A Verifiable Random Function With Short Proofs and Keys

Yevgeniy Dodis[1] and Aleksandr Yampolskiy[2]

[1] Department of Computer Science, New York University, USA. dodis@cs.nyu.edu
[2] Department of Computer Science, Yale University, USA.
aleksandr.yampolsky@yale.edu

Abstract. We give a simple and efficient construction of a verifiable random function (VRF) on groups equipped with a bilinear mapping. Our construction is direct; it bypasses an expensive Goldreich-Levin transformation from a unique signature to a VRF in contrast to prior works of Micali-Rabin-Vadhan [MRV99] and Lysyanskaya [Lys02]. Our proofs of security are based on a decisional bilinear Diffie-Hellman inversion assumption (DBDHI), previously used in [BB04a] to construct an identity based encryption scheme. Our VRF's proofs and keys have constant size in contrast to proofs and keys of VRFs in [Lys02] and [Dod03], which are linear in the size of the message. We operate over an elliptic group, which is significantly shorter than the multiplicative group $\mathbb{Z}_n^*$ used in [MRV99], yet we achieve the same security. Furthermore, our scheme can be made distributed and proactive.

Key words. Verifiable random functions, Unique signatures, Short keys and proofs, Bilinear groups.

1 Introduction

The notion of a **verifiable random function** (VRF) was introduced by Micali, Rabin, and Vadhan [MRV99]. A VRF is a pseudo-random function that provides a non-interactively verifiable proof for the correctness of its output. Given an input value x, the knowledge of the secret key SK allows one to compute $y = F_{SK}(x)$ together with the proof of correctness π_x. This proof convinces every verifier that the value $y = F_{SK}(x)$ is indeed correct with respect to the public key of the VRF. We can view VRFs as a commitment to an exponential number of random-looking bits.

Since their introduction, VRFs have found useful applications in protocol design. To give a few examples, VRFs were used in [MR01] to construct a one-time sound RZK (resettable zero-knowledge) protocol in three rounds [MR01]. In [MR02], VRFs were used to construct a non-interactive lottery system employed in micropayments [MR02]. Recently, Jarecki and Shmatikov constructed a verifiable transaction escrow scheme, which preserves users' anonymity while enabling automatic de-escrow, again with the help of VRFs [JS04].

Unfortunately, despite their utility, VRFs are not very well studied. As of this moment, there exist only a handful of constructions in the standard model (*viz.*, [MRV99, Lys02, Dod03]). With the exception of [Dod03], these works first construct a **verifiable unpredictable function** (VUF) (also called a **unique signature**), whose output is hard to predict but does not necessarily look random. They use an expensive Goldreich-Levin hardcore bit [GL89] to convert a VUF into a VRF; this loses a factor in security. The size of proofs and keys of VRFs in [Lys02, Dod03] is linear in input size, which may be undesirable in resource-constrained environments. VRF of [MRV99] operates over a large multiplicative group $\mathbb{Z}_n^*$ which has to be very large to achieve reasonable security. Furthermore, before the VRF can be computed, inputs have to be mapped to primes in a complicated fashion.

In this paper, we construct a simple VRF on groups equipped with bilinear maps. These groups, recently discovered by Joux and Nguyen [JN01], have the property that decisional Diffie-Hellman (DDH) assumption (given g, g^a, and g^b, distinguish g^{ab} from random) is easy, but computational Diffie-Hellman (CDH) assumption (given g, g^a, and g^b, compute g^{ab}) seems to be hard. This fact gives us many useful properties like verifiability. Our construction is direct; it does not use a Goldreich-Levin hardcore bit, saving several factors in security. The inputs need not be primes or codewords of some special encoding. Our VRF has proofs and keys of constant size, regardless of the size of the message. It can also be made distributed and proactive.

We begin in Section 2 by formalizing the notions of a **verifiable random function** (VRF) and a **verifiable unpredictable function** (VUF). We also review the definition of bilinear groups.

Our proofs of security rely on two assumptions, which we describe in Section 3. Roughly, they are:

- q**-Diffie-Hellman inversion assumption** (q-DHI) states that no efficient algorithm can compute $g^{1/x}$ on input $\left(g, g^x, \ldots, g^{(x^q)}\right)$ [MSK02];
- q**-decisional Diffie-Hellman inversion assumption** (q-DBDHI) states that no efficient algorithm can distinguish $e(g,g)^{1/x}$ from random even after seeing $\left(g, g^x, \ldots, g^{(x^q)}\right)$ [BB04a].

(Here $e(\cdot,\cdot)$ is a bilinear map, which we define later.)

In Section 4, we give our constructions and analyze their efficiency.

First, in Section 4.1, we show how to turn a signature due to Boneh and Boyen [BB04b] into a VUF for messages of small length. On input x and a secret key SK, the signature is $\textsc{Sign}_{SK}(x) = g^{1/(x+SK)}$. We could then use the approach of prior works [MRV99, Lys02] to convert this VUF into a VRF Specifically, we could use the Goldreich-Levin hardcore bit [GL89] to convert this VUF into a VRF with output size 1, amplify the output size to match the size of the input, and then follow a tree-based construction to get a VRF with arbitrary input size.

Instead, we prefer to construct a VRF directly in Section 4.2, saving several fators in security. We give a simple direct VRF construction for small inputs, which is secure under the q-DBDHI assumption. On input x and a secret key SK, our VRF computes $(F_{SK}(x), \pi(x))$, where $F_{SK}(x) = e(g,g)^{1/(x+SK)}$ is the VRF value and $\pi(x) = g^{1/(x+SK)}$ is the proof of correctness. We can apply a collision-resistant hash function to large inputs to transform our VRF into a VRF with unrestricted input length. By making the group size sufficiently large, we can construct a VRF with inputs of size roughly 160 bits, which is the length of SHA-1 digests. In theory, we do not have to assume existence of collision-resistant functions, and could also apply a variant of a generic tree transformation to amplify the input size. Even though keys and proofs no longer have constant size, the resulting VRF still has shorter keys and proof sizes than in [MRV99, Lys02] constructions. We analyze how large the group has to be and how our VRF compares with other constructions in Section 4.4.

Evaluating the VRF at a single server is a performance bottleneck and a single point of failure. Naturally, in Section 5, we sketch how to make our VRF distributed and proactive [Dod03].

In Section 6, we analyze the q-DBDHI assumption in the generic group model à la Shoup [Sho97]. We show that if the adversary can distinguish $e(g,g)^{1/x}$ from random with probability $\frac{1}{2} + \epsilon$, he will need to perform (at least) $\Omega(\sqrt{\epsilon p/q})$ generic group operations in a group of size p.

Conclusions and suggestions for future work appear in Section 7.

2 Definitions

Before presenting our results, we review some basic definitions and assumptions.

Let k be a security parameter. We employ the standard cryptographic model in which protocol participants are modeled by probabilistic Turing machines (PPTMs), whose running time is polynomial in k. Hereafter, we use $\text{negl}(k)$ to refer to a negligible function in the security parameter k.[3]

2.1 VRFs and VUFs

Let $a : \mathbb{N} \mapsto \mathbb{N} \cup \{*\}$ and $b, s : \mathbb{N} \mapsto \mathbb{N}$ be any functions for which $a(k), b(k), s(k)$ are computable in $\text{poly}(k)$ time (except when a takes tha value $*$).[4]

Definition 1. *A function family $F_{(\cdot)}(\cdot) : \{0,1\}^{a(k)} \mapsto \{0,1\}^{b(k)}$ is a family of VRFs if there exist algorithms $(\textsc{Gen}, \textsc{Prove}, \textsc{Ver})$ such that $\textsc{Gen}(1^k)$ outputs a pair of keys (PK, SK); $\textsc{Prove}_{SK}(x)$ outputs a pair $\big(F_{SK}(x), \pi_{SK}(x)\big)$, where $F_{SK}(x)$ is the function value and $\pi_{SK}(x)$ is the proof of correctness; and $\textsc{Ver}_{PK}(x, y, \pi)$ verifies that $y = F_{SK}(x)$ using the proof π. Formally, we require:*

1. ***Uniqueness:*** *no values $(PK, x, y_1, y_2, \pi_1, \pi_2)$ can satisfy $\textsc{Ver}_{PK}(x, y_1, \pi_1) = \textsc{Ver}_{PK}(x, y_2, \pi_2)$.*

[3] A function $\text{negl}(k) : \mathbb{N} \mapsto (0,1)$ is **negligible** if for every $c > 0$, for all sufficiently large k, $\text{negl}(k) < 1/k^c$. See any standard reference, such as [GB99], for details.

[4] When $a(k)$ takes the value of $*$, it means the VRF is defined for inputs of all length.

2. ***Provability:*** *if* $(y, \pi) = \textsc{Prove}_{SK}(x)$, *then* $\textsc{Ver}_{PK}(x, y, \pi) = 1$.
3. ***Pseudorandomness:*** *for any PPT algorithm* $A = (A_E, A_J)$, *which runs for a total of* $s(k)$ *steps when its first input is* 1^k, *and does not query the oracle on* x,

$$\Pr\left[\ b = b' \left| \begin{array}{c} (PK, SK) \leftarrow \textsc{Gen}(1^k); (x, st) \leftarrow A_E^{\textsc{Prove}(\cdot)}(PK); \\ y_0 = F_{SK}(x); y_1 \leftarrow \{0,1\}^{b(k)}; \\ b \leftarrow \{0,1\}; b' \leftarrow A_J^{\textsc{Prove}(\cdot)}(y_b, st) \end{array} \right. \right] \leq \frac{1}{2} + negl(k)$$

Intuitively, the definition says that no value of the function can be distinguished from random, even after seeing any other function values together with corresponding proofs.

A close relative of a VRF is a verifiable unpredictable function (VUF). It can be thought of as a signature scheme in which a unique (at most one) signature is accepted by the verification algorithm for every message and public key.

Definition 2. *A function family* $F_{(\cdot)}(\cdot) : \{0,1\}^{a(k)} \mapsto \{0,1\}^{b(k)}$ *is a family of VUFs, if there exist algorithms* $(\textsc{Gen}, \textsc{Sign}, \textsc{Ver})$ *which satisfy the uniqueness and provability properties of Definition 1, but the pseudorandomness property is replaced by a weaker property:*

3' . ***Unpredictability:*** *for any PPT algorithm* A, *which runs for a total of* $s(k)$ *steps, and does not query the oracle on* x,

$$\Pr\left[y = F_{SK}(x) \middle| (PK, SK) \leftarrow \textsc{Gen}(1^k); (x, y) \leftarrow A^{\textsc{Sign}(\cdot)}(PK) \right] \leq negl(k)$$

For exact security bounds, we will occasionally say that $F_{(\cdot)}(\cdot)$ is an $(s'(k), \epsilon'(k))$ secure VRF (*resp.*, VUF) if no adversary $\mathcal{A}$ running in time $s'(k)$ (and therefore making at most $s'(k)$ oracle queries) can break the pseudorandomness (*resp.*, unpredictability) property with $\epsilon'(k)$ advantage.

2.2 Bilinear groups

We shall use bilinear maps to construct a verification algorithm for our VRF. We briefly review their properties below (see [DBS04] for a more comprehensive treatment).

Consider two (multiplicative) cyclic groups $\mathbb{G}$ and $\mathbb{G}_1$ of prime order p. Let g be a generator of $\mathbb{G}$. Roughly speaking, a mapping is **bilinear** if it is linear with respect to each of its variables:

Definition 3. *An (admissible) bilinear map is a map* $e : \mathbb{G} \times \mathbb{G} \mapsto \mathbb{G}_1$ *with the following properties:*

1. **Bilinear:** *for all* $u, v \in \mathbb{G}$ *and* $x, y \in \mathbb{Z}$, *we have* $e(u^x, v^y) = e(u, v)^{xy}$.
2. **Non-degenerate:** $e(g, g) \neq 1$.
3. **Computable:** *there is an efficient algorithm to compute* $e(u, v)$ *for all* $u, v \in \mathbb{G}$.

We say that a group $\mathbb{G}$ is bilinear if the group action in $\mathbb{G}$ is efficiently computable and there exists a group $\mathbb{G}_1$ and an admissible bilinear map $e : \mathbb{G} \times \mathbb{G} \mapsto \mathbb{G}_1$. Henceforth, we shall use $\mathbb{G}^*$ to stand for $\mathbb{G} \backslash \{1_{\mathbb{G}}\}$.

Note that an admissible bilinear map provides an algorithm for solving the decisional Diffie-Hellman problem (DDH) in $\mathbb{G}$. Specifically, to determine whether (g, g^x, g^y, g^z) is a DDH tuple, we can check if $e(g^x, g^y) = e(g, g^z)$. Such maps can be constructed from Weil and Tate pairings on elliptic curves or abelian varieties [BF01, JN01, Gal01].

3 Complexity assumptions

We now state the hardness assumptions on which our constructions are based. In what follows, we let $\mathbb{G}$ be a bilinear group of prime order p, and let g be its generator.

3.1 Diffie-Hellman inversion assumption

Our VUF construction relies on the Diffie-Hellman inversion (DHI) assumption, originally proposed in [MSK02]. The q-DHI problem on $\mathbb{G}$ asks: given the $(q+1)$-tuple $\left(g, g^x, \ldots, g^{(x^q)}\right) \in (\mathbb{G}^*)^{q+1}$ as input, to compute $g^{1/x}$. Formally, an algorithm $\mathcal{A}$ has advantage ϵ in solving q-DHI in $\mathbb{G}$ if

$$\Pr\left[\mathcal{A}(g, g^x, \ldots, g^{(x^q)}) = g^{1/x}\right] \geq \epsilon,$$

where probability is taken over the coin tosses of $\mathcal{A}$ and the random choice of $x \in \mathbb{Z}_p^*$.

Definition 4. *(q-DHI assumption) We say that (t, q, ϵ)-DHI assumption holds in $\mathbb{G}$ if, no t-time algorithm $\mathcal{A}$ has advantage at least ϵ in solving the q-DHI problem in $\mathbb{G}$.*

Boneh and Boyen [BB04a] pointed out that the q-DHI assumption implies the $(q+1)$-generalized Diffie-Hellman assumption (GDH), on which many cryptographic constructions are based (*e.g.*, [NR97,BS02,STW96] as well as the VUF by Lysyanskaya [Lys02]).

3.2 Decisional bilinear Diffie-Hellman inversion assumption

In order to construct a VRF directly, we need to make a decisional bilinear Diffie-Hellman inversion assumption (DBDHI). It was previously used in [BB04a] to construct a selective-ID secure identity based encryption scheme.

The q-DBDHI problem asks: given the tuple $\left(g, g^x, \ldots, g^{(x^q)}\right)$ as input, to distinguish $e(g, g)^{1/x}$ from random. Formally, an algorithm $\mathcal{A}$ has advantage ϵ in solving the q-DBDHI problem if

$$\left| \Pr\left[\mathcal{A}(g, g^x, \ldots, g^{(x^q)}, e(g,g)^{1/x}) = 1\right] - \Pr\left[\mathcal{A}(g, g^x, \ldots, g^{(x^q)}, \mathit{\Gamma}) = 1\right]\right| \leq \epsilon,$$

where the probability is over the internal coin tosses of $\mathcal{A}$ and the choice of $x \in \mathbb{Z}_p^*$ and $\mathit{\Gamma} \in \mathbb{G}_1$.

Definition 5. *(q-DBDHI assumption) We say that the (t, q, ϵ)-DBDHI assumption holds in $\mathbb{G}$ if no t-time algorithm $\mathcal{A}$ has advantage at least ϵ in solving the q-DBDHI problem in $\mathbb{G}$.*

Clearly, q-DBDHI is a stronger assumption than q-DHI. To provide more confidence in its validity, we analyze this assumption in the generic group model in Section 6.

4 Our constructions

In Section 4.1, we begin by showing how a signature scheme due to Boneh and Boyen [BB04b] can be turned into a VUF with small inputs. We forego an expensive transformation from VUFs to VRFs using a Goldreich-Levin hardcore bit. Instead, we construct our VRF directly in Section 4.2 on inputs of small size. We show how input size can be extended in Section 4.3. We analyze our constructions' efficiency in Section 4.4.

Fix input length $a(k)$, output length $b(k)$, and security $s(k)$. For notational convenience, we will usually omit the security parameter k, writing, for example, a or s, instead of $a(k)$ or $s(k)$. Let $\mathbb{G}$ $\;(|\mathbb{G}| = p)$ be a bilinear group, where p is a k bit prime. Let g be a generator of $\mathbb{G}$. Throughout, we shall assume that messages can be encoded as elements of $\mathbb{Z}_p^*$.

4.1 A verifiable unpredictable function

In order to build the intuition for our next proof, we show how to construct a simple VUF (Gen, Sign, Ver) with small (superlogarithmic) inputs.

$$h^{\frac{1}{x+\beta}}$$

- **Algorithm** Gen(1^k): Chooses a secret $s \in_r \mathbb{Z}_p^*$ and sets the secret key to $SK = s$ and a public key to $PK = g^s$.
- **Algorithm** Sign$_{SK}(x)$: Outputs the signature Sign$_{SK}(x) = g^{1/(x+SK)}$. Note that the proof is embedded in the output value so we do not need to include it explicitly.
- **Algorithm** Ver$_{PK}(x, y)$: Outputs 1 if $e(g^x \cdot PK, y) = e(g, g)$; otherwise, outputs 0. Indeed, if the VRF value y was correctly computed, we have:

$$e(g^x \cdot PK, y) = e(g^x g^s, g^{1/(x+s)}) = e(g, g).$$

Boneh and Boyen proved this signature scheme to be existentially unforgeable under weak chosen message attacks (*i.e.*, the adversary is *non-adaptive*). To do so, they used a q-SDH assumption which claims that on input $\left(g, g^x, \ldots, g^{(x^q)}\right)$, no efficient algorithm can compute $\left(c, g^{1/(x+c)}\right)$ for any $c \in \mathbb{Z}_p^*$. In our proof, we use a weaker q-DHI assumption (which implies the q-SDH assumption) to prove the scheme's security against *adaptive* adversaries. We use a trick of Micali-Rabin-Vadhan [MRV99], which restricts messages to be of slightly superlogarithmic size (in security parameter k). This enables us to enumerate all possible messages and to respond to adversary's queries adaptively.

Theorem 1. *Suppose the* $(s(k), 2^{a(k)}, \epsilon(k))$*-DHI assumption holds in a bilinear group* $\mathbb{G}$ ($|\mathbb{G}| = p$)*. Let the input size be* $a(k)$ *and output size be* $b(k) = \log_2 p$*. Then* (Gen, Sign, Ver) *is a* $(s'(k), \epsilon'(k))$ *unique signature, where* $s'(k) = s(k)/(2^{a(k)} poly(k))$ *and* $\epsilon'(k) = \epsilon(k) \cdot 2^{a(k)}$.

Proof. It is easy to see that uniqueness and provability properties of Definition 2 are satisfied. We thus concentrate on residual unpredictability.

We shall use a shortcut and write $q = 2^{a(k)}$. Suppose there exists an adversary $\mathcal{A}$, running in time $s'(k)$, which guesses the value of the function at an unseen point with non-negligible probability $\epsilon'(k)$. We shall construct an algorithm $\mathcal{B}$ that by interacting with $\mathcal{A}$ breaks the q-DHI assumption with non-negligible probability.

Input to the reduction: Algorithm $\mathcal{B}$ is given a tuple $\left(g, g^\alpha, \ldots, g^{(\alpha^q)}\right) \in (\mathbb{G}^*)^{q+1}$, for some unknown $\alpha \in \mathbb{Z}_p^*$. Its goal is to compute $g^{1/\alpha}$.

Key generation: We guess that $\mathcal{A}$ will output a forgery on message $x_0 \in_r \{0,1\}^{a(k)}$. We are right with probability $1/2^{a(k)}$; error probability can be decreased by repeating the algorithm sufficiently many times. Let $\beta = \alpha - x_0$.[5] We don't know what β is because α is secret. However, we can use the Binomial Theorem to compute $\left(g^\beta, \ldots, g^{(\beta^q)}\right)$ from $\left(g^\alpha, \ldots, g^{(\alpha^q)}\right)$. Because $a(k) = \log(s(k))$, we can enumerate all possible inputs in $s(k)$ time. Let $f(z)$ be the polynomial

Directly assign $X_0 = 0$

$$f(z) = \prod_{w \in \{0,1\}^a, w \neq x_0} (z + w) = \sum_{j=0}^{q-1} c_j z^j \text{ (for some coefficients } c_0, \ldots, c_{q-1}).$$

We can compute

$$h = g^{f(\beta)} = \prod_{j=0}^{q-1} \left(g^{(\beta^j)}\right)^{c_j} \text{ and } h^\beta = \prod_{j=1}^{q} \left(g^{(\beta_j)}\right)^{c_{j-1}}.$$

Finally, we set h to be the generator and give $PK = h^\beta$ to $\mathcal{A}$. The secret key is $SK = \beta$, which we don't know ourselves.

[5] We should really have written $\beta = \alpha - \psi(x_0)$, where $\psi : \{0,1\}^a \mapsto \mathbb{Z}_p^*$. For sake of readability, we shall abuse the notation.

Responding to oracle queries: Without loss of generality, we assume that $\mathcal{A}$ never repeats a query. Consider the ith query $(1 \leq i < q)$ on message x_i. If $x_i = x_0$, then we fail. Otherwise, we must compute $\textsc{Sign}_{SK}(x_i) = h^{1/(x_i+\beta)}$. Let $f_i(z)$ be the polynomial

$$f_i(z) = f(z)/(z + x_i) = \sum_{j=0}^{q-2} d_j z^j \text{ (for some coefficients } d_0, \dots, d_{q-2}).$$

We can compute

$$g^{f_i(z)} = \prod_{j=0}^{q-2} \left(g^{(\beta^j)}\right)^{d_j} = h^{1/(x_i+\beta)}$$

and return it as the signature.

Outputting the forgery: Eventually, $\mathcal{A}$ outputs a forgery (x^*, σ^*). If $x^* \neq x_0$, then our simulation failed. Because the signature is unique, we must have $\sigma^* = h^{1/(x_0+\beta)} = g^{f(\beta)/(x_0+\beta)}$. Compute

If $x_0 = 0$. then

$$f(z)/(z + x_0) = \sum_{j=0}^{q-2} \gamma_j z^j + \frac{\gamma_{-1}}{z + x_0},$$

$\sigma^* = h^{\frac{1}{(x_0+\beta)}} = h^{\frac{1}{\beta}} = h^{\frac{1}{\alpha}}$

where $\gamma_{-1} \neq 0$. Hence,

$$\left(\sigma^* \cdot \prod_{j=0}^{q-2} \left(g^{(\beta^i)}\right)^{-\gamma_i}\right)^{1/\gamma_{-1}} = g^{1/(x_0+\beta)} = g^{1/\alpha}.$$

Let $\epsilon'(k) = \epsilon(k) \cdot 2^{a(k)}$ and $s'(k) = s(k)/(2^{a(k)} \cdot \text{poly}(k))$. To finish the proof, note that algorithm $\mathcal{B}$ succeeds with probability $\epsilon'(k)/2^{a(k)} = \epsilon(k)$. Its running time is dominated by answering oracle queries, and each query takes $(2^{a(k)} - 2) \cdot \text{poly}(k)$ time to answer. Therefore, $\mathcal{B}$ will run in roughly $s'(k) \cdot 2^{a(k)}\text{poly}(k) = s(k)$ time.

□

Remark 1. The security reduction of Theorem 1 is not tight. It allows to construct VUFs with input roughly $a(k) = \Omega(\log s(k))$. In theory, this means that the input size we can achieve might be only slightly superlogarithmic in k (similar to [MRV99]). First, it might be reasonable to assume subexponential hardness of the q-DHI assumption which will immediately allow one to support input of size $k^{\Omega(1)}$. Also, by utilizing a collision-resistant hash function, we will anyway only need to construct VUFs with relatively small input size such as 160 bits. Indeed, in Section 4.4, we show that our construction seems to yield practical and secure VUF for inputs of arbitrary length already when $k = 1000$ bits.

4.2 A verifiable random function

Our main contribution is a direct construction of a verifiable random function from a slightly stronger q-DBDHI assumption. The VRF $(\textsc{Gen}, \textsc{Prove}, \textsc{Ver})$ is as follows.

Algorithm $\textsc{Gen}(1^k)$: Selects a random $s \in \mathbb{G}$ and sets the secret key to $SK = s$ and the public key to $PK = g^s$.

Algorithm $\textsc{Prove}_{SK}(x)$: We let $\textsc{Prove}_{SK}(x) = \big(F_{SK}(x), \pi_{SK}(x)\big)$ where $F_{SK}(x) = e(g,g)^{1/(x+SK)}$ is the VRF output and $\pi_{SK}(x) = g^{1/(x+SK)}$ is the proof of correctness.

Algorithm $\textsc{Ver}_{PK}(x, y, \pi)$: To verify whether y was computed correctly, check if $e(g^x \cdot PK, \pi) = e(g,g)$ and whether $y = e(g, \pi)$. If both checks succeed, output 1; otherwise, output 0.

We can prove this scheme to be secure (in the sense of Definition 1) for small inputs (superlogarithmic in k). We then show how to convert it into a VRF with unrestricted input size.

Theorem 2. *Suppose the $(s(k), 2^{a(k)}, \epsilon(k))$-decisional BDHI assumption holds in a bilinear group $\mathbb{G}$ ($|\mathbb{G}| = p$). Let the input size be $a(k)$ and output size be $b(k) = \log_2 p$. Then* $(\textsc{Gen}, \textsc{Prove}, \textsc{Ver})$*, as defined above, is a $(s'(k), \epsilon'(k))$ verifiable random function with $s'(k) = s(k)/(2^{a(k)} \cdot poly(k))$ and $\epsilon'(k) = \epsilon(k) \cdot 2^{a(k)}$.*

Proof. It is trivial to show that uniqueness and provability properties of Definition 1 are satisfied. We thus concentrate on the pseudorandomness property.

We shall use $q = 2^{a(k)}$ as a shortcut. For sake of contradiction, suppose there exists an algorithm $\mathcal{A} = (A_E, A_J)$, which runs in time $s'(k)$, and can distinguish between $F_{SK}(x) = e(g,g)^{1/(x+s)}$ (for some x) and a random element in $\mathbb{G}_1$ with probability at least $1/2 + \epsilon'(k)$. We shall construct an algorithm $\mathcal{B}$ that uses $\mathcal{A}$ to break the q-DBDHI assumption in $\mathbb{G}$.

Input to the reduction: Algorithm $\mathcal{B}$ is given a tuple $(g, g^{\alpha}, \ldots, g^{(\alpha^q)}, \mathit{\Gamma}) \in (\mathbb{G}^*)^{q+1} \times \mathbb{G}_1$, where $\mathit{\Gamma}$ is either $e(g,g)^{1/\alpha} \in \mathbb{G}_1$ or a random element in $\mathbb{G}_1$. Its goal is to output 1 if $\mathit{\Gamma} = e(g,g)^{1/\alpha}$ and 0 otherwise.

Key generation: We guess that $\mathcal{A}$ will choose to distinguish the VRF value on message $x_0 \in \{0,1\}^{a(k)}$. Let $\beta = \alpha - x_0$ (see footnote 5). We generate the public and private keys for algorithm $\mathcal{A}$ as in the proof of Theorem 1. Using the Binomial Theorem, we compute the tuple $\big(g^{\beta}, \ldots, g^{(\beta^q)}\big)$. We define

$$f(z) = \prod_{w \in \{0,1\}^a, w \neq x_0} (z + w) = \sum_{j=0}^{q-1} c_j z^j.$$

This enables us to compute the new base

$$h = g^{f(\beta)} = \prod_{j=0}^{q-1} \left(g^{(\beta^j)}\right)^{c_j}.$$

Finally, we give $PK = h^{\beta} = \prod_{j=1}^{q} \left(g^{(\beta^j)}\right)^{c_{j-1}}$ as the public key to $\mathcal{A}$. The secret key is $SK = \beta$, which we don't know.

Remark 2. We can pre-compute $e(g, g)$ and $e(h, h)$ to speed up subsequent computation.

Responding to oracle queries: Consider the ith query ($1 \leq i < q$) on message x_i. If $x_i = x_0$, we fail. Otherwise, we must respond with the corresponding proof $\pi_{SK}(x_i)$ and a VRF value $F_{SK}(x_i)$.
As in Theorem 1, we define

$$f_i(z) = f(z)/(z + x_i) = \sum_{j=0}^{q-2} d_j z^j \text{ (for some coefficients } d_0, \ldots, d_{q-2}).$$

We can thus compute

$$\pi_{SK}(x_i) = \prod_{j=0}^{q-2} \left(g^{(\beta^j)}\right)^{d_j} = h^{1/(\beta + x_i)}$$

and

$$F_{SK}(x_i) = e(h, \pi_{SK}(x_i)) = e(h, h)^{1/(\beta + x_i)},$$

and return them to algorithm $\mathcal{A}$.

Challenge: Eventually, $\mathcal{A}$ outputs a message x^* on which it wants to be challenged. If $x^* \neq x_0$, then we fail. Otherwise, $\mathcal{A}$ claims to be able to distinguish $e(h, h)^{1/(\beta + x_0)} = e(h, h)^{1/\alpha}$ from a random element in $\mathbb{G}_1$. Recall that

$$f(z) = \sum_{i=0}^{q-1} c_i z^i.$$

Because $f(z)$ is not divisible by $(z + x_0)$, we have:

$$\begin{aligned} f'(z) &= f(z)/(z + x_0) - \frac{\gamma}{z + x_0} \\ &= \sum_{j=0}^{q-2} \gamma_j z^j \text{ (for some } \gamma \neq 0 \text{ and coefficients } \gamma_0, \ldots, \gamma_{q-2}). \end{aligned}$$

Let $\mathit{\Gamma}_0$ be

$$\begin{aligned} \mathit{\Gamma}_0 &= \left(\prod_{i=0}^{q-1} \prod_{j=0}^{q-2} e\left(g^{(\beta^i)}, g^{(\beta^j)}\right)^{c_i \gamma_j} \right) \cdot \left(\prod_{m=0}^{q-2} e\left(g, g^{(\beta^t)}\right)^{\gamma \cdot \gamma_m} \right) \\ &= e\left(g^{f(\beta)}, g^{f'(\beta)}\right) \cdot e\left(g^{\gamma}, g^{f'(\beta)}\right) \qquad (1) \\ &= e(g, g)^{(f(\beta)^2 - \gamma^2)/\alpha}. \end{aligned}$$

Set $\mathit{\Gamma}^* = \mathit{\Gamma}^{(\gamma^2)} \cdot \mathit{\Gamma}_0$. Notice that if $\mathit{\Gamma} = e(g, g)^{1/\alpha}$, then $\mathit{\Gamma}^* = e(g^{f(\beta)}, g^{f(\beta)/\alpha}) = e(h, h)^{1/\alpha}$. Meanwhile, if $\mathit{\Gamma}$ is uniformly distributed, then so is $\mathit{\Gamma}^*$. We give $\mathit{\Gamma}^*$ to algorithm $\mathcal{A}$.

Remark 3. We can see from Equation (1) that it takes only two bilinear map evaluations to compute $\varGamma_0$.

Guess: Algorithm $\mathcal{A}$ makes some more queries to which we respond as before. Finally, $\mathcal{A}$ outputs a guess $b \in \{0, 1\}$. We return b as our guess as well.

The running time of the reduction is dominated by simulating oracle queries. Per every query, we must perform one bilinear map evaluation (this takes $\mathrm{poly}(k)$ time) and $(2^a - 2)$ multiplications and exponentiations (this takes $2^a \cdot \mathrm{poly}(k)$ time). Because $\mathcal{A}$ can make at most $s'(k)$ queries, the running time of $\mathcal{B}$ is altogether $s'(k)(2^{a(k)} \cdot \mathrm{poly}(k))$. The advantage of $\mathcal{B}$ in this experiment is $\epsilon'(k)/2^{a(k)}$. Setting $s'(k) = s(k)/(2^{a(k)} \cdot \mathrm{poly}(k))$ and $\epsilon'(k) = \epsilon(k) \cdot 2^{a(k)}$ completes the proof. □

4.3 Extending the input size

Hashing the input. We constructed a VRF (Gen, Prove, Ver) with inputs of small size $a(k) = \varOmega(\log(k))$ and output size $b(k)$. We point out a trivial observation that if we have an unpredictable or verifiable random function function $\textsc{Prove}_{SK}(\cdot) : \{0,1\}^a \mapsto \{0,1\}^b$ and a collision-resistant hash function $H(\cdot) : \{0,1\}^c \mapsto \{0,1\}^a$ $(c > a)$, then their composition $\textsc{Prove}_{SK}(H(\cdot)) : \{0,1\}^c \mapsto \{0,1\}^b$ is secure. Although our security reduction is relatively loose, we can make the size of bilinear groups large enough (we give precise numbers in Section 4.4) to have inputs of length roughly $a(k) = 160$ bits, the length of SHA-1 digests. Restriction to small inputs is therefore not limiting because we can always hash longer inputs.

Tree construction. Although, we recommend using the previous construction (by making the group large enough), in theory, we could always use the generic tree construction (which is inefficient) to extend the input length. Then, we do not have to assume the existence of a collision-resistant hash function; having a universal hash function suffices:

Proposition 1 ([MRV99]). *If there is a VRF with input length $a(k)$, output length 1, and security $s(k)$, then there is a VRF with unrestricted input length, output length 1 and security at least* $\min(s(k)^{1/5}, 2^{a(k)/5})$.

The proof begins by converting a VRF with output length 1 into a VRF with output length $(a-1)$. This transformation loses a factor of a in security. Because our VRF has output length much larger than 1, we can omit this step. Instead, we apply a universal hash function to VRF's output and let the VRF's value be the first $(a-1)$ bits of hash function's output (it is easily seen that these bits will be pseudo-random as well).

The rest of the transformation proceeds as usual: The root of the binary tree is labeled with 0^{a-1} and children of node y are labeled with short VRF's values on inputs $(y \circ 0)$ and $(y \circ 1)$. The tree's depth is precisely the input size $|x|$. Computing the VRF value on input x (of unrestricted length) amounts to computing labels of nodes on the path from the root of the tree to the leaf corresponding to x. The proof of correctness is the tuple of short VRF's proofs, one proof per each node on the path traced by x.[6]

We note that both of the aforementioned techniques can be used to convert the VUF in Section 4.1 into a VUF with unrestricted input length.

4.4 Efficiency

We now compare efficiency of our construction with efficiency of prior VRF constructions. Fix inputs to be $a(k) = 160$ bits, the length of SHA-1 digests, and let $q = 2^{a(k)}$.

Our VRF. According to Theorem 2, if $(s(k), q, \epsilon(k))$-DBDHI holds on $\mathbb{G}$, then our VRF is secure against adversaries running in time $s'(k) = s(k)/(2^{a(k)} \cdot \mathrm{poly}(k))$ that have advantage $\epsilon'(k) = \epsilon(k) \cdot 2^{a(k)}$. To be generous, we instantiate $\epsilon'(k) = 2^{-80}$, $s'(k) = 2^{80}$, and $\mathrm{poly}(k) = 2^{30}$. Then, we have: $\epsilon(k) = 2^{-240}$ and $s(k) = 2^{270}$. Suppose there is no better algorithm for breaking the q-DBDHI assumption than a generic group algorithm. Then, by Theorem 3 (which we prove in Section 6), for these security parameters, a bilinear group must have size:

$$\begin{aligned} p &\geq \frac{2(s(k)+q+3)^2 q}{\epsilon(k)} \\ &= \frac{2\left(2^{270}+2^{160}+3\right)^2 2^{160}}{2^{-240}} \\ &\approx 2^{940}. \end{aligned}$$

Therefore, making the group size be a 1,000 bit prime seems sufficient to guarantee security of the VRF that takes 160 bit inputs. Proofs and keys consist of a single group element and will roughly be 125 bytes each. We can generate such groups using the standard parameter generator of [BF01].

VRF by [MRV99]. Micali-Rabin-Vadhan VRF operates over a multiplicative group $\mathbb{Z}_n^*$, where $n = pq$ is a k-bit RSA modulus. The fastest general-purpose factoring algorithm today is the number field sieve [BLZ94]; it takes approximately $O\left(e^{1.9223(k^{1/3}(\log k)^{2/3})}\right)$ time to factor a k bit number. The RSA based VUF (not even a VRF) constructed in [MRV99] has security $s'(k) = s(k)/(2^{a(k)} \cdot \mathrm{poly}(k))$ where $s(k)$ is hardness of RSA. Letting $s'(k) = 2^{80}$ and $\mathrm{poly}(k) = 2^{30}$ as before, we obtain RSA security lower bound $s(k) = 2^{80} \cdot (2^{160} \cdot 2^{30}) = 2^{270}$. Because RSA is only secure as long as we cannot factor n, to get 270 bits of security, we need n to be a k-bit number, where

$$1.9223k^{1/3}(\log k)^{2/3} = 270.$$

[6] The inputs have to be prefix-free for this tree construction to work; this can be accomplished using techniques of [MRV99].

Hence, n must be at least $14,383$ bits long if we want to use this VUF on 160 bit inputs. After following the tree construction, proofs for 160 bit inputs will have size 280 kilobytes.

VRF by [Dod03] and VUF by [Lys02]. These constructions work on elliptic curve groups, whose size is usually a 160 bit prime. At the bare minimum, 160 bit messages yield keys and proofs of size $160 \cdot 160 = 25,600$ bits, which is about 3.2 kilobytes. In fact, they will probably have larger size due to use of error-correcting codes and other encoding expansions.

To summarize, none of the prior VRF constructions come close to the 1,000 bit proofs and keys of our construction. Even if our VRF is used with the generic tree construction, the keys and proofs contain only one group element per bit of the input ($|x|$ elements altogether). This is less than $|x|$ group elements per bit of the input ($|x|^2$ elements altogether) contained by keys and proofs of a VRF in (Section 5, [Lys02]).

5 Distributed VRF

We point out that our VUF/VRF constructions can be easily made distributed (or even proactive). Indeed, both of the constructions simply amount to a secure computation of the function $\pi_s(x) = g^{1/(x+s)}$ when the servers have shares of the secret s. Since it is well known how to do multiparty addition, inversion, and exponentiation [BIB89, BoGW88], this extension follows immediately. We notice however that unlike the construction of Dodis [Dod03], our distributed VUF/VRF is interactive.

6 Generic Security of the q-DBDHI Assumption

In this section, we examine the q-DBDHI assumption in the generic group model [Sho97].

In the generic group model, elements of $\mathbb{G}$ and $\mathbb{G}_1$ are encoded as unique random strings. We define an injective function $\theta : \mathbb{Z}_p \mapsto \{0,1\}^*$, which maps $a \in \mathbb{Z}_p$ to the string representation $\theta(g^a)$ of $g^a \in \mathbb{G}$. Similarly, we define $\theta_1 : \mathbb{Z}_p \mapsto \{0,1\}^*$ for $\mathbb{G}_1$. The encodings are such that non-group operations are meaningless. There exist three oracles which compute the group action in $\mathbb{G}$, the group action in $\mathbb{G}_1$, and the bilinear pairing $e : \mathbb{G} \times \mathbb{G} \mapsto \mathbb{G}_1$ from elements' encodings.

Theorem 3. *Let $\mathcal{A}$ be an algorithm that solves the q-DBDHI problem. Suppose that $x \in \mathbb{Z}_p^*$ and encoding functions θ, θ_1 are chosen at random. If $\mathcal{A}$ makes at most q_G queries to oracles computing the group action in $\mathbb{G}, \mathbb{G}_1$ and the bilinear mapping $e : \mathbb{G} \times \mathbb{G} \mapsto \mathbb{G}_1$, then*

$$\left|\Pr\left[\begin{array}{c}\mathcal{A}\big(p,\theta(1),\theta(x),\ldots,\theta(x^q),\\ \theta_1(\varGamma_0),\theta_1(\varGamma_1)\big) = b\end{array}\middle|\begin{array}{c} b \overset{r}{\leftarrow} \{0,1\};\\ \varGamma_b \leftarrow 1/x;\ \varGamma_{1-b} \overset{r}{\leftarrow} \mathbb{Z}_p^*\end{array}\right] - \frac{1}{2}\right| \leq \frac{2(q_G+q+3)^2 q}{p}.$$

Proof. Instead of letting $\mathcal{A}$ interact with the actual oracles, we play the following game.

We maintain two lists: $L = \{ (F_i, s_i) : i = 0, \ldots, t-1\}$ and $L' = \{ (F'_i, s'_i) : i = 0, \ldots, t'-1\}$. Here $s_i, s'_i \in \{0,1\}^*$ are encodings and $F_i, F'_i \in \mathbb{Z}_p[X, \mathit{\Gamma}_0, \mathit{\Gamma}_1]$ are multivariate polynomials in $X, \mathit{\Gamma}_0$, and $\mathit{\Gamma}_1$. The total length of lists at step $\tau \leq q_G$ in the game must be

$$t + t' = \tau + q + 3. \tag{2}$$

At the beginning of the game, we initialize the lists to $F_0 = 1, F_1 = X, \ldots, F_q = X^q$ and $F'_0 = \mathit{\Gamma}_0, F'_1 = \mathit{\Gamma}_1$. The corresponding encodings are set to arbitrary distinct strings in $\{0,1\}^*$. The lists have length $t = q+1$ and $t' = 2$.

We start the game by providing $\mathcal{A}$ with encodings $(s_0, \ldots, s_q, s'_0)$. Algorithm $\mathcal{A}$ begins to issue oracle queries. We respond to them in the standard fashion:

Group action: Given a multiply/divide bit and two operands s_i and s_j $(0 \leq i, j < t)$, we compute $F_t = F_i \pm F_j$ accordingly. If $F_t = F_l$ for some $l < t$, we set $s_t = s_l$. Otherwise, we set s_t to a random string in $\{0,1\}^* \backslash \{s_0, \ldots, s_{t-1}\}$, and increment t by 1. Group action in G_1 is computed similarly, except we operate on list L'.

Bilinear pairing: Given two operands s_i and s_j $(0 \leq i, j < t)$, we compute the product $F_{t'} = F_i F_j$. If $F_{t'} = F_l$ for some $l < t'$, we set $s_{t'} = s_l$. Otherwise we set it to a random string in $\{0,1\}^* \backslash \{s_0, \ldots, s_{t'-1}\}$. We then increment t' by 1.

After making at most q_G queries, $\mathcal{A}$ halts with a guess $\hat{b} \in \{0,1\}$. We now choose $x, y \xleftarrow{r} \mathbb{Z}_p^*$ and consider $\mathit{\Gamma}_b \leftarrow 1/x, \mathit{\Gamma}_{1-b} = y$ for both choices of b. Our simulation is perfect and reveals nothing to $\mathcal{A}$ about b unless the values that we chose for indeterminates give rise to some non-trivial equality relation. Specifically, algorithm $\mathcal{A}$ wins the game if for any $F_i \neq F_j$ or any $F'_i \neq F'_j$, either of these hold:

1. $F_i(x, 1/x, y) - F_j(x, 1/x, y) = 0$
2. $F_i(x, y, 1/x) - F_j(x, y, 1/x) = 0$
3. $F'_i(x, 1/x, y) - F'_j(x, 1/x, y) = 0$
4. $F'_i(x, y, 1/x) - F'_j(x, y, 1/x) = 0$

Notice that $\mathcal{A}$ can never engineer an encoding of an element whose corresponding polynomial would have a $1/X$ term unless he is explicitly given it. Therefore, we can only get a non-trivial equality relation as a result of numerical cancellation.

For all i, we have: $\deg(F_i) \leq q$ and $\deg(F'_i) \leq 2q$. We can use the Schwartz-Zippel Theorem [Sch80] to bound the probability of a cancelation. It tells us that for all i, j, $\Pr[F_i - F_j = 0] \leq q/p$ and $\Pr[F'_i - F'_j = 0] \leq 2q/p$. Thus $\mathcal{A}$'s advantage is

$$\begin{aligned}\epsilon &\le 2 \cdot \left(\binom{t}{2} \frac{q}{p} + \binom{t'}{2} \frac{2q}{p} \right) \\ &< 2(q_G + q + 3)^2 \frac{q}{p} \ \ \text{(using Equation (2))} \\ &= O\left(\frac{q_G^2 q + q^3}{p} \right).\end{aligned}$$

□

It turns out that in a generic group model algorithm $\mathcal{A}$ that solves the q-DBDHI problem has advantage, which is roughly twice as much as an advantage of an algorithm solving the q-SDH problem (see [BB04b], Section 5); the asymptotic complexities are the same.

The following corollary is immediate.

Corollary 1. *Any adversary that breaks the q-DBDHI assumption with probability* $\nu = \frac{1}{2} + \epsilon$ $(0 < \epsilon < 1/2)$ *in generic groups of order* p *such that* $q < o(\sqrt[3]{p})$ *requires* $\Omega(\sqrt{\epsilon p/q})$ *generic group operations.*

7 Conclusion

We have presented a simple and efficient construction of a verifiable random function. Our VRF's proofs and keys have constant size regardless of the size of the input. Our proofs of security are based on a bilinear Diffie-Hellman inversion assumption, which seems reasonable given current state of knowledge. We also demonstrated that our scheme can be instantiated with elliptic groups of very reasonable size which makes our constructions quite practical.

References

[BB04a] Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity based encryption without random oracles. In *Advances in Cryptology—EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–238. Berlin: Springer-Verlag, 2004.

[BB04b] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In *Advances in Cryptology—EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 56–73. Berlin: Springer-Verlag, 2004.

[BF01] Dan Boneh and Matt Franklin. Identity-based encryption from the Weil pairing. *Lecture Notes in Computer Science*, 2139:213–229, 2001.

[BIB89] Judit Bar-Ilan and Donald Beaver. Non-cryptographic fault-tolerant computing in a constant number of rounds. In *Proceedings of the ACM Symposium on Principles of Distributed Computation*, pages 201–209, 1989.

[BLZ94] Johannes A. Buchmann, J. Loho, and J. Zayer. An implementation of the general number field sieve. *Lecture Notes in Computer Science*, 773:159–166, 1994.

[BoGW88] Michael Ben-or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computing. In *Proceedings of the 20th Annual ACM Symposium on the Theory of Computing*, pages 1–10, 1988.

[BS02] Dan Boneh and Alice Silverberg. Application of multilinear forms to cryptography. Cryptology ePrint Archive, Report 2002/080, 2002. `http://eprint.iacr.org/2002/080/`.

[DBS04] Ratna Dutta, Rana Barua, and Palash Sarkar. Pairing-based cryptographic protocols : a survey. Cryptology ePrint Archive, Report 2004/064, 2004. `http://eprint.iacr.org/2004/064/`.

[Dod03] Yevgeniy Dodis. Efficient construction of (distributed) verifiable random functions. In *Proceedings of 6th International Workshop on Theory and Practice in Public Key Cryptography*, pages 1–17, 2003.

[Gal01] Steven D. Galbraith. Supersingular curves in cryptography. *Lecture Notes in Computer Science*, 2248:495–513, 2001.

[GB99] S. Goldwasser and M. Bellare. Lecture notes on cryptography. Summer Course "Cryptography and Computer Security" at MIT, 1996–1999, 1999.

[GL89] Oded Goldreich and Leonid Levin. A hard-core predicate for all one-way functions. In *Proceedings of the 21th Annual ACM Symposium on the Theory of Computing*, pages 25–32, 1989.

[JN01] Antoine Joux and Kim Nguyen. Separating Decision Diffie-Hellman from Diffie-Hellman in cryptographic groups. Cryptology ePrint Archive, Report 2001/003, 2001. `http://eprint.iacr.org/2001/003/`.

[JS04] Stanislaw Jarecki and Vitaly Shmatikov. Handcuffing big brother : an abuse-resilient transaction escrow scheme. In *Advances in Cryptology - Proceedings of EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 590–608. Springer-Verlag, 2004.

[Lys02] Anna Lysyanskaya. Unique signatures and verifiable random functions from DH-DDH separation. In *Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology*, pages 597–612, 2002.

[MR01] Silvio Micali and Leonid Reyzin. Soundness in the public-key model. *Lecture Notes in Computer Science*, 2139:542–565, 2001.

[MR02] Silvio Micali and Ronald L. Rivest. Micropayments revisited. In *CT-RSA*, pages 149–163, 2002.

[MRV99] Silvio Micali, Michael O. Rabin, and Salil P. Vadhan. Verifiable random functions. In *Proceedings of the 40th IEEE Symposium on Foundations of Computer Science*, pages 120–130, 1999.

[MSK02] Shigeo Mitsunari, Ryuichi Sakai, and Masao Kasahara. A new traitor tracing. *IEICE Trans. Fundamentals*, pages 481–484, 2002.

[NR97] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *Proceedings of the 38th IEEE Symposium on Foundations of Computer Science*, pages 458–467, 1997.

[Sch80] Jacob T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the Association for Computing Machinery*, 27:701–717, 1980.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. *Lecture Notes in Computer Science*, 1233:256–266, 1997.

[STW96] Michael Steiner, Gene Tsudik, and Michael Waidner. Diffie-Hellman key distribution extended to group communication. In *Proceedings of the 3rd ACM Conference on Computer and Communications Security*, pages 31–37, 1996.