Simulatable VRFs with Applications to Multi-Theorem NIZK

Melissa Chase and Anna Lysyanskaya

Computer Science Department
Brown University
Providence, RI 02912
{mchase,anna}@cs.brown.edu

Abstract. This paper introduces simulatable verifiable random functions (sVRF). VRFs are similar to pseudorandom functions, except that they are also verifiable: corresponding to each seed SK, there is a public key PK, and for $y = F_{PK}(x)$, it is possible to prove that y is indeed the value of the function seeded by SK. A *simulatable* VRF is a VRF for which this proof can be simulated, so a simulator can pretend that the value of $F_{PK}(x)$ is any y.
Our contributions are as follows. We introduce the notion of sVRF. We give two constructions: one from general assumptions (based on NIZK), but inefficient, just as a proof of concept; the other construction is practical and based on a special assumption about composite-order groups with bilinear maps. We then use an sVRF to get a direct transformation from a single-theorem non-interactive zero-knowledge proof system for a language L to a multi-theorem non-interactive proof system for the same language L.

1 Introduction

It has been more than twenty years since the discovery of zero-knowledge proofs. In that time, they have attracted interest from the theoretical computer science community (leading to the study of interactive proof systems and PCPs), theoretical cryptography community, and, more recently, cryptographic practice.

The proof protocols that have been implemented so far [Bra99,CH02,BCC04], even though zero-knowledge in spirit, are not, strictly speaking, zero-knowledge proofs as we usually define them. Typically, they are honest-verifier interactive zero-knowledge proofs (sometimes, actually, arguments of knowledge) with the interactive step removed using the Fiat-Shamir paradigm [FS87,GK03]. Interaction is an expensive resource, and so using a heuristic such as the Fiat-Shamir transform in order to remove interaction is more attractive than using an interactive proof.

SINGLE-THEOREM NIZK In contrast to the Fiat-Shamir-based protocols adopted in practice, that do not in fact provide more than just a heuristic security guarantee [GK03], there are also well-known provable techniques for achieving zero-knowledge in non-interactive proofs. Blum et al. [BFM88,DMP88,BDMP91] introduced the notion of a *non-interactive zero-knowledge* (NIZK) proof system.

In such a proof system, some parameters of the system are set up securely ahead of time. Specifically, a common random string σ is available to all participants. The prover in such a proof system is given an $x \in L$ for some language L and a witness w attesting that $x \in L$. (For example, L can be the language of all pairs (n, e) e is relatively prime to $\phi(n)$. The witness w can be the factorization of n.) The prover computes a proof π, and the proof system is zero-knowledge in the following sense: the simulator can pick its own σ' for which it can find a proof π' for the statement $x \in L$. The values (σ', π') output by the simulator are indistinguishable from (σ, π) that are generated by first picking a random σ and then having the honest prover produce π for $x \in L$ using witness w. Blum et al. also gave several languages L with reasonably efficient NIZK proof systems.

Let us explain the Blum et al. NIZK proof system for the language L in the example above: $L = \{(n, e) \mid \gcd(\phi(n), e) = 1\}$. First, recall that if e is not relatively prime to $\phi(n)$, then the probability that for a random $x \in \mathbb{Z}_n^*$ there exists y such that $y^e = x \bmod n$ is upper-bounded by 1/2. On the other hand, if e *is* relatively prime to $\phi(n)$, then for all $x \in \mathbb{Z}_n^*$ there exists such a y. So the proof system would go as follows: parse the common random string σ as a sequence $z_1, \ldots, z_\ell$ of elements of $\mathbb{Z}_n^*$, and for each z_i, compute y_i such that $y_i^e = z_i \bmod n$. The proof π consists of the values $y_1, \ldots, y_\ell$. The verifier simply needs to check that each y_i is the e^{th} root of z_i. For any specific instance (n, e), the probability (over the choice of the common random string σ) that a cheating prover can come up with a proof that passes the verification is $2^{-\ell}$. By the union bound, letting $\ell = k(|n| + |e|)$ guarantees that the probability, over the choice of σ, that a cheating prover can find an instance (n, e) and a proof π passing the verification, is negligible in k.

Note that the proof system described above, although expensive, is not prohibitively so. Proof systems of this type have been shown to yield themselves to further optimizations [DCP97]. So why aren't such proofs attractive in practice?

MULTI-THEOREM NIZK The problem with NIZK as initially defined and explained above was that one proof π completely used up the common random string σ, and so to produce more proofs, fresh common randomness was required. Blum et al. [BDMP91] showed a single-prover multi-theorem NIZK proof system for 3SAT, and since 3SAT is NP-complete, the result followed for any language in NP, assuming quadratic residuocity. Feige, Lapidot and Shamir [FLS99] constructed a multi-prover, multi-theorem NIZK proof system for all NP based on trapdoor permutations. Recently, Groth, Ostrovsky and Sahai [GOS06] gave a multi-theorem NIZK proof system for circuit satisfiability with very compact common parameters and achieving perfect zero-knowledge (with computational soundness), based on the assumption that the Boneh, Goh, Nissim [BGN05] cryptosystem is secure.

In each of the multi-theorem NIZK results mentioned above, to prove that $x \in L$ for a language L in NP, the prover would proceed as follows: first, reduce x to an instance of the right NP-complete problem, also keeping track of the witness w. Then invoke the multi-theorem NIZK proof system constructed for this NP-complete problem. In other words, even if the language L itself had an efficient *single-theorem* NIZK, existing multi-theorem NIZK constructions have no way of exploiting it. The Feige et al. result, which is the most attractive because it is based on general assumptions, is especially bad in this regard: their construction explicitly includes a step that transforms every instance x into a new instance x' via a Cook-Levin reduction. These reductions are what makes NIZK prohibitively expensive to be considered for use in practice.

In this paper, we give a construction for achieving multi-theorem NIZK for any language L based on single-theorem NIZK for L, without having to reduce instances of L to instances of any NP-complete languages. This construction is based on a new building block: a *simulatable verifiable random function* (sVRF).

Simulatable VRFs. Verifiable random functions (VRFs) were introduced by Micali, Rabin, and Vadhan [MRV99]. They are similar to pseudorandom functions [GGM86], except that they are also verifiable. That is to say, associated with a secret seed SK, there is a public key PK, domain D_{PK}, range R_{PK} and a function $F_{PK}(\cdot)\ : D_{PK} \mapsto R_{PK}$ such that (1) $y = F_{PK}(x)$ is efficiently computable given the corresponding SK; (2) a proof $\pi_{PK}(x)$ that this value y corresponds to the public key PK is also efficiently computable given SK; such a proof can exist only for a unique value y; (3) based purely on PK and oracle calls to $F_{PK}(\cdot)$ and the corresponding proof oracle, no adversary can distinguish the value $F_{PK}(x)$ from a random value without explicitly querying the function on input x. Several constructions of VRFs in the plain model exist [MRV99,Lys02,Dod02,DY05]. In the common-random-string model, Goldwasser and Ostrovsky [GO92] showed that existence of VRFs (with polynomial-size domains; one can also call such VRFs verifiable pseudorandom generators, or VPRGs) is a necessary and sufficient condition for multi-theorem NIZK for all NP. Dwork and Naor [DN00] showed that (approximate) VPRGs in the standard model are necessary and sufficient for zaps (zaps are witness-indistinguishable proof protocols consisting of two rounds; the first round is a message from the verifier to the prover than can be reused for future instances).

We introduce *simulatable* VRFs (sVRFs). In the common parameters model, $F_{PK}(\cdot)$ is a VRF in the sense defined above for all honest settings of the common parameters. However, there is also a way to simulate the common parameters such that, corresponding to a PK, for any $x \in D_{PK}$, $y \in R_{PK}$, it is possible to simulate a proof π that $F_{PK}(x) = y$. The resulting simulation is indistinguishable from the view obtained when the parameters are set up correctly.

Using an sVRF to transform single-theorem NIZK to multi-theorem NIZK. A simulatable VRF with domain of size $\ell(k)$ and binary range allows a prover to come up with a fresh verifiably random string R of appropriate length $\ell(k)$ every time he wants to prove a new theorem. He simply comes up with a new PK for a VRF, and evaluates F_{PK} on input i to obtain the ith bit of R, R_i. The VRF allows him to prove that R was chosen correctly. He can then XOR R with a truly random public string σ_1 to obtain a string σ to be used in a single-theorem NIZK. The resulting construction is zero-knowledge because of the simulatability properties of both the sVRF and the single-theorem NIZK. It is sound because σ_1 is a truly random string, and so it inherits the soundness from the single-theorem NIZK (note that it incurs a penalty in the soundness error). Note that because our sVRF construction is in the public parameters model, the resulting multi-theorem proof system is also in the public parameters model (rather than the common random string model).

Since we give an *efficient* instantiation of sVRFs, our results essentially mean that studying efficient single-theorem NIZK proof systems for languages of interest is a good idea, because our construction gives an *efficient* transformation from such proof systems to multi-theorem ones.

Using an sVRF instead of the random oracle. An sVRF shares some characteristics with a programmable random oracle: assuming that the parameters of the system were picked by the simulator, the simulator can *program* it to take certain values on certain inputs. One cannot necessarily use it instead of the hash function in constructions where the adversary gets the code for the hash function. But it turns out that it can sometimes replace the random oracle in constructions where the adversary is allowed oracle access to the hash function and requires some means to be sure that the output is correct. For example, using an sVRF instead of H in the RSA-FDH construction [BR93,Cor00] would make the same proof of security hold without the random oracle. Of course, it is not a useful insight: an sVRF is already a signature, so it is silly to use it as a building block in constructing another signature. The reason we think the above observation is worth-while is that it is an example of when using an sVRF instead of an RO gives provable instead of heuristic guarantees.

Constructing an sVRF. Our main result is a direct construction of a simulatable VRF based on the Subgroup Decision assumption (SDA) [BGN05], and an assumption related to the Q-BDHI assumption [BB04b]. Dodis and Yampolskiy [DY05] used the Q-BDHI assumption to extend the Boneh-Boyen short signature scheme [BB04a] and derive a VRF. The Dodis-Yampolskiy VRF is of the form $F_s(x) = e(g, g)^{1/(s+x)}$, where g is a generator of some group G_1 of prime order q, and $e : G_1 \times G_1 \mapsto G_2$ is a bilinear map. The secret key is s while the public key is g^s. The DY proof that $y = F_s(x)$ is the value $\pi = g^{1/(s+x)}$ whose correctness can be verified using the bilinear map.

Our sVRF is quite similar, only it is in a composite-order group with a bilinear map: the order of G_1 is an RSA modulus $n = pq$. This is what makes simulatability possible. In our construction, the public parameters consist of (g, A, D, H), all generators of G_1. As before, the secret key is s, but now the public key is A^s. $F_s(x) = e(H, g)^{1/(s+x)}$, and the proof is a randomized version of the DY proof: $\pi = (\pi_1, \pi_2, \pi_3)$, where $\pi_1 = H^{r/(s+x)}/D^r$, $\pi_2 = g^{1/r}$ and $\pi_3 = A^{(s+x)/r}$. It turns out that, when A generates the entire G_1, there is a unique $y = F_s(x)$ for which a proof exists. However, when A belongs to the order-p subgroup of G_1 (as is going to be the case when the system parameters are picked by the simulator), the verification tests correctness only as far as the order-p subgroup is concerned, and so the order-q component of $F_s(x)$ is unconstrained. The proof of security requires that a strengthening of Q-BDHI hold for the prime-order subgroups of G_1, and that the SDA assumption holds so that A picked by the simulator is indistinguishable from the correct A.

We also give, as proof of concept, a construction under general assumptions, based on multi-theorem NIZK.

Organization of the rest of this paper. In Section 2 we define sVRFs. In Section 3 we give an sVRF construction based on general assumptions as a proof of concept. In Section 4 we give our main result and its proof of security. Finally, in Section 5 we give the transformation from single-theorem NIZK to multi-theorem NIZK using sVRFs.

2 On Defining sVRFs

We begin by adapting the definition of Micali, Rabin and Vadhan [MRV99] in the public parameters model.

Definition 1 (VRF in the public parameters model). *Let $Params(\cdot)$ be an algorithm generating public parameters $\mathtt{p}$ on input security parameter 1^k. Let $D(\mathtt{p})$ and $R(\mathtt{p})$ be families of efficiently samplable domains for all $\mathtt{p} \in Params$. The set of algorithms $(G, \mathtt{Eval}, \mathtt{Prove}, \mathtt{Verify})$ constitutes a verifiable random function (VRF) for parameter model Params, input domain $D(\cdot)$ and output range $R(\cdot)$ if*

Correctness *Informally, correctness means that the verification algorithm $\mathtt{Verify}$ will always accept $(\mathtt{p}, PK, x, y, \pi)$ when $y = F_{PK}(x)$, and π is the proof of this fact generated using $\mathtt{Prove}$. More formally, $\forall k, \mathtt{p} \in Params(1^k)$, $x \in D(\mathtt{p})$,*

$$\Pr[(PK, SK) \leftarrow G(\mathtt{p}); y = \mathtt{Eval}(\mathtt{p}, SK, x); \pi \leftarrow \mathtt{Prove}(\mathtt{p}, SK, x); \\ b \leftarrow \mathtt{Verify}(\mathtt{p}, PK, x, y, \pi) \ : \ b = 1] = 1$$

Pseudorandomness *Informally, pseudorandomness means that, on input $(\mathtt{p}, PK)$, even with oracle access to $\mathtt{Eval}(\mathtt{p}, SK, \cdot)$ and $\mathtt{Prove}(\mathtt{p}, SK, \cdot)$, no adversary can distinguish $F_{PK}(x)$ from a random element of $R(\mathtt{p})$ without explicitly querying for it. More formally, $\forall$ PPT $\mathcal{A}$, $\exists$ negligible ν such that*

$$\begin{aligned} &\Pr[\mathtt{p} \leftarrow Params(1^k); (PK, SK) \leftarrow G(\mathtt{p}); \\ &(Q_e, Q_p, x, \mathtt{state}) \leftarrow \mathcal{A}^{\mathtt{Eval}(\mathtt{p}, SK, \cdot), \mathtt{Prove}(\mathtt{p}, SK, \cdot)}(\mathtt{p}, PK); \\ &y_0 = \mathtt{Eval}(\mathtt{p}, SK, x); y_1 \leftarrow R(\mathtt{p}); b \leftarrow \{0,1\}; \\ &(Q'_e, Q'_p, b') \leftarrow \mathcal{A}^{\mathtt{Eval}(\mathtt{p}, SK, \cdot), \mathtt{Prove}(\mathtt{p}, SK, \cdot)}(\mathtt{state}, y_b) \\ &\quad : \ b' = b \wedge x \notin (Q_e \cup Q_p \cup Q'_e \cup Q'_p)] \le 1/2 + \nu(k) \end{aligned}$$

where Q_e and Q_p denote, respectively, the contents of the query tape that records $\mathcal{A}$'s queries to its $\mathtt{Eval}$ and $\mathtt{Prove}$ oracles in the first query phase, and Q'_e and Q'_p denote the query tapes in the second query phase.

Verifiability *For all k, for all $\mathtt{p} \in Params(1^k)$, there do not exist $(PK, x, y_1, \pi_1, y_2, \pi_2)$ such that $y_1 \neq y_2$, but $\mathtt{Verify}(\mathtt{p}, PK, x, y_1, \pi_1) = \mathtt{Verify}(\mathtt{p}, PK, x, y_2, \pi_2) = ACCEPT$.*

Note that verifiability in the definition above can be relaxed so as to only hold computationally (as opposed to unconditionally).

Simulatability, as defined below, is the novel aspect of sVRFs, setting them apart from VRFs as previously defined. First, we give the definition, and then we discuss variations.

Definition 2 (Simulatable VRF). *Let* $(Params, G, \texttt{Eval}, \texttt{Prove}, \texttt{Verify})$ *be a VRF (according to Definition 1). They constitute a simulatable VRF if there exist algorithms* $(SimParams, SimG, SimProve)$ *such that for all PPT* $\mathcal{A}$*,* $\mathcal{A}$*'s views in the following two games are indistinguishable:*

Game Real $\texttt{p} \leftarrow Params(1^k)$ *and then* $\mathcal{A}(\texttt{p})$ *gets access to the following oracle* $\mathcal{R}$*: On query* NewPK*,* $\mathcal{R}$ *obtains and stores* $(PK, SK) \leftarrow G(\texttt{p})$*, and returns* PK *to* $\mathcal{A}$*. On query* (PK, x)*,* $\mathcal{R}$ *verifies that* (PK, SK) *has been stored for some* SK*. If not it returns "error". If so, it returns* $y = \texttt{Eval}(\texttt{p}, SK, x)$ *and* $\pi \leftarrow \texttt{Prove}(\texttt{p}, SK, x)$*.*

Game Simulated $(\texttt{p}, t) \leftarrow SimParams(1^k)$*, and then* $\mathcal{A}(\texttt{p})$ *gets access to the following oracle* $\mathcal{S}$*: On query* NewPK*,* $\mathcal{S}$ *obtains and stores* $(PK, SK) \leftarrow SimG(\texttt{p}, t)$*, and returns* PK *to* $\mathcal{A}$*. On query* (PK, x)*,* $\mathcal{S}$ *verifies if* (PK, SK) *has been stored for some* SK*. If not, it returns "error". If so,* $\mathcal{S}$ *(1) checks if* x *has previously been queried, and if so, returns the answer stored; (2) otherwise,* $\mathcal{S}$ *obtains* $y \leftarrow R(\texttt{p})$ *and* $\pi \leftarrow SimProve(\texttt{p}, SK, x, y, t)$*, and returns and stores* (y, π)*.*

2.1 Simplifying the Definition

The games in the above definition need to store multiple public keys and secret keys, as well as responses to all the queries issued so far, and consistently respond to multiple queries corresponding to all these various keys. It is clear that this level of security is desirable: we want an sVRF to retain its security properties under composition with other instances within the same system. A natural question is whether we can simplify the games by restricting the adversary to just one NewPK query or just one (PK, x) query per PK without weakening the security guarantees. In fact, the four possible combinations of such restrictions yield four distinct security notions, as we show in the full version of this paper.

Although we cannot simplify Definition 2 in this way, we can give a seemingly simpler definition (one that only allows one NewPK query from the adversary) that is strictly stronger than Definition 2 in that it requires that the adversary cannot distinguish the real game from the simulated one, *even with the knowledge of the trapdoor* t.

Definition 3 (Trapdoor-indistinguishable sVRF). *Let* $(Params, G, \texttt{Eval}, \texttt{Prove}, \texttt{Verify})$ *be a VRF (as in Definition 1). They constitute a trapdoor-indistinguishable (TI) sVRF if there exist algorithms* $(SimParams, SimG, SimProve)$ *such that the distribution* $Params(1^k)$ *is computationally indistinguishable from the distribution* $SimParams(1^k)$ *and for all PPT* $\mathcal{A}$*,* $\mathcal{A}$*'s views in the following two games are indistinguishable:*

Game Real Proofs $(\mathtt{p}, t) \leftarrow \mathit{SimParams}(1^k)$, $(PK, SK) \leftarrow G(\mathtt{p})$ *and then* $\mathcal{A}(\mathtt{p}, t, PK)$ *gets access to the following oracle* $\mathcal{R}$*: On query* x*,* $\mathcal{R}$ *returns* $y = \mathtt{Eval}(\mathtt{p}, SK, x)$ *and* $\pi \leftarrow \mathtt{Prove}(\mathtt{p}, SK, x)$*.*

Game Simulated Proofs $(\mathtt{p}, t) \leftarrow \mathit{SimParams}(1^k)$, $(PK, SK) \leftarrow \mathit{SimG}(\mathtt{p}, t)$*, and then* $\mathcal{A}(\mathtt{p}, t, PK)$ *gets access to the following oracle* $\mathcal{S}$*: On query* x*,* $\mathcal{S}$ *(1) checks if* x *has previously been queried, and if so, returns the answer stored; (2) otherwise, obtains* $y \leftarrow R(\mathtt{p})$ *and* $\pi \leftarrow \mathit{SimProve}(\mathtt{p}, SK, x, y, t)$*, and returns and stores* (y, π)*.*

By a fairly standard hybrid argument, we have the following lemma (see the full version for the proof):

Lemma 1. *If* $(\mathit{Params}, G, \mathtt{Eval}, \mathtt{Prove}, \mathtt{Verify})$ *is a TI-sVRF, it is an sVRF.*

2.2 Weak Trapdoor-Indistinguishable sVRF

We now define a somewhat weaker notion of TI sVRFs, in which a simulator can only give fake proofs for those values of the output range that it has sampled itself in some special way.

Definition 4 (Weak TI-sVRF). *Let* $(G, \mathtt{Eval}, \mathtt{Prove}, \mathtt{Verify})$ *be a VRF in the* $\mathit{Params}(1^k)$ *model with domain* $D(\cdot)$ *and range* $R(\cdot)$*. They constitute a weak trapdoor-indistinguishable (TI) sVRF if there exist algorithms* $(\mathit{SimParams}, \mathit{SimG}, \mathit{SimProve}, \mathit{SimSample})$ *such that the distribution* $\mathit{Params}(1^k)$ *is computationally indistinguishable from the distribution* $\mathit{SimParams}(1^k)$ *and for all PPT* $\mathcal{A}$*,* $\mathcal{A}$*'s views in the following two games are indistinguishable:*

Game Real Proofs $(\mathtt{p}, t) \leftarrow \mathit{SimParams}(1^k)$, $(PK, SK) \leftarrow G(\mathtt{p})$ *and then* $\mathcal{A}(\mathtt{p}, t, PK)$ *gets access to the following oracle: On query* x*, the oracle returns* $y = \mathtt{Eval}(\mathtt{p}, SK, x)$ *and* $\pi \leftarrow \mathtt{Prove}(\mathtt{p}, SK, x)$*.*

Game Simulated Proofs $(\mathtt{p}, t) \leftarrow \mathit{SimParams}(1^k)$, $(PK, SK) \leftarrow \mathit{SimG}(\mathtt{p}, t)$*, and then* $\mathcal{A}(\mathtt{p}, t, PK)$ *gets access to the following oracle: On query* x*, the oracle (1) checks if* x *has previously been queried, and if so, returns the answer stored; (2) otherwise, obtains* $(y, w) \leftarrow \mathit{SimSample}(\mathtt{p}, t, SK, x)$ *and* $\pi \leftarrow \mathit{SimProve}(\mathtt{p}, SK, x, y, t, w)$*, and returns and stores* (y, π)*.*

We now show that a weak TI-sVRF where *SimSample* outputs a uniformly random element of a sufficiently large set can be converted to a TI-sVRF with binary range. Let $(G, \mathtt{Eval}, \mathtt{Prove}, \mathtt{Verify})$ be a weak TI-sVRF in the *Params* model with domain $D(\mathtt{p})$, and range $R(\mathtt{p}) \subseteq \{0,1\}^{m(k)}$ for some polynomial m for all $\mathtt{p} \in \mathit{Params}(1^k)$. Consider the following algorithms:

Params$^*(1^k)$ Pick $r \leftarrow \{0,1\}^{m(k)}$, $\mathtt{p} \leftarrow \mathit{Params}(1^k)$; return $\mathtt{p}^* = (r, \mathtt{p})$.

G^* On input $\mathtt{p}^* = (r, \mathtt{p})$, output $(PK^*, SK^*) \leftarrow G(\mathtt{p})$.

$\mathtt{Eval}^*$ **and** $\mathtt{Prove}^*$ On input $\mathtt{p}^* = (r, \mathtt{p})$, SK^*, and $x \in D(\mathtt{p})$, compute $y = \mathtt{Eval}(\mathtt{p}, SK^*, x)$. Let $y^* = y \cdot r$, where by "$\cdot$", we denote the inner product, i.e. $y \cdot r = \bigoplus_{i=1}^{|y|} y_i r_i$. $\mathtt{Eval}^*$ outputs y^*. $\mathtt{Prove}^*$ picks $\pi \leftarrow \mathtt{Prove}(\mathtt{p}, SK^*, x)$ and outputs $\pi^* = (\pi, y)$.

$\mathtt{Verify}^*$ On input $\mathtt{p}^* = (r, \mathtt{p})$, PK^*, $x \in D(\mathtt{p})$, $y^* \in \{0,1\}$, $\pi^* = (\pi, y)$: accept iff $\mathtt{Verify}(\mathtt{p}, PK, x, y, \pi)$ accepts and $y^* = r \cdot y$.

Lemma 2. *Suppose* $(G, \mathtt{Eval}, \mathtt{Prove}, \mathtt{Verify})$ *is a weak TI-sVRF with* $(\mathit{SimParams}, \mathit{SimSample}, \mathit{SimG}, \mathit{SimProve})$ *as in Definition 4. Let* ρ *be such that for all* $(\mathtt{p}, t) \in \mathit{SimParams}(1^k)$*, for all* $x \in D(\mathtt{p})$*, for all* $(SK, PK) \in \mathit{SimG}(\mathtt{p}, t)$*,* $|\mathit{SimSample}(\mathtt{p}, t, SK, x)| \geq \rho(k)$*, and SimSample is a uniform distribution over its support. Let* μ *be such that for all* $\mathtt{p} \in \mathit{Params}(1^k)$*,* $|D(\mathtt{p})| \leq \mu(k)$*. If there exists a negligible function* ν *such that* $\mu(k)\rho(k)^{-\frac{1}{3}} = \nu(k)$ *then* $(G^*, \mathtt{Eval}^*, \mathtt{Prove}^*, \mathtt{Verify}^*)$ *as constructed above are a TI-sVRF in the* Params^* *model with domain* $D(\mathtt{p})$*, and range* $\{0,1\}$*.*

Proof. Correctness, verifiability and pseudorandomness follow easily from the respective properties of the weak TI-sVRF (recall that a weak TI-sVRF is still a VRF – the "weak" part refers to simulatability only). In particular, pseudorandomness follows by standard techniques such as the leftover hash lemma.

We must show TI-simulatability. We first prove a useful claim. Consider specific values $(\mathtt{p}, t) \in \mathit{SimParams}(1^k)$, $(PK, SK) \in \mathit{SimG}(\mathtt{p}, t)$. Since t and SK are fixed, the distributions $R'(x) = \mathit{SimSample}(\mathtt{p}, t, SK, x)$ and $\mathit{Bad}(x) = \{r \in \{0,1\}^{m(k)} : |\Pr[y \leftarrow R'(x) : y \cdot r = 1] - .5| \geq |R'(x)|^{-\frac{1}{3}}\}$ are well-defined. In English, $\mathit{Bad}(x)$ is the set of those r's for which the random variable $y \cdot r$ (where y is sampled uniformly at random from $R'(x)$, i.e. sampled according to $\mathit{SimSample}(\mathtt{p}, t, SK, x)$) is biased by at least $|R'(x)|^{-\frac{1}{3}}$ from a random bit.

Claim. $\forall x \in D(\mathtt{p})$, $\Pr[r \leftarrow \{0,1\}^{m(k)} : r \in \mathit{Bad}(x)] \leq |R'(x)|^{-\frac{1}{3}}$.

Proof. (Of claim.) Suppose $x \in D(\mathtt{p})$ is fixed. Let $\mathit{Weight}(r) = \sum_{y \in R'(x)} y \cdot r$. By definition of $\mathit{Bad}(x)$, $r \in \mathit{Bad}(x)$ if and only if $|\mathit{Weight}(r)/|R'(x)| - .5| \geq |R'(x)|^{-\frac{1}{3}}$. It is easy to see that, if the probability is taken over the choice of r, then $\mathit{Exp}[\mathit{Weight}(r)/|R'(x)|] = .5$. On the other hand, for any pair $y_1 \neq y_2 \in R'(x)$, $y_1 \cdot r$ is independent from $y_2 \cdot r$, and so $\mathit{Weight}(r) = \sum_{y \in R'(x)} y \cdot r$ is a sum of pairwise independent random variables. Thus, $\mathit{Var}[\mathit{Weight}(r)] = \sum_{y \in R'(x)} \mathit{Var}[y \cdot r] = |R'(x)|/4$, and $\mathit{Var}[\mathit{Weight}(r)/|R'(x)|] = 1/4|R'(x)|$. Plugging Exp and Var for $\mathit{Weight}(r)/|R'(x)|$ into Chebyshev's inequality, we get $\Pr[|\mathit{Weight}(r)/|R'(x)| - .5| \geq |R'(x)|^{-\frac{1}{3}}\}] \leq |R'(x)|^{-\frac{1}{3}}$ which completes the proof.

Now we will show that the simulatability property holds. Consider the following algorithms:

SimParams** On input 1^k, obtain $(\mathtt{p}, t) \leftarrow \mathit{SimParams}(1^k)$, $r \leftarrow \{0,1\}^{m(k)}$. Output $\mathtt{p}^* = (r, \mathtt{p})$, $t^* = t$.

SimG** On input $(\mathtt{p}^*, t^*)$, where $\mathtt{p}^* = (r, \mathtt{p})$ obtain $(PK, SK) \leftarrow \mathit{SimG}(\mathtt{p}, t^*)$. Output $PK^* = PK, SK^* = SK$.

SimProve** On input $(\mathtt{p}^*, SK^*, x, y^*, t^*)$ where $\mathtt{p}^* = (r, \mathtt{p})$, repeat the following up to k times until $y \cdot r = y^*$: $(y, w) \leftarrow \mathit{SimSample}(\mathtt{p}, t, SK, x)$. If after k calls to $\mathit{SimSample}$, $y \cdot r \neq y^*$, output "fail". Else obtain $\pi \leftarrow \mathit{SimProve}(\mathtt{p}, t, SK, x, (y, w))$. Output $\pi^* = (\pi, y)$.

We define two intermediate games in which the adversary is given an oracle that is similar to Game Simulated Proofs from the TI-sVRF definition in that it does not use `Eval` and `Prove`; instead of `Eval`, it uses *SimSample* (from the weak TI-sVRF definition) to obtain (y, w), and then outputs $y^* = y \cdot r$. The two games generate the proofs in different ways: Game Intermediate Real Proof just uses w and *SimProve* of the weak TI-sVRF definition to generate π, while Game Intermediate Simulated Proof uses *SimProve** defined above. More precisely:

Game Intermediate Real Proofs $(\mathtt{p}^*, t^*) \leftarrow \mathit{SimParams}^*(1^k)$, $(PK^*, SK^*) \leftarrow \mathit{SimG}^*(\mathtt{p}^*, t^*)$, and then $\mathcal{A}(\mathtt{p}^*, t^*, PK^*)$ gets access to the following oracle: On query x, the oracle (1) checks if x has previously been queried, and if so, returns the answer stored; (2) otherwise, obtains $(y, w) \leftarrow \mathit{SimSample}(\mathtt{p}, t, SK, x)$, $y^* = y \cdot r$, and $\pi \leftarrow \mathit{SimProve}(\mathtt{p}, SK, x, y, t, w)$, $\pi^* = (\pi, y)$, and returns and stores (y^*, π^*).

Game Intermediate Simulated Proofs $(\mathtt{p}^*, t^*) \leftarrow \mathit{SimParams}^*(1^k)$, $(PK^*, SK^*) \leftarrow \mathit{SimG}^*(\mathtt{p}^*, t^*)$, and then $\mathcal{A}(\mathtt{p}^*, t^*, PK^*)$ gets access to the following oracle: On query x, the oracle (1) checks if x has previously been queried, and if so, returns the answer stored; (2) otherwise, obtains $(y', w') \leftarrow \mathit{SimSample}(\mathtt{p}, t, SK, x)$, $y^* = y' \cdot r$, and $\pi^* \leftarrow \mathit{SimProve}^*(\mathtt{p}, SK, x, y^*, t)$, and returns and stores (y^*, π^*).

We now argue that these intermediate games are indistinguishable from Game Real Proofs and Game Simulated Proofs as specified by the definition of TI-sVRF, instantiated with (*SimParams*, *SimG*, *SimSample*, *SimProve*) that follow from simulatability of our weak TI-sVRF, and with (*SimParams**, *SimG**, *SimProve**) defined above. First, it is straightforward to see that an adversary distinguishing between Game Real Proofs and Game Intermediate Real Proofs directly contradicts the simulatability property of weak TI-sVRFs.

The only difference between Game Intermediate Simulated Proofs and Game Simulated Proofs, is the choice of the bit y^*: in the former, it is chosen using *SimSample*, i.e. indistinguishably from the way it is chosen in the real game. In the latter, it is chosen at random. If we condition on the event that for all x, $r \notin \mathit{Bad}(x)$, these two distributions are statistically close.

The only thing left to show is that the two intermediate games defined above are indistinguishable. If we condition on the event that we never fail, then the two games are identical. Note that if for all x, $r \notin \mathit{Bad}(x)$, then the probability that we fail on a particular query is $\leq (1/2 + |R'(x)|^{-\frac{1}{3}})^k$ which is negligible.

Thus we have shown that if the probability that $r \in \mathit{Bad}(x)$ for some x is negligible, then Game Real Proofs is indistinguishable from Game Simulated Proofs. By the union bound, combined with the claim, $\Pr[r \leftarrow \{0,1\}^{m(k)} : \exists x \in D(\mathtt{p}) \text{ such that } r \in \mathit{Bad}(x)] \leq |D(\mathtt{p})||R'(x)|^{-\frac{1}{3}}$, which is equal to $\nu(k)$ by the premise of the lemma. □

From Lemmas 1 and 2, we see that from a weak TI-sVRF satisfying the conditions of Lemma 2, we can construct an equally efficient sVRF with range $\{0, 1\}$.

Remark. Note that, even though the support of $\mathit{SimSample}(\mathtt{p}, t, \mathit{SK}, x)$ is quite large, the construction above only extracts one bit of randomness from it. Although it can be easily extended to extract a logarithmic number of random bits, there does not seem to be a black-box construction extracting a superlogarithmic number of bits. Suppose ext is a procedure that extracts ℓ bits from y, so $y^* = \mathit{ext}(y)$ is of length ℓ. Then how would $\mathit{SimProve}^*$ work to generate a proof that y^* is correct? It needs to call $\mathit{SimProve}(\mathtt{p}, \mathit{SK}, x, y, t, w)$ for some y such that $y^* = \mathit{ext}(y)$ and w is an appropriate witness. It seems that the only way to obtain such a pair (y, w) is by calling $\mathit{SimSample}(\mathtt{p}, t, \mathit{SK}, x)$; in expectation, 2^ℓ calls to $\mathit{SimSample}$ are needed to obtain an appropriate pair (y, w); if ℓ is superlogarithmic, this is prohibitively inefficient.

3 Construction Based on General Assumptions

In the common-random-string (CRS) model, sVRFs can be constructed from any one-way function and an unconditionally sound multi-theorem non-interactive zero-knowledge proof system ($\mathtt{NIZKProve}, \mathtt{NIZKVerify}$) for NP (we review the notion of NIZK in Section 5). Pseudorandom functions (PRFs) can be obtained from one-way functions [HILL99,GGM86] (in the sequel, by $F_s(x)$ we denote a PRF with seed s and input x). In the CRS model, one-way functions also imply unconditionally binding computationally hiding non-interactive commitment [Nao91] (in the sequel, denoted as $\mathtt{Commit}(x, q, r)$, where x is the value to which one commits, q is the public parameter, and r is the randomness). We describe the construction below. In the full version, we prove it is an sVRF.

Params Corresponding to the security parameter k, choose a common random string σ of length $\ell(k)$, where $\ell(k)$ bits suffice for multi-theorem NIZK [BDMP91,FLS99,GOS06]. Choose a random $2k$-bit string q as the public parameter for the Naor commitment scheme. The parameters are $\mathtt{p} = (\sigma, q)$.

Domain and range The function has domain $D(\mathtt{p}) = \{0,1\}^{p_1(k)}$, and range $R(\mathtt{p}) = \{0,1\}^{p_2(k)}$, where p_1 and p_2 are functions bounded by a polynomial.

G Pick a random seed s for a pseudorandom function $F_s : \{0,1\}^{p_1(k)} \mapsto \{0,1\}^{p_2(k)}$. Let $\mathit{PK} = \mathtt{Commit}(s, q, r)$, $\mathit{SK} = (s, r)$, where r is the randomness needed for the commitment.

`Eval` On input x, output $y = F_s(x)$.

`Prove` On input x, run `NIZKProve` using CRS σ to output a NIZK proof π of the following statement: $\exists (s, r) \mid \mathit{PK} = \mathtt{Commit}(s, q, r) \wedge y = F_s(x)$.

`Verify` On input (PK, y, π), verify the proof π using the `NIZKVerify` algorithm.

4 Efficient Construction

We first present a construction for a weak TI-sVRF with a large output range. As we have shown, this can then be transformed into an sVRF with range $\{0,1\}$. The security relies on the following assumptions.

Definition 5 **((Q,ν)-BDHI [BB04a]).** *A family $\mathcal{G}$ of groups satisfies the $(Q(k),\nu(k))$-bilinear Diffie-Hellman inversion assumption if no PPT $\mathcal{A}$, on input $(instance, challenge)$ can distinguish if its challenge is of type 1 or type 2 with advantage asymptotically higher than $\nu(k)$ where instance and challenge are defined as follows: instance $= (G_1, G_2, q, e, g, g^{\alpha}, g^{\alpha^2}, g^{\alpha^3}, \ldots, g^{\alpha^{Q(k)}})$ where q is a prime of length poly(k), G_1, G_2 are groups of order q returned by $\mathcal{G}(q)$, $e : G_1 \times G_1 \to G_2$ is a bilinear map, $g \leftarrow G_1$, $\alpha \leftarrow \mathbb{Z}_q^*$, challenge of type 1 is $e(g,g)^{\frac{1}{\alpha}}$, while challenge of type 2 is $e(g,g)^R$ for random $R \leftarrow \mathbb{Z}_q^*$.*

Definition 6 **((Q,ν)-BDHBI).** *An family $\mathcal{G}$ of groups satisfies the $(Q(k),\nu(k))$ bilinear Diffie-Hellman* basegroup *inversion assumption if no PPT $\mathcal{A}$, on input $(instance, challenge)$ can distinguish if its challenge is of type 1 or type 2 with advantage asymptotically higher than $\nu(k)$, where instance and challenge are defined as follows: instance $= (G_1, G_2, q, e, g, g^{\alpha}, g^{\alpha^2}, g^{\alpha^3}, \ldots, g^{\alpha^{Q(k)}}, g^{\beta})$ where q is a prime of length poly(k), G_1, G_2 are groups of order q returned by $\mathcal{G}(q)$, $e : G_1 \times G_1 \to G_2$ is a bilinear map, $g \leftarrow G_1$, $\alpha \leftarrow \mathbb{Z}_q^*$, $\beta \leftarrow \mathbb{Z}_q^*$, challenge of type 1 is $g^{\frac{1}{\alpha\beta}}$, while challenge of type 2 is g^R for random $R \leftarrow \mathbb{Z}_q^*$.*

The assumption in Definition 6 is a new assumption which can be shown to imply Q-BDHI. We will assume that it holds for the prime order subgroup of composite order bilinear groups that can be efficiently instantiated [BGN05].

Definition 7 (SDA [BGN05]). *A family $\mathcal{G}$ of groups satisfies the subgroup decision assumption if no PPT $\mathcal{A}$, on input $(instance, challenge)$ can distinguish if its challenge is of type 1 or type 2, where instance and challenge are defined as follows: instance $= (G_1, G_2, n, e, h)$ where $n = pq$ is a product of two primes of length poly(k) (for k a sec. param.), G_1, G_2 are groups of order n returned by $\mathcal{G}(q,p)$, $e : G_1 \times G_1 \to G_2$ is a bilinear map, h is a random generator of G_1, challenge of type 1 is g, a random generator of G_1, while challenge of type 2 is g_p, a random order-p element of G_1.*

The weak TI-sVRF construction is as follows:

Params On input 1^k, choose groups G_1, G_2 of order $n = pq$ for primes p, q, where $|p|$ and $|q|$ are polynomial in k, with bilinear map $e : G_1 \times G_1 \to G_2$. Choose random generators g, H, A, D for G_1. *Params* will output $\mathtt{p} = (G_1, G_2, n, e, g, H, A, D)$.

Domain and range The input domain $\mathcal{D}(\mathtt{p})$ consists of integers $1 \leq x \leq l(k)$ where $l(k) < 2^{|q|-1}$ (We will later see the connection between $l(k)$ and $Q(k)$ by which our assumption is parameterized.) Note that $\mathcal{D}(\mathtt{p})$ depends only on k, not on $\mathtt{p}$. $R(\mathtt{p}) = G_2$.

G On input $\mathtt{p}$, pick $s \leftarrow \mathbb{Z}_n^*$, output $SK = s$, $PK = A^s$.

`Eval` On input $(\mathtt{p}, SK, x)$, output $e(H, g)^{\frac{1}{s+x}}$.

`Prove` On input $(\mathtt{p}, SK, x)$, pick $r \leftarrow \mathbb{Z}_n^*$, and output $\pi = (\pi_1, \pi_2, \pi_3)$, where $\pi_1 = H^{\frac{r}{s+x}}/D^r$, $\pi_2 = g^{\frac{1}{r}}$, $\pi_3 = A^{\frac{x+s}{r}}$.

`Verify` On input $(\mathtt{p}, SK, x, y, \pi)$, parse $\pi = (\pi_1, \pi_2, \pi_3)$ and verify that $e(\pi_1, \pi_2)$ $e(D, g) = y$, $e(\pi_3, g) = e(A^x PK, \pi_2)$, $e(\pi_1, \pi_3)e(D, A^x PK) = e(H, A)$.

Theorem 1. *(G, $\mathtt{Eval}$, $\mathtt{Prove}$, $\mathtt{Verify}$) as described above constitute a weak TI-sVRF for parameter model Params, input domain $\mathcal{D}$ of size l, and output range G_2 (where G_2 is as output by Params) under the SDA assumption combined with the $(l(k), \nu(k)/l^2(k))$-BDHBI, where ν is an upper bound on the asymptotic advantage that any probabilistic polynomial-time algorithm has in breaking the simulatability game of Definition 4.*

Proof. **Correctness** follows from construction.
Verifiability: Suppose there exists an adversary who, given parameters $\mathtt{p} = (G_1, G_2, n, e, g, H = g^h, A = g^a, D = g^d)$ generated by *Params* can produce $PK, y, y', \pi = (\pi_1, \pi_2, \pi_3), \pi' = (\pi'_1, \pi'_2, \pi'_3)$ such that $\mathtt{Verify}(\mathtt{p}, PK, y, \pi) = \mathtt{Verify}(\mathtt{p}, PK, y', \pi') = 1$. Then we will show that $y = y'$.

Let $\lambda, \mu, \mu', \sigma, \phi, \theta, \sigma', \phi', \theta' \in \mathbb{Z}_n$ be the exponents such that $PK = g^\lambda, y = g^\mu, y' = g^{\mu'}, \pi_1 = g^\sigma, \pi_2 = g^\phi, \pi_3 = g^\theta, \pi'_1 = g^{\sigma'}, \pi'_2 = g^{\phi'}, \pi'_3 = g^{\theta'}$.

If the verifications succeed, then we get that the following equations hold in $\mathbb{Z}_n$: $\sigma\phi + d = \mu, \quad \theta = (ax + \lambda)\phi, \quad \theta\sigma + d(ax + \lambda) = ha$.

Solving this system of equations gives us: $ha = \mu(ax+\lambda)$. Similarly, if (y', π') satisfy the verification equations, then we know that $ha = \mu'(ax + \lambda)$. H, A are generators for G_1, so $h, a \in \mathbb{Z}_n^*$, and therefore, $\mu'(ax + \lambda) \in \mathbb{Z}_n^*$, and $\mu(ax + \lambda) \in \mathbb{Z}_n^*$. This in turn means that $\mu', \mu, (ax + \lambda) \in \mathbb{Z}_n^*$.

From the solutions to the above equations, we know $\mu(ax + \lambda) = \mu'(ax + \lambda)$. Since $(ax + \lambda) \in \mathbb{Z}_n^*$, we can compute a unique inverse $(ax + \lambda)^{-1}$, and conclude that $\mu = \mu'$, and $y = y'$.

Note that this argument relies crucially on the fact that $h, a \in \mathbb{Z}_n^*$. In our simulation, we will instead choose $a = 0 \mod q$, which will allow us to avoid this binding property.
Pseudorandomness follows under the Q-BDHI Assumption from pseudorandomnesss of the Dodis-Yampolskiy VRF [DY05].
Simulatability: Consider the following simulator algorithms:

SimParams(1^k) Choose groups G_1, G_2 of order $n = pq$ for prime p, q, where $|p|$ and $|q|$ are polynomial in k, with bilinear map $e : G_1 \times G_1 \to G_2$. Let G_p be the order p subgroup of G_1, and let G_q be the order q subgroup of G_1. Let $(A, g_p, H_p, D_p) \leftarrow G_p^4$ and $(g_q, H_q, D_q) \leftarrow G_q^3$. Let $g = g_p g_q$, $H = H_p H_q$, and $D = D_p D_q$. Output $\mathtt{p} = (G_1, G_2, n, e, g, H, A, D)$, $t = (g_p, g_q, H_p, H_q, D_p, D_q)$.
This is identical to *Params* except that $A \in G_p$, so that the verification algorithm cannot properly verify the G_q components of y and π.

SimG$(\mathtt{p}, t)$ $(SK, PK) \leftarrow G(\mathtt{p})$.

SimSample On input $(\mathtt{p}, t, SK, x)$, pick $w \leftarrow \mathbb{Z}_q^*$.
Let $y = e(H_p, g_p)^{\frac{1}{s+x}} e(g_q, g_q)^w$. Output (y, w). (Note y's G_p component will be correct, while its G_q component will be random.)

SimProve On input $(\mathtt{p}, SK, x, y, t, w)$, pick $r \leftarrow \mathbb{Z}_n^*$;
let $\pi_1 = (H_p^{\frac{r}{s+x}} / D_p^r)(g_q^{wr} / D_q^r)$, $\pi_2 = g^{\frac{1}{r}}$, $\pi_3 = A^{\frac{x+s}{r}}$. Output $\pi = (\pi_1, \pi_2, \pi_3)$. (Note that π's G_p components are correct, while its G_q components are chosen so as to allow us to fake the proof.)

Lemma 3. *The distribution* $Params(1^k)$ *is indistinguishable from the distribution* $SimParams(1^k)$ *by the Subgroup Decision Assumption.*

Proof. The only difference between these two distributions is that in *Params*, A is chosen at random from G_1, and in *SimParams*, A is chosen at random from G_p. Thus, these two distributions are indistinguishable by the Subgroup Decision assumption by a straightforward reduction. □

Lemma 4. *For the algorithms described above, Game Real Proofs and Game Simulated Proofs (as in Definition 4) are indistinguishable with advantage more that* $\nu(k)$ *by the* $(l(k), \nu(k)/l^2(k))$*-BDHBI assumption.*

Before we prove this lemma, we will describe and prove an intermediate assumption that follows from the assumptions that we have already made. We state this assumption in terms of any prime order bilinear group. However, we will later assume that this assumption (and the Q-BDHBI assumption) holds over the prime order subgroup of a composite order bilinear group.

Definition 8 (**(Q,ν)-Intermediate assumption**). *A family $\mathcal{G}$ of groups satisfies the $(Q(k), \nu(k))$-intermediate assumption if for all subsets X of $\mathbb{Z}_{2^{a(k)-1}}$ (where $a(k)$ is a polynomial), of size $Q(k)-1$ for all $x^* \in \mathbb{Z}_{2^{a(k)-1}} \setminus X$, no PPT $\mathcal{A}$, on input $(instance, challenge)$ can distinguish if its challenge is of type 1 or type 2 with advantage asymptotically higher than $\nu(k)$, for instance and challenge defined as follows: instance $= (G_1, G_2, q, e, g, H, D, \{(H^{r_x \frac{1}{s+x}}/D^{r_x}, g^{\frac{1}{r_x}})\}_{\forall x \in X})$ where q is an $a(k)$-bit prime, G_1, G_2 are groups of order q returned by $\mathcal{G}(q)$, $e : G_1 \times G_1 \to G_2$ is a bilinear map, $(g, H, D) \leftarrow G_1^3$, and $\{r_x\}_{x \in X}$ and s were all picked at random from $\mathbb{Z}_q^*$; challenge of type 1 is $(H^{r^* \frac{1}{s+x^*}}/D^{r^*}, g^{\frac{1}{r^*}})$ where $r^* \leftarrow \mathbb{Z}_q^*$, while challenge of type 2 is (g^{R_1}, g^{R_2}) for R_1 and R_2 random from $\mathbb{Z}_q^*$.*

Lemma 5. (l,ν)*-BDHBI assumption implies* (l,ν)*-intermediate assumption.*

Proof. Suppose there exists an adversary $\mathcal{A}$ who breaks the intermediate assumption for set X of cardinality $l-1$, and $x^* \notin X$. Then we show an algorithm $\mathcal{B}$ that can break l-BDHBI Assumption.

Algorithm $\mathcal{B}$ will behave as follows: Receive $G, q, e, g, g^{\alpha}, \dots g^{\alpha^l}, g^{\beta}$, and $Z = g^{\frac{1}{\alpha\beta}}$ or $Z = g^R$ for random $R \in Z_q^*$.

Choose random values $\Delta_1, \Delta_2 \leftarrow \mathbb{Z}_q^*$. Implicitly, let $\gamma = \gamma(\alpha) = \Delta_1(\alpha - \Delta_2)\prod_{x \in X}(\alpha + (x - x^*))$. Compute $H = g^{\gamma}$. Note that since this exponent is just an l degree polynomial in α, we can compute this value using $g, \dots g^{\alpha^l}$. If we implicitly define $s = \alpha - x^*$, we will get $H = g^{\Delta_1(\alpha - \Delta_2)\prod_{x \in X}(s+x)}$. (Note that now we know neither s, nor α explicitly.) Note that because of Δ_1, H is uniformly distributed over G_1, and is independent of g. Now we want to provide D. Implicitly we will define $d = \frac{\gamma - \delta}{\alpha}$, where $\delta = \Delta_1 \Delta_2 \prod_{x \in X}(x - x^*)$ is the constant term of the polynomial in α (represented by $\gamma(\alpha)$). Note now that δ is a quantity $\mathcal{B}$ can compute, while d is only defined implicitly. Since d is a polynomial expression in α, $D = g^d$ can be expressed as a sum of terms

$g, g^{\alpha}, \ldots, g^{\alpha^{l-1}}$, and computed using the given values. Finally, note that, because of Δ_2, D is uniformly distributed over G_1, and is independent of (g, H).

For all $\hat{x} \in X$: Let $\gamma'(\hat{x}) = \Delta_1(\alpha - \Delta_2)\prod_{x \in X, x \neq \hat{x}}(\alpha + (x - x^*)) = \frac{\gamma}{\hat{x}+s}$. Compute $v = g^{\gamma'(\hat{x})} = g^{\frac{\gamma}{s+\hat{x}}} = H^{\frac{1}{s+\hat{x}}}$. We then choose a random $r_{\hat{x}} \leftarrow \mathbb{Z}_n^*$. We compute and output $(v^{r_{\hat{x}}}/D^{r_{\hat{x}}}, g^{\frac{1}{r_{\hat{x}}}})$.

For x^*: Implicitly define $r^* = \frac{1}{\beta}$. Compute $u_1 = Z^{\delta}$. If $Z = g^{\frac{1}{\alpha\beta}}$, then this is equal to $g^{\frac{\delta}{\alpha\beta}} = g^{\frac{\gamma}{\alpha\beta} - \frac{\gamma-\delta}{\alpha\beta}} = g^{\frac{\gamma}{\alpha\beta}}/g^{\frac{\gamma-\delta}{\alpha\beta}} = H^{r^*\frac{1}{s+x^*}}/D^{r^*}$. Otherwise, this is equal to g^{R_1} for random R_1. Compute $u_2 = (g^{\beta}) = (g^{\frac{1}{r^*}})$. Output (u_1, u_2).

Finally, if $\mathcal{A}$ guesses that he received $(H^{r^*\frac{1}{s+x^*}}/D^{r^*}, g^{\frac{1}{r^*}})$, $\mathcal{B}$ guesses that $Z = g^{\frac{1}{\alpha\beta}}$, else that $Z = g_q^R$. If $\mathcal{A}$'s guess is correct, then $\mathcal{B}$'s guess is correct. □

Proof. (of Lemma 4) We first define a series of hybrid games:

Game Hybrid i**:** Obtain $(p, t) \leftarrow \mathit{SimParams}(1^k)$, and $(\mathit{PK}, \mathit{SK}) \leftarrow \mathit{SimG}(\mathtt{p}, t)$ and then $\mathcal{A}(\mathtt{p}, t, \mathit{PK})$ gets access to the following oracle: The oracle begins by storing $j = 0$. On query x, the oracle (1) checks if x has previously been queried, and if so, returns the answer stored. Otherwise, (2) if $j < i$ the oracle obtains $(y, w) \leftarrow \mathit{SimSample}(\mathtt{p}, t, \mathit{SK}, x)$ and $\pi \leftarrow \mathit{SimProve}(\mathtt{p}, \mathit{SK}, x, y, w, t)$, returns and stores (y, π), and increments j. (3) Or if $j \geq i$, the oracle computes $y = \mathtt{Eval}(\mathtt{p}, \mathit{SK}, x)$ and $\pi \leftarrow \mathtt{Prove}(\mathtt{p}, \mathit{SK}, x)$, returns and stores (y, π) and increments j.

Note that in this case, $G(\mathtt{p})$ is identical to $\mathit{SimG}(\mathtt{p}, t)$ for all $\mathtt{p}, t$, so Game Hybrid 0 is identical to Game Real Proofs. Game Hybrid Q, where Q is the maximum number of distinct oracle queries (not including repeated queries) that the adversary is allowed to make, is identical to Game Simulated Proofs. Thus, we have only to show the following lemma:

Lemma 6. *Suppose the (l, ν)-BDHBI Assumption holds in one of the two subgroups of a composite bilinear group. Then, when the size of the domain is at most l, no PPT adversary can distinguish Game Hybrid $i-1$ from Game Hybrid i with advantage higher than νl.*

Proof. Suppose there exists an adversary $\mathcal{A}$ who can distinguish Game Hybrid $i-1$ from Game Hybrid i when the domain $\mathcal{D}$ is of size l. Then we show an algorithm $\mathcal{B}$ that can break the l-intermediate assumption with advantage ϵ.

First we make a guess x^* about which input $\mathcal{A}$ will give in its ith distinct oracle query. Since $|\mathcal{D}| = l$, and all values given to $\mathcal{A}$ will be independent of x^*, we will be correct with probability $1/l$.

Now, we will show an algorithm $\mathcal{B}$, which can, with nonnegligible probability, break the intermediate assumption for set $X = \mathcal{D} \setminus \{x^*\}$ and the x^* chosen above. $\mathcal{B}$ will receive $G, p, q, e, g_p, g_q, H_q, D_q, \{(H_q^{\frac{r_x}{s_q+x}}/D_q^{r_x}, g_q^{\frac{1}{r_x}})\}_{\forall x \in X}, (Z_1, Z_2)$ for $g_q, H_q, D_q \leftarrow G_q$, and randomly chosen (but unknown) $\{r_x\}_{x \in X}, s_q \leftarrow \mathbb{Z}_q^*$. Here, either $(Z_1, Z_2) = (H_q^{r^*\frac{1}{s_q+x^*}}/D_q^{r*}, g_q^{\frac{1}{r^*}})$ or $(Z_1, Z_2) = (g_q^{R_1}, g_q^{R_2})$ for random $R_1, R_2 \leftarrow \mathbb{Z}_q^*$.

First, $\mathcal{B}$ prepares the parameters as follows: Choose $H_p, A, D_p \leftarrow G_p$ and compute $g = g_p g_q$, $H = H_p H_q$, $D = D_p D_q$. Set $\mathtt{p} = (G_1, G_2, n, e, g, H, A, D)$. Let $s_p \leftarrow \mathbb{Z}_p^*$, and $PK = A^{s_p}$. Implicitly, set $s \in \mathbb{Z}_n^*$ to the the element such that $s \mod p = s_p$, and $s \mod q = s_q$. $\mathcal{B}$ sends $\mathtt{p}$ and trapdoor $t = (g_p, g_q, H_p, H_q, D_p, D_q)$ to $\mathcal{A}$.

Now $\mathcal{B}$ must answer $\mathcal{A}$'s queries. We assume (WLOG) that $\mathcal{A}$ does not repeat queries.

When $\mathcal{A}$ sends its j^{th} query, $\hat{x}$, $\mathcal{B}$ proceeds as follows:
If $j < i$: if $\hat{x} = x^*$, then $\mathcal{B}$ has guessed wrong about which value $\mathcal{A}$ will choose in his ith distinct query (if it is used again later, it will be repeated and thus not distinct), so $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ chooses a random $w' \in \mathbb{Z}_q^*$. Let $y = e(H_p^{\frac{1}{s_p+\hat{x}}}, g_p)e(H_q, g_q)^{w'}$. Choose a random $r \leftarrow \mathbb{Z}_n^*$. Let $\pi_1 = (H_p^{r\frac{1}{s_p+\hat{x}}}/D_p^r)(H_q^{w'r}/D_q^r)$. Let $\pi_2 = g^{\frac{1}{r}}$ and $\pi_3 = A^{\frac{\hat{x}+s_p}{r}}$. If we implicitly set $w = w'h_q$, (where $H_q = g_q^{h_q}$) then these value will be distributed as in the output of *SimSample* and *SimProve*. Output $(y, \pi = (\pi_1, \pi_2, \pi_3))$.

If $j = i$: If $\hat{x} \neq x^*$, then $\mathcal{B}$ has guessed wrong, so it aborts. Otherwise, choose random $r_p \leftarrow \mathbb{Z}_p^*$. Implicitly set $r \in \mathbb{Z}_n^*$ to be the element such that $r \mod q = r^*$ and $r \mod p = r_p$. Compute $\pi_1 = H_p^{r_p\frac{1}{x^*+s_p}}/D_p^{r_p}Z_1$. Note that, if $Z_1 = H_q^{r^*\frac{1}{s_q+x^*}}/D_q^{r^*}$, then this is equal to $H^{\frac{r}{s+x^*}}/D^r$. Otherwise, this is equal to $H_p^{r_p\frac{1}{x^*+s_p}}/D_p^{r_p}g_q^{R_1}$. Now compute $\pi_2 = g_p^{\frac{1}{r_p}}Z_2$, if $Z_2 = g_q^{\frac{1}{r^*}}$, then this value will be $g^{\frac{1}{r}}$. Otherwise it will be $g_p^{\frac{1}{r_p}}g_q^{R_2}$ Compute $\pi_3 = A^{\frac{s_p+x^*}{r_p}} = A^{\frac{s+x^*}{r}}$. Finally, compute $y = e(\pi_1, \pi_2)e(D, g)$. Output $(y, \pi = (\pi_1, \pi_2, \pi_3))$ to the adversary.

If $j > i$, we know $\hat{x} \neq x^*$, and $\hat{x} \in X$. Let $V_1 = H_q^{r_{\hat{x}}\frac{1}{s_q+\hat{x}}}/D_q^{r_{\hat{x}}}$, and $V_2 = g^{\frac{1}{r_{\hat{x}}}}$, as provided in $\mathcal{B}$'s input. $\mathcal{B}$ chooses a random $r_p \leftarrow \mathbb{Z}_p^*$. Implicitly, set $r \in \mathbb{Z}_n^*$ for this query to be the element such that $r \mod p = r_p$, and $r \mod q = r_{\hat{x}}$. $\mathcal{B}$ computes $\pi_1 = (H_p^{r_p\frac{1}{s_p+\hat{x}}}/D_p^{r_p})V_1 = H^{r\frac{1}{s+\hat{x}}}/D^r$, $\pi_2 = g_p^{\frac{1}{r_p}}V_2 = g^{\frac{1}{r}}$, and $\pi_3 = A^{\frac{s_p+\hat{x}}{r_p}} = A^{\frac{\hat{x}+s}{r}}$. Finally, $\mathcal{B}$ computes $y = e(\pi_1, \pi_2)e(D, g)$ and outputs $(y, \pi = (\pi_1, \pi_2, \pi_3))$ to $\mathcal{A}$.

Finally, $\mathcal{B}$ gets $\mathcal{A}$'s guess bit b. If $\mathcal{A}$ guesses that this is Game Hybrid $i-1$, $\mathcal{B}$ guesses that $(Z_1, Z_2) = (H_q^{r^*\frac{1}{s_q+x^*}}/D_{q^*}^{r_*}, g_q^{\frac{1}{r^*}})$; otherwise $\mathcal{B}$ guesses that $(Z_1, Z_2) = (g_q^{R_1}, g_q^{R_2})$. If $\mathcal{A}$ guesses correctly, $\mathcal{B}$'s guess will also be correct.

$\mathcal{B}$ has a $\frac{1}{l}$ probability of not aborting. Suppose that when $\mathcal{B}$ aborts, it returns a random bit. Then $\mathcal{B}$'s quess is correct with probability $(1-\frac{1}{l}) * \frac{1}{2} + \frac{1}{l} * (\frac{1}{2}+\epsilon) = \frac{1}{2} + \frac{\epsilon}{l}$, where ϵ is $\mathcal{A}$'s advantage. Thus, if $\mathcal{A}$'s advantage is $\epsilon > \nu l$ then $\mathcal{B}$'s advantage is higher than ν, contradicting the assumption. □

For the theorem to follow, we observe that the overall reduction from breaking the simulatability game to breaking the BDHBI assumption uses at most $(l+1)$ hybrids, and so the adversary's advantage ϵ translates into the reduction's advantage ϵ/l^2 in breaking BDHBI. □

Remark. Since the construction above satisfies the premise of Lemma 2, it can be converted to an sVRF with binary range using the construction in Section 2.2.

5 Multi-Theorem NIZK from One-Theorem NIZK via sVRFs

Here, we omit the definition of single-theorem and multi-theorem NIZK, but refer the reader to Blum et al. [BDMP91] and Feige, Lapidot, Shamir [FLS99]. Instead, we informally sketch this definition:

Algorithms `NIZKProve` and `NIZKVerify` The algorithm `NIZKProve` takes as input the common random string σ of length $\ell(k)$, and values (x, w), $|x| \leq q(k)$, such that $x \in L$, and w is a witness to this. `NIZKProve` outputs a *proof* Π. `NIZKVerify` is the algorithm that takes (σ, x, Π) as input, and outputs *ACCEPT* or *REJECT*.

Perfect completeness For all $x \in L$, for all witnesses w for x, for all values of the public random string σ, and for all outputs π of $\texttt{NIZKProve}(\sigma, x, w)$, $\texttt{NIZKVerify}(\sigma, x, \pi) = ACCEPT$.

Soundness $s(k)$ For all adversarial prover algorithms $\mathcal{A}$, for a randomly chosen σ, the probability that $\mathcal{A}$ can produce (x, π) such that $x \notin L$ but $\texttt{NIZKVerify}(\sigma, x, \pi) = ACCEPT$, is $s(k)$.

Single-theorem ZK There exists an algorithm `SimProveOne` that, on input 1^k and $x \in L$, $|x| \leq q(k)$, outputs simulated CRS σ^S together with a simulated proof Π^S, such that (σ^S, Π^S) are distributed indistinguishably from (σ, Π) produced by generating a random CRS σ, and obtaining Π by running `NIZKProve`.

Multi-theorem ZK There exist algorithms `SimCRS` and `NIZKSimProve`, as follows: $\texttt{SimCRS}(1^k)$ outputs (σ, s). For all x, $\texttt{NIZKSimProve}(\sigma, s, x)$ outputs a *simulated proof* Π^S. Even for a sequence of adversarially and adaptively picked $(x_1, \ldots, x_m)$ (m is polynomial in k), if for all $1 \leq i \leq m$, $x_i \in L$, then the simulated proofs $\Pi_1^S, \ldots, \Pi_m^S$ are distributed indistinguishably from proofs $\Pi_1, \ldots, \Pi_m$ that are computed by running $\texttt{NIZKProve}(\sigma, x_i, w_i)$, where w_i is some witness that $x_i \in L$.

Suppose that, for a language L, we are given a single-theorem NIZK proof system (`ProveOne`, `VerOne`) in the CRS model, with perfect completeness and unconditional soundness error $s(k)$. Let $\ell(k)$ denote the function such that an $\ell(k)$-bit random string serves as the CRS for this proof system. Let $q(k)$ denote the polynomial upper bound on the size of the input x. Suppose also that we are given a simulatable VRF (G, `Eval`, `Prove`, `Verify`) in the parameter model *Params*, whose domain is $[1, \ell(k)]$, with range $\{0, 1\}$. Consider the following construction for multi-theorem NIZK in the common reference string model for instances of size k:

Generate common parameters The algorithm `NIZKParams`: Obtain $\sigma_1 \leftarrow \{0,1\}^{\ell(k)}$. Let $\texttt{p} \leftarrow Params(1^k)$. The values $(\sigma_1, \texttt{p})$ are the parameters of the system.

Prove The algorithm `NIZKProve`: On input instance $x \in L$ with witness w, and common parameters $(\sigma_1, \mathtt{p})$ do: Obtain $(PK, SK) \leftarrow G(1^k, \mathtt{p})$. Let R be the $\ell(k)$-bit string computed as follows: for $1 \leq i \leq \ell(k)$, $R_i = \mathtt{Eval}(\mathtt{p}, SK, i)$, where R_i denotes the ith bit of R. For $1 \leq i \leq \ell(k)$, let $\pi_i \leftarrow \mathtt{Prove}(\mathtt{p}, SK, i)$. Let $\sigma = \sigma_1 \oplus R$. Obtain $\Pi' \leftarrow \mathtt{ProveOne}(\sigma, x, w)$. Output the proof $\Pi = (PK, R, \pi_1, \ldots, p_{\ell(k)}, \Pi')$.

Verify The algorithm `NIZKVerify`: On input x and Π, and common parameters $(\sigma_1, \mathtt{p})$, do: (1) for $1 \leq i \leq \ell(i)$, check that $\mathtt{Verify}(\mathtt{p}, PK, i, R_i, \pi_i)$ accepts; (2) let $\sigma = \sigma_1 \oplus R$; check that $\mathtt{VerOne}(\sigma, x, \Pi')$ accepts; if all these checks passed, accept, otherwise, reject.

Theorem 2. *If for a language L, $(\mathtt{ProveOne}, \mathtt{VerOne})$ is a single-theorem NIZK proof system in the $\ell(k)$-bit CRS model for instances of length up to $q(k)$ with perfect completeness and unconditional soundness error $s(k)$, and $(G, \mathtt{Eval}, \mathtt{Prove}, \mathtt{Verify})$ in the parameter model $Params(1^k)$, is a strong simulatable VRF with domain $[1, \ell(k)]$ and range $\{0, 1\}$, then the above construction is a multi-theorem NIZK proof system in the public parameters model that comprises the $\ell(k)$-bit CRS and $Params(1^k)$, with perfect completeness and unconditional soundness error $s(k)2^{u(k)}$, where u denotes the bit length of a PK output by $G(\mathtt{p})$ on input $\mathtt{p} \leftarrow Params(1^k)$.*

Proof. (Sketch) The perfect completeness property follows from the perfect completeness property of the single-theorem NIZK.

Let us show the multi-theorem zero-knowledge property. Recall that, by the definition of (strong) sVRF, we have a simulator consisting of *SimParams*, *SimG* and *SimProve* such that, if (PK, SK) were generated by *SimG*, then for a randomly sampled y from the range of the sVRF, and for any x in the domain, *SimProve* can generate a fake proof that $y = \mathtt{Eval}(SK, x)$. (See Section 2.)

Also recall that by the definition of NIZK, there exists a simulator `SimProveOne` such that no adversary $\mathcal{A}$ can distinguish between the following two distributions for any $x \in L$ and any witness w for x: (1) choose $\sigma \leftarrow \{0, 1\}^{\ell(k)}$, and let $\Pi \leftarrow \mathtt{ProveOne}(\sigma, x, w)$; give (σ, Π) to $\mathcal{A}$; (2) $(\sigma, \Pi) \leftarrow \mathtt{SimProveOne}(1^k, x)$; give (σ, Π) to $\mathcal{A}$.

Consider the following simulator S for our multi-theorem NIZK construction. The simulator will consist of `SimCRS` that generates the simulated parameters, and of `NIZKSimProve` that generates the simulated proof. `SimCRS` works as follows: generate $(\mathtt{p}, t) \leftarrow SimParams$, and $\sigma_1 \leftarrow \{0, 1\}^{\ell(k)}$; publish $(\sigma_1, \mathtt{p})$ as the parameters of the system. `NIZKSimProve` works like this: generate $(\sigma, \Pi') \leftarrow \mathtt{SimProveOne}(1^k, x)$. Then let $R = \sigma \oplus \sigma_1$. Let $(PK, SK) \leftarrow SimG(\mathtt{p}, t)$. For $1 \leq i \leq \ell(k)$, let $\pi_i = SimProve(\mathtt{p}, SK, x, R_i, t)$. Output the proof $\Pi = (PK, R, \pi_1, \ldots, p_{\ell(k)}, \Pi')$. In the full version, we show that the view that the adversary obtains in the simulation is indistinguishable from the view obtained when interacting with the prover.

We now show soundness. We are given that, for $\sigma \leftarrow \{0, 1\}^{\ell(k)}$, the probability that there exists $x \notin L$ and a proof Π' such that $\mathtt{Verify}(\sigma, x, \Pi') = 1$, is $s(k)$.

Consider $\mathtt{p} \leftarrow Params$, and $(PK, SK) \leftarrow G(1^k)$. Let R be as defined in `NIZKProve`: $R_i = \mathtt{Eval}(SK, i)$. Note that by the verifiability property of the sVRF, there is a *unique* R for which there exists a proof of correctness $(\pi_1, \ldots, \pi_{\ell(k)})$. The probability, over the choice of σ_1, that there exists $x \notin L$ and a proof Π' such that $\texttt{Verify}(R \oplus \sigma_1, x, \Pi') = 1$ (if such an x exists, we say that PK is *bad* for σ_1), is still $s(k)$, since we first fixed $\texttt{p}$ and PK, and then randomly chose σ_1. By the union bound, since there are $2^{u(k)}$ possible PK's, for every $\texttt{p}$, the probability that there exists a bad PK for a particular σ_1, is $s(k)2^{u(k)}$. $\square$

Remark. Note that if an NIZK proof system is in the hidden-random-string (HRS) model (such as those due to Feige, Lapidot and Shamir [FLS99] and Kilian and Petrank [KP98]), then we can take advantage of it as follows: the hidden random string can be obtained the way that σ is currently obtained by the prover in the construction above; only in the construction above, the prover reveals the entire string σ and the proof that each bit of σ is computed correctly; while in the HRS model, the prover only reveals the subset of bits of the hidden random string that he needs to reveal. This observation was inspired by Dwork and Naor's construction of zaps from VRFs and verifiable PRGs [DN00] based on NIZK using HRS model. We give more details on consequences in the HRS model in the full version.

Acknowledgments: We thank Leo Reyzin and Markulf Kohlweiss for helpful discussions. We thank UCLA's Institute for Pure and Applied Mathematics for hosting us while part of this research was carried out. Melissa Chase is supported by NSF grant CNS-0374661 and NSF Graduate Research Fellowship. Anna Lysyanskaya is supported by NSF CAREER grant CNS-0374661 and NSF grant CNS-0627553.

References

[BB04a] D. Boneh and X. Boyen. Efficient selective id secure identity based encryption without random oracles. In *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 223–238.

[BB04b] D. Boneh and X. Boyen. Short signatures without random oracles. In *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 54–73.

[BCC04] E. Brickell, J. Camenisch, and L. Chen. Direct anonymous attestation. In *11th ACM CCS*, pages 225–234. ACM press, 2004.

[BDMP91] M. Blum, A. De Santis, S. Micali, and G. Persiano. Non-interactive zero-knowledge. *SIAM Journal of Computing*, 20(6):1084–1118, 1991.

[BFM88] M. Blum, P. Feldman, and S. Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *20th Annual ACM STOC*, pages 103–112, May 1988.

[BGN05] D. Boneh, E. Goh, and K. Nissim. Evaluating 2-DNF formulas on ciphertexts. In *TCC 2005*, volume 3378 of *LNCS*, pages 325–341.

[BR93] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *1st ACM CCS*, pages 62–73. ACM press, 1993.

[Bra99] S. Brands. *Rethinking Public Key Infrastructure and Digital Certificates—Building in Privacy.* PhD thesis, Eindhoven Institute of Technology, Eindhoven, The Netherlands, 1999.

[CH02] J. Camenisch and E. Van Herreweghen. Design and implementation of the *idemix* anonymous credential system. Technical Report Research Report RZ 3419, IBM Research Division, May 2002.

[Cor00] J.-S. Coron. On the exact security of full domain hash. In *CRYPTO 2000*, volume 1880 of *LNCS*, pages 229–235.

[DCP97] A. De Santis, G. Di Crescenzo, and G. Persiano. Randomness-efficient non-interactive zero-knowledge (extended abstract). In *ICALP*, pages 716–726, 1997.

[DMP88] A. De Santis, S. Micali, and G. Persiano. Non-interactive zero-knowledge proof systems. In *CRYPTO '87*, volume 293 of *LNCS*, pages 52–72.

[DN00] C. Dwork and M. Naor. Zaps and their applications. In *FOCS*, pages 283–293, 2000.

[Dod02] Y. Dodis. Efficient construction of (distributed) verifiable random functions. In *PKC*, volume 2567 of *LNCS*, pages 1–17.

[DY05] Y. Dodis and A. Yampolskiy. A verifiable random function with short proofs and keys. In *PKC 2005*, pages 416–432.

[FLS99] U. Feige, D. Lapidot, and A. Shamir. Multiple noninteractive zero knowledge proofs under general assumptions. *SIAM Journal on Computing*, 29(1):1–28, 1999.

[FS87] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO '86*, volume 263 of *LNCS*, pages 186–194.

[GGM86] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, October 1986.

[GK03] S. Goldwasser and Y. Tauman Kalai. On the (in)security of the Fiat-Shamir paradigm. In *44th FOCS*, pages 102–115. IEEE Computer Society Press, 2003.

[GO92] S. Goldwasser and R. Ostrovsky. Invariant signatures and non-interactive zero-knowledge proofs are equivalent. In *CRYPTO '92*, volume 740 of *LNCS*, pages 228–24.

[GOS06] J. Groth, R. Ostrovsky, and A. Sahai. Perfect non-interactive zero knowledge for NP. In *Eurocrypt '06*, volume 4004 of *LNCS*, pages 339-358.

[HILL99] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal of Computing*, 28(4):1364–1396, 1999.

[KP98] J. Kilian and E. Petrank. An efficient noninteractive zero-knowledge proof system for NP with general assumptions. *Journal of Cryptology*, 11(1):1–27, 1998.

[Lys02] A. Lysyanskaya. Unique signatures and verifiable random functions from the DH-DDH separation. In *CRYPTO 2002*, volume 2442 of *LNCS*, pages 597–612.

[MRV99] S. Micali, M. Rabin, and S. Vadhan. Verifiable random functions. In *40th FOCS*, pages 120–130. IEEE Computer Society Press, 1999.

[Nao91] M. Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):51–158, 1991.