Weak Verifiable Random Functions

Zvika Brakerski[1], Shafi Goldwasser[1,2,⋆],
Guy N. Rothblum[2,⋆⋆], and Vinod Vaikuntanathan[2,3,⋆⋆⋆]

[1] Weizmann Institute of Science
[2] CSAIL, MIT
[3] IBM Research

Abstract. Verifiable random functions (VRFs), introduced by Micali, Rabin and Vadhan, are pseudorandom functions in which the owner of the seed produces a public-key that constitutes a commitment to all values of the function and can then produce, for any input x, a proof that the function has been evaluated correctly on x, preserving pseudorandomness for all other inputs. No public-key (even a falsely generated one) should allow for proving more than one value per input.

VRFs are both a natural and a useful primitive, and previous works have suggested a variety of constructions and applications. Still, there are many open questions in the study of VRFs, especially their relation to more widely studied cryptographic primitives and constructing them from a wide variety of cryptographic assumptions.

In this work we define a natural relaxation of VRFs that we call weak verifiable random functions, where pseudorandomness is required to hold only for randomly selected inputs. We conduct a study of weak VRFs, focusing on applications, constructions, and their relationship to other cryptographic primitives. We show:

- **Constructions.** We present constructions of weak VRFs based on a variety of assumptions, including general assumptions such as (enhanced) trapdoor permutations, as well as constructions based on specific number-theoretic assumptions such as the Diffie-Hellman assumption in bilinear groups.
- **Separations.** Verifiable random functions (both weak and standard) cannot be constructed from one-way permutations in a black-box manner. This constitutes the first result separating (standard) VRFs from any cryptographic primitive.
- **Applications.** Weak VRFs capture the essence of constructing non-interactive zero-knowledge proofs for all **NP** languages.

[⋆] Supported by NSF grants CCF-0514167, CCF-0635297, NSF-0729011, the Israel Science Foundation 700/08 and the Chais Family Fellows Program.

[⋆⋆] Supported in part by NSF grants CCF-0635297, CNS-0430336, NSF-0729011, Israel Science Foundation 700/08 and a Symantec Graduate Fellowship.

[⋆⋆⋆] Supported in part by NSF grants CCF-0635297 and Israel Science Foundation 700/08.

1 Introduction

Verifiable random functions (VRFs) were introduced by Micali, Rabin and Vadhan [1]. A VRF is a pseudorandom function (see Goldreich, Goldwasser and Micali [2]), that also enables a verifier to verify, given input x, output y and a proof, that the function has been computed correctly on x. The VRF's seed (or secret key) SK is associated with a public key PK. As usual, SK can be used to compute the function's output $y = F_{SK}(x)$ on input x, but it is also used to generate a *proof of correctness* $\pi = \Pi_{SK}(x)$. This proof can be used in conjunction with PK to convince a verifier that y is indeed the correct output on input x with respect to the public-key PK. Further, it is guaranteed that the verifier cannot accept two different values for an input x, even if PK is generated dishonestly. In other words, any string interpreted as a public-key of a VRF constitutes a commitment to at most one output per input.

VRFs are a natural primitive that combines the properties of pseudorandomness and verifiability; however, constructions of VRFs have been few and far between (see related work). In particular, unlike many central and natural cryptographic primitives, there are no known constructions of VRFs based on more *general assumptions*, such as the existence of a one-way function, or even the strong assumption of trapdoor permutations.

1.1 This Work

We propose a relaxation of VRFs, which we call *weak verifiable random functions* (or WVRF). Informally, a weak VRF is similar to a VRF, except that while VRFs require the output to be pseudo-random for *adversarially chosen* inputs, weak VRFs only require pseudorandomness to hold for *random inputs* (see Section 2 for the formal definition). Thus, weak VRFs are a natural relaxation of VRFs, analogous to the relaxation of *weak* pseudorandom functions (without verifiability), proposed by Naor and Reingold [3].

This work is a study of the power of weak VRFs. We present applications to building non-interactive zero-knowledge proofs by showing that the existence of non-interactive zero-knowledge proofs for all of **NP** in the common random string model is essentially equivalent to the existence of weak verifiable random functions in the *standard* model (i.e. without setup assumptions). Thus, we provide a new and conceptually simpler methodology for constructing and analyzing non-interactive zero-knowledge proof systems. We proceed by showing constructions of weak verifiable random functions from a variety of cryptographic assumptions (ones that are not known to imply standard VRFs). Finally, we present a black-box separation from other widely-studied cryptographic primitives (namely one-way permutations). These separations are also the first known separations of standard VRFs from any other cryptographic primitive. We proceed with an overview of each of these contributions.

Weak VRFs and NIZK. We begin by showing an intimate connection between weak VRFs and the study of non-interactive zero-knowledge proofs (NIZKs) for all **NP** languages (with an efficient-prover and in the common random string model). In one direction, we show that a weak VRF can be used to construct a NIZK for any **NP** language. This construction is based on the methodology of Feige, Lapidot and Shamir [4], and shows that weak VRFs can be used to implement their hidden-bits model. In a nutshell, to implement the hidden-bits model, we need a method for a prover (with a secret key) and a verifier (with a public key) to interpret a common random string as a sequence of bit commitments. Towards this end, we give the verifier a public key that guarantees that the commitments are binding, and the prover a secret key that allows non-interactive de-commitment. Weak VRFs are a natural solution to this problem. Modulo the technical details, we can implement the hidden-bits model by taking a random bit sequence x to be a commitment to the output of the weak VRF on input x. The verifier, given the weak VRF's public key, is guaranteed binding, and the pseudo-randomness (even though it only holds for *random* inputs) guarantees hiding. There is a subtle technical problem with the above construction, where the prover may choose a particularly badly formed public key, this is resolved using a certification technique due to Bellare and Yung [5].

Lemma 1. *If there is a family of weak verifiable random functions, then there are efficient-prover non-interactive zero-knowledge proofs in the common random string model for all of* **NP***.*

In the other direction, we show that given a construction of (efficient-prover) NIZKs for all **NP** languages, and given also an injective one-way function, we can construct a weak VRF.[4] First recall that the existence of injective OWFs implies the existence of (ordinary) pseudorandom functions (PRFs) and non-interactive perfectly binding commitment schemes. The WVRF is as follows: the generator produces a seed for a PRF and uses it as secret key. The public key is a commitment to that seed. For a random input we use a part of the input as an input to the PRF. The weak VRF's output will be the output of the PRF on this input. The second part of the input is used as a CRS for a NIZK proof. The novelty of this construction is observing that since pseudorandomness should only hold when the adversary sees randomly distributed samples, the *input* can be used as an "honest" source of randomness (CRS in this case). Weak pseudorandomness follows, therefore, from the zero-knowledge property.

Lemma 2. *If there exist injective one-way functions and efficient-prover non-interactive zero-knowledge proofs in the common random string model for all of* **NP***, then there is a family of weak verifiable random functions.*

Thus, weak VRFs and efficient prover NIZK proof systems are essentially equivalent. We hope that weak VRFs, being a "clean" and natural primitive, will prove to be a useful tool or abstraction leading to new constructions of NIZK proof systems. See Section 3 for all details of the above relationship.

[4] In fact, we can replace the injective one-way function with a standard one-way function by either (a) allowing the prover and verifier to be non-uniform or (b) using derandomization assumptions (see Barak, Ong and Vadhan [6]).

Constructing Weak VRFs. We show efficient constructions of weak VRFs based on a variety of assumptions. We consider both general assumptions, such as the existence of enhanced trapdoor permutations, and specific number-theoretic assumptions such as the bilinear decisional Diffie-Hellman (BDDH) assumption. We note that none of the assumptions we use to construct weak VRFs are known to imply the existence of (strong) VRFs.

The first construction is a black-box construction using any (enhanced) trapdoor permutation, obtaining WVRFs with arbitrary long (polynomial) output length.[5] The second construction (with a single output bit) is based on the bilinear Diffie-Hellman assumption. In contrast, all known constructions of (strong) VRFs from bilinear maps make substantially stronger assumptions.

We note that these constructions are implicit in known constructions of non-interactive zero-knowledge proofs from trapdoor permutations in [4] and based on bilinear maps on elliptic curves in [7]. This is not surprising in light of the equivalence to NIZK proof systems, and in fact further reinforces our belief that weak VRFs are a natural primitive to focus on when constructing non-interactive zero-knowledge proofs.

See Section 4 for the full details of all the above constructions.

Black-Box Separations. Finally, we initiate a study of the relationship between weak VRFs and more extensively studied cryptographic primitives. We show that there are no black-box constructions of weak VRFs from one-way permutations (OWPs). We note that, given that both pseudorandom functions and signature schemes can be (black-box) constructed even from one-way functions [2, 8–10], one might have hoped that it is possible to combine both the pseudorandomness and the verifiability properties and to construct weak VRFs (or even standard, strong VRFs) from one-way functions. However, while in signature schemes verifiability is guaranteed if the public-key has been selected properly, in VRFs we require verifiability even for adversarial ones. Indeed, our result shows that this cannot be done. We note that this is also the first separation result for (strong) VRFs, and thus sheds light on their complexity as well.

Formally, we show a black-box separation (see related work) between weak VRFs and OWPs. In fact we show this even for weak verifiable unpredictable functions, where pseudo-randomness is replaced with unpredictability.

Theorem 1 (informal). *There is no black-box construction of weak VRFs from OWPs.*

We prove this theorem by showing that any black-box construction of weak VRF fails with respect to some oracle implementing a OWP. From this result it follows immediately that VRFs also cannot be black-box reduced to OWPs.

This result differs from previous work on black-box separations (see related work below) in several ways. Unlike the seminal result of Impagliazzo and Rudich separating one-way permutations and key agreement protocols [11], we do not

[5] Note that we can always increase the output length of weak VRFs by concatenating multiple instances, but here this is done without increasing the key lengths.

show a relativizing separation. That is, we do not prove that there exists a oracle relative to which OWPs exist but weak VRFs do not. Moreover, our method is also different from the one introduced by Gertner, Malkin and Reingold [12] for proving a separation between trapdoor predicates and trapdoor functions. There, each construction is tailored with an oracle relative to which it is proved insecure. Informally speaking, we show that there exists an oracle $\mathcal{O}$ (that implements a OWP and encodes an **NP**-hard oracle) such that any construction that is correct (namely, both complete and sound) with respect to all one-way permutation oracles, fails to be pseudorandom with respect to $\mathcal{O}$ (see the proof intuition below for more details).

We remark that since we show that (enhanced) trapdoor permutations imply weak VRFs, it also follows from our separation that there is no black-box construction of such trapdoor permutations from one-way permutations. While this was already known as a corollary from [11], our proof is simpler (albeit somewhat more restricted).

Proof Intuition. Our adversary works by “reverse engineering” the oracle queries made by the secret and public key generator of the weak VRF. Essentially, what we want to do, is use the oracle (that can solve **NP**-hard problems) to find a “fake” secret-key corresponding to the given public key. This fake secret key will later be used (by the adversary) to predict the outputs of the weak VRF. The intuition is that these predictions (by the fake secret key) should be correct because they match the same public key as the one used with the “real” secret key. Note that this holds because we are assuming here strong “completeness” of the key generation algorithm: namely, it *always* (with probability 1) generates a valid secret-public key pair, and so even the fake secret key that we found “should” generate correct outputs. We need this completeness property (which all our constructions possess) to prove our separation result.

The problem we face is that the key-generator is itself an oracle circuit, and thus we cannot simply find a fake secret key that corresponds to the given public key and the given OWP oracle. We can, however, use our ability to solve **NP**-hard problems (via the given oracle) to find a fake secret key that matches the given public key and a (slightly) different oracle, by changing the oracle’s answers on queries made by the key generator. Consider an oracle query that was made by the generator, and consider the case of changing the answer for this query in a way that does not affect the public key. This change may affect some values of the function, and so we are no longer guaranteed that the function’s output on the real and fake secret key are the same. Note, however, that the function’s output can only change when the verification algorithm makes an oracle query to a value whose output has changed. This follows from the weak VRF’s soundness: the verification algorithm must make such an oracle query in order to tell the two cases apart. Therefore, if we take enough random samples, run the verification algorithm and “collect” its OWP oracle queries, we’d have a bank of “common” queries that holds all queries that affect a large portion of the values of the function. Then we can find, using the **NP**-hardness of the oracle, a secret key and simulated values to all other queries made by the generator (answers to the “common” queries are unchanged) that yield the same public-key. Changing these “uncommon” values, however, would not affect the value of the function almost anywhere, and thus we can use the fake secret key we found to predict the value of the function on a random point.

1.2 Related Work

Verifiable Random Functions. Bellare and Goldwasser [13] present a signature scheme based on combining a PRF and a NIZK proof system. While their scheme implies a PRF with verifiability properties, a falsely generated verification-key may enable the prover to make the verifier accept more than one output per input. Thus this construction falls short of the soundness requirement for a VRF.

Known constructions of VRFs are due to [1] based on strong RSA, Lysyanskaya [14] based on a strong version of the Diffie-Hellman assumption in bilinear groups, Dodis [15] based on the sum-free generalized DDH assumption, and Dodis and Yampolskiy [16] based on the bilinear Diffie-Hellman inversion assumption.[6]

Variants of VRFs have also been proposed and used for various applications, for instance the notion of of simulatable VRFs, introduced by Chase and Lysyanskaya [17], which were used to compile any single-theorem non-interactive zero-knowledge proof for a language L into a many-theorem non-interactive zero-knowledge proof for the same L. We stress that simulatable VRFs are defined in the public-parameters model and are incomparable to standard VRFs.

Non-Interactive Zero-Knowledge and Related Primitives. Non-interactive zero-knowledge proofs (NIZK)[7], introduced by Blum, Feldman and Micali [18], are proof systems where the prover P sends a single message to the verifier V to convince V of an (**NP**) statement, while conveying no more knowledge to the verifier except that the statement is true. NIZK proof systems have been immensely useful, including in the construction of non-malleable and chosen-ciphertext secure encryption schemes [19–25], signature schemes [13] and more. There have been a handful of constructions of NIZK proof systems from the time they were introduced, based on specific number theoretic assumptions and based on general assumptions (see below).

Specific number-theoretic assumptions that imply NIZKs include quadratic residuosity [18], the computational Diffie-Hellman assumption on (prime-order) bilinear groups due to Canetti, Halevi and Katz [7], and constructions due to Groth, Ostrovsky and Sahai based on the subgroup-decisional assumption on

[6] We note that both [1] and [16] construct VRFs with polynomial-size domains and later extend it to arbitrary domains via a tree-based construction, which impacts their efficiency.

[7] We stress that here we are dealing with computational NIZK *proofs*, as opposed to statistical NIZK *arguments*. In the latter, the soundness property holds only against cheating provers that are computationally bounded.

composite-order bilinear groups [26], and on the decisional linear assumption on (prime-order) bilinear groups [27].

Many works have investigated constructions of NIZK proofs based on *general assumptions* – these include the construction of [4] based on (enhanced) trapdoor permutations (improved by Kilian and Petrank [28] for better efficiency), and more recently, the construction of [7] based on what they call verifiable trapdoor predicates. Both these constructions use primitives that have an explicit trapdoor structure which may not be inherent. Dwork and Naor [29] showed that NIZK proof systems can be constructed using a (strong) VRF (Dodis and Puniya [30] show an alternative construction by going through their notion of a verifiable random permutation). Our construction is from a *weak VRF* and is much more direct.

Goldwasser and Ostrovsky [31] proposed a new primitive called invariant signature schemes and showed that in the CRS model they are *equivalent* to non-interactive zero-knowledge proofs for all of **NP**. In [29], verifiable pseudorandom generators (VPRGs) are presented and shown to be equivalent to NIZK proofs in the CRS model. A VPRG is essentially a pseudorandom generator, such that the owner of the seed can post proofs of correctness for subsets of the generated bits, while maintaining hiding of the rest of them. It is shown that the existence of VPRGs in the CRS model is equivalent to the existence of NIZK in that model. Approximate VPRG is a variant of this notion, where the soundness requirement of the proof is relaxed. Approximate VPRGs in the *standard model* exist if and only if NIZK exists in the *CRS model*.

Black-Box Separations. A black-box reduction from primitive A to primitive B is essentially a construction of A that uses an oracle to B, such that the security of B implies the security of A. Most known reductions between cryptographic primitives are black-box. In [11], it was shown for the first time that it is possible to rule out the existence of black-box reductions between some primitives.[8] Such proofs of impossibility are referred to as *black-box separations.* Their result has been followed by many others, showing impossibility of various classes of black-box reductions between various cryptographic primitives and protocols. For a classification of black-box reductions (and separations), we refer the reader to the work of Reingold, Trevisan and Vadhan [32].

2 Preliminaries and Definitions

Verifiable Random Functions. We use the definition of verifiable random functions from Micali, Rabin and Vadhan [1]. A key feature of VRFs is their soundness property: soundness requires that no two distinct values can be proven to be $F_{SK}(x)$, for *any* PK, *and any* x, even ones that are adversarially chosen. This

[8] Specifically, [11] show that there is no *relativizing* reduction from secure key-agreement protocols to one-way permutations, thus ruling out black-box reductions where the proof of correctness is also black box. This has been improved by [32] to also rule out cases where the proof of correctness has "some" non-black-boxness.

is a crucial difference between VRFs and other cryptographic primitives such as encryption and digital signatures, where the public/secret-keys are assumed to be chosen correctly. For a formal definition of VRFs, we refer the reader to [1].

Weak Verifiable Random Functions. Weak verifiable random functions maintain the key feature of VRFs, namely that even if the public-key PK is adversarially chosen, it is impossible for the adversary (even one who knows SK) to prove that $y = F_{SK}(x)$ and $y' = F_{SK}(x)$ for two different y and y'. However, in the case of weak VRFs, we relax this condition slightly by saying that *for every* PK, the completeness and soundness conditions hold *for most inputs* x (and not necessarily all the inputs, as in the case of standard VRFs). We stress that there are *no public-keys* PK for which the completeness and/or the soundness conditions fail on a large fraction of inputs.

The other major difference between the definitions of weak VRFs and standard VRFs is in the pseudorandomness condition: whereas in the case of VRFs, pseudorandomness holds against an adversary that can adaptively choose inputs x and obtain evaluations $F_{SK}(x)$, the weak VRF adversary gets evaluations of $F_{SK}(x)$ on *random values* x. This is in the spirit of weak PRFs presented in [3].

Definition 1 (Weak Verifiable Random Function). *A family of functions* $\mathcal{F} = \{f_s : \{0,1\}^{n(k)} \rightarrow \{0,1\}^{m(k)}\}_{s\in\{0,1\}^k}$ *is a family of* weak verifiable random functions *with security parameter* k *if there exist algorithms* (G, F, Π, V) *such that: the key-generation algorithm* $G(1^k)$ *is a PPT algorithm that outputs a pair of keys* (PK, SK)*; the function-evaluator* $F_{SK}(x)$ *is a deterministic algorithm that outputs* $f_{SK}(x)$*; the Prover* $\Pi_{SK}(x)$ *is a deterministic algorithm that outputs a proof of correctness* π *and the Verifier* $V(PK, x, y, \pi)$ *is a PPT algorithm that either accepts or rejects a purported proof* π *of the statement "*$y = F_{SK}(x)$*".*

We require the following:

1. (Relaxed) Completeness: *for all* $(PK, SK) \leftarrow G(1^k)$ *and for* all but a 2^{-k} fraction of x's, *if* $y = F_{SK}(x)$ *and* $\pi = \Pi_{SK}(x)$*, then* $\Pr[V(PK, x, y, \pi) = \mathsf{accept}] \geq 1 - 2^{-k}$*. The probability here is taken over the random coins of* V*.*
2. (Relaxed) Soundness: *for all* PK *and for all but a* 2^{-k} *fraction of* x*'s, and for all* y_1, y_2, π_1, π_2 *such that* $y_1 \neq y_2$*,* $\Pr[V(PK, x, y_i, \pi_i) = \mathsf{accept}] \leq 2^{-k}$ *for at least one* $i \in \{1, 2\}$*.*
3. Weak Randomness: *let* A *be a PPT algorithm, and let* $p(k)$ *be any polynomial. Then, the probability that* A *succeeds in the following experiment is at most* $\frac{1}{2} + \mathsf{negl}(k)$*:*
   $(PK, SK) \leftarrow G(1^k)$
   Choose $x_1, x_2, \ldots, x_{p(k)} \overset{R}{\leftarrow} \{0,1\}^{n(k)}$*,* $x^* \overset{R}{\leftarrow} \{0,1\}^{n(k)}$ *and* $b \overset{R}{\leftarrow} \{0,1\}$*.*
   If $b = 0$*, set* $y^* = F_{SK}(x^*)$*, otherwise set* $y^* \overset{R}{\leftarrow} \{0,1\}^{m(k)}$
   $b' \leftarrow A(1^k, PK, \{x_i, F_{SK}(x_i), \Pi_{SK}(x_i)\}_{i=1}^{p(k)}, x^*, y^*)$
   A succeeds *if* $b' = b$*.*

Weak verifiable unpredictable functions (VUF) are the same as the above, except that the weak randomness requirement is replaced by the weak unpredictability requirement below.

$3'$. Weak Unpredictability: *Consider the following experiment with the adversary A, and let $p(k)$ be any polynomial.*
$(PK, SK) \leftarrow G(1^k)$
Choose $x_1, x_2, \ldots, x_{p(k)} \leftarrow \{0,1\}^{n(k)}$, $x^* \leftarrow \{0,1\}^{n(k)}$.
$y^* \leftarrow A(1^k, PK, \{x_i, F_{SK}(x_i), \Pi_{SK}(x_i)\}_{i=1}^{p(k)}, x^*)$
A succeeds *if* $y^* = F_{SK}(x^*)$. *We require that the probability that A succeeds is at most* $\mathsf{negl}(k)$.

3 Weak Verifiable Random Functions and NIZK Proofs

In this section, we show that weak verifiable random functions and non-interactive zero-knowledge proofs are essentially equivalent. First, in Lemma 2, we construct a weak VRF given a non-interactive zero-knowledge (NIZK) proof system for all of **NP** (with an efficient prover, in the common random string model) and an injective one-way function. Secondly, in Lemma 1, we construct NIZK proof systems for every **NP** language, given any weak VRF.

Lemma 2 (restated). *If there exist injective one-way functions and efficient-prover non-interactive zero-knowledge proofs in the common random string model for all of* **NP** *, then there is a family of weak verifiable random functions.*

Proof. The construction is very similar to the construction of a signature scheme from (enhanced) trapdoor permutations, due to [13]: the difference is that in [13], the common random string for the NIZK proof system is part of the public-key of the resulting signature scheme, whereas in our case, it is part of the *input.* Informally speaking, the reason for this difference is that in a signature scheme, the public-key is completely trusted, whereas this is not the case for (both strong and weak) VRFs.

The key-generation algorithm picks s and s', two independent seeds for a pseudorandom function. The public-key PK for the WVRF is the commitment of the seed s, using randomness ρ. The secret-key SK is (s, ρ, s'). Namely, $PK = \textsc{com}(s; \rho)$ and $SK = (s, \rho, s')$. The function $F_{SK}(r||x)$ parses its input as r and x and outputs $f_s(x)$. The proof generator Π does the following: define the **NP** language

$$L = \{(PK, x, y) \mid \exists s, \rho \text{ such that } PK = \textsc{com}(s; \rho) \text{ AND } y = f_s(x)\}$$

Π runs the prover algorithm for the NIZK proof system for the language L using r as the common random string. The randomness of the prover is $f_{s'}(x)$, and the output of the prover is the proof π. It is easy to see that given SK and x, the proof-generator Π is deterministic. The verifier V, given PK, x, y and π, runs the NIZK verifier on input the statement (PK, x, y) and the proof π and accepts if and only if the NIZK verifier accepts.

This construction assumes a pseudorandom function (which can be constructed from any one-way function [2, 8]) and a non-interactive commitment scheme (which can be constructed from any injective one-way function, see Blum and Micali [33]).

Completeness of the WVRF follows from the perfect completeness of the NIZK proof system. Pseudorandomness follows via a standard hybrid argument, which we omit for lack of space.

Relaxed soundness follows from the perfect binding of the commitment scheme and the soundness of the NIZK proof system. Slightly more precisely, given any PK, there is at most one s such that $PK \in \textsc{com}(s;\cdot)$ (where $\textsc{com}(s;\cdot)$ denotes the set of all commitments of the string s). Thus, for all $y' \neq f_s(x)$, it follows that $(PK, x, y') \notin L$. By the soundness of the NIZK proof system, this means that with high probability over the input (that is, over r) the verifier will not accept any purported proof of the statement (PK, x, y') with high probability over its coin-tosses. □

Next, we show how to construct NIZK proofs for all of NP from any weak VRF. We do this by implementing the hidden-bit model of [4] using any weak VRF.

Lemma 1 (restated). *If there is a family of weak verifiable random functions, then there are efficient-prover non-interactive zero-knowledge proofs in the common random string model for all of* **NP**.

Informally, the idea for implementing the hidden-bits proof system is to let the prover P choose a pair of keys (PK, SK) for the weak verifiable random function, and let the hidden bits $(b_1, \ldots, b_m)$ (for some $m = \text{poly}(n)$) be defined as $b_i = F_{SK}(r_i)$, where $(r_1, \ldots, r_m)$ is the first part of the common random string. The prover can reveal any subset of the bits, simply by giving the verifier the proof $\Pi_{SK}(r_i)$ for the corresponding bits.

One potential problem is that the prover can select (PK, SK) depending on the common random string and potentially violate soundness. This is solved in the standard way of [4] by reducing the soundness error of the NIZK proof in the hidden-bit model.

A more subtle problem is that the prover may select (PK, SK) such that $F_{SK}(\cdot)$ is heavily unbalanced, thus introducing a bias into the distribution of the hidden bits. We handle this in a way that is similar to a certification procedure developed in [5].

We refer the reader to the full version [34] for a complete proof of this lemma.

4 Constructions of Weak Verifiable Random Functions

In this section, we show two efficient constructions of weak verifiable random functions (WVRF), as outlined in the introduction.

Construction from Trapdoor Permutations. For simplicity, we describe the construction from any (enhanced) *certified* trapdoor permutation, namely given a function f, it is possible in polynomial time to check that f indeed defines a one-to-one and onto function. This construction can be made to work with any (enhanced) trapdoor permutation, using a certification procedure of Bellare and Yung [5].

Let (f, f^{-1}) be an enhanced certified trapdoor permutation. Then, the construction of a WVRF (G, F, Π, V) is as follows: The key-generation algorithm G, on input 1^k, outputs $PK = f$, and $SK = f^{-1}$, where (f, f^{-1}) is a random trapdoor permutation together with its trapdoor. Let $f^{-i}(x)$ denote the result of f^{-1} applied i times to the input x. F_{SK} parses its input as (x, r), and outputs $(b_1, \ldots, b_\ell)$ where $b_i = \langle f^{-i}(x), r \rangle$. $\Pi_{SK}(x, r)$ outputs $f^{-(\ell+1)}(x)$. The verification algorithm V, given $PK, (x, r)$ and y, accepts if and only if $\langle f^i(\pi), r \rangle = b_{\ell-i+1}$ for all $1 \leq i \leq \ell$ and $f^{\ell+1}(\pi) = x$.

To sketch the proof of this construction, observe that perfect completeness is immediate. Soundness follows from the fact that f is a (certified) permutation. Pseudorandomness follows from the one-wayness of f, as well as the fact that we use the Goldreich-Levin hardcore bit.

Construction from the Computational Diffie-Hellman Assumption in Gap-DDH groups. Let $\mathbb{G}$ and $\mathbb{G}'$ be groups of prime order q, with a bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}'$. Let g be a generator of $\mathbb{G}$. The WVRF (G, F, Π, V) is defined as follows: the key-generation algorithm $G(1^k)$ outputs $PK = g^a$ and $SK = a$, where a is a random element in $\mathbb{Z}_q$. $F_{SK}(r)$ uses r to sample a random element R in $\mathbb{G}$, [9] and outputs a hardcore bit of R^a (for example, the most significant bit of R^a). $\Pi_{SK}(r)$ simply outputs R^a. The verification algorithm, on input PK, x, y and π, accepts if $e(PK, x) = e(g, \pi)$ and y is the hardcore bit of π.

The fact that this is a weak VRF follows from the Diffie-Hellman assumption. The formal proof is omitted from this extended abstract.

5 Separations

In this section, we show a black-box separation between weak verifiable unpredictable functions (weak VUFs) and one-way permutations. Recall that both weak and standard VRFs are in particular also weak VUFs, and that weak VRFs can be constructed in a fully black-box manner from (enhanced) trapdoor permutations (eTDPs, see Section 4). This result, therefore, implies a separation between weak VRFs, standard VRFs and eTDPs and one-way permutations.

Technically, we show that there is no *semi black-box* reduction (a notion defined in [32], included below) from a weak VUF to a one-way permutation. In other words, we show that for every construction of a weak VUF from a one-way permutation, there is an oracle (which possibly depends on the construction) such that the construction fails with respect to the oracle.[10]

[9] In the case where $\mathbb{G}$ is a subgroup of an elliptic curve group, the sampling can be done efficiently. See the full version [34] for details.

[10] We note that our reduction does not preclude a relativizing reduction. Ruling out a relativizing reduction involves constructing an oracle relative to which no secure weak VUF exists. For more details on the different types of black-box reductions, see [32].

Definition 2 ([32]). *A tuple of oracle algorithms* (G, F, Π, V) *is a* Semi-BB *reduction from weak verifiable unpredictable functions to one-way permutations:*

- **Correctness.** *For every permutation* $\mathcal{O}$*,* $(G^{\mathcal{O}}, F^{\mathcal{O}}, \Pi^{\mathcal{O}}, V^{\mathcal{O}})$ *has (relaxed) completeness and soundness as in Definition 1.*
- **Security.** *For every permutation* $\mathcal{O}$*, if there exists a PPT oracle machine* A *such that* $A^{\mathcal{O}}$ *predicts* $(G^{\mathcal{O}}, F^{\mathcal{O}}, \Pi^{\mathcal{O}}, V^{\mathcal{O}})$ *in the sense of Definition 1, then there exists a PPT oracle machine* S *such that* $S^{\mathcal{O}}$ *inverts* $\mathcal{O}$ *with non-negligible probability.*

Using the definition above, we can formally state our claim. We show that the following holds.

Theorem 1 (formally stated). *There is no semi black-box reduction from a weak VUF to a one-way permutation. Namely, for every construction* (G, F, Π, V) *of a weak VUF, there is an oracle* $\mathcal{O}$ *such that* $(G^{\mathcal{O}}, F^{\mathcal{O}}, \Pi^{\mathcal{O}}, V^{\mathcal{O}})$ *is, in the terms of Definition 2, either incorrect or insecure.*

In the remaining of this section, we provide a sketch of the proof of Theorem 1 (Section 5.1, for the full proof, see the full version of this paper) and conclude with some remarks on limits and extensions of the proof (Section 5.2).

5.1 Proof Sketch of Theorem 1

The proof proceeds by contradiction. Fix (towards contradiction) some semi black-box reduction (see Definition 2) (G, F, Π, V). For any oracle $\mathcal{O}$ that implements a one-way permutation, $(G^{\mathcal{O}}, F^{\mathcal{O}}, \Pi^{\mathcal{O}}, V^{\mathcal{O}})$ is a weak VUF. For any such reduction, we show an oracle $\mathcal{O}$ and an adversary $A^{\mathcal{O}}$ that breaks the weak unpredictability of the defined weak VUF (w.r.t $\mathcal{O}$). Throughout the proof, let t_G (resp. t_V) denote the (polynomial in k) running times of $G^{\mathcal{O}}$ (resp. $V^{\mathcal{O}}$). For simplicity, we will assume that the verifier $V^{\mathcal{O}}$ is deterministic, throughout the rest of this proof.[11]

The oracle $\mathcal{O}$ is similar to the one presented in [32].[12] Roughly speaking, $\mathcal{O}$ both implements a one-way permutation (that is, no adversary with oracle access to $\mathcal{O}$ can compute x given $\mathcal{O}(x)$, for a random $x \in \{0,1\}^n$), and is **NP**-hard (namely, with oracle access to $\mathcal{O}$, it is possible to decide every language in **NP**). A formal statement follows.

Proposition 1 (implicit in [32]). *There exists an oracle* $\mathcal{O}$ *which is (i)* A length preserving permutation*; (ii)* One-way: *there exists no PPT oracle machine* A *s.t.* $A^{\mathcal{O}}$ *inverts* $\mathcal{O}$*; and (iii)* **NP**-hard: *for any* **NP** *relation* $\mathcal{R}$*, there exists a polynomial-time oracle machine* B *that for any* x *where* $\exists y.(x, y) \in \mathcal{R}$*,* $B^{\mathcal{O}}$ *finds such a* y*, namely:* $(x, B^{\mathcal{O}}(x)) \in \mathcal{R}$*.*

[11] However, see remark on handling probabilistic verifiers in Section 5.2.

[12] In [11], two oracles are used: a random oracle and a **PSPACE**-complete oracle. [32] show how this can be simplified into one oracle that is both a one-way permutation and is **PSPACE**-hard (the same argument holds for **NP**-hardness as well).

We want to use the power of $\mathcal{O}$ to construct an adversary that predicts the weak VUF. Given a public-key PK, this can be done by finding a secret-key SK' such that (PK, SK') is a possible output of $G^{\mathcal{O}}(1^k)$ (this follows from completeness and soundness of the weak VUF). However, this requires finding a witness for an $\mathbf{NP}^{\mathcal{O}}$ relation, a task beyond the powers of our oracle. We thus relax the requirement. We present an **NP** relation that enables finding a secret-key SK' and an oracle $\mathcal{O}'$ such that (PK, SK') is a possible output of $G^{\mathcal{O}'}(1^k)$. Furthermore, $\mathcal{O}'$ is only a slight modification of $\mathcal{O}$: $\mathcal{O}'$ and $\mathcal{O}$ agree on almost all inputs, and particularly on a set of "significant" inputs. We then show that such $SK', \mathcal{O}'$ can be used to predict the weak VUF. A detailed description follows.

We define an **NP**-relation $\mathcal{R}$ that will enable us to find SK' and a transcript of oracle query/answers (which will define $\mathcal{O}'$) that are consistent with PK and with a predefined query bank (a set of queries and answers from $\mathcal{O}$). The query bank will formally be represented by a set of queries $Q \subseteq \{0,1\}^*$ and a function $f_Q : Q \to \{0,1\}^*$ mapping them to answers. The input of $\mathcal{R}$, therefore, is formally denoted $z = (1^k, PK, Q, f_Q)$ (where 1^k is the security parameter). The corresponding witness consists of the new secret key SK', along with the rest of the information that enables simulating the generation of (PK, SK'): the randomness r that G uses, and the queries not in Q that G made, along with their respective answers. These are represented by $D \subseteq \{0,1\}^*$, $f_D : D \to \{0,1\}^*$ (the same way as Q). We require that $|r|, |D| \leq t_G(k)$. Formally, the witness for relation $\mathcal{R}$ is denoted $w = (r, SK', D, f_D)$ and $(z, w) \in R$ if (PK, SK') are produced by an execution of G with security parameter 1^k and randomness r, which makes oracle queries in $Q \cup D$, and gets answers according to f_Q, f_D. The verification procedure $\mathrm{Ver}_{\mathcal{R}}(z, w)$ for $\mathcal{R}$ simply simulates G for at most $t_G(k)$ steps and checks that (z, w) are consistent with the above.

Using an **NP**-hard oracle, we can compute a witness of $\mathcal{R}$ for any input, if such exists. We further note that if f_Q is consistent with $\mathcal{O}$, and if PK was in fact generated by $G^{\mathcal{O}}(1^k)$, then there always exists at least one witness for that input: the one that contains the actual random tape, secret key and oracle query/answers that were used in the generation of PK.

We are now ready to describe the adversary algorithm $A^{\mathcal{O}}$ (recall that A has oracle access to $\mathcal{O}$). For the remainder of the proof, fix the (PK, SK) generated by $G(1^k)$ for the unpredictability challenge.

The Adversary Algorithm The adversary A operates in two stages. In the first stage, the *"exploration stage"*, the adversary receives a public-key PK for the weak VUF, as well as polynomially many evaluations of $F^{\mathcal{O}}_{SK}, \Pi^{\mathcal{O}}_{SK}$ on random inputs x_i. The adversary tries to learn the random-oracle queries that are "significant" in computing the function, and outputs a bank of oracle queries and answers. In the second stage, the *"conquering stage"*, the adversary (using the bank of queries) constructs a secret-key SK' and an (implicit) oracle $\mathcal{O}'$ for the same PK such that $F^{\mathcal{O}'}_{SK'}$ and $F^{\mathcal{O}}_{SK}$ coincide on most inputs. This enables the adversary to predict the value of $F^{\mathcal{O}}_{SK}$ on most inputs, in turn. The description of A follows.

The **exploration stage** of the adversary A.

INPUT: 1^k, PK and $\{x_i, y_i, \pi_i\}_{i=1}^{k^2 t_G(k)}$, where $(PK, SK) \leftarrow G^{\mathcal{O}}(1^k)$, $y_i = F_{SK}^{\mathcal{O}}(x_i)$ and $\pi_i = \Pi_{SK}^{\mathcal{O}}(x_i)$.

OUTPUT: A bank of queries consisting of a set of queries Q and a mapping $f_Q : Q \to \{0,1\}^*$ matching answer $f_Q(q) = \mathcal{O}(q)$ to every answer q.

ALGORITHM:

1. Initialize the bank of queries $Q, f_Q = \emptyset$.
2. For $i = 1, \ldots, k^2 \cdot t_G(k)$ run $V^{\mathcal{O}}(PK, x_i, y_i, \pi_i)$. Save all the query-answer pairs made by V to the oracle $\mathcal{O}$ into the query bank (Q, f_Q). Output (Q, f_Q).

The **conquering stage** of the adversary A.

INPUT: PK, query bank (Q, f_Q) and a challenge $x^* \overset{R}{\leftarrow} \{0,1\}^{n(k)}$.

OUTPUT: $y^* \in \{0,1\}^{m(k)}$.

ALGORITHM:

1. Let $z = (1^k, PK, Q, f_Q)$ be an input for **NP** relation $\mathcal{R}$ described above, we can use $\mathcal{O}$ (which is **NP**-hard) to compute a witness $w = (r, SK', D, f_D)$ such that $(z, w) \in \mathcal{R}$ (as we mentioned, such witness must exist).
2. For all $q \in \{0,1\}^*$, define

$$\mathcal{O}'(q) = \begin{cases} f_D(q), \, q \in D, \\ \mathcal{O}(q), \;\; \text{otherwise.} \end{cases}$$

   Note that $\mathcal{O}'(q)$ can be computed in polynomial time given access to $\mathcal{O}, D, f_D$. Using SK' and $\mathcal{O}'$, return $y^* = F_{SK'}^{\mathcal{O}'}(x^*)$.[13]

Analysis of the Adversary Recall that we fixed PK, SK. We first define a notion of "frequent oracle queries" of the verification algorithm (with respect to PK and SK) and show that the bank of queries (Q, f_Q) that the adversary outputs in the exploration stage contains all the frequent oracle queries of the verification algorithm, with high probability.

We define the frequency $\mathsf{freq}(q)$ of a query q to the oracle $\mathcal{O}$ (with respect to PK and SK) to be the fraction of x's for which the verification algorithm, on input PK, x, $F_{SK}(x)$ and $\Pi_{SK}(x)$, makes the query q to the oracle $\mathcal{O}$ during its execution. More precisely,

$$\mathsf{freq}(q) = \Pr_{x \in \{0,1\}^{n(k)}} \left[V^{\mathcal{O}}(PK, x, F_{SK}^{\mathcal{O}}(x), \Pi_{SK}^{\mathcal{O}}(x)) \text{ makes query } q \text{ to } \mathcal{O} \right]$$

A query q is called α-frequent if $\mathsf{freq}(q) \geq 1/\alpha$. Let $\mathcal{F}_{\alpha(k)}$ be the set of all $\alpha(k)$-frequent queries. That is, $\mathcal{F}_{\alpha(k)} = \{q : \mathsf{freq}(q) \geq 1/\alpha(k)\}$. The following lemma states that the exploration stage succeeds in finding all frequent queries with very high probability.

[13] We remark that $\mathcal{O}'$ as defined is not necessarily a permutation, so $F_{SK'}^{\mathcal{O}'}(x^*)$ may not be well defined. In the full version [34] we show how this is fixed by defining a permutation $\mathcal{O}'$ s.t. $|\{q : \mathcal{O}(q) \neq \mathcal{O}'(q)\}| \leq 2|D|$. For the remaining of the analysis, we assume that $\mathcal{O}'$ is a permutation.

Lemma 3 (exploration stage). *Let $\alpha(k) = k \cdot t_G(k)$. With probability at least $1 - \text{poly}(k) \cdot e^{-k}$, at the end of the exploration stage of A, $Q \supseteq \mathcal{F}_{\alpha(k)}$.*

Proof. Consider an $\alpha(k)$-frequent query $q \in \mathcal{F}_{\alpha(k)}$. By definition, $\mathsf{freq}(q) \geq 1/\alpha(k)$. That is, for at least $1/\alpha(k)$ fraction of x's, $V^{\mathcal{O}}(PK, x, f_{SK}(x), \mathit{\Pi}_{SK}(x))$ makes the query q to the oracle $\mathcal{O}$. Since the bank of queries Q contains all the oracle queries made by V on $k\alpha(k)$ random inputs x_i, the probability that q is not in the bank of queries is exponentially small. More precisely,

$$\Pr[q \notin Q] \leq (1 - 1/\alpha(k))^{k\alpha(k)} \leq e^{-k}$$

Union bounding over all queries in $\mathcal{F}_{\alpha(k)}$ shows that with probability all but $\left|\mathcal{F}_{\alpha(k)}\right| \cdot e^{-k}$, Q contains $\mathcal{F}_{kt_G(k)}$ (where the probability is over the randomness of x_i). Now, $\left|\mathcal{F}_{\alpha(k)}\right| \leq \alpha(k) \cdot t_V(k) = k \cdot t_G(k) \cdot t_V(k)$ by simple counting. This completes the proof. □

The next lemma states that assuming the exploration stage completed properly, in the conquering stage A breaks the weak unpredictability of the VUF. We note that $\mathcal{O}'$ is one-way because it only differs from $\mathcal{O}$ on polynomially many inputs.

Lemma 4 (conquering stage). *Let Q, f_Q be an output of the exploration stage of A s.t. $Q \supseteq \mathcal{F}_{kt_G(k)}$. Then the conquering stage runs in $\text{poly}(k)$ time and predicts $F^{\mathcal{O}}_{SK}(x^*)$ with probability at least $1 - 1/k$.*

Proof. We define an input $x \in \{0,1\}^{n(k)}$ to be *indifferent* (with respect to PK and SK) if the execution of the verification algorithm (with oracle access to $\mathcal{O}$), on input $(PK, x, F^{\mathcal{O}}_{SK}(x), \mathit{\Pi}^{\mathcal{O}}_{SK}(x))$ makes no oracle query $q \in D$ (recall that D is the set of queries computed in step 1 of the conquering stage). In other words, this execution of the verification algorithm is indifferent to whether it is given oracle access to $\mathcal{O}$ or $\mathcal{O}'$.

The following claims establish that all but a $1/k$ fraction of the inputs x are indifferent; and that for every indifferent input x, $F^{\mathcal{O}}_{SK}(x) = F^{\mathcal{O}'}_{SK'}(x)$. In other words, if x^* is an indifferent input, then the adversary (which outputs $F^{\mathcal{O}'}_{SK'}(x^*)$) succeeds in predicting $F^{\mathcal{O}}_{SK}(x^*)$. It follows that the adversary succeeds with probability $1 - \frac{1}{k}$.

Claim. Let $\mathcal{I}$ denote the set of indifferent inputs (with respect to $\mathcal{O}, PK$ and SK). Then, $\Pr_{x \in \{0,1\}^{n(k)}}[x \in \mathcal{I}] \geq 1 - 1/k$.

Proof: By definition of our **NP** relation $\mathcal{R}$, $Q \cap D = \emptyset$. Thus, if $Q \supseteq \mathcal{F}_{kt_G(k)}$ then $\mathcal{F}_{kt_G(k)} \cap D = \emptyset$. If we fix some query $q \in D$, then $q \notin \mathcal{F}_{kt_G(k)}$, meaning

$$\mathsf{freq}(q) = \Pr_{x \in \{0,1\}^{n(k)}} \left[V^{\mathcal{O}}(PK, x, F^{\mathcal{O}}_{SK}(x), \mathit{\Pi}^{\mathcal{O}}_{SK}(x)) \text{ makes query } q\right] \leq 1/(kt_G(k))\,.$$

Applying the union bound over all $|D| \leq t_G(k)$ queries in D yields

$$\Pr_{x \in \{0,1\}^{n(k)}} \left[V^{\mathcal{O}}(PK, x, F^{\mathcal{O}}_{SK}(x), \mathit{\Pi}^{\mathcal{O}}_{SK}(x)) \text{ makes any query } q \in D\right] \leq \frac{|D|}{kt_G(k)} \leq \frac{1}{k}\,,$$

and the claim follows. ■

Claim. For all $x^* \in \mathcal{I}$, $F^{\mathcal{O}}_{SK}(x^*) = F^{\mathcal{O}'}_{SK'}(x^*)$.

Proof: For simplicity, assume that for every one-way permutation $\mathcal{O}$, the construction $(G^{\mathcal{O}}, F^{\mathcal{O}}, \Pi^{\mathcal{O}}, V^{\mathcal{O}})$ is correct for every input x.[14]

By the completeness of the weak VUF with respect to $\mathcal{O}$, we have that $V^{\mathcal{O}}(PK, x^*, F^{\mathcal{O}}_{SK}(x^*), \Pi^{\mathcal{O}}_{SK}(x^*))$ accepts. Since no queries in D are made during this computation, then clearly it would run in the exact same way with oracle access to $\mathcal{O}'$ rather than to $\mathcal{O}$. Thus,

$$V^{\mathcal{O}'}(PK, x^*, F^{\mathcal{O}}_{SK}(x^*), \Pi^{\mathcal{O}}_{SK}(x^*)) = V^{\mathcal{O}}(PK, x^*, F^{\mathcal{O}}_{SK}(x^*), \Pi^{\mathcal{O}}_{SK}(x^*)) = \mathsf{accept}\ .$$

Since $\mathcal{O}'$ is a OWP, $(G^{\mathcal{O}'}, F^{\mathcal{O}'}, \Pi^{\mathcal{O}'}, V^{\mathcal{O}'})$ is a weak VUF, which in particular, means that it satisfies the completeness and soundness properties. By its completeness, we get that $V^{\mathcal{O}'}(PK, x^*, F^{\mathcal{O}'}_{SK'}(x^*), \Pi^{\mathcal{O}'}_{SK'}(x^*))$ accepts. Soundness with respect to $\mathcal{O}'$ guarantees, therefore, that $F^{\mathcal{O}}_{SK}(x^*) = F^{\mathcal{O}'}_{SK'}(x^*)$. ■ □

Combining Lemmas 3, 4 (using the union bound) yields that A succeeds in predicting $F^{\mathcal{O}}_{SK}(x^*)$ with probability at least $1 - 1/k - \text{poly}(k) \cdot e^{-k}$, contradictory to the alleged security of the reduction. Theorem 1 follows. □

5.2 Additional Remarks

- **Handling Probabilistic Verifiers.** The analysis above disregarded the fact that the verifier V may not return the correct answer, with some small probability. Essentially, we handle this issue by using amplification by applying sequential repetition and then using a single random tape for all inputs. For details, we refer the reader to the full version [34].
- **On Requiring Perfect Completeness.** In the definition of VRF and weak VRF, we required that completeness holds for any (PK, SK) generated by G. When allowing *relaxed* completeness in Definition 1, the relaxation was over the *inputs* and not the keys. While this definition is frequently used, in some cases (e.g. [1]) the definition is so that the generator is allowed to output "bad" keys (ones that have no completeness for almost any input) with very small probability.
  While our proof does not cover such constructions, we notice that all known constructions (including that of [1]), can be presented as having perfect completeness in a **ZPP** sense. That is, where the generator is allowed to run for expected polynomial time rather than worst-case. Our construction can be slightly altered to work for such constructions as well.
- **Separating Trapdoor Permutations from OWPs.** As mentioned above, since there is a black-box reduction from weak VRFs to eTDPs, our separation also implies a Semi-BB separation of eTDPs from OWPs. The work of [11] implies a result that is stronger in two aspects: their separation (appended with a modification due to [32]) implies a ∀∃Semi-BB separation

[14] This is as opposed to relaxed completeness and soundness as in Definition 1 which hold for almost all inputs.

(see definition in [32]), and they show a separation from key-agreement. Our result, on the other hand, seems simpler and does not use heavy probability-theoretic machinery.

- **Other Types of Black-box Separations.** Our result as presented does not rule out a relativizing reduction. To rule out a relativizing reduction, we must exhibit an oracle $\mathcal{O}$ relative to which no weak VUF exists. We show, essentially, that for every construction, there is an oracle that makes the construction fail as a weak VUF. Our adversary, however, works by generating a slightly different OWP, which is efficiently computable from the old one, and plugging it into the same construction. To get a separation, we require correctness (but not necessarily security) for the modified oracle. Therefore, while our separation only rules out Semi-BB reductions in the general case, it can also be interpreted as ruling out ∀∃Semi-BB reductions if correctness holds for any OWP.
- **Inefficient Proof Generators.** We notice that while the adversary uses the code of oracle algorithms G, F, V, its use of Π is black-box only. Therefore, we additionally obtain that a semi-BB reduction is impossible even when the proof generator Π is allowed to be inefficient. [15]

References

1. Micali, S., Rabin, M., Vadhan, S.: Verifiable random functions. focs **00** (1999) 120
2. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. J. ACM **33**(4) (1986) 792–807
3. Naor, M., Reingold, O.: Synthesizers and their application to the parallel construction of pseudo-random functions. J. Comput. Syst. Sci. **58**(2) (1999)
4. Feige, U., Lapidot, D., Shamir, A.: Multiple noninteractive zero knowledge proofs under general assumptions. SIAM J. Comput. **29**(1) (1999) 1–28
5. Bellare, M., Yung, M.: Certifying permutations: Noninteractive zero-knowledge based on any trapdoor permutation. J. Cryptology **9**(3) (1996) 149–166
6. Barak, B., Ong, S.J., Vadhan, S.P.: Derandomization in cryptography. SIAM J. Comput. **37**(2) (2007) 380–400
7. Canetti, R., Halevi, S., Katz, J.: A forward-secure public-key encryption scheme. J. Cryptology **20**(3) (2007) 265–294
8. Håstad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A pseudorandom generator from any one-way function. SIAM J. Comput. **28**(4) (1999) 1364–1396
9. Naor, M., Yung, M.: Universal one-way hash functions and their cryptographic applications. In: STOC. (1989) 33–43
10. Rompel, J.: One-way functions are necessary and sufficient for secure signatures. In: STOC. (1990) 387–394
11. Impagliazzo, R., Rudich, S.: Limits on the provable consequences of one-way permutations. In: STOC '89: Proceedings of the twenty-first annual ACM symposium on Theory of computing, New York, NY, USA, ACM (1989) 44–61

[15] We note that there is a construction of a weak VRF from one-way permutations, using inefficient-prover non-interactive zero-knowledge for **NP** [4]. However, this construction is inherently *non-black-box.* Our separation result shows that this *must* be the case.

12. Gertner, Y., Malkin, T., Reingold, O.: On the impossibility of basing trapdoor functions on trapdoor predicates. In: FOCS '01: Proceedings of the 42nd IEEE symposium on Foundations of Computer Science, Washington, DC, USA, IEEE Computer Society (2001) 126
13. Bellare, M., Goldwasser, S.: New paradigms for digital signatures and message authentication based on non-interative zero knowledge proofs. In: CRYPTO. (1989) 194–211
14. Lysyanskaya, A.: Unique signatures and verifiable random functions from the dh-ddh separation. In: CRYPTO. (2002) 597–612
15. Dodis, Y.: Efficient construction of (distributed) verifiable random functions. In: Public Key Cryptography. (2003) 1–17
16. Dodis, Y., Yampolskiy, A.: A verifiable random function with short proofs and keys. In: Public Key Cryptography. (2005) 416–431
17. Chase, M., Lysyanskaya, A.: Simulatable vrfs with applications to multi-theorem nizk. In: CRYPTO. (2007) 303–322
18. Blum, M., Feldman, P., Micali, S.: Non-interactive zero-knowledge and its applications (extended abstract). In: STOC. (1988) 103–112
19. Blum, M., Feldman, P., Micali, S.: Proving security against chosen cyphertext attacks. In: CRYPTO. (1988) 256–268
20. Rackoff, C., Simon, D.R.: Cryptographic defense against traffic analysis. In: STOC. (1993) 672–681
21. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: STOC. (1990) 427–437
22. Dolev, D., Dwork, C., Naor, M.: Nonmalleable cryptography. SIAM J. Comput. **30**(2) (2000) 391–437
23. Sahai, A.: Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In: FOCS. (1999) 543–553
24. Pass, R., Shelat, A., Vaikuntanathan, V.: Construction of a non-malleable encryption scheme from any semantically secure one. In: CRYPTO. (2006) 271–289
25. Cramer, R., Hanaoka, G., Hofheinz, D., Imai, H., Kiltz, E., Pass, R., Shelat, A., Vaikuntanathan, V.: Bounded cca2-secure encryption. In: ASIACRYPT. (2007) 502–518
26. Groth, J., Ostrovsky, R., Sahai, A.: Perfect non-interactive zero knowledge for np. In: EUROCRYPT. (2006) 339–358
27. Groth, J., Ostrovsky, R., Sahai, A.: Non-interactive zaps and new techniques for nizk. In: CRYPTO. (2006) 97–111
28. Kilian, J., Petrank, E.: An efficient non-interactive zero-knowledge proof system for np with general assumptions. Journal of Cryptology **11** (1998) 1–27
29. Dwork, C., Naor, M.: Zaps and their applications. SIAM J. Comput. **36**(6) (2007) 1513–1543
30. Dodis, Y., Puniya, P.: Feistel networks made public, and applications. In: EUROCRYPT. (2007) 534–554
31. Goldwasser, S., Ostrovsky, R.: Invariant signatures and non-interactive zero-knowledge proofs are equivalent (extended abstract). In: CRYPTO. (1992) 228–245
32. Reingold, O., Trevisan, L., Vadhan, S.P.: Notions of reducibility between cryptographic primitives. In: TCC. (2004) 1–20
33. Blum, M., Micali, S.: How to generate cryptographically strong sequences of pseudo random bits. In: FOCS. (1982) 112–117
34. Brakerski, Z., Goldwasser, S., Rothblum, G., Vaikuntanathan, V.: Weak verifiable random functions. MIT CSAIL Technical Report (2008)