keys, $\{PK_\ell, SK_\ell\}_{\ell \in 1\ldots j-1, j+1\ldots k}$, and sends the public key $PK_1 \ldots PK_{j-1}, PK, PK_{j+1} \ldots PK_k$ to A_{xor} as the public key for construction π^{LR}. Then A_f chooses a random query number $i \leftarrow \{1, q\}$. It honestly evaluates the LR construction for queries $1 \ldots i-1$, querying its oracle $F(SK, \cdot)$ for the j^{th} round function in every query. For query number i made by A_{xor}, it honestly evaluates the LR construction upto the j^{th} round value R_j^i. If R_j^i is a previously defined value then A_f exits with failure, otherwise it sends the round value R_j^i as its query to the challenge VRF oracle. On getting the challenge oracle response Y, A_f compares it with all XORs of upto c previously defined round values. If there is a match then it outputs 1 indicating that $Y = F(SK, R_j^i)$ otherwise it outputs 0 indicating that it is a random value. The advantage of A_f can be computed as follows,

$$Adv(A_f) = Pr[(A_f \text{ outputs } 1) \wedge (Y = F(SK, R_j^i))] + Pr[(A_f \text{ outputs } 0) \wedge (Y \text{ is random})] - \frac{1}{2}$$

Here we assume, without loss of generality, that the above advantage is positive. If ϵ_{xor} is the advantage of A_{xor} in the XOR attack game then these probabilities can be computed as,

$$\begin{aligned} Pr\left[(A_f \text{ outputs } 1) \wedge (Y = F(SK, R_j^i))\right] &= Pr\left[(A_f \text{ outputs } 1) \,\middle|\, (Y = F(SK, R_j^i))\right] \cdot Pr\left[Y = F(SK, R_j^i)\right] \\ &\geq \frac{\epsilon_{xor}}{qk} \cdot \frac{1}{2} = \frac{\epsilon_{xor}}{2qk} \end{aligned}$$

Similarly,

$$\begin{aligned} Pr\left[(A_f \text{ outputs } 0) \wedge (Y \text{ is random})\right] &= Pr\left[(A_f \text{ outputs } 0)\,|(Y \text{ is random})\right] \cdot Pr\left[Y \text{ is random}\right] \\ &\geq \left[1 - \frac{(qk)^c}{2^n}\right] \cdot \frac{1}{2} \end{aligned}$$

Thus, we can compute the advantage of A_f to be,

$$Adv(A_f) \geq \left(\frac{\epsilon_{xor}}{2qk} - \frac{(qk)^c}{2^{n+1}}\right)$$

□

C Proof of claim 4.3

Let us denote the round functions used in π^{LR} by $\{f_j = (G^i, F^i, V^i\}_{j=1\ldots k}$. And when we say that the round function f_j is 4-XOR resistant, it means that for a polynomial number (q) of queries made to π^{LR}, none of the new round function values outputted by f_j are XORs of previously defined round values. Let XOR_{f_j} denote the event that the round function f_j is not 4-XOR resistant, i.e. the VRP adversary is able to force an XOR dependency involving round function f_j. From above, we know that if the maximum advantage of a VRF adversary in a pseudorandomness game with f_j is ϵ_{f_j}, then

$$Pr[XOR_{f_j}] \leq qk \cdot \left(2\epsilon_{f_j} + \frac{(qk)^4}{2^n}\right)$$

For simplicity, we will take the maximum advantage of a XOR attack against any of the round functions used in π^{LR} to be ϵ_{xor}. Now say there exists a VRP adversary A_π that has a non-negligible distinguishing advantage ϵ_π in the pseudorandomness game with π^{LR}. Our "XOR-free" VRP adversary A'_π simply runs A_π and rejects if A_π finds a round function value that is an XOR of upto 4 previously existing round values. If no such XORs occur, then A'_π simply outputs the prediction of A_π. Thus,

$$\begin{aligned} Adv(A_\pi) &= \left(\sum_{j=1}^{k} Pr\left[XOR_{f_j} \wedge \left(\bigwedge_{j'=1}^{j-1} \overline{XOR}_{f_{j'}}\right)\right]\right) + Pr\left[A_\pi \text{ wins} \wedge \left(\bigwedge_{j=1}^{k} \overline{XOR}_{f_j}\right)\right] \\ &= \left(\sum_{j=1}^{k} Pr\left[XOR_{f_j} \wedge \left(\bigwedge_{j'=1}^{j-1} \overline{XOR}_{f_{j'}}\right)\right]\right) + Adv(A'_\pi) \\ \implies Adv(A'_\pi) &\geq \epsilon_\pi - \sum_{j=0}^{k-1} \epsilon_{xor}(1-\epsilon_{xor})^j \\ &= \epsilon_\pi - \left(1 - (1-\epsilon_{xor})^k\right) \end{aligned}$$

Thus if ϵ_π (advantage of the VRP adversary A_π) is non-negligible and ϵ_{xor} is negligible, then the advantage of the XOR free adversary A'_π is also non-negligible.

D Proof of claim 4.4

Understanding why this is true involves five main steps:

- The VRF adversary A_f plugs the VRF f into a "simulated" construction π^{LR} that the VRP adversary A_π is allowed to attack. But plugging f just anywhere in the LR construction does not work. We get the best results if f is used as the middle round function $f_{\frac{k}{2}}$ in π^{LR}.

- A_f enters the "challenge phase" of its attack, exactly when A_π sends its challenge query. And the challenge query of A_f is the $\left(\frac{k}{2}\right)^{th}$ round value in the challenge query made by A_π to π^{LR}.
- We need to ensure that the $\left(\frac{k}{2}\right)^{th}$ round value in A_π is a new round value. If not then it implies that A_f has already queried the VRF f on this query and its attack fails. Here we make crucial use of the *combinatorial lemma* 3.1.
- Another important thing to ensure is that if the challenge query for A_f is chosen to be random (instead of the VRF value), then the challenge query that A_f gives to A_π (by computing the "simulated" construction π^{LR}) is also random. But we essentially get from the properties of the iterated Feistel construction.
- We also need to show that once the challenge query response is known to A_π, it cannot force the VRF adversary A_f to provide a proof on the $\left(\frac{k}{2}\right)^{th}$ round value of this query. This is because A_f itself does not know a proof for this round value. This part again makes use of the *combinatorial lemma* 3.1.

The VRF adversary is given access to the function evaluator F for the VRF f. It honestly initializes $k-1$ independent VRFs $f_1, \ldots, f_{\frac{k}{2}-1}, f_{\frac{k}{2}+1}, \ldots f_k$ using the corresponding VRF generators. And simulates the VRP construction π^{LR} using the LR construction $\Psi_{f_1,\ldots,f_{\frac{k}{2}-1},f,f_{\frac{k}{2}+1},\ldots f_k}$, and gives the VRP adversary A_π oracle access to this VRP construction. When A_π makes a (forward or inverse) query to the VRP oracle, then A_f honestly evaluates $\Psi_{f_1,\ldots,f_k}$ (using the VRF oracle for f to compute the middle round function).

Let us denote the queries made by A_π before the challenge phase as $(b_i, x_i)_{i\in\{1,q_1-1\}}$, the challenge query of A_π as (b_{q_1}, x_{q_1}) and the queries after the challenge phase as $(b_i, x_i)_{i\in\{q_1+1,q\}}$. And denote the corresponding round values for these queries as $\{R^i_0, \ldots, R^i_{k+1}\}_{i\in\{1,q\}}$. For the challenge query (b_{q_1}, x_{q_1}), the VRF adversary again computes the construction π^{LR} honestly except the round function value $f(R_{\frac{k}{2}})$. A_f sends this round value as its challenge query in the VRF game it plays with f. Without loss of generality, say the challenge query made by A_π is a forward query, i.e. $b_{q_1} = 0$. Then A_f sends $R^{q_1}_k \parallel R^{q_1}_{k+1}$ as the challenge response to A_π.

Problems could occur if round value $R^{q_1}_{\frac{k}{2}}$ collides with a round value $R^i_{\frac{k}{2}}$, for $i < \frac{q}{2}$. But then we can construct a $q_1 \times (k+2)$ matrix Φ using the round values $\{R^i_0, \ldots, R^i_{k+1}\}_{i\in\{1,q_1\}}$. And since, by assumption, A_π remains "4-XOR free", the matrix Φ is a matrix representation for an LR construction with 4-XOR resistant round functions. By lemma 3.1, we can deduce that the number of rows in this matrix, i.e. q_1, is at least $\mathsf{Fibonacci}\left(\frac{k}{4}\right)$. This is super-polynomial in the security parameter λ if $k = \omega(\log(\lambda))$, and contradicts the fact that A_π makes a polynomial number of queries.

Thus the round value $R^{q_1}_{\frac{k}{2}}$ is a new one, and hence the corresponding round function value looks random irrespective of the choice of the VRF challenge oracle (i.e. random value or VRF value). Similarly all the round values $R_{\frac{k}{2}+1} \ldots R_{k+1}$ are new round values with very high probability, since the converse would imply a collision for one of the VRFs $f_{\frac{k}{2}+1} \ldots f_k$. And such a collision will also imply that the colliding round function value can be represented as an XOR of previously existing round values. Thus, we can deduce that the last two round values (R_k, R_{k+1}) (i.e. the response given to A_π for its challenge query) look random if the VRF challenge response is a random value. A similar argument can be carried out if the challenge query made by A_π is an inverse query.

If, following the challenge phase, the VRP adversary A_π forces A_f to give a proof for the round value $R^{q_1}_{\frac{k}{2}}$ then it would mean that for some $i' > q_1$ the round value $R_{\frac{k}{2}i'}$ collides with $R^{q_1}_{\frac{k}{2}}$. But then we can construct

an $i' \times (k+2)$ matrix Φ using the round values $\{R_0^i, \ldots, R_{k+1}^i\}_{i \in \{1,i'\}}$. This is a matrix representation for the LR construction of π^{LR}, whose round functions are 4-XOR resistant by assumption. And by lemma 3.1, we deduce that the number of rows in Φ (i.e. i') is at least $\mathsf{Fibonacci}\left(\frac{k}{4}\right)$. This implies the number of queries made by A_π is super-polynomial in λ, if $k = \omega(\log(\lambda))$.

Thus, in conclusion, if the VRP adversary A_π (that is 4-XOR free) has advantage ϵ_π in the VRP attack game with the construction π^{LR} then there is a VRF adversary that has an advantage $\epsilon_f = \epsilon_\pi$ in the VRF attack game with one of the (secure) VRFs used in the construction π^{LR}, provided A_π makes a polynomial number of queries.

E NIZK and Hidden Bits Model proofs

We essentially follow the definitions of NIZK and hidden bits model given in [12]. For completeness, we give the definitions here,

Definition 2 (Non-iteractive Zero-Knowledge Proofs) *A pair of probabilistic machines (P, V) is called a non-interactive zero-knowledge proof system for a language L if V is polynomial time and the following conditions hold,*

- Completeness: *For every $x \in L$,*

$$Pr\left[V(x, R, P(x, R)) = 1\right] \geq \frac{2}{3}$$

  where R is a random variable uniformly distributed in $\{0,1\}^{poly(|x|)}$.

- Soundness: *For every $x \notin L$ and every algorithm B,*

$$Pr\left[V(x, R, B(x, R)) = 1\right] \leq \frac{1}{3}$$

  where R is a random variable uniformly distributed in $\{0,1\}^{poly(|x|)}$.

- Zero-Knowledge: *There exists a polynomial p and a probabilistic polynomial-time algorithm M such that the ensembles $\left\{(x, U_{p(|x|)}, P(x, U_{p(|x|)}))\right\}_{x \in L}$ and $\{M(x)\}_{x \in L}$ are computationally indistinguishable, where U_m is a random variable uniformly distributed over $\{0,1\}^m$.*

The random string R above is called the common reference string.

Now let us give the definition of Hidden Bits Model proofs.

Definition 3 (Hidden Bits Model Proofs) *A pair of probabilistic algorithms (P, V) is called a hidden bits proof system for L if V is polynomial time and the following two conditions hold:*

- Completeness: *For every $x \in L$,*

$$Pr\left[V(x, R_I, I, \pi) = 1\right] \geq \frac{2}{3}$$

  where $(I, \pi) \stackrel{def}{=} P(x, R)$, R is a random variable uniformly distributed over $\{0,1\}^{poly(|x|)}$, and R_I is a sub-string of R at positions $I \subseteq \{1, 2, \ldots, poly(|x|)\}$. That is $R_I = r_{i_1} \ldots r_{i_t}$, where $R = r_1 \ldots r_t$ and $I = (i_1, \ldots, i_t)$.

- Soundness: *For every $x \notin L$ and every algorithm B,*

$$Pr\left[V(x, R_I, I, \pi) = 1\right] \leq \frac{1}{3}$$

where $(I, \pi) \stackrel{def}{=} P(x, R)$, R is a random variable uniformly distributed over $\{0,1\}^{poly(|x|)}$, and R_I is a sub-string of R at positions $I \subseteq \{1, 2, \ldots, poly(|x|)\}$.

In both cases, I is called the set of revealed bits *and π is called the* certificate. *Zero-Knowledge is defined as in the case of NIZK proofs with the exception that we need to simulate $(x, R_I, P(x, R)) = (x, R_I, I, \pi)$ rather than $(x, R, P(x, R))$.*

F Proof of claim 6.1

Given the VRF (G, F, V), we add a special key pair (PK', SK') to the public key space of this VRF. And we modify the VRF to get a new VRF (G', F', V') as follows:

- The VRF generator G' either runs G or returns the special key pair (PK', SK') (with negligible probability).
- The VRF evaluator F' checks to see if it gets the special secret key SK'. If so, then it outputs the all zero string for any input. Else, it simply runs the original VRF evaluator F.
- The VRF verification algorithm V' also checks if the public key provided is the special public key PK'. If so, then it accepts if the output is the all zero string. Otherwise, it runs the original verification algorithm V.

It is not hard to see that the new VRF (G', F', V') is secure if the original VRF is secure. This is because the VRF generator G' outputs the special key pair only with a negligible probability. Indeed, if this VRF is used in the NIZK proof of [10], then a malicious prover can cheat by choosing this special public key to generate its proofs.

G Proof of claim 6.2

We will start by describing the NIZK proof system (P', V') for the language L. Let (P, V) be the hidden-bits proof system for L and $\pi = (G_\pi, \Pi, V_\pi)$ is a VRP on $\{0,1\}^n$. We will denote by $b_1 : \{0,1\}^n \to \{0,1\}$, the predicate that returns the first bit of its input. Now let the common input for (P', V') be $x \in \{0,1\}^n$ and the common reference string be $s = (s_1, \ldots, s_m)$, where each $s_i \in \{0,1\}^n$.

- *NIZK prover P':*
  1. Generate a VRP public/secret key pair $(PK, SK) \leftarrow G(1^\lambda)$.
  2. Compute $r_i \leftarrow b_1(\Pi_1(0, SK, s_i))$ for $i = 1 \ldots m$.
  3. Invoke the HBS prover P to get $(I, \tau) \leftarrow P(x, r_1, \ldots, r_m)$.
  4. Output $(PK, I, \tau, y_I, proof_I)$, where for $I = (i_1, \ldots, i_t)$, $y_I \stackrel{def}{=} (\Pi_1(0, SK, s_{i_1}), \ldots, \Pi_1(0, SK, s_{i_t}))$ and $proof_I \stackrel{def}{=} (\Pi_2(0, SK, s_{i_1}), \ldots, \Pi_2(0, SK, s_{i_t}))$

- *NIZK verifier V':* Given the output $(PK, I, \tau, y_I, proof_I)$ of the prover, the verifier performs the following operations:

    1. Perform the sequence of verifications $V_\pi(PK, s_{i_j}, y_j, proof_j)$, for each $i_j \in I$. If any of the verifications return NO, exit with failure.
    2. Compute $r_i = b_1(y_i)$ for $i = 1 \ldots t$, and let $r = r_1, \ldots, r_t$.
    3. Invoke V on (x, r, I, τ) and accept if and only if V accepts.

Completeness of the proof system (P', V') is implied directly from the completeness of the HBS proof system (P, V), since for the honest prover P' the hidden bits $r_i = b_1(\Pi_1(0, SK, s_i))$ will be uniformly random and V' accepts if and only if V accepts.

Soundness of the proof system (P', V') is again implied from the soundness of (P, V). Although we need to make sure that a malicious prover cannot cheat by choosing a fake public key $\widetilde{PK}$ for the VRP π. However, note that for any public key (even a maliciously chosen one) the VRP π still gives a permutation. This will in turn imply that each bit of the hidden bit string is random and unbiased. This is sufficient for the proof of Feige et al [8] to go through, except that in this case we have several possible permutations (one for each possible key) in comparison to [8] where only a unique permutation is considered. But this problem can be resolved easily by applying a union bound over all the possible public keys of the VRP π and amplifying the error probability by making the CRS sufficiently (although still polynomially) longer.

In order to show that the proof system (P', V') is zero knowledge, we will design a simulator S' for P' using the simulator S for hidden-bits model prover P. The simulator S' simulates the view of the verifier V' as follows:

1. Generate a public/secret key pair $(PK, SK) \leftarrow G(1^\lambda)$.
2. Run the hidden-bits model simulator S to get a simulated output (x, r_I, I, τ), where (I, τ) is the simulated output of P ,r_I denotes the revealed bits of the hidden-bit string and x is the common input.
3. For each bit r_j revealed by S, generate a uniformly random n bit string y_j such that $r_j = b_1(y_j)$. And set the corresponding n bit block in the CRS as $s_{i_j} = \Pi_1(1, SK, y_j)$ (i.e. $\pi_S^{-1} K(y_j)$). Set the remaining n-bit blocks in the CRS at random.
4. Output $(x, s, PK, I, y_I, \tau, proof_I)$, where $proof_i = \Pi_2(1, SK, y_j)$ for all $i \in I$.

It is not hard to see that if a distinguisher has a non-negligible advantage in distinguishing the view of the verifier in the proof system (P', V') and the output of S' then we can also use it to design a similar distinguisher for the corresponding hidden-bits model simulator S that also succeeds with non-negligible probability.