Bilinear maps in Verifiable Random Functions

Ananth Raghunathan*
ananthr@stanford.edu
Stanford University

Abstract

One of the biggest reasons for the popularity and versatility of elliptic curves in cryptography, besides the lack of "better-than-black-box" discrete log algorithms, is the presence of a bilinear map. In this short paper, we look into the definitions, motivations, and constructions of VRFs and note that groups equipped with a bilinear map can be used to construct a variety of different VRFs.

*This paper is my final paper for the course at Stanford titled "Elliptic Curves in Cryptography" by David Freeman (http://www.stanford.edu/class/cs259c/). Results presented here are paraphrased from various papers. Any mistakes and inaccuracies are solely my own and feedback is much appreciated.

1 Definitions

In this section, we define pairings and verifiable random functions. In crypto literature elliptic curve groups are generally treated multiplicatively and we will use the same convention.

1.1 Pairings

We briefly review the necessary facts about bilinear maps and bilinear map groups [Mil04]. Let $\mathbb{G}$ and $\mathbb{G}_{\mathbf{T}}$ be two (multiplicative) cyclic groups of prime order p and let g be a generator of $\mathbb{G}$. A bilinear map is the modified Weil pairing $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_{\mathbf{T}}$ with the following properties:

1. Bilinear: for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}$ we have $\hat{e}(u^a, v^b) = \hat{e}(u, v)^{ab}$.
2. Non-degenerate: $\hat{e}(g, g) \neq 1$.

We say that $\mathbb{G}$ is a bilinear group if the group operation in $\mathbb{G}$ can be computed efficiently and there exists a group $\mathbb{G}_{\mathbf{T}}$ and an efficiently computable bilinear map $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_{\mathbf{T}}$ as above. Note that $\hat{e}(\cdot, \cdot)$ is symmetric since $\hat{e}(g^a, g^b) = \hat{e}(g, g)^{ab} = \hat{e}(g^b, g^a)$.

1.2 VRFs

Verifable Random Functions, introduced by Micali, Rabin, and Vadhan [MRV99] are Pseudorandom Functions (PRFs)[1] where the party holding the secret key can produce a non-interactive proof that the PRF was evaluated correctly. The proof should not interfere with the pseudorandom properties of the PRF. To formalize this notion, we take the following definition directly from [BMR10].

Definition. A VRF is an efficiently computable function $F : K \times X \to Y$ equipped with three algorithms:

1. $\mathsf{Gen}(1^\lambda)$ outputs a pair of keys $(\mathsf{pk}, \mathsf{sk})$ for a security parameter λ.
2. $\mathsf{Prove}(\mathsf{sk}, x)$ computes $\big(F(\mathsf{sk}, x), \pi(\mathsf{sk}, x)\big)$, where $\pi(\mathsf{sk}, x)$ is a proof of correctness.
3. $\mathsf{Ver}(\mathsf{pk}, x, y, \pi)$ verifies that $y = F(\mathsf{sk}, x)$ using the proof π.

Security for a VRF is defined using two experiments between a challenger and a Q-query adversary $\mathcal{A}$. For $b \in \{0, 1\}$, the challenger in Exp_b works as follows:

> The challenger chooses a random key $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$.
> The adversary (adaptively) sends queries $x_1, \ldots, x_Q$ in X to the challenger who responds with $F(\mathsf{sk}, x_i)$ and proof $\pi(\mathsf{sk}, x_i)$ for $i = 1, \ldots, Q$.
> The adversary issues a special challenge query x^*. If $b = 0$, the challenger responds with $F(\mathsf{sk}, x^*)$. If $b = 1$, the challenger chooses random $y \in Y$ and responds with y.
> Eventually the adversary outputs a bit $b' \in \{0, 1\}$.

For $b \in \{0, 1\}$ let W_b be the probability that $\mathcal{A}$ outputs 1 in Exp_b. Define $\mathsf{VRF}_{\mathsf{adv}}[\mathcal{A}, F] := |W_0 - W_1|$.

[1] Informally, a PRF is a keyed function family, a random member of which is indistinguishable from a random function on the same domain and range. Its security can be formally defined in a manner largery similar to the VRF security game described later in this section.

Definition 1.1. A VRF is said to be secure if it satisfies the following properties.

1. **Pseudorandomness:** For every efficient adversary $\mathcal{A}$, $\mathsf{VRF}_{\mathrm{adv}}[\mathcal{A}, F]$ is a negligible function.
2. **Verifiability:** For $(y, \pi) \leftarrow \mathsf{Prove}(\mathsf{sk}, x)$, $\mathsf{Ver}(\mathsf{pk}, x, y, \pi) = 1$.
3. **Uniqueness:** no values of $(\mathsf{pk}, x, y_1, y_2, \pi_1, \pi_2)$ satisfy $\mathsf{Ver}_{\mathsf{pk}}(x, y_1, \pi_1) = \mathsf{Ver}_{\mathsf{pk}}(x, y_2, \pi_2) = 1$ for $y_1 \neq y_2$.

Applications of VRFs. VRFs have a variety of interesting applications, partially because they allow a short commitment to an exponential number of pseudorandom bits. Abdalla et al. [ACF09] and Hohenberger-Waters [HW10] provide a nice summary of applications where VRFs are used as a building block. These include resettable zero-knowledge proofs, micropayment schemes, updatable zero-knowledge databases and verifiable transaction escrow schemes. It also appears likely that suitable VRFs could be a useful alternative in several applications which, as part of the system, output the value of the PRF together with a proof (non-interactive or otherwise) that the evaluation was correct with some additional properties. Examples of this might include compact e-cash, keyword search, set intersection protocols and adaptive oblivious transfer protocols.

2 Preliminaries

2.1 Relation to signatures

Digital signatures are a standard primitive in cryptography. In a digital signature scheme, the secret key holder first publishes (or commits to) some public information referred to as the verification key. A signature scheme is correct if arbitrary third parties can *verify* a signature on a claimed message only using this public information. Security requires that given the public information, and signatures on arbitrary messages, the adversary is unable to compute the secret key, or more strongly, the adversary cannot construct a *forgery*—a new message and a successful signature on that message.

With this definition in mind, it is fairly straightforward to see several parallels between VRFs and signature schemes. Indeed, traditional signature schemes are always non-interactive and the public verification keys are committed to ahead of time. One might venture a guess that setting $f(x) = \sigma(x)$ for a signature scheme that produces signatures $\sigma(x)$ on input message x might suffice. However, such schemes do not directly give rise to VRFs for two reasons as was observed in [MRV99].

1. There may be many valid signatures for a given string x (violating the unique provability requirement). This is because signature schemes usually do not require uniqueness.
2. Unforgeability can only show that $\sigma(x)$ is unpredictable, not necessarily pseudorandom.

Indeed, in the recent decade, starting from the first Identity-Based Encryption scheme (IBE) in [BF01], there have been several signature schemes based on bilinear groups equipped with a pairing [BLS04, BBS04, BB04, BB08]. Thus, elliptic curve groups seem like a good candidate choice for constructing VRFs. The first difficulty, the lack of uniqueness, is not difficult to tackle as there are constructions of unique signature schemes in literature ([Lys02]). Constructing *pseudorandom* outputs from signatures is rather more challenging. Thus, before each construction in the paper, we will briefly discuss hardness assumptions that allow us to achieve pseudorandomness.

VUFs. When the concept of VRFs were first introduced by Micali, Rabin, and Vadhan [MRV99], pairing-based cryptography was still in its infancy. As seen above, all the signature schemes cited were constructed only later. Thus, the authors described the following notion: We define a weaker variation of VRFs called *Verifiable Unpredictable Functions* (VUFs) where in the security game, instead of an adversary requiring to distinguish between Exp_b for $b \in \{0,1\}$, during the *challenge phase*, the experiment picks a random message $x^* \xleftarrow{R} X$ (such that x^* was not queried before) and provides $F(\mathsf{sk}, x^*)$ to the adversary. And the adversary wins the game when he outputs x^*.

Then informally, we can state the following: Given a unique signature scheme (i.e., a scheme that accepts a unique signature for every message given any fixed, but even improperly chosen public key) we can use the above idea to construct a VUF. To use VUFs to construct VRFs, we require answers to two questions:

1. Do verifiable unpredictable functions imply verifiable pseudorandom functions?
2. Can we construct verifiable unpredictable functions?

2.2 A black-box construction from VUFs

In the paper that introduced the concept of VRFs ([MRV99]) the authors first constructed a VUF from the RSA assumption and then used a generic transformation from the Goldreich-Levin theorem [GL89]. Given a VUF $f(\cdot)$, the VRF $f'(\cdot)$ is defined by $f'(x) = \langle f(x), r\rangle$, where r is a binary vector chosen uniformly and placed in the public key and $\langle \cdot, \cdot \rangle$ denotes inner product $(\mathrm{mod}\ 2)$. The proof that $f'(x) = \sigma$ consists of a value v such that $\langle v, r\rangle = \sigma$ and a proof that $f(x) = v$. To show that the VRF is pseudorandom, we need to argue that a PPT adversary that predicts $f'(x) = \langle f(x), r\rangle$ at an unseen value with probability at least $1/2 + \text{non-negl}(|x|)$ will be able to reconstruct $f(x)$ with probability at least non-negl$(|x|)$.

The basic idea behind constructing VUFs follows largely the idea given in Section 2.1 and the basic signature scheme that the authors use is RSA signatures. However, this construction does not use pairings and only works on a *small domain*, i.e., the domain of inputs X is small. It is also very inefficient in its reductions and requires an inefficient tree-based construction of the resulting VRF to get a VRF with arbitrary input size and small output size. In some sense, the inefficiency of this above construction is expected given its generality and the fact that it has to convert *unpredictability* into much stronger property of *pseudorandomness*. This means that the resulting VRF constructions are very bulky and inelegant. Thus, we do not elaborate further on this construction.

2.3 Groups with easy DDH

Recollect that for a group $\mathbb{G}$ with a generator g of order p, a 4-tuple $(X, Y, Z, W) \in \mathbb{G}^4$ is a DDH tuple iff for some $\alpha \in \mathbb{Z}_p$, $Y = X^\alpha$ and $W = Z^\alpha$. The DDH problem asks: Is a given 4-tuple (g, X, Y, Z) a DDH tuple or not. The "search version" of this decision problem, called the computational Diffie Hellman problem asks: given a 3-tuple (g, X, Y) find $Z \in \mathbb{G}$ such that (g, X, Y, Z) is a DDH tuple.

A common recurring theme in the constructions that follow will be groups where CDH or a variant of the assumption is hard (usually we require stronger assumptions than CDH to hold) but DDH is easy. To see how DDH might be easy consider a group $\mathbb{G}$ with an efficiently computable map $\hat{e}(\cdot, \cdot)$. Given a 4-tuple (X, Y, Z, W), to check whether it is a DDH tuple or not, simply check if $\hat{e}(X, W) = \hat{e}(Y, Z)$ or not. It is easy to see from the correctness of the weil pairing that the test succeeds iff the given tuple is a DDH tuple. We will use this idea repeatedly in the rest of the paper.

3 Small domain constructions

Unlike in Section 2.2, where we first constructed a verifiable function that was not sufficiently pseudorandom and then worked on making them "more" random, in this section, we describe constructions that first start off with a PRF and then augment the constructions to provide verifiability.

A note on hardness assumptions. All the constructions below require variants of the DDH assumption to prove security. [NR97] showed that DDH suffices to construct PRFs, but it is unclear whether assumptions that do not involve a bilinear map can be used to construct *efficient* VRFs. In fact, the main theme of this paper is to show that for efficient constructions, the only technique used in crypto literature involves using pairings.

Unfortunately directly assuming DDH in groups equipped with a bilinear map is false. We already saw this in Section 2.3. Thus, our assumptions must implicitly require all PPT adversaries to fail to distinguish two distributions (almost always in the target group $\mathbb{G}_{\mathbf{T}}$), distinguishing whom must be a task that is "harder" than DDH. A formal treatment of this general idea is presented in [Dod03, Section 2.2].

However, at this juncture, we note that the theme of this paper is not an analysis of the hardness assumptions, but rather to study how pairings allow us to *verify* the PRF constructions. Although the assumptions are usually constructed (and justified) to closely match the constructions. the randomized self-reducibility of such assumptions allow us to phrase them as *non-interactive* assumptions[2] which give us VRFs secure under an *interactive*[3] security model.

3.1 Dod03 construction

Dodis in [Dod03] constructs a PRF (which is easily modified to give a VRF) from a variant of the DDH assumption called sum-free DDH ([Dod03, Def. 4]). The scheme described below also requires an efficiently computable code $C : \{0,1\}^{\ell} \to \{0,1\}^{L}$ that takes ℓ-bit inputs and produces L-bit codewords (for some $L > \ell$). The code C is constructed such that the codewords are 4-wise independent ([Dod03, Theorem 1]). Thus, $X = \{0,1\}^{\ell}, K = (\mathbb{G} \times \mathbb{Z}_p^{L+1})$, and $Y = \mathbb{G}$. Here ℓ and L decide the strength of the sum-free DDH-assumption that is required for pseudorandomness.

1. $\mathsf{Gen}(1^{\lambda})$: Choose $(\mathbb{G}, g, p)$ a group with a bilinear map and a generator g of order p. Pick random $a_1, \ldots, a_{L+1} \xleftarrow{R} \mathbb{Z}_q$. Set $h = g^{a_{L+1}}, A_i = h^{a_i}$. Set the public key $\mathsf{pk} = (\mathbb{G}, q, g, h, y_i)$ and the secret key $\mathsf{sk} = (\mathsf{pk}, \{a_i\})$.

2. $\mathsf{Prove}(\mathsf{sk}, x)$: Output $(\sigma_1, \ldots, \sigma_L)$ where

$$\sigma_j = g^{\prod_{i \le j, C(x)_i = 1} a_i}$$

   for $1 \le j \le L$. In particular, $F(\mathsf{sk}, x) = \sigma_L$ and $(\sigma_1, \ldots, \sigma_{L-1})$ is the proof $\pi(\mathsf{sk}, x)$.

3. $\mathsf{Ver}(\mathsf{pk}, x, y, \pi)$: Let $y = \sigma_L$. Set $\sigma_0 = g$. Check for $i = 1, \ldots, L$ that $(\sigma_{i-1}, \sigma_i, h, A_i)$ form a DDH tuple when $C(x)_i = 1$ or that $\sigma_{i-1} = \sigma_i$ if $C(x)_i = 0$. Output 1 if all tests are successful; otherwise output 0.

[2] Loosely, assumptions whose challengers do not interact with the adversary breaking the assumption.
[3] Note that the security definition in Section 1.2 requires interactions.

The pseudorandomness of the above scheme follows from the sum-free DDH assumption and an appropriately chosen code C. Verifiability follows in a fairly straightforward manner from how Prove and Ver work. The advantage of a group with a bilinear map is in checking whether a given tuple is a DDH or a random tuple.

3.2 DY05 construction

Dodis and Yampolskiy in [DY05] construct a VRF that requires the ℓ-decisional bilinear Diffie-Hellman Inversion (ℓ-DBDHI) assumption ([DY05, Def. 5]). Informally, ℓ-DBDHI asks: given the tuple $(g, g^x, \ldots, g^{(x^\ell)})$ as input, distinguish $\hat{e}(g,g)^{1/x}$ from random. An adversary $\mathcal{A}$ is said to break the ℓ-DBDHI assumption with advantage ε if:

$$\left|\Pr\left[\mathcal{A}\left(g, g^x, \ldots, g^{(x^\ell)}, \hat{e}(g,g)^{1/x}\right) = 1 \mid x \xleftarrow{R} {\mathbb{Z}_p}^*\right]\right| - \left|\Pr\left[\mathcal{A}\left(g, g^x, \ldots, g^{(x^\ell)}, y\right) = 1 \mid y \xleftarrow{R} \mathbb{G}_{\mathbf{T}}\right]\right| \leq \varepsilon.$$

Construction. In the construction described below, $X = \{1, \ldots, \ell\}, K = {\mathbb{Z}_p}^*,$ and $Y = \mathbb{G}$.

1. $\mathsf{Gen}(1^\lambda)$: Choose $(\mathbb{G}, p, g)$ a group with a bilinear map and a generator g of order p. Next, choose a secret $s \xleftarrow{R} {\mathbb{Z}_p}^*$ and set $\mathsf{pk} = (\mathbb{G}, p, g, g^s)$ and $\mathsf{sk} = (\mathsf{pk}, s)$.
2. $\mathsf{Prove}(\mathsf{sk}, x)$: Output $(\hat{e}(g,g)^{1/(x+\mathsf{sk})}, g^{1/(x+\mathsf{sk})})$. Here $\hat{e}(g,g)^{1/(x+\mathsf{sk})}$ is $F(\mathsf{sk}, x)$ and $g^{1/(x+\mathsf{sk})}$ is $\pi(\mathsf{sk}, x)$.
3. $\mathsf{Ver}(\mathsf{pk}, x, y, \pi)$: To verify whether y was computed correctly, check if $\hat{e}(g^x \cdot \mathsf{pk}, \pi) = \hat{e}(g,g)$ and whether $y = \hat{e}(g, \pi)$. If both checks succeed, output 1; otherwise, output 0.

As the construction described above is the easiest to analyze and is very instructive towards showing how to prove VRFs, we will give a brief outline of the theorem and proof. Let DY refer to the above VRF construction.

Theorem 3.1 (Paraphrased from [DY05, Theorem 2])**.** *Suppose that for a constant $\varepsilon > 0$ there are no PPT adversaries with ℓ-DBDHI advantage $> \varepsilon$, then there are no PPT adversaries $\mathcal{A}$ against the DY05 VRF such that $\mathsf{VRF}_{adv}[\mathcal{A}, \mathsf{DY}] > 2^\ell \varepsilon$.*

Proof Outline. As in all crypto proofs, we prove this by deriving a contradiction. We assume there exists a PPT adversary $\mathcal{A}$ that can distinguish between $\hat{e}(g,g)^{1/(x+s)}$ for some x and a random element in $\mathbb{G}_{\mathbf{T}}$ with probability at least ε', then we can construct a PPT adversary $\mathcal{B}$ that breaks the ℓ-DBDHI challenge with advantage at least $2^{-\ell}\varepsilon'$. It is easy to see that this completes the proof.

The challenger $\mathcal{B}$ given $(g, g^\alpha, \ldots, g^{\alpha^\ell}, \Gamma)$ must output 1 if $\Gamma = \hat{e}(g,g)^{1/\alpha}$ and 0 otherwise. To do this, $\mathcal{B}$ chooses to distinguish the VRF on a specific message x_0 and let $\beta = \alpha - x_0$. $\mathcal{B}$ knows neither β nor α. The key point however is this; given $(g, g^\alpha, \ldots, g^{\alpha^\ell})$ $\mathcal{B}$ uses the binomial theorem to compute

$$(g^\beta, \ldots, g^{\beta^\ell}) \tag{1}$$

without knowing α or β! With this, the challenger can define its own new base $h(\beta)$ to use, instead of g in the VRF. This step requires that the input domain $\{1, \ldots, \ell\}$ be at most the number of terms in $\{(g^{\alpha^i})\}$. To

see how to compute h, note that if $f(z) = \prod(z + w)$ for all $w \neq x_0$ is expressed as a sum of powers of z, then $\mathcal{B}$ can compute $h = g^{f(\beta)}$ using (1) and the coefficients computed above.

Now, $\mathcal{B}$ can answer queries on all inputs by $\mathcal{A}$ except x_0. Doing this is fairly similar to what was just done. On input x instead of $f(z)$ the challenger must compute $f'(z) = f(z)/(z + x)$ once again as sum of powers of z and can then reply to query x using these coefficients with (1) appropriately.

Finally, when the adversary gives us a challenge query x^* if $x^* \neq x_0$ $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ returns Γ. A little bit of algebraic manipulation suffices to show that if Γ were $\hat{e}(g, g)^{1/\alpha}$ then $\mathcal{B}$'s reply looks like the VRF reply, else it looks like a random reply. Thus, whatever bit the adversary $\mathcal{A}$ outputs (as a distinguisher), $\mathcal{B}$ does the same. If $\mathcal{B}$ does not abort, its advantage is ε'. Choosing a random x_0 ensures that $\mathcal{B}$ does not abort with at least $2^{-\ell}$ probability. Thus, $\mathcal{B}$'s advantage over its random coins is $2^{-\ell}\varepsilon'$.

∎

4 Large domain constructions

Although we do not go into details of the parameters in the constructions in Section 3, we note that both the Dod03 and the DY05 constructions only work when the input domain X is small. In the former construction, efficient codes C that are 4-wise independent require that $L \geq 2\ell$ and for sum-free DDH to be hard, L must be small. In the latter construction, $|X| = \ell$ and directly relates to the hardness assumption. As seen in Theorem 3.1, ℓ must be at most $\log \lambda$, the security parameter for the result to be meaningful.

In this section, we see how to process more than one "block" of input at a time without a significant degradation in the parameter of the security assumption. This allows us to increase $|X|$ at a small expense in security. We informally note that the pairing function allows us to "chain proofs" on a single block in a secure way. When processing more than one block, this chaining of proofs is necessary to bind $F(\mathsf{sk}, x)$ to the *entire* input rather than a single block.

4.1 HW10 construction

Hohenberger and Waters [HW10] construct the first large-domain VRFs. They require the ℓ-decisional Diffie-Hellman Exponent problem to be hard. The problem informally asks:

$$\text{given } (g, h, g^x, \ldots, g^{x^{\ell-1}}, g^{\ell+1}, \ldots, g^{2\ell}) \text{ distinguish } \hat{e}(g, h)^{x^\ell} \text{ from random.}$$

In other words, given 2ℓ exponents of a secret x with a "gap" at the ℓ^{th} exponent, distinguish a pairing of this gap and a $h \in \mathbb{G}$ from a random $\mathbb{G}_{\mathbf{T}}$ element. In this scheme, $X = \{0,1\}^n$, $K = \mathbb{Z}_p^{n+1}$, and $Y = \mathbb{G}_{\mathbf{T}}$.

1. $\mathsf{Gen}(1^\lambda)$: Choose $(\mathbb{G}, p, g, h)$ a group with a bilinear map and random generators g, h of order p. Next, choose random values $u_0, \ldots, u_n \in \mathbb{Z}_p$ and set $U_i = g^{u_i}$. Output $\mathsf{pk} = (\mathbb{G}, p, g, h, \{U_i\})$ and $\mathsf{sk} = (\mathsf{pk}, \{u_i\})$.

2. $\mathsf{Prove}(\mathsf{sk}, x)$: On input $x = (x_1 x_2 \cdots x_n)$, for $i = 1, \ldots, n$ compute $\pi_i = g^{\prod_{j=1}^{i} u_j^{x_j}}$. Then, compute $\pi_0 = g^{u_0 \prod_{j=1}^{n} x_j^{u_j}}$. Output
$$(\hat{e}(\pi_0, h), \pi_0, \pi_1, \ldots, \pi_n).$$
Thus, $F(\mathsf{sk}, x) = \hat{e}(\pi_0, h)$ and $\pi(\mathsf{sk}, x) = (\pi_0, \ldots, \pi_n)$.

3. $\mathsf{Ver}(\mathsf{pk}, x, y, \pi)$: First check that $\hat{e}(\pi_1, g) = e(g, g)$ if $x_1 = 0$ or $e(U_1, g)$ otherwise. Then, for $i = 2, \ldots, n$, check that $\hat{e}(\pi_i, g) = \hat{e}(\pi_{i-1}, g)$ if $x_i = 0$ or $\hat{e}(\pi_{i-1}, U_i)$ otherwise. Finally check that $\hat{e}(\pi_0, g) = \hat{e}(\pi_n, U_0)$ and $\hat{e}(\pi_0, h) = y$. Return 1 iff all checks are successful; 0 otherwise.

The algebraic nature of $F(\mathsf{sk}, x)$ allows us to efficiently compute the VRF by first computing the exponent and then doing only a single costly exponentiation and pairing operation independent of the parameter n.

Also, we can see that in processing more than one block of input (more than one bit) we successively verify π_i using π_{i-1}. The proof of security is rather involved, but requires $\ell = 4Q(n+1)$ against a Q-query adversary. We note that ℓ-DBDHE is a (possibly strictly) weaker assumption (the scheme enjoys stronger security) than 2ℓ sum-free DDH.

4.2 BMR10 construction

Boneh, Montgomery, and Raghunathan in [BMR10] (in the context of a generic framework to extend PRFs) constructed an efficient VRF secure under the ζ-DBDHI assumption (explained before in Section 3.2). Although the DBDHI assumption is a (possibly strictly) stronger assumption than DBDHE, the scheme constructed below is secure for a tighter parameter $\zeta = n\ell$ for constant n. Unlike [HW10], this is independent of Q. As in [Dod03], the number of blocks n can be extended to be arbitrary using an appropriately chosen code. The resulting scheme requires $\zeta = 8n\ell^2$ for n-bit inputs (as in [HW10]). We describe the simpler scheme. Here, $X = \{1, \ldots, \ell\}^n, K = (\mathbb{G} \times \mathbb{Z}_p^n),$ and $Y = \mathbb{G}_{\mathbf{T}}$.

1. $\mathsf{Gen}(1^\lambda)$: Choose $(\mathbb{G}, p, g, u)$ a group with a bilinear map and random generators g, u of order p. Next, choose $s_1, \ldots, s_n \xleftarrow{R} \mathbb{Z}_p$ and set $t_i = g^{s_i}$. Set $\mathsf{pk} = (\mathbb{G}, p, g, u, \{t_i\})$ and $\mathsf{sk} = (\mathsf{pk}, s_1, \ldots, s_n)$.

2. $\mathsf{Prove}(\mathsf{sk}, x)$: On input $x = (x_1 x_2 \cdots x_n)$, for $i = 1, \ldots, n$ compute $\pi_i = g^{1/\prod_{j=1}^{i}(x_j + s_j)}$. Then, compute $F(\mathsf{sk}, x) = \hat{e}(\pi_n, u)$. Output

$$(\hat{e}(\pi_n, u), \pi_1, \ldots, \pi_n).$$

   Thus, $\pi(\mathsf{sk}, x) = (\pi_0, \ldots, \pi_n)$.

3. $\mathsf{Ver}(\mathsf{pk}, x, y, \pi)$: Set $\pi_0 := g$. For $i = 1, \ldots, n$ check that $\hat{e}(\pi, g^{x_i} \cdot t_i) = \hat{e}(\pi_{i-1}, g)$. Finally check that $\hat{e}(\pi_n, u) = y$. Return 1 iff all checks are successful; 0 otherwise.

5 Summary

In this short paper, we define and motivate the need for constructing verifiable random functions (VRFs). First we show a close relation to signature schemes and cite a few construction of signature schemes from bilinear groups. Next, we observe that groups over which DDH is easy, but other decision problems are hard serve as very useful building blocks for VRFs. Indeed, as seen in this paper, elliptic curve groups equipped with a bilinear map are spectacularly successful in constructing VRFs. In the paper, we give four concrete instantiations that include two small-domain [Dod03, DY05] and two large-domain [HW10, BMR10] VRFs, but note that there are several other constructions [Lys02, CL07]. We also give a proof outline of the VRF constructed in [DY05].

We conclude by saying that pairings have shown to be an extremely useful tool in cryptography. One of the places where the stand out is in constructing verifiable random functions which in turn are used in several varied applications as mentioned earlier in this paper. In [BMR10], the authors give a generic construction of PRFs (without the verification property) that convert PRFs with small input to PRFs that have much larger inputs. This technique requires that the underlying PRFs satisfy some minimal constraints. Extending this to VRFs will be a useful result.

References

[ACF09] Michel Abdalla, Dario Catalano, and Dario Fiore. Verifiable random functions from identity-based key encapsulation. In *EUROCRYPT'09*, pages 554–571, 2009.

[BB04] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In *Advances in Cryptology—EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 56–73. Springer-Verlag, 2004.

[BB08] Dan Boneh and Xavier Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *Journal of Cryptology*, 21(2):149–77, 2008.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *Advances in Cryptology—CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, 2004.

[BF01] Dan Boneh and Matt Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *Advances in Cryptology—CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–29. Springer-Verlag, 2001.

[BLS04] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. *J. Cryptology*, 17(4):297–319, 2004.

[BMR10] Dan Boneh, Hart William Montgomery, and Ananth Raghunathan. Algebraic pseudorandom functions with improved efficiency from the augmented cascade. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM Conference on Computer and Communications Security*, pages 131–140. ACM, 2010.

[CL07] Melissa Chase and Anna Lysyanskaya. Simulatable VRFs with applications to multi-theorem NIZK. In *CRYPTO'07*, pages 303–322, 2007.

[Dod03] Yevgeniy Dodis. Efficient construction of (distributed) verifiable random functions. In *Public Key Cryptography*, pages 1–17, 2003.

[DY05] Yevgeniy Dodis and Aleksandr Yampolskiy. A verifiable random function with short proofs and keys. In *Public Key Cryptography*, pages 416–431, 2005.

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In David S. Johnson, editor, *STOC*, pages 25–32. ACM, 1989.

[HW10] Susan Hohenberger and Brent Waters. Constructing verifiable random functions with large input spaces. In *Eurocrypto 2010*, 2010.

[Lys02] Anna Lysyanskaya. Unique signatures and verifiable random functions from the DH-DDH separation. In *Advances in Cryptology—CRYPTO 2002*, LNCS. Springer-Verlag, 2002.

[Mil04] Victor Miller. The Weil pairing, and its efficient calculation. *Journal of Cryptology*, 17(4), 2004.

[MRV99] Silvio Micali, Michael O. Rabin, and Salil P. Vadhan. Verifiable random functions. In *FOCS*, pages 120–130, 1999.

[NR97] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *FOCS'97*, pages 458–67, 1997.