Algebraic Pseudorandom Functions with Improved Efficiency from the Augmented Cascade*

DAN BONEH† HART MONTGOMERY‡
ANANTH RAGHUNATHAN§

Department of Computer Science, Stanford University
{dabo,hartm,ananthr}@cs.stanford.edu

September 8, 2020

Abstract

We construct an algebraic pseudorandom function (PRF) that is more efficient than the classic Naor-Reingold algebraic PRF. Our PRF is the result of adapting the cascade construction, which is the basis of HMAC, to the algebraic settings. To do so we define an augmented cascade and prove it secure when the underlying PRF satisfies a property called parallel security. We then use the augmented cascade to build new algebraic PRFs. The algebraic structure of our PRF leads to an efficient large-domain Verifiable Random Function (VRF) and a large-domain simulatable VRF.

1 Introduction

Pseudorandom functions (PRFs), first defined by Goldreich, Goldwasser, and Micali [GGM86], are a fundamental building block in cryptography and have numerous applications. They are used for encryption, message integrity, signatures, key derivation, user authentication, and many other cryptographic mechanisms. Beyond cryptography, PRFs are used to defend against denial of service attacks [Ber96, CW03] and even to prove lower bounds in learning theory.

In a nutshell, a PRF is indistinguishable from a truly random function. We give precise definitions in the next section. The fastest PRFs are built from block ciphers like AES and security is based on ad-hoc interactive assumptions. In 1996, Naor and Reingold [NR97] presented an elegant PRF whose security can be deduced from the hardness of the Decision Diffie-Hellman problem (DDH) defined in the next section. The Naor-Reingold PRF takes as input an m-bit string $b = b_1 \dots b_m \in \{0,1\}^m$ and a secret key $(h, x_1, \dots, x_m)$ and outputs

$$F_{\text{NR}}\big(\underbrace{(h, x_1, \dots, x_m)}_{key}, \underbrace{(b_1 \dots b_m)}_{input}\big) := h^w \quad \text{where} \quad w := \prod_{i=1}^{m} x_i^{b_i} \,. \tag{1}$$

*An extended abstract [BMR10] appears in ACM CCS 2010.
†Supported by NSF and the Packard Foundation.
‡Supported by a Stanford Graduate Fellowship.
§Supported by a Stanford School of Engineering Fellowship.

We define this PRF more precisely in Section 4.1. Evaluating this PRF amounts to m modular multiplications plus one exponentiation. This PRF was recently generalized by Lewko and Waters [LW09] to work in groups where DDH may be easy, but where a weaker assumption called k-linear may hold. While this has clear security benefits, there is a cost in performance compared to Naor-Reingold.

The algebraic structure of the Naor-Reingold PRF leads to several beautiful applications that are much harder to construct with generic PRFs built from block ciphers. Some examples include Verifiable Random Functions (VRFs) [HW10], oblivious PRFs (used for private keyword search [FIPR05] and secure computation of set-intersection [JL09]), and distributed PRFs [NR97], to name a few. Another algebraic PRF due to Dodis and Yampolskiy [DY05] (based on the signature scheme from [BB04c]) also has many useful applications. However, this PRF is only known to be secure when the domain is small (i.e. polynomial size).

Our results. We describe a new algebraic PRF that has the same domain as Naor-Reingold, but requires fewer multiplications to evaluate and uses shorter private keys. For parameters ℓ and n our PRF takes inputs $(u_1, \ldots, u_n)$ in $\mathbb{Z}_\ell^n$ along with a key $(h, x_1, \ldots, x_n)$ and outputs

$$F\big(\ (h, x_1, \ldots, x_n),\ (u_1 \ldots u_n)\ \big) := h^{1/w} \quad \text{where} \quad w := \prod_{i=1}^{n}(x_i + u_i)\,. \tag{2}$$

For a domain of size 2^m we have $n = m/\log_2 \ell$ and therefore evaluating this PRF requires a factor of $\log_2 \ell$ fewer multiplications than (1) to compute w. Since computing w often takes roughly the same time as the final exponentiation, evaluating this PRF is about twice as fast as evaluating the Naor-Reingold PRF. The secret key is shorter by a factor of $\log_2 \ell$. We prove security of this PRF from the ℓ-DDH assumption defined in the next section. The larger ℓ gets the stronger the assumption becomes and therefore one should keep ℓ small. Setting $\ell = 16$ or 256 for example is a reasonable choice.

Techniques. We prove security of the PRF by developing a PRF composition theorem that generalizes the classic cascade construction of Bellare, Canetti and Krawczyk [BCK96b]. The cascade construction, shown in Figure 1(a), constructs a PRF with a large domain from a PRF with a small domain and is the basis for the NMAC and HMAC PRFs [BCK96a, Bel06].

Unfortunately, the cascade construction is insufficient for our purposes because it requires the output of the underlying PRF to be at least as long as the secret key. We therefore define the augmented cascade, shown in Figure 1(b), which eliminates this requirement by using supplemental secret information in every block. The augmented cascade can be applied directly to PRFs whose output is much smaller than the secret key. Suprisingly, security of the augmented cascade does not follow from security of the underlying PRF. We therefore develop a sufficient condition on the underlying PRF, called parallel security, that implies security of the augmented cascade.

Armed with the augmented cascade theorem, we build our large-domain PRF by plugging the Dodis-Yampolskiy small-domain PRF [DY05] into the augmented cascade. To prove security, we prove that the Dodis-Yampolskiy PRF is parallel secure. As a short aside, we show the power of the augmented cascade theorem by using it to quickly prove security of the Naor-Reingold and Lewko-Waters PRFs.

Verifiable Random Functions. The algebraic structure of the PRF in (2) enables many of the same applications as the Naor-Reingold PRF. In Sections 6.2 and 7 we show how to convert this PRF into an efficient Verifiable Random Function (VRF) with a large domain in groups with a bilinear map. A VRF, as defined in [MRV99], is a PRF that also outputs a proof that it was evaluated correctly. VRFs give signature schemes where every message has a unique signature. They were also used to construct e-cash schemes [BCKL09, ASM07].

Hohenberger and Waters [HW10] recently constructed an elegant VRF with a large domain. Our VRF is a little less efficient, but surprisingly is based on a weaker assumption. Their construction requires an assumption where the problem instance has size $O(mQ)$ where 2^m is the size of the domain and Q is the number of adversarial queries. We only require a problem instance of size $O(m)$. Our security proof makes use of admissible hash functions as in [BB04b]. We also describe a large-domain *simulatable* VRF, as defined in [CL07].

2 Preliminaries

2.1 Pseudorandom Functions

We begin by reviewing the definition of pseudorandom functions [GGM86]. Informally, a pseudorandom function is an efficiently computable function such that no efficient adversary can distinguish the function from a truly random function given only black-box access.

More precisely, a PRF is an efficiently computable function $F : K \times X \to Y$ where K is called the key space, X is called the domain, and Y is called the range. Security for a PRF is defined using two experiments between a challenger and an adversary $\mathcal{A}$. For $b \in \{0,1\}$ the challenger in Exp_b works as follows.

When $b = 0$ the challenger chooses a random key $k \in K$ and sets $f(\cdot) := F(k, \cdot)$.
When $b = 1$ the challenger chooses a random function $f : X \to Y$.
The adversary (adaptively) sends input queries $x_1, \ldots, x_q$ in X to the challenger and the challenger responds with $f(x_1), \ldots, f(x_q)$. 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.

Definition 1. *A PRF $F : K \times X \to Y$ is secure if for all efficient adversaries $\mathcal{A}$ the quantity*

$$\mathsf{PRF}_{adv}[\mathcal{A}, F] := |W_0 - W_1|$$

is negligible.

As usual, one makes the terms "efficient" and "negligible" precise using asymptotic notation by equating efficient with probabilistic polynomial time and equating negligible with functions smaller than all inverse polynomials. Here, we use non-asymptotic language to simplify the notation.

2.2 Complexity assumptions

Notation. In this section and in Section 4.2 it is convenient to use vector notation defined as follows. Let $\mathbb{G}$ be a group of prime order p with generator g.

- for vectors $\bar{g} = (g_1, \ldots, g_n) \in \mathbb{G}^n$ and $\bar{x} = (x_1, \ldots, x_n) \in \mathbb{Z}_p^n$ define $\bar{g}^{\bar{x}} := \left(g_1^{x_1}, \ldots, g_n^{x_n}\right) \in \mathbb{G}^n$. For a scalar $g \in \mathbb{G}$ define $g^{\bar{x}} := (g^{x_1}, \ldots, g^{x_n})$.
- for a matrix $A = (a_{i,j}) \in \mathbb{Z}_p^{n \times m}$ and a vector $\bar{g} \in \mathbb{G}^m$ define

$$A \cdot \bar{g} := \bar{h} \in \mathbb{G}^n \quad \text{where} \quad h_i := \prod_{j=1}^{m} g_j^{a_{i,j}} \quad \text{for } i = 1, \ldots, n.$$

  and for a scalar $g \in \mathbb{G}$ define $g^A := \left(g^{(a_{i,j})}\right) \in \mathbb{G}^{n \times m}$.

We also use $[k]$ to denote the set $\{1, \ldots, k\}$.

The k-linear assumption. Let V_k be the linear subspace of $\mathbb{Z}_p^{k+1}$ containing all vectors orthogonal to $(-1, 1, 1, \ldots, 1)$; its dimension is k. A vector $v = (v_0, \ldots, v_k)$ is in V_k if v_0 is the sum of the remaining coordinates. When $k = 1$ a vector $v = (v_0, v_1)$ is in V_1 if and only if $v_0 = v_1$.
For an algorithm $\mathcal{A}$ define

$$\mathsf{LIN}_{\mathrm{adv}}^{(k)}[\mathcal{A}, \mathbb{G}] := \left| \Pr[\mathcal{A}(\bar{g},\ \bar{g}^{\bar{x}}) = 1] - \Pr[\mathcal{A}(\bar{g},\ \bar{g}^{\bar{y}}) = 1] \right|$$

where $\bar{g}$ is uniform in $\mathbb{G}^{k+1}$, $\bar{x}$ is uniform in V_k, and $\bar{y}$ is uniform in $\mathbb{Z}_p^{k+1}$.

Definition 2. *For $k \geq 1$ we say that the k-linear assumption holds for the group $\mathbb{G}$ if for all efficient algorithms $\mathcal{A}$ the advantage $\mathsf{LIN}_{adv}^{(k)}[\mathcal{A}, \mathbb{G}]$ is negligible.*

The 1-linear assumption is identical to the standard Decision Diffie-Hellman (DDH) problem in $\mathbb{G}$ and we write $\mathsf{DDH}_{\mathrm{adv}}[\mathcal{A}, \mathbb{G}]$ to denote $\mathsf{LIN}_{\mathrm{adv}}^{(1)}[\mathcal{A}, \mathbb{G}]$. For $k = 2$ we obtain the decision linear assumption defined in [BBS04]. For larger k we obtain the generalized linear assumption defined in [Sha07, HK07].

It is not difficult to show that if the k-linear assumption holds for $\mathbb{G}$ then so does the ℓ-linear assumption for all $\ell > k$. It is believed that the larger k is the weaker the assumption becomes. In particular, the 2-linear assumption may hold in groups where the 1-linear assumption (a.k.a DDH) is false.

The k-DDH assumption. For $x \in \mathbb{Z}_p$ let $\mathrm{pow}(x, k)$ be the vector $(1, x, x^2, \ldots, x^k) \in \mathbb{Z}_p^{k+1}$. The k-DDH assumption states that $g^{1/x}$ is indistinguishable from a random group element given $g^{\mathrm{pow}(x,k)}$. More precisely, for an algorithm $\mathcal{A}$ define

$$\mathsf{DDH}_{\mathrm{adv}}^{(k)}[\mathcal{A}, \mathbb{G}] := \left| \Pr[\mathcal{A}(g^{\mathrm{pow}(x,k)},\ g^{1/x}) = 1] - \Pr[\mathcal{A}(g^{\mathrm{pow}(x,k)},\ h) = 1] \right|$$

where g, h are uniform in $\mathbb{G}$ and x is uniform in $\mathbb{Z}_p$. When $x = 0$ we define $g^{1/x}$ to be 1 in $\mathbb{G}$.

Definition 3. *For $k \geq 1$ we say that the k-DDH assumption holds for the group $\mathbb{G}$ if for all efficient algorithms $\mathcal{A}$ the advantage $\mathsf{DDH}_{adv}^{(k)}[\mathcal{A}, \mathbb{G}]$ is negligible.*

This assumption was previously used in [BB04a, DY05] where it was called k-DDHI. The 1-DDH assumption implies the standard DDH assumption. Moreover, for $k > 1$ the k-DDH assumption implies the ℓ-DDH assumption for $\ell < k$.

A hierarchy. From the facts stated above we obtain a hierarchy of complexity assumptions from the k-linear and k-DDH assumptions:

$$\ldots \leq k\text{-lin} \leq \ldots \leq 1\text{-lin} \equiv \text{DDH} \leq 1\text{-DDH} \leq \ldots \leq k\text{-DDH} \leq \ldots$$

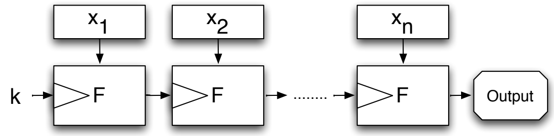
The assumptions becomes stronger as one moves from left to right. In the generic group model this hierarchy can be shown not to collapse [BBG05].

A useful lemma. We will need the following lemma from [BHHO08] (Lemma 1). Let $\mathbb{Z}_p^{n\times m}$ be the set of $n \times m$ matrices over $\mathbb{Z}_p$ and let $\mathrm{Rk}_1(\mathbb{Z}_p^{n\times m})$ be the set of matrices in $\mathbb{Z}_p^{n\times m}$ of rank 1.

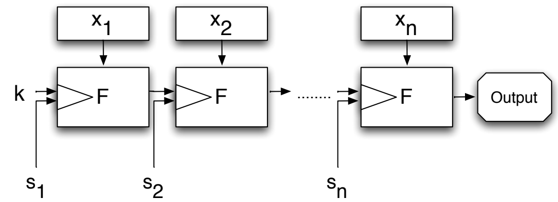
Let $\mathbb{G}$ be a group of order p with generator g. Let A_0 be uniform in $\mathrm{RK}_1(\mathbb{Z}_p^{n\times m})$ and A_1 be uniform in $\mathbb{Z}_p^{n\times m}$. For an algorithm $\mathcal{A} : \mathbb{G}^{n\times m} \to \{0, 1\}$ define

$$\mathrm{adv}[\mathcal{A}] := \left| \Pr[\mathcal{A}(g^{(A_0)}) = 1] - \Pr[\mathcal{A}(g^{(A_1)}) = 1] \right|$$

The following lemma shows that when DDH is hard in $\mathbb{G}$, no efficient adversary can distinguish a rank 1 matrix in the exponent from a random matrix in the exponent.



(a) The cascade construction, key $= k$



(b) The augmented cascade, key $= (s_1, \ldots, s_n, k)$

Figure 1: Cascade and augmented cascade

Lemma 1. *For every algorithm $\mathcal{A}$ there exists an algorithm $\mathcal{B}$ with about the same running time as $\mathcal{A}$ so that*

$$adv[\mathcal{A}] \leq \min(m, n) \cdot \textsf{DDH}_{adv}[\mathcal{B}, \mathbb{G}]$$

2.3 Bilinear maps

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

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

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

The k-BDH assumption. For $x \in \mathbb{Z}_p$, let $g^{\mathrm{pow}(x,k)}$ be the vector $(g, g^x, \ldots, g^{(x^k)})$. The k-BDH assumption states that $e(g, u)^{1/x}$ is indistinguishable from a random group element in $\mathbb{G}_{\mathrm{T}}$ given u and $g^{\mathrm{pow}(x,k)}$. More precisely, for an algorithm $\mathcal{A}$ define

$$\textsf{BDH}^{(k)}_{\mathrm{adv}}[\mathcal{A}, \mathbb{G}] := \left| \Pr[\mathcal{A}(u, g^{\mathrm{pow}(x,k)}, e(g^{1/x}, u)) = 1] - \Pr[\mathcal{A}(u, g^{\mathrm{pow}(x,k)}, \gamma) = 1] \right|$$

where g, u are uniform in $\mathbb{G}$, x is uniform in $\mathbb{Z}_p$, and γ is uniform in $\mathbb{G}_T$.

Definition 4. *For $k \geq 1$, we say that the k-BDH assumption holds for the group $\mathbb{G}$ if for all efficient algorithms $\mathcal{A}$, the advantage $\textsf{BDH}^{(k)}_{adv}[\mathcal{A}, \mathbb{G}]$ is negligible.*

3 The Augmented Cascade

3.1 The cascade PRF

The cascade pseudorandom function, defined in [BCK96b], constructs a secure PRF with domain X^n from a secure PRF with domain X. The cascade construction is shown in Figure 1(a). More precisely, let $F : K \times X \to K$ be a secure PRF. We define the cascade of F denoted $F^{*n} : K \times X^n \to K$ as:

input: key $k_0 \in K$, and $(x_1, \ldots, x_n) \in X^n$

for $i = 1, \ldots, n$ do:
  $k_i \leftarrow F(k_{i-1}, x_i)$

output k_n

Here the output range of F must equal the key space K.

Cascade is the basis for the NMAC and HMAC message authentication codes [BCK96b, Bel06]. Cascade is a generalization of the GGM PRF [GGM86], which can be viewed as a method to convert a PRF with a 1-bit domain into a PRF with an n-bit domain. The security of the cascade construction is stated concretely in the following theorem, which is shown in [BCK96b].

Theorem 2. *For every q-query PRF adversary $\mathcal{A}$ attacking F^{*n} there exists a q-query PRF adversary $\mathcal{B}$ attacking F such that*

$$\textsf{PRF}_{adv}[\mathcal{A}, F^{*n}] \leq nq \cdot \textsf{PRF}_{adv}[\mathcal{B}, F]$$

where $\mathcal{B}$ runs in about the same time as $\mathcal{A}$.

3.2 Augmented Cascade PRF

The cascade construction works with a PRF F whose output is as long as the PRF key. When constructing algebraic PRFs, the starting point is often a PRF F whose output is shorter than required for cascade. We therefore need to augment the output of F so that its output is a valid key for F. Consider a PRF F operating on the following spaces:

$$F : \underbrace{(S \times K)}_{\text{key}} \times X \to K$$

Notice that the key for F is a pair in (S, K) while the output is in K and therefore not a complete key. In the augmented cascade we append a fresh random string to the output to make it into a valid key.

We define the augmented cascade, denoted $\hat{F}^{*n}$, as a function

$$\hat{F}^{*n} : \underbrace{(S^n \times K)}_{\text{key}} \times X^n \to K$$

The function's domain is X^n and its keys are tuples of the form $(s_1, \ldots, s_n, k) \in S^n \times K$. The augmented cascade is shown in Figure 1(b) and is defined as follows:

input: key $(s_1, \ldots, s_n, k_0) \in S^n \times K$, and
  value $(x_1, \ldots, x_n) \in X^n$

for $i = 1, \ldots, n$ do:
  $k_i \leftarrow F(\ (s_i, k_{i-1}),\ x_i)$

output k_n

Security. Unfortunately, the augmented cascade can be insecure even if the underlying function F is a secure PRF. For example, F can be a secure PRF even if it ignores the part of the key in K (i.e. F only uses the part of the key in S). In this case, since we ignore k_i (for all i), the last block of the augmented cascade construction is evaluated independently of the first $n-1$ blocks. Thus, the resulting augmented cascade construction $\hat{F}^{*n}$ ignores the first $n-1$ input blocks and hence cannot be a secure PRF. In the next two sections we establish sufficient conditions for security of the augmented cascade.

3.3 Parallel composition security

In Theorem 3 below we will show that the augmented cascade is a secure PRF provided that the underlying function F satisfies a property we call *parallel security*. This property says that F remains a secure PRF when the adversary has access to multiple instances of the function with different but related keys.

For a function $F : (S \times K) \times X \to K$ and an integer $q > 0$ we define q related keys $(s, k_1), \ldots, (s, k_q)$ where $s \in S$ and $k_1, \ldots, k_q \in K$. These keys are related since they all share the same s. We say that the function F is q-parallel secure if the resulting set of q functions is indistinguishable from q random independent functions.
More precisely, let $F^{(q)}$ be the function: $F^{(q)} : (S \times K^q) \times \big(X \times [q]\big) \to K$ defined by

$$F^{(q)}\big(\ \underbrace{(s, k_1, \ldots, k_q)}_{\text{key}}\ ,\ \underbrace{(x, i)}_{\text{input}}\big) := F\big(\ (s, k_i)\ ,\ x\big)$$

Here $i \in [q]$ selects the key (s, k_i) to be used in the function F. Thus, $F^{(q)}$ emulates q functions F whose keys are (s, k_i) for $i = 1, \ldots, q$.

Definition 5. *We say that $F : (S \times K) \times X \to K$ is a q-parallel secure PRF if $F^{(q)}$ is a secure PRF.*

The function F need not be q-parallel secure even if it is secure as a PRF. For example, as above, a secure PRF $F : (S \times K) \times X \to K$ that ignores the part of the key in K (i.e. only uses the S part of the key) is clearly not 2-parallel secure. Even when S is small (e.g. $S = \{0, 1\}$) the function F may not be 2-parallel secure even though F is a secure PRF.

3.4 Security of the augmented cascade

We now prove security of the augmented cascade provided that the underlying PRF is parallel-secure.

Theorem 3. *If F is q-parallel secure then the augmented cascade $\hat{F}^{*n}$ is a secure PRF against q-query adversaries.*
In particular, for every q-query PRF adversary $\mathcal{A}$ attacking $\hat{F}^{*n}$ there is a q-query PRF adversary $\mathcal{B}$ attacking $F^{(q)}$ such that

$$\textsf{PRF}_{\textsf{adv}}[\mathcal{A}, \hat{F}^{*n}] \le n \cdot \textsf{PRF}_{\textsf{adv}}[\mathcal{B}, F^{(q)}]$$

where $\mathcal{B}$ runs in about the same time as $\mathcal{A}$.

The proof uses a hybrid argument similar to the proof of the original cascade [BCK96b], but is sufficiently different to require its own proof.

Proof of Theorem 3. Given an adversary $\mathcal{A}$ we construct an adversary $\mathcal{B}$ as required. The intuition for the construction of $\mathcal{B}$ comes from the following sequence of $n + 1$ hybrid experiments between a challenger and adversary $\mathcal{A}$. In hybrid i, the challenger replaces the first i stages of the augmented cascade with a truly random function, while the last $n - i$ stages are carried out as in the standard augmented cascade.

More precisely, for $i = 0, \ldots, n$ define the challenger in hybrid experiment $\mathcal{P}_i$ as follows:

setup: the challenger chooses a random function $f : X^i \to K$ and random keys $s_1, \ldots, s_n$ in S.

queries: to respond to a query $(x_1, \ldots, x_n) \in X^n$ from $\mathcal{A}$ do:
  let $k_i \leftarrow f(x_1, \ldots, x_i) \in K$

for $j = i+1, \ldots, n$ do:
$\quad k_j \leftarrow F(\ (s_j, k_{j-1}),\ x_j)$

send k_n to $\mathcal{A}$

For $i = 0, \ldots, n$, let W_i be the probability that $\mathcal{A}$ outputs 1 in hybrid number i. Observe that in hybrid $\mathcal{P}_0$ the adversary $\mathcal{A}$ interacts with the function $\hat{F}^{*n}$ while in hybrid $\mathcal{P}_n$ the adversary interacts with a random function $f : X^n \to K$. Therefore,

$$\mathsf{PRF}_{\text{adv}}[\mathcal{A}, \hat{F}^{*n}] = |W_n - W_0|$$

It follows by the standard hybrid argument that there exists a $t \in [1, n]$ such that

$$\mathsf{PRF}_{\text{adv}}[\mathcal{A}, \hat{F}^{*n}] \leq n \cdot |W_{t-1} - W_t| \tag{3}$$

We construct a q-query PRF adversary $\mathcal{B}$ such that

$$\mathsf{PRF}_{\text{adv}}[\mathcal{B}, F^{(q)}] = |W_{t-1} - W_t| \tag{4}$$

Combining (3) and (4) proves the theorem.

Adversary $\mathcal{B}$ emulates the challenger in hybrid $\mathcal{P}_t$ or $\mathcal{P}_{t+1}$. This requires $\mathcal{B}$ to emulate a random function $f : X^t \to K$. To do so, it is convenient to describe $\mathcal{B}$ using an associative array T that maps elements of X^t to numbers in $\{1, \ldots, q\}$. Initially the array T is empty.

Adversary $\mathcal{B}$ interacts with its $F^{(q)}$ challenger and emulates a $\hat{F}^{*n}$ challenger for $\mathcal{A}$. $\mathcal{B}$ works as follows:

setup: $T \leftarrow \emptyset$, ctr $\leftarrow 0$, choose random $s_{t+2}, \ldots, s_n$ in S

queries: to respond to a query for $(x_1, \ldots, x_n) \in X^n$ from $\mathcal{A}$ do:

if $T[x_1 \ldots x_t] = \perp$ (i.e. $x_1 \ldots x_t$ is a new prefix)
$\quad$ increment ctr by 1 and set $T[x_1 \ldots x_t] :=$ ctr

let $u \leftarrow T[x_1 \ldots x_t] \in \{1, \ldots, q\}$
$\mathcal{B}$ queries its $F^{(q)}$ challenger at (x_{t+1}, u) and obtains some $k_{t+1} \in K$

note: k_{t+1} is either random in K or is equal to $F((s, k^*_u), x_{t+1})$ for some random key (s, k^*_u) chosen by $\mathcal{B}$'s challenger.

for $j = t+2, \ldots, n$ do: (finish the cascade)
$\quad k_j \leftarrow F(\ (s_j, k_{j-1}),\ x_j)$

send k_n to $\mathcal{A}$

eventually $\mathcal{A}$ outputs a bit $b' \in \{0, 1\}$. $\mathcal{B}$ outputs the same bit and terminates.

Since $\mathcal{A}$ makes at most q queries the variable u is always in the range $[1, q]$ and therefore all of $\mathcal{B}$'s queries to its challenger are in the proper range.

When $\mathcal{B}$'s challenger emulates a random function then $\mathcal{B}$ emulates a $\mathcal{P}_{t+1}$ challenger to adversary $\mathcal{A}$. When $\mathcal{B}$'s challenger emulates $F^{(q)}$ then $\mathcal{B}$ emulates a $\mathcal{P}_t$ challenger to adversary $\mathcal{A}$. Therefore (4) holds which completes the proof of the theorem. $\square$

4 Existing Algebraic PRFs

We briefly review two existing algebraic PRFs in the literature and explain how their security neatly follows from the security of the augmented cascade construction.

4.1 The Naor-Reingold PRF

We start with the Naor-Reingold PRF [NR97]. Let $\mathbb{G}$ be a group of order p and let $f : (\mathbb{Z}_p \times \mathbb{G}) \times \{0,1\} \to \mathbb{G}$ be the function

$$f\big((x,h), b\big) := h^{(x^b)} = \begin{cases} h & \text{if } b = 0 \\ h^x & \text{if } b = 1 \end{cases} \tag{5}$$

Plugging f into the augmented cascade we obtain the following PRF whose domain is $\{0,1\}^n$ and range is $\mathbb{G}$:

$$F_{\text{NR}} := \hat{f}^{*n}\big(\underbrace{(x_1, \dots, x_n, h)}_{\text{key}}, \underbrace{(b_1 \dots b_n)}_{\text{input}}\big) = h^{\left(x_1^{b_1} \cdots x_n^{b_n}\right)}$$

To show that F_{NR} is a secure PRF it suffices to show that f is parallel secure. Naor and Reingold do so implicitly in their proof. We state this in the following lemma.

Lemma 4. *If the DDH assumption holds for the group $\mathbb{G}$ then the function f defined in (5) is q-parallel secure for all q polynomial in the security parameter.*

Proof. To prove that f is q-parallel secure we need to show that $f^{(q)}$ is a secure PRF. The function $f^{(q)}$ has domain $\{0,1\} \times [q]$ which is a set of size $2q$. Hence, it suffices to show that enumerating the $2q$ outputs of $f^{(q)}$ gives a secure pseudorandom generator. In particular, all we need to show is that

$$G(x, h_1, \dots, h_q) := (h_1,\ h_1^x, \dots, h_q,\ h_q^x)$$

is a secure PRG, assuming DDH holds in $\mathbb{G}$. This is a direct application of the random self reduction of DDH [NR97]. For completeness, we briefly review the reduction.

Let $\mathcal{A}$ be an algorithm that distinguishes the output of G on a random seed from a random tuple in G^{2q}. We build an algorithm $\mathcal{B}$ that breaks DDH in $\mathbb{G}$. Given a tuple (g, h, u, v) as input, algorithm $\mathcal{B}$ chooses random $a_1, \dots, a_q$ and $b_1, \dots, b_q$ in $\mathbb{Z}_p$ and computes

$$\big(g^{a_1} u^{b_1},\ h^{a_1} v^{b_1},\ \dots,\ g^{a_q} u^{b_q},\ h^{a_q} v^{b_q}\big) \in \mathbb{G}^{2q} \tag{6}$$

Naor and Reingold show that if (g, h, u, v) is a DDH tuple then (6) is distributed as the output of G on a random seed. If (g, h, u, v) is a random tuple then (6) is random in $\mathbb{G}^{2q}$. Algorithm $\mathcal{B}$ runs $\mathcal{A}$ on the tuple (6) and outputs whatever $\mathcal{A}$ outputs. Then $\mathsf{DDH}_{\text{adv}}[\mathcal{B}, \mathbb{G}] = \mathsf{PRF}_{\text{adv}}[\mathcal{A}, f^{(q)}]$ as required. The running time overhead of $\mathcal{B}$ is polynomial in q. $\square$

Combining Theorem 3 with Lemma 4 proves that the function F_{NR} is a secure PRF whenever DDH holds in $\mathbb{G}$.

4.2 The Lewko-Waters PRF

Lewko and Waters construct a PRF from the k-linear assumption [LW09]. While their PRF is not as efficient as the PRF of Naor and Reingold, their construction can remain secure in groups where DDH is false.

Let $\mathbb{G}$ be a group of order p. Let $k > 0$ be a parameter and define $f : (\mathbb{Z}_p^{k\times k} \times \mathbb{G}^k) \times \{0,1\} \to \mathbb{G}^k$ as the function

$$f\big((A,h),b\big) := A^b \cdot h = \begin{cases} h & \text{if } b = 0 \\ A \cdot h & \text{if } b = 1 \end{cases} \tag{7}$$

Recall that the notation $A \cdot h$ is defined in Section 2.2. Plugging f into the augmented cascade we obtain the following PRF whose domain is $\{0,1\}^n$:

$$F_{\text{LW}} := \hat{f}^{*n}\big((A_1,\ldots,A_n,h),\ (b_1\ldots b_n)\big) = (A_1^{b_1}\cdots A_n^{b_n})\cdot h\ \in \mathbb{G}^k$$

To show that F_{LW} is a secure PRF it suffices to show that f is parallel secure. Lewko-Waters do so implicitly in their proof. We state this in the following lemma.

Lemma 5. *If the k-linear assumption holds for the group $\mathbb{G}$, then the function f with parameter k defined in (7) is q-parallel secure for all q polynomial in the security parameter.*

Proof sketch. As in the proof of Lemma 4, it suffices to show that

$$G(A,h_1,\ldots,h_q) := (h_1,\ A\cdot h_1,\ldots,h_n,\ A\cdot h_n)$$

is a secure pseudorandom generator, assuming k-linear holds in $\mathbb{G}$. To prove this, one first shows that this G is a secure PRG when A is a random row vector in $\mathbb{Z}_p^k$. This uses the random self reduction of the k-linear problem described in [LW09]. Then one extends this to a $k\times k$ matrix using a hybrid argument over the k rows of the matrix A. Both ingredients are given in the Lewko-Waters proof of security. □

Combining Theorem 3 with Lemma 5 proves that the function F_{LW} with parameter k is a secure PRF whenever the k-linear assumption holds in $\mathbb{G}$.

5 A New Algebraic PRF

Our starting point is a secure PRF due to Dodis and Yampolskiy [DY05] with a domain of size ℓ for some small ℓ. The PRF is proven secure under the ℓ-DDH assumption. Recall that we use $[\ell]$ to denote the set $\{1,\ldots,\ell\}$ and consider the PRF $f : (\mathbb{Z}_p \times \mathbb{G}) \times [\ell] \to \mathbb{G}$ defined as follows:

$$f(\ \underbrace{(s,h)}_{\text{key}},\ x\ ) := h^{1/(s+x)} \tag{8}$$

As before we define $h^{1/0} = 1$. Dodis and Yampolskiy prove the following theorem.

Theorem 6 ([DY05])**.** *Suppose the ℓ-DDH assumption holds in $\mathbb{G}$. Then f is a secure PRF provided the domain size ℓ is polynomial in the security parameter.*

In particular, for every PRF adversary $\mathcal{A}$ there is an ℓ-DDH algorithm B such that

$$\textsf{PRF}_{adv}[\mathcal{A},f] = \textsf{DDH}^{(\ell)}_{adv}[\mathcal{B},\mathbb{G}] \quad \textit{and} \quad \textit{time}(\mathcal{B}) = \textit{time}(\mathcal{A}) + O(\ell\cdot T)$$

where T is the maximum time for exponentiation in $\mathbb{G}$.

Plugging f into the augmented cascade we obtain a PRF whose domain $[\ell]^n$ has exponential size. The resulting PRF is defined as follows:

$$F := \hat{f}^{*n}(\underbrace{(s_1,\ldots,s_n,h)}_{\text{key}},\ \underbrace{(x_1,\ldots,x_n)}_{\text{input}}) := h^{\left[1/\prod_{i=1}^{n}(s_i+x_i)\right]} \tag{9}$$

As discussed in the introduction, this PRF is more efficient than the Naor-Reingold PRF since it processes $\log_2 \ell$ bits per block rather than just one bit per block. The cost of this increased efficiency is reliance on a stronger assumption, namely ℓ-DDH.

Theorem 7. *The PRF defined in (9) is secure assuming the ℓ-DDH assumption holds in $\mathbb{G}$.*

To prove the theorem it suffices to show that f defined in (8) is parallel secure; namely that $f^{(q)}$ is a secure PRF for all polynomial q. We state this in the following lemma.

Lemma 8. *If the function f defined in (8) is a secure PRF and the DDH assumption holds in $\mathbb{G}$ then f is q-parallel secure for all q polynomial in the security parameter.*

In particular, for every PRF adversary $\mathcal{A}$ there are adversaries $\mathcal{B}_1$ and $\mathcal{B}_2$, whose running time is about the same as $\mathcal{A}$'s up to a polynomial factor, such that

$$\mathsf{PRF}_{adv}[\mathcal{A}, f^{(q)}] \le \mathsf{PRF}_{adv}[\mathcal{B}_1, f] + q \cdot \mathsf{DDH}_{adv}[\mathcal{B}_2, \mathbb{G}]$$

Note that the DDH assumption is implied by the k-DDH assumption and hence the DDH assumption used in Lemma 8 does not add an assumption beyond the one already used to prove that the underlying f is a secure PRF.

Proof of Lemma 8. Our goal is to show that $f^{(q)}$ is a secure PRF. We present the proof as a sequence of three games between a challenger and a PRF adversary $\mathcal{A}$ that attacks $f^{(q)}$. For $i = 0, 1, 2$, let W_i be the probability that $\mathcal{A}$ outputs 1 at the end of Game i.

Game 0. The challenger in this game behaves as a standard challenger presenting the adversary with an oracle for the pseudorandom function $f^{(q)}$ with a random key $(s, h_1, \ldots, h_q)$.

Game 1. The challenger in this game chooses a random function $u : [\ell] \to \mathbb{G}$. It also chooses random $r_1, \ldots, r_q$ in $\mathbb{Z}_p$. Now, given a query $(x, i) \in [\ell] \times [q]$ from the adversary, the challenger responds with $u(x)^{r_i}$.

We show that Games 0 and 1 are indistinguishable, assuming f is a secure PRF. In particular, there is a PRF adversary $\mathcal{B}_1$, whose running time is about the same as $\mathcal{A}$'s, such that

$$|W_0 - W_1| = \mathsf{PRF}_{\text{adv}}[\mathcal{B}_1, f] \tag{10}$$

Adversary $\mathcal{B}_1$ interacts with a PRF challenger for f and plays the role of an $f^{(q)}$ PRF challenger for $\mathcal{A}$. Adversary $\mathcal{B}_1$ works as follows:

choose random $r_1, \ldots, r_q$ in $\mathbb{Z}_p$.

given a query $(x, i) \in [\ell] \times [q]$ from $\mathcal{A}$ do:
  issue a query to $\mathcal{B}_1$'s challenger with input x and
    obtain y in response.
  respond on $\mathcal{A}$ with y^{r_i}.

finally, output whatever $\mathcal{A}$ outputs.

When $\mathcal{B}_1$'s challenger emulates an oracle for the function f with random key (s, h) it responds to query x with $y = h^{1/(s+x)}$. For $i = 1, \ldots, q$ define $h_i := h^{r_i}$. Then $\mathcal{B}_1$'s response to $\mathcal{A}$'s query for (x, i) is simply $h_i^{1/(s+x)}$ which is precisely $f^{(q)}(\ (s, h_1, \ldots, h_q),\ (x, i)\ )$. Hence, in this case $\mathcal{B}_1$ emulates a Game 0 challenger for $\mathcal{A}$.

When $\mathcal{B}_1$'s challenger emulates a random function $u : [\ell] \to \mathbb{G}$ then $\mathcal{B}_1$'s response to $\mathcal{A}$'s query for (x, i) is simply $u(x)^{r_i}$ which is precisely how a Game 1 challenger would respond. These two arguments prove (10), as required.

Game 2. The challenger presents the adversary with an oracle for a random function $w : [\ell] \times [q] \to \mathbb{G}$.

We use Lemma 1 to argue that Games 1 and 2 are indistinguishable assuming the DDH assumption holds in $\mathbb{G}$. In particular, there is a DDH algorithm $\mathcal{B}_2$ such that

$$|W_1 - W_2| \leq q \cdot \mathsf{DDH}_{\text{adv}}[\mathcal{B}_2, \mathbb{G}] \tag{11}$$

Let $(x_1, i_1), \ldots, (x_q, i_q)$ be $\mathcal{A}$'s queries to its challenger. Recall that in Game 1 the challenger responds to $\mathcal{A}$'s queries using a random function $u : X \to G$ and random $r_1, \ldots, r_q \in \mathbb{Z}_p$. Let $A \in \mathbb{Z}_p^{q \times q}$ be the matrix $A := (r_i s_j)_{1 \leq i,j \leq q}$. Clearly A has rank 1.

In Game 1 the adversary is given q entries in the matrix $g^A \in \mathbb{G}^{q \times q}$. In Game 2 the adversary is given q random values in $\mathbb{G}$ which we treat as q entries in a random $q \times q$ matrix in $\mathbb{G}$. By Lemma 1 there is an algorithm $\mathcal{B}_2$ that satisfies (11), as required.

Summary. Combining (10) and (11) shows that

$$\mathsf{PRF}_{\text{adv}}[\mathcal{A}, f^{(q)}] = |W_0 - W_2| \leq |W_0 - W_1| + |W_1 - W_2| \leq \mathsf{PRF}_{\text{adv}}[\mathcal{B}_1, f] + q \cdot \mathsf{DDH}_{\text{adv}}[\mathcal{B}_2, \mathbb{G}]$$

which completes the proof of the theorem. $\square$

The proof of Theorem 7 follows by combining Theorem 3 with Lemma 8, which shows that the function F with parameter ℓ is a secure PRF whenever the ℓ-DDH assumption holds.

6 Verifiable Random Functions

Verifable Random Functions, introduced by Micali, Rabin, and Vadhan [MRV99], are PRFs 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.

We give two VRF constructions from the augmented cascade. In this section, for a parameter ℓ, we use the Dodis-Yampolskiy small-domain VRF to construct VRFs for a domain of size ℓ^n for *constant* n. Security is based on the $n\ell$-BDH assumption in bilinear groups. In comparison, the core Dodis-Yampolskiy construction requires the ℓ^n-BDH assumption for a VRF on a domain of size ℓ^n.

In Section 7 we construct a VRF for a domain of size 2^m for arbitrary m from the $O(m)$-BDH assumption. Our construction makes use of admissible hash functions introduced in [BB04b].

Hohenberger and Waters [HW10] recently constructed an elegant large domain VRF from the Naor-Reingold PRF for a domain of size 2^m for arbitrary m. Security against a Q-query adversary relies on the $O(mQ)$-BDHE assumption, where t-BDHE is an assumption of the same flavor as the t-BDH assumption. While the efficiency of our VRF is worse than that of Hohenberger and Waters, the required complexity assumption is weaker: $O(m)$ vs. $O(mQ)$. The proof techiques for the two constructions are quite different. Hohenberger and Waters use the pile-up approach of Waters [Wat05] while we use admissible hash functions [BB04b].

Other VRFs include Abdalla et al. [ACF09] who give a construction using the m-wBDH assumption in blinear groups for a domain of size 2^m. The construction is limited to polynomial size domains since security degrades exponentially in m. Early VRFs outputting one bit were given by Lysyanskaya [Lys02] and Dodis [Dod03] based on stronger assumptions.

6.1 Definition of VRFs

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

- $\mathsf{Gen}(1^\lambda)$ outputs a pair of keys $(\mathsf{pk}, \mathsf{sk})$ for a security parameter λ.
- $\mathsf{Prove}(\mathsf{sk}, x)$ outputs $\big(F(\mathsf{sk}, x),\ \pi\big)$, where $\pi = \pi(\mathsf{sk}, x)$ is a proof of correctness.
- $\mathsf{Verify}(\mathsf{pk}, x, y, \pi)$ verifies that $y = F(\mathsf{sk}, x)$ using the proof π, and outputs 0 or 1 accordingly.

Security for a VRF is defined using two experiments, Exp_0 and Exp_1, that interact with an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$. For $b \in \{0, 1\}$, experiment Exp_b is defined as:

$$
\begin{aligned}
&(\mathsf{pk}, \mathsf{sk}) \xleftarrow{R} \mathsf{Gen}(1^\lambda)\\
&(x^*, \mathsf{state}) \xleftarrow{R} \mathcal{A}_1^{\mathcal{O}(\cdot)}(\mathsf{pk})\\
&y_0 \leftarrow F(\mathsf{sk}, x^*), \quad y_1 \xleftarrow{R} Y\\
&b' \xleftarrow{R} \mathcal{A}_2^{\mathcal{O}(\cdot)}(y_b,\ \mathsf{state})\\
&\text{output } b'
\end{aligned}
$$

where the oracle $\mathcal{O}(x)$, for $x \in X$, is defined as $\mathcal{O}(x) := \mathsf{Prove}(\mathsf{sk}, x)$. Moreover, $\mathcal{A}$ must never query $\mathcal{O}$ at x^*. For $b \in \{0, 1\}$ let W_b be the probability that $\mathcal{A}$ outputs 1 in Exp_b.
Define $\mathsf{VRF}_{\mathrm{adv}}[\mathcal{A}, F] := |W_0 - W_1|$.

Definition 6. *A VRF is said to be secure if it satisfies the following properties.*

1. **Pseudorandom:** *For every efficient adversary $\mathcal{A}$, $\mathsf{VRF}_{adv}[\mathcal{A}, F]$ is a negligible function of λ.*
2. **Correct:** *For all $x \in X$, if $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{R} \mathsf{Gen}(1^\lambda)$ and $(y, \pi) \xleftarrow{R} \mathsf{Prove}(\mathsf{sk}, x)$ then* $\Pr\big[\mathsf{Verify}(\mathsf{pk}, x, y, \pi) = 1\big] = 1$.
3. **Unique:** *no values of $(\mathsf{pk}, x, y_1, y_2, \pi_1, \pi_2)$ satisfy $\mathsf{Verify}(\mathsf{pk}, x, y_1, \pi_1) = \mathsf{Verify}(\mathsf{pk}, x, y_2, \pi_2) = 1$ for $y_1 \neq y_2$.*

6.2 Building a VRF using the augmented cascade

We construct a secure VRF with domain of size ℓ^n using the $n\ell$-BDH assumption and the augmented cascade. Evaluating the VRF takes n multiplications and one exponentiation. Our VRF is built from the augmented cascade using the Dodis-Yampolskiy VRF as the underlying function. The Dodis-Yampolskiy VRF uses pairings and outputs elements in $\mathbb{G}_\mathrm{T}$ while its key uses elemets in $\mathbb{G}$. We therefore need to slightly tweak the augmented cascade to compensate for the difference between $\mathbb{G}_\mathrm{T}$ and $\mathbb{G}$, but this is easily done.

The VRF is parameterized by two positive integers n and ℓ, has domain $[\ell]^n$, and is defined as follows:

Algorithm $\mathsf{Gen}(1^\lambda)$: Fix a group $\mathbb{G}$ of prime order p with a bilinear pairing. Choose random generators $g, u \in \mathbb{G}$ and random values $s_1, s_2, \ldots, s_n \stackrel{\mathrm{R}}{\leftarrow} \mathbb{Z}_p$. Set $t_i := g^{s_i}$ for $i = 1, \ldots, n$ and output the keys

$$\mathsf{pk} = (g, u, t_1, \ldots, t_n), \quad \mathsf{sk} = (g, u, s_1, \ldots, s_n).$$

Function $F : (\mathbb{G}^2 \times \mathbb{Z}_p^n) \times [\ell]^n \to \mathbb{G}_{\mathrm{T}}$. On input sk and $\mathbf{x} = (x_1, \ldots, x_n) \in [\ell]^n$ output

$$F(\mathsf{sk}, \mathbf{x}) := e(\, g^{[1/\prod_{i=1}^n (x_i + s_i)]},\ u)\ .$$

Algorithm $\mathsf{Prove}(\mathsf{sk}, \mathbf{x})$: This algorithm outputs $F(\mathsf{sk}, \mathbf{x})$ along with a proof π as follows: for $i = 1$ to n, compute $\pi_i = g^{[1/\prod_{j=1}^i (x_j + s_j)]} \in \mathbb{G}$ and output the proof $\pi := (\pi_1, \pi_2, \ldots, \pi_n) \in \mathbb{G}^n$.

Algorithm $\mathsf{Verify}(\mathsf{pk}, \mathbf{x}, y, \pi)$: First verify that the proof π contains legal encodings of elements in $\mathbb{G}$. Next, check that

$$e(\pi_i,\ g^{x_i}\, t_i) = e(\pi_{i-1}, g) \quad \text{for } i = 1, \ldots, n,$$

where $\pi_0 := g$. Finally, check that $e(\pi_n, u) = y$, where y is the output of the VRF. Verify returns 1 iff all the checks are satisfied.

6.3 Proof of VRF security

We prove security for a polynomial size domain. For a domain of size ℓ^n we use the $n\ell$-BDH assumption.

Theorem 9. *Let $\mathbb{G}$ be a bilinear group of order p, and let n, ℓ be positive integers with $2 \le \ell < p$. If the $n\ell$-BDH assumption holds in $\mathbb{G}$, and ℓ^n is polynomial in the security parameter, then the VRF defined in Section 6.2 is secure. In particular, for every VRF adversary $\mathcal{A}$ there is a $n\ell$-BDH algorithm $\mathcal{B}$, whose running time is about the same as $\mathcal{A}$'s, such that*

$$\mathit{VRF}_{adv}[\mathcal{A}, F] \le \ell^n \cdot \mathit{BDH}_{adv}^{(n\ell)}[\mathcal{B}, \mathbb{G}].$$

Proof. Correctness of the VRF is straightforward. *Uniqueness* follows from the group structure: for any input there is only one group element in $\mathbb{G}$ that is a valid output, and moreover, it is not possible (even for an unbounded adversary) to devise a valid proof for another element. It remains to prove *pseudorandomness*.

Intuition. $\mathcal{B}$ chooses a random $\mathbf{b}^* = (b_1^*, \ldots, b_n^*) \in [\ell]^n$. Consider the $n \times \ell$ matrix where the $(i, j)^{\text{th}}$ entry holds some polynomial in $\mathbb{Z}_p[z]$. A query $\mathbf{b} \in [\ell]^n$ from the adversary defines a path through this matrix that visits exactly one cell in every row (corresponding to each coordinate of $\mathbf{b}$). The random vector $\mathbf{b}^*$ defines n special cells called "mines," one mine per row. Then $\mathcal{B}$ constructs a public key that lets it answer all queries from the adversary that do not visit *all* n mines. If the adversary's challenge query hits each and every mine — which happens with probability ℓ^{-n} — then $\mathcal{B}$ can use the adversary to solve the given $n\ell$-BDH instance. We now formalize this intuition.

Let $\mathcal{A}$ be a VRF adversary attacking F. We construct the following algorithm $\mathcal{B}$ that breaks the $n\ell$-BDH assumption in $\mathbb{G}$ with advantage $\mathsf{VRF}_{\text{adv}}[\mathcal{A}, F]/\ell^n$.

Input: Algorithm $\mathcal{B}$ is given a tuple $(g, u, g^x, \ldots, g^{(x^{n\ell})}, y) \in \mathbb{G}^{n\ell+2} \times \mathbb{G}_{\mathrm{T}}$ and is to determine if y is $e(g, u)^{1/x}$ or drawn randomly from $\mathbb{G}_{\mathrm{T}}$.

Key generation: Algorithm $\mathcal{B}$ begins by choosing a random $\mathbf{b}^* = (b_1^*, \ldots, b_n^*) \xleftarrow{R} [\ell]^n$ and random $r_1, \ldots, r_n \xleftarrow{R} \mathbb{Z}_p^*$. It constructs an instance of the VRF as follows: First, $\mathcal{B}$ constructs the polynomials

$$p_i(z) = \prod_{a \in [\ell]} (r_i z + a - b_i^*) \qquad \text{and} \qquad p(z) = z^{-1} \prod_{i=1}^{n} p_i(z)$$

in $\mathbb{Z}_p[z]$. Observe that every $p_i(z)$ is divisible by $r_i z$, and therefore the product of all $p_i(z)$ is divisible by z^n. Hence, $p(z)$ is in $\mathbb{Z}_p[z]$ and is divisible by z^{n-1}. Moreover, $p(z)$ is not divisible by z^n. Write

$$p(z) = \sum_{j=0}^{n\ell-1} c_j \cdot z^j$$

for some c_j in $\mathbb{Z}_p$. $\mathcal{B}$ then constructs the generator h as:

$$h = g^{p(x)} = \prod_{j=0}^{n\ell-1} \left(g^{(x^j)^{c_j}} \right) \in \mathbb{G}.$$

Next, for $i = 1, \ldots, n$, algorithm $\mathcal{B}$ constructs the public key values

$$t_i = h^{(r_i x - b_i^*)} = g^{p(x)(r_i x - b_i^*)} = \prod_{j=0}^{n\ell} \left(g^{(x^j)} \right)^{d_j}$$

where $p(z)(r_i z - b_i^*) = \sum_{j=0}^{n\ell} d_j z^j$. It sends the public key $\mathsf{pk} = (h, u, t_1, \ldots, t_n)$ to $\mathcal{A}$. The secret key values $s_1, \ldots, s_n \in \mathbb{Z}_p$ corresponding to this public key (and are unknown to $\mathcal{B}$) are:

$$s_i := r_i x - b_i^* \quad \text{for } i = 1, \ldots, n.$$

These values are statistically close to uniform in $\mathbb{Z}_p$ thanks to the random choice of $r_1, \ldots, r_n$ in $\mathbb{Z}_p^*$. Hence pk is indistinguishable from a random public key in the real scheme.

Responding to Oracle Queries: Consider a query from $\mathcal{A}$ for some input $\mathbf{b} = (b_1, \ldots, b_n) \in [\ell]^n$. If $\mathbf{b} = \mathbf{b}^*$, algorithm $\mathcal{B}$ aborts the simulation and outputs $\perp$. We show that when $\mathbf{b} \neq \mathbf{b}^*$, our $\mathcal{B}$ can successfully answer the query. First $\mathcal{B}$ constructs n polynomials, $p^{(1)}, \ldots, p^{(n)} \in \mathbb{Z}_p[z]$ as

$$p^{(j)}(z) = \frac{p(z)}{(r_1 z + b_1 - b_1^*) \cdots (r_j z + b_j - b_j^*)} = \sum_{k=0}^{n\ell-j-1} d_{j,k} \cdot z^k \quad \in \mathbb{Z}_p[z] \tag{12}$$

for some constants $d_{j,k} \in \mathbb{Z}_p$, $j = 1, \ldots, n$. This $p^{(j)}(z)$ is a polynomial in $\mathbb{Z}_p[z]$ because $p(z)$ is divisible by the denominator in (12), unless the denominator is a multiple of z^n, which only happens when $\mathbf{b} = \mathbf{b}^*$.

Now, for $j = 1, \ldots, n$ our $\mathcal{B}$ computes

$$\pi_j = h^{1/\prod_{k=1}^{j}(s_k + b_k)} = g^{p^{(j)}(x)} = \prod_{k=0}^{n\ell-j-1} \left(g^{(x^j)} \right)^{d_{j,k}} \in \mathbb{G}.$$

Let $\pi := (\pi_1, \ldots, \pi_n)$. Observe that

$$e(\pi_n, u) = e\left(h^{1/\left[\prod_{k=1}^{n}(s_k + b_k)\right]},\ u\right) = F(\mathsf{sk}, \mathbf{b})$$

and hence $e(\pi_n, u)$ is the value of the function $F(\mathsf{sk}, \cdot)$ at the input $\mathbf{b}$. It sends to $\mathcal{A}$ the response $\big(e(\pi_n, u),\ \pi\big)$.

Challenge: Eventually, $\mathcal{A}$ outputs an input $\hat{\mathbf{b}} \in [\ell]^n$ on which it wants to be challenged. If $\hat{\mathbf{b}} \neq \mathbf{b}^*$, then $\mathcal{B}$ aborts and outputs $\perp$. If $\hat{\mathbf{b}} = \mathbf{b}^*$, then $\mathcal{B}$ proceeds as follows.

Since $\mathcal{A}$ is a VRF adversary, it can distinguish between

$$F(\mathsf{sk}, \mathbf{b}^*) = e(h, u)^{1/\prod_{i=1}^{n}(s_i + b_i^*)} = e(h, u)^{1/x^n \prod_{i=1}^{n} r_i} \in \mathbb{G}_{\mathrm{T}}$$

and a random element in $\mathbb{G}_{\mathrm{T}}$ with advantage $\mathsf{VRF}_{\mathrm{adv}}[\mathcal{A}, F]$. Now, recall that $p(z)$ is divisible by z^{n-1} but not by z^n. Therefore, there are scalars $\rho \neq 0$ and $\rho_0, \ldots, \rho_{n\ell-n-1}$ in $\mathbb{Z}_p$ such that:

$$s(z) := \frac{p(z)}{z^n \prod_{i=1}^{n} r_i} = \frac{1}{z} \cdot \underbrace{\left[\frac{p(z)}{z^{n-1} \prod_{i=1}^{n} r_i}\right]}_{\text{in } \mathbb{Z}_p[z]} = \frac{\rho}{z} + \sum_{j=0}^{n\ell-n-1} \rho_j z^j \ .$$

Now, using the challenge $y \in \mathbb{G}_{\mathrm{T}}$, algorithm $\mathcal{B}$ computes:

$$y^* := y^{\rho} \cdot \prod_{j=0}^{n\ell-n-1} e\left(g^{(x^j)})^{\rho_j},\ u\right) \in \mathbb{G}_{\mathrm{T}}.$$

If $y \xleftarrow{\mathrm{R}} \mathbb{G}_{\mathrm{T}}$, then y^* is random in $\mathbb{G}_{\mathrm{T}}$ because $\rho \neq 0$. However, if $y = e(g, u)^{1/x}$ then y^* satisfies

$$y^* = e(g, u)^{s(x)} = \ e(h, u)^{1/x^n \prod_{i=1}^{n} r_i} = F(\mathsf{sk}, \mathbf{b}^*).$$

Now $\mathcal{B}$ responds to $\mathcal{A}$'s challenge query with y^*.

Guess: If needed, algorithm $\mathcal{A}$ makes more queries at inputs different from $\mathbf{b}^*$, to which $\mathcal{B}$ responds as before. Finally, $\mathcal{A}$ outputs a guess bit $b' \in \{0, 1\}$. $\mathcal{B}$ outputs b' as its guess.

Success probability. The running time of $\mathcal{B}$ is dominated by responding to oracle queries. Its running time is at most a polynomial factor beyond the running time of $\mathcal{A}$.

If $\mathcal{A}$'s challenge query is $\mathbf{b}^*$, then $\mathcal{B}$ solves the given $n\ell$-BDH challenge with the same advantage as adversary $\mathcal{A}$ has against the VRF. Moreover, a response to an adversary oracle query for $\mathbf{b} \in [\ell]^n$ reveals nothing about $\mathbf{b}^*$ other than $\mathbf{b} \neq \mathbf{b}^*$. Then, a standard argument shows that $\mathcal{A}$'s challenge query is $\mathbf{b}^*$ with probability ℓ^{-n}. Hence,

$$\mathsf{BDH}_{\mathrm{adv}}^{(n\ell)}[\mathcal{B}, \mathbb{G}] \geq \mathsf{VRF}_{\mathrm{adv}}[\mathcal{A}, F]/\ell^n$$

which completes the proof of the theorem. □

7 VRFs with large input domains

In this section, we show how to construct a secure VRF with an input domain of $\{0, 1\}^m$ for arbitrary m. Security depends on the $O(m)$-BDH assumption. Evaluating the VRF requires about $O(m)$ multiplications and one exponentiation. Our construction uses error correcting codes with a large minimum distance, also known as low rate codes.

Definition 7. *A $(m, n, d)_\ell$-error correcting code is an injective function $H : \{0,1\}^m \to [\ell]^n$ such that for all distinct $c_1, c_2 \in \{0,1\}^m$ we have:*

$$\mathrm{HDist}\left(H(c_1), H(c_2)\right) \geq d,$$

where $\mathrm{HDist}(\cdot,\cdot)$ denotes the hamming distance between the codewords (the number of coordinates where the two codewords differ). We say that an error-correcting code is efficient *if the function H is efficiently computable.*

Let $\mathrm{Bin}(n, p)$ be a Binomial random variable with parameters n and p. The Gilbert-Varshamov bound proves the existence of codes with a large minimum distance d, and a positive rate, as long as $d < n(1 - 1/\ell)$.

Lemma 10. *For all integers $n, \ell \geq 2$, and $0 \leq d < n(1 - 1/\ell)$, there exists a $(m, n, d)_\ell$-error correcting code with $m \geq -\log_2(\Pr\left[Bin(n, 1/\ell) \geq n - d\right])$.*

The following corollary states the existence of large minimum distance code more explicitly.

Corollary 11. *Let $\ell \geq 2$ be an integer and $\epsilon \in (\frac{1}{\ell}, 1)$. Then there is a constant $c = c(\epsilon)$ such that for all $m \geq 1$ and $n \geq cm$, there is a $(m, n, d)_\ell$-error correcting code with $d \geq n(1 - \epsilon)$.*

We will use the fact that $n = O(m)$ is sufficient for the existence of a $(m, n, d)_\ell$-error correcting code with $d \geq n(1 - \epsilon)$. Concretely, it suffices to use a VRF with a 256-bit domain so that a large input can be hashed with SHA256 into this domain. As an example instantiation of Lemma 10, the Gilbert-Varshamov bound shows that there is a $(256, n, d)_{128}$-error correcting code with $n = 1024$ and $d \geq 0.9n$. Other examples are listed in Table 1.

We will need the function H defining the error-correcting code to be efficiently computable. We can do so using an appropriate pseudorandom function, or by using an explicit low rate error-correcting code from [ABN+92].

7.1 VRF construction

We now describe a VRF with input domain $\{0,1\}^m$. The construction uses an efficient $(m, n, d)_\ell$ error correcting code $H : \{0,1\}^m \to [\ell]^n$ with a minimum distance $d \geq n(1 - \epsilon)$, for some parameters n, ℓ, and $\epsilon \in (\frac{1}{\ell}, 1)$. The value of ϵ will affect the tightness of the security reduction. The smaller ϵ is the tighter the reduction. Recall that $n = O(m)$.

Algorithm $\mathsf{Gen}(1^\lambda)$: Fix a group $\mathbb{G}$ of prime order p with a bilinear pairing. Select random generators $g, u \in \mathbb{G}$, random values $s_1, s_2, \ldots, s_n \in \mathbb{Z}_p$, and set $t_i = g^{s_i}$. Output the keys:

$$\mathsf{pk} = (g, u, t_1, \ldots, t_n), \quad \mathsf{sk} = (g, u, s_1, \ldots, s_n).$$

Function $F : (\mathbb{G}^2 \times \mathbb{Z}_p^n) \times \{0,1\}^m \to \mathbb{G}_\mathrm{T}$. On input sk and $x \in \{0,1\}^m$, output:

$$F(\mathsf{sk}, x) := e\left(g^{\left[1/\prod_{i=1}^{n}(H(x)_i + s_i)\right]},\ u\right)$$

where $H(x)_i$ refers to the i^{th} coordinate of $H(x) \in [\ell]^n$.

Algorithm $\mathsf{Prove}(\mathsf{sk}, x)$: On input sk and x, output $F(\mathsf{sk}, x)$ along with a proof π as follows. For $i = 1, \ldots, n$ compute $\pi_i = g^{[1/\prod_{j=1}^{i}(H(x)_j + s_j)]}$. Output the proof:

$$\pi := (\pi_1, \pi_2, \ldots, \pi_n) \quad \in \mathbb{G}^n\ .$$

ℓ	ϵ	$n_{\min}$	n	w	multiplier of $\mathsf{BDH}^{(n\ell)}_{\text{adv}}[\mathcal{B},\mathbb{G}]$ in (13) (smaller is better)
128	0.1	1022	1024	46	$2^{19}Q$
256	0.1	732	768	31	$2^{25}Q$
256	0.05	2081	2112	44	$2^{12}Q$

Table 1: Concrete bounds for the tightness of the reduction in (13) for different values of n, ℓ and ϵ, using $m = 256$ and $Q = 2^{48}$. The scheme uses a $(m, n, d)_\ell$ error correcting code with $d \geq n(1-\epsilon)$. The $n_{\min}$ column is the smallest n for which such a code exists by the Gilbert-Varshamov bound (Lemma 10). The n column is the value of n used to compute the right most column. The w column refers to a parameter used in the proof of Theorem 12.

Algorithm $\mathsf{Verify}(\mathsf{pk}, x, y, \pi)$: First verify that π contains legal encodings of elements in $\mathbb{G}$. Next, check that:

$$e(\pi_i,\ g^{H(x)_i} \cdot t_i) = e(\pi_{i-1}, g),$$

where $\pi_0 := g$. Finally, check that $e(\pi_n, u) = y$, where y is the output of the VRF. Verify returns 1 iff all the checks are true.

Security. The scheme above is the same as the scheme in the previous section, where the input x to the function is replaced by $H(x) \in [\ell]^n$. Hence, correctness and uniqueness follow in the same way. We next prove that it is pseudorandom.

Theorem 12. *Let $\epsilon \in (\frac{1}{\ell}, 1)$ and let Q and $\ell \geq 2$ be integers. Then there are constants $c = c(\ell, \epsilon, Q)$, $m_0 = m_0(\ell, \epsilon, Q)$, and $\tau = \tau(\epsilon) \geq 1$, such that for all $m \geq m_0$ and $n \geq cm$,*

- *there is an $(m, n, d)_\ell$-error correcting code $H : \{0,1\}^m \to [\ell]^n$ with $d > n(1-\epsilon)$, and*
- *the VRF constructed in Section 7.1 is secure under the $n\ell$-BDH assumption. In particular, for every VRF adversary $\mathcal{A}$ that makes at most Q queries, there is an $n\ell$-BDH algorithm $\mathcal{B}$ such that*

$$\mathsf{VRF}_{adv}[\mathcal{A}, F] \leq (Qn)^\tau \cdot \mathsf{BDH}^{(n\ell)}_{adv}[\mathcal{B}, \mathbb{G}]. \tag{13}$$

Our security proof introduces a factor of $(Qn)^\tau$ to the success probability of breaking the hardness assumption. The smaller ϵ is, the smaller τ becomes, but τ is always greater than 1. A factor of at least Q is necessary to prove the security of any VRF (or a unique signature scheme) with an exponential size domain based on a non-interactive assumption, as shown in [BJLS16, HJK12, Cor02].

Table 1 gives concrete bounds for the tightness of the reduction in (13) for different values of n, ℓ and ϵ, using $m = 256$ and $Q = 2^{48}$. The quantity in the right most column replaces the multiplier $(Qn)^\tau$ in (13). The table is calculated using expressions derived in Appendix A.

Proof intuition. Recall the intuition behind the proof of Theorem 9. In the proof of Theorem 9 algorithm $\mathcal{B}$ chose n random cells, called mines, in an $n \times \ell$ matrix, one mine per row. $\mathcal{B}$ then constructed public parameters that enable it to answer any query that does not visit all n mines. We showed that an adversary $\mathcal{A}$ whose challenge query visits all n mines can be used to solve the given $n\ell$-BDH instance. Since the fraction of challenge queries that do not cause the adversary to abort is exponentially small in n (i.e. ℓ^{-n}), this proof technique works only for small n.

In this proof, algorithm $\mathcal{B}$ first sets a parameter w that is close to n/ℓ. It then lays n random mines in an $n \times \ell$ matrix, as before, one mine per row. However, here $\mathcal{B}$ constructs public parameters that let it answer any query from $\mathcal{A}$ that visits fewer than w mines (instead of fewer than n mines). This means that it can answer fewer adversary queries than before. However, an adversary whose challenge query visits exactly w (rather than all n) mines, can be used to solve the given $n\ell$-BDH instance. This means that $\mathcal{B}$ is more likely to be able to use the adversary's challenge query compared to before. Since non of $\mathcal{A}$'s queries should visit w or more mines, w is chosen so that this condition holds with probability about $(1/Q)$. The purpose of the code H is to mitigate against an adversary $\mathcal{A}$ that forces $\mathcal{B}$ to abort by constructing highly correlated queries.

Proof. Let $\mathcal{A}$ be an adversary that distinguishes the VRF from a random function with non-negligible probability. We construct an algorithm $\mathcal{B}$ that solves the $n\ell$-BDH instance.

Input: Algorithm $\mathcal{B}$ is given a tuple $(g, u, g^x, \ldots, g^{(x^{n\ell})}, y) \in \mathbb{G}^{n\ell+2} \times \mathbb{G}_{\mathrm{T}}$ and needs to determine if y is $e(g,u)^{1/x}$ or y is random in $\mathbb{G}_{\mathrm{T}}$.

Key generation: $\mathcal{B}$ sets $w := \lfloor n/\ell + \Delta\sqrt{n/\ell}\,\rfloor$, where $\Delta := \frac{3}{1-\epsilon}\sqrt{\ln(2Qn)}$. It then constructs the VRF parameters as follows. $\mathcal{B}$ begins by choosing a random $\mathbf{b}^* = (b_1^*, \ldots, b_n^*) \xleftarrow{\mathrm{R}} [\ell]^n$, and random $r_1, \ldots, r_n \xleftarrow{\mathrm{R}} \mathbb{Z}_p^*$. It constructs the VRF public parameters by first constructing the polynomials

$$p_i(z) = \prod_{a\in[\ell]} (r_i z + a - b_i^*) \qquad \text{and} \qquad p(z) = z^{-(n-w+1)} \prod_{i=1}^{n} p_i(z) = \sum_{j=0}^{n\ell-(n-w+1)} c_j \cdot z^j,$$

for some coefficients c_j in $\mathbb{Z}_p$. As in the proof of Theorem 9, the product of all the $p_i(z)$ is divisible by z^n. Therefore $p(z)$ is in $\mathbb{Z}[z]$, and is divisible by z^{w-1}, but is not divisible by z^w. This will ensure that $\mathcal{B}$ cannot answer a query that matches $\mathbf{b}^*$ at w or more positions. $\mathcal{B}$ then computes the VRF public parameters h and $t_1, \ldots, t_n$ as in the proof of Theorem 9. For $i = 1, \ldots, n$:

$$h := g^{p(x)} = \prod_{j=0}^{n\ell-(n-w+1)} \left(g^{(x^j)^{c_j}}\right) \in \mathbb{G}$$

$$t_i := h^{(r_i x - b_i^*)} = g^{p(x)(r_i x - b_i^*)} = \prod_{j=0}^{n\ell-n+w} \left(g^{(x^j)}\right)^{d_j} \in \mathbb{G} \qquad \text{where} \quad \sum_{j=0}^{n\ell-n+w} d_j z^j = p(z)(r_i z - b_i^*).$$

It sends the public key $\mathsf{pk} = (h, u, t_1, \ldots, t_n)$ to $\mathcal{A}$. The secret keys $s_1, \ldots, s_n \in \mathbb{Z}_p$ (unknown to $\mathcal{B}$) that correspond to pk are $s_i := r_i x - b_i^* \in \mathbb{Z}_p$, for $i = 1, \ldots, n$.

Responding to Oracle Queries: $\mathcal{B}$ responds to queries in a manner almost identical to the one in the proof of Theorem 9. Let $q \in \{0,1\}^m$ be a query from $\mathcal{A}$. Our $\mathcal{B}$ first computes $\mathbf{b} := H(q)$. If $\mathrm{HDist}(\mathbf{b}, \mathbf{b}^*) \leq (n-w)$, meaning that the two vectors agree on w or more coordinates, then $\mathcal{B}$ cannot answer the query. In this case $\mathcal{B}$ aborts the simulation and outputs a random bit in $\{0,1\}$. Otherwise, $\mathcal{B}$ evaluates the function and responds to $\mathcal{A}$ in a manner identical to the one in the proof of Theorem 9.

Challenge: Eventually, $\mathcal{A}$ outputs a point $q^* \in \{0,1\}^m$ on which it wants to be challenged.

If $\mathrm{HDist}(H(q^*), \mathbf{b}^*) \neq (n-w)$, $\mathcal{B}$ aborts the simulation and outputs a random bit as its guess.

Otherwise, $\mathrm{HDist}(H(q^*), \mathbf{b}^*) = (n - w)$. Let $S \subseteq [n]$ be the set of w coordinates where $H(q^*)$ and $\mathbf{b}^*$ match. Define

$$v(z) := \prod_{i \in S} r_i \cdot \left[\prod_{i \notin S} (r_i z - \mathbf{b}^*{}_i + H(q^*)_i) \right] \in \mathbb{Z}_p[z]$$

of degree $(n - w)$. Note that $v(z)$ is not divisible by z. Then since $\mathcal{A}$ is a VRF adversary, it can distinguish between

$$F(\mathsf{sk}, q^*) = e(h,u)^{1/\left[\prod_{i=1}^{n}(s_i + H(q^*)_i)\right]} = e(h,u)^{1/(x^w \cdot v(x))}$$

and a random element in $\mathbb{G}_{\mathrm{T}}$ with advantage $\mathsf{VRF}_{\mathsf{adv}}[\mathcal{A}, F]$.

Recall that $p(z)$ is divisible by z^{w-1} but not by z^w. Thus, there are scalars $\rho \neq 0$ and $\rho_0, \ldots, \rho_{n\ell - 2n + 2w - 1}$ in $\mathbb{Z}_p$ such that:

$$s(z) := \frac{p(z)}{z^w \cdot v(z)} = \frac{1}{z} \cdot \underbrace{\left[\frac{p(z)}{z^{w-1} \cdot v(z)} \right]}_{\text{in } \mathbb{Z}_p[z]} = \frac{\rho}{z} + \sum_{j=0}^{n\ell - 2n + 2w - 1} \rho_j z^j \quad \in \mathbb{Z}_p[z].$$

Now, using the BDH challenge y, algorithm $\mathcal{B}$ computes:

$$y^* = y^\rho \cdot \prod_{j=0}^{n\ell - 2n + 2w - 1} e\left((g^{(x^j)})^{\rho_j},\ u \right) \in \mathbb{G}_{\mathrm{T}}.$$

Now, if $y \overset{\mathrm{R}}{\leftarrow} \mathbb{G}_{\mathrm{T}}$, then y^* is distributed randomly in $\mathbb{G}_{\mathrm{T}}$ because $\rho \neq 0$. However, if $y = e(g,u)^{1/x}$, then

$$y^* = e(g,u)^{s(x)} = e(g,u)^{p(x)/(x^w \cdot v(x))} = e(h,u)^{1/(x^w \cdot v(x))} = F(\mathsf{sk}, q^*).$$

Now, $\mathcal{B}$ responds to $\mathcal{A}$ with the value y^* as computed above.

Guess: If needed, algorithm $\mathcal{A}$ makes more queries at inputs different from $\mathbf{b}^*$, to which $\mathcal{B}$ responds as before. Finally, $\mathcal{A}$ outputs a guess bit $b' \in \{0, 1\}$. $\mathcal{B}$ outputs this b' as its guess.

Success Probability. Let $\mathbf{q} = (q_1, \ldots, q_Q, q^*)$ be the tuple of $Q+1$ queries from $\mathcal{A}$. To bound the probability that $\mathcal{B}$ aborts during the simulation, we can delay the choice of $\mathbf{b}^* = (b_1^*, \ldots, b_n^*)$ in $[\ell]^n$ to the end of the simulation, and analyze the probability that a random $\mathbf{b}^*$ would have caused an abort given the observed query tuple $\mathbf{q}$ from the adversary.

Define the abort indicator function:

$$\mathsf{abort}(\mathbf{q}, \mathbf{b}^*) = \begin{cases} 1 & \text{if } \mathrm{HDist}(H(q^*), \mathbf{b}^*) \neq n - w \ \ \bigvee_{i=1}^{Q} \left(\mathrm{HDist}(H(q_i), \mathbf{b}^*) \leq (n - w) \right); \\ 0 & \text{otherwise.} \end{cases}$$

The function $\mathsf{abort}(\mathbf{q}, \mathbf{b}^*)$ evaluates to 0 if the queries $\mathbf{q}$ will not cause a regular abort for the given choice of $\mathbf{b}^*$. Define the success probability over all possible choices of $\mathbf{b}^* \in [\ell]^n$ as

$$\zeta(\mathbf{q}) := \Pr_{\mathbf{b}^*}\big[\mathsf{abort}(\mathbf{q}, \mathbf{b}^*) = 0\big].$$

Let $\zeta_{\min} := \min_{\mathbf{q}}[\zeta(\mathbf{q})]$. We prove the following bound on $\zeta_{\min}$.

Lemma 13. *Under the assumptions of Theorem 12, for all query tuples $\mathbf{q}$ we have $\zeta(\mathbf{q}) \geq \zeta_{\min} \geq (1/Qn)^\tau$ for some constant $\tau = \tau(\epsilon)$.*

Proof. The proof is given in Appendix A. ◻

If $\mathcal{B}$ does not abort, then $\mathcal{A}$'s final output lets $\mathcal{B}$ solve the $n\ell$-BDH problem with the same advantage as $\mathcal{A}$. However, to complete the analysis we need to introduce a standard artificial abort condition, as in [Wat05, HW10]. This is needed to ensure that the probability that $\mathcal{B}$ does not abort is about the same for every choice of adversary query tuple $\mathbf{q}$. We refer to [Wat05, HW10] for the details.

The end result is that

$$\mathsf{BDH}_{\text{adv}}^{(n\ell)}[\mathcal{B}, \mathbb{G}] \geq \zeta_{\min} \cdot \mathsf{VRF}_{\text{adv}}[\mathcal{A}, F] \geq (1/Qn)^{\tau} \cdot \mathsf{VRF}_{\text{adv}}[\mathcal{A}, F]$$

which completes the proof of Theorem 12. ◻

7.2 Simulatable VRFs

Chase and Lysyanskaya [CL07] introduced *simulatable* VRFs (sVRF), which they used to convert single-theorem non-interactive zero knowledge (NIZK) to many-theorem NIZK. Their simulatable VRF, secure under the k-BDH assumption and the subgroup decision assumption (SDA), has a polynomial size domain. We briefly outline how the augmented cascade gives a large-domain sVRF using the same assumptions.

Stated informally, Chase and Lysyanskaya show that by modifying the proof π of the Dodis-Yampolskiy VRF, there exist algorithms $(\mathsf{SimG}, \mathsf{SimSample}, \mathsf{SimProve})$ (analogous to $(\mathsf{Gen}, F, \mathsf{Prove})$ in the definition of VRFs) and a way to simulate parameters $\mathsf{SimParam}$ with the following properties:

1. $\mathsf{SimSample}$, using the parameters output by $\mathsf{SimParams}$, produces a random distribution (that is indistinguishable from the distribution of the outputs of F, since F is a sVRF).
2. $\mathsf{SimProve}$ is able to simulate proofs for these random outputs that are indistinguishable from proofs produced by Prove, and any adversary that is able to distinguish between the simulated proofs and real proofs can be used to break SDA.

The augmented cascade theorem generalizes to sVRFs and can be used to construct large-domain sVRFs from small-domain ones, provided the underlying sVRF has parallel-security. The simulatability of the sVRF makes it possible to push the hybrid proof of the augmented cascade (Theorem 3) to the settings of sVRFs. We note that this was not possible for VRFs since the simulator cannot provide proofs in the hybrid experiments. Now, plugging the Chase-Lysyanskaya sVRF into this augmented cascade, we obtain a large-domain sVRF.

8 Conclusions

We presented a generalization of the cascade construction called the augmented cascade. We used the augmented cascade to construct large-domain PRFs from small-domain algebraic PRFs. The augmented cascade provides a unified framework for analysing the constructions of Naor-Reingold and Lewko-Waters. We used the augmented cascade to extend the Dodis-Yampolskiy PRF to a PRF on large domains, resulting in the most efficient algebraic PRF to date.

The new large-domain PRF can be converted into a large-domain VRF in a bilinear group and proven secure based on the m-BDH assumption for some parameter m that depends on the domain size. For small domains the resulting VRF uses a weaker assumption than its Dodis-Yampolskiy origin. We obtain an efficient large domain VRF using error correcting codes. The algebraic structure of these constructions will likely find many applications, as was the case for the Naor-Reingold PRF. As an example, we briefly noted a simulatable-VRF for large domains.

Acknowledgments

We thank David Niehues for a helpful discussion of Section 7.

References

[ABN+92] Noga Alon, Jehoshua Bruck, Joseph Naor, Moni Naor, and Ron M. Roth. Construction of asymptotically good low-rate error-correcting codes through pseudo-random graphs. *IEEE Transactions on Information Theory*, 38(2):509–, 1992.

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

[ASM07] Man Ho Au, Willy Susilo, and Yi Mu. Practical compact e-cash. In *proc. of ACISP'07*, pages 431–445, 2007.

[BB04a] Dan Boneh and Xavier Boyen. Efficient selective-ID identity based encryption without random oracles. In *Advances in Cryptology—EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 223–38. Springer-Verlag, 2004.

[BB04b] Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In Matt Franklin, editor, *Advances in Cryptology—CRYPTO 2004*, volume 3152 of *LNCS*, pages 443–59. Springer-Verlag, 2004.

[BB04c] 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.

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In *Advances in Cryptology—EUROCRYPT 2005*, volume 3494 of *LNCS*. Springer-Verlag, 2005.

[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.

[BCK96a] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In *CRYPTO'96*, pages 1–15, 1996.

[BCK96b] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Pseudorandom functions revisited: The cascade construction and its concrete security. In *FOCS'96*, 1996.

[BCKL09] Mira Belenkiy, Melissa Chase, Markulf Kohlweiss, and Anna Lysyanskaya. Compact e-cash and simulatable vrfs revisited. In *Pairing'09*, pages 114–131, 2009.

[Bel06] Mihir Bellare. New proofs for NMAC and HMAC: Security without collision-resistance. In *Crypto'06*, pages 602–619, 2006.

[Ber96] Dan Bernstein. Syn cookies, 1996. `http://cr.yp.to/syncookies.html`.

[BHHO08] Dan Boneh, Shai Halevi, Michael Hamburg, and Rafail Ostrovsky. Circular-secure encryption from decision diffie-hellman. In *CRYPTO'08*, pages 108–125, 2008.

[BJLS16] Christoph Bader, Tibor Jager, Yong Li, and Sven Schäge. On the impossibility of tight cryptographic reductions. In *EUROCRYPT 2016*, volume 9666 of *Lecture Notes in Computer Science*, pages 273–304. Springer, 2016.

[BMR10] Dan Boneh, Hart Montgomery, and Ananth Raghunathan. Algebraic pseudorandom functions with improved efficiency from the augmented cascade. In *ACM Conference on Computer and Communications Security—CCS 2010 (to appear)*, 2010.

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

[Cor02] Jean-Sébastien Coron. Optimal security proofs for PSS and other signature schemes. In *EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 272–287. Springer, 2002.

[CW03] Scott Crosby and Dan Wallach. Denial of service via algorithmic complexity attacks. In *12th Usenix Security Symposium*, 2003.

[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.

[FIPR05] Michael Freedman, Yuval Ishai, Benny Pinkas, and Omer Reingold. Keyword search and oblivious pseudorandom functions. In *TCC*, pages 303–324, 2005.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 34(4):792–807, 1986.

[HJK12] Dennis Hofheinz, Tibor Jager, and Edward Knapp. Waters signatures with optimal security reduction. In *PKC 2012*, volume 7293 of *Lecture Notes in Computer Science*, pages 66–83. Springer, 2012.

[HK07] Dennis Hofheinz and Eike Kiltz. Secure hybrid encryption from weakened key encapsulation. In *CRYPTO'07*, pages 553–571, 2007.

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

[JL09] Stanislaw Jarecki and Xiaomin Liu. Efficient oblivious pseudorandom function with applications to adaptive ot and secure computation of set intersection. In *TCC'09*, pages 577–594, 2009.

[LW09] Allison Lewko and Brent Waters. Efficient pseudorandom functions from the decisional linear assumption and weaker variants. In *ACM CCS*, pages 112–120, 2009.

[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.

[Sha07] H. Shacham. A cramer-shoup encryption scheme from the linear assumption and from progressively weaker linear variants. Cryptology ePrint Archive, Report 2007/074, 2007. `http://eprint.iacr.org/`.

[Wat05] Brent Waters. Efficient identity-based encryption without random oracles. In *Advances in Cryptology—EUROCRYPT 2005*, volume 3494 of *LNCS*. Springer-Verlag, 2005.

A Proof of Lemma 13

Proof. Let $\mathbf{q}$ be some fixed query tuple from the adversary. Suppose $\mathbf{b}^* \in [\ell]^n$ is chosen uniformly at random at the end of the simulation. In this probability space, let A_i be the event that query number i issued by the adversary, for $i = 1, \ldots, Q$, cannot be answered by algorithm $\mathcal{B}$. Let A^* be the event that $\mathcal{B}$ does not abort on the challenge query. Then the probability that $\mathcal{B}$ does not abort is

$$\Pr[\text{Success}] = \zeta(\mathbf{q}, \mathbf{b}^*) = \Pr\left[\bar{A}_1 \wedge \ldots \wedge \bar{A}_Q \wedge A^*\right].$$

We can bound this using the inclusion-exclusion principle:

$$\Pr[\text{Success}] = \Pr\left[\bar{A}_1 \wedge \ldots \wedge \bar{A}_Q \wedge A^*\right] \geq \Pr\left[A^*\right] - \sum_{i=1}^{Q} \Pr\left[A_i \wedge A^*\right]. \tag{14}$$

Our goal is to show that when

$$w := \left\lceil (n/\ell) + \Delta \cdot \sqrt{n/\ell}\, \right\rceil \in [n] \quad \text{where } \Delta := \frac{3}{1-\epsilon} \cdot \sqrt{\ln(2Qn)}, \tag{15}$$

then for $i = 1, \ldots, n$

$$\Pr\left[A_i \wedge A^*\right] \leq (1/2Q) \cdot \Pr[A^*], \quad \text{and} \tag{16}$$

$$\Pr[A^*] \geq (1/Qn)^\tau \quad \text{for some constant } \tau > 0. \tag{17}$$

Then by (14) we get

$$\Pr[\text{Success}] \;\geq\; \Pr[A^*] - Q \cdot (1/2Q) \cdot \Pr[A^*] \;=\; \Pr[A^*]/2 \;\geq\; \tfrac{1}{2}(1/Qn)^\tau,$$

which proves the lemma. Table 1 uses (14) to give concrete bounds on $\Pr[\text{Success}]$ for various values of n, ϵ and w. It does so using the exact bounds on $\Pr[A^*]$ and $\Pr\left[A_i \wedge A^*\right]$ in (18) and (19) below.

It remains to prove that (16) and (17) hold when w is as in (15). We start with (17). Let $\text{Bin}(n,p)$ be a Binomial random variable with parameters n and p. First, observe that:

$$\Pr[A^*] = \Pr\big[\text{Bin}(n, 1/\ell) = w\big] = \binom{n}{w} \cdot \left(\frac{1}{\ell}\right)^w \left(1 - \frac{1}{\ell}\right)^{n-w}. \tag{18}$$

By properties of the binomial distribution $\mathrm{Bin}(n, 1/\ell)$, we know that for w in (15) we have

$$\Pr[A^*] \geq \frac{1}{\sqrt{2n}} \cdot e^{-(\Delta^2)} = \frac{1}{\sqrt{2n}} \left(\frac{1}{2Qn}\right)^{\tau'} \qquad \text{where } \tau' := 9/(1-\epsilon)^2,$$

for a sufficiently large n, which proves (17).

Next, we prove (16), which takes a bit more work. Let X be a hypergeometric random variable with parameters n, w, k, so that for $u \in \{0, \ldots, w\}$ we have $\Pr[X = u] = \binom{k}{u}\binom{n-k}{w-u}/\binom{n}{w}$.

Claim 14. *Let* $t(u) := n - k - (w - u)$. *Then*

$$\Pr\left[A_i \wedge A^*\right] = \Pr[A^*] \cdot \sum_{u=0}^{w} \Pr\big[X = u\big] \cdot \Pr\big[\mathit{Bin}\big(t(u), \tfrac{1}{\ell-1}\big) \geq w - u\big]. \tag{19}$$

Let's first prove the claim, and then prove (16).

Proof. Define the following sets:

- $I \subseteq [n]$: the set of coordinates where q_i and q^* match. Let $k := |I|$.
- $I_1 \subseteq [n]$: the set of coordinates outside of I where a mine hits q^*.
- $I_2 \subseteq [n]$: the set of coordinates outside of I where a mine hits q_i.

Since the code H has minimum distance $d > n(1-\epsilon)$, for some $\epsilon \in (\frac{1}{\ell}, 1)$, we know that $k = |I| \leq \epsilon n$. We also know that I_1 and I_2 are disjoint.

Both A_i and A^* occur simultaneously if and only if (i) there are exactly u mines in I for some $u \in \{0, \ldots, w\}$, (ii) there are exactly $w - u$ mines in I_1, and (iii) there are at least $w - u$ mines in I_2. Recall that $t(u) := n - k - (w - u)$. Then,

$$\begin{aligned}
\Pr\left[A_i \wedge A^*\right] &= \sum_{u=0}^{w} \underbrace{\Pr\left[\mathrm{Bin}\big(k, \tfrac{1}{\ell}\big) = u\right]}_{\text{part (i)}} \cdot \underbrace{\binom{n-k}{w-u} \cdot \left(\frac{1}{\ell}\right)^{w-u}}_{\text{part (ii)}} \cdot \underbrace{\sum_{v=w-u}^{t(u)} \binom{t(u)}{v} \left(\frac{1}{\ell}\right)^{v} \left(1 - \frac{2}{\ell}\right)^{t(u)-v}}_{\text{part (iii)}} \\
&= \sum_{u=0}^{w} \binom{k}{u}\binom{n-k}{w-u}\left(\frac{1}{\ell}\right)^{w}\left(1-\frac{1}{\ell}\right)^{n-w} \cdot \left[\frac{\sum_{v=w-u}^{t(u)} \binom{t(u)}{v}\left(\frac{1}{\ell}\right)^{v}\left(1-\frac{2}{\ell}\right)^{t(u)-v}}{\left(1-\frac{1}{\ell}\right)^{t(u)}}\right] \\
&= \sum_{u=0}^{w} \binom{k}{u}\binom{n-k}{w-u}\left(\frac{1}{\ell}\right)^{w}\left(1-\frac{1}{\ell}\right)^{n-w} \cdot \left[\sum_{v=w-u}^{t(u)} \binom{t(u)}{v}\left(\frac{1}{\ell-1}\right)^{v}\left(1-\frac{1}{\ell-1}\right)^{t(u)-v}\right] \\
&= \Pr[A^*] \cdot \sum_{u=0}^{w} \Pr\big[X = u\big] \cdot \Pr\big[\mathrm{Bin}\big(t(u),\ \tfrac{1}{\ell-1}\big) \geq w - u\big]
\end{aligned}$$

and the claim follows. ∎

Next, let's prove (16). The two random variables in (19), namely X and $\mathrm{Bin}\big(t(u), \frac{1}{\ell-1}\big)$, are concentrated around their means, wk/n and $(n+u-k-w)/(\ell-1)$, respectively. As we will see, for an appropriate choice of w, these means are far enough apart so that, for all $u \in \{0,\ldots,w\}$ either $\Pr\big[X=u\big]$ is small or $\Pr\big[\mathrm{Bin}\big(t(u), \frac{1}{\ell-1}\big) \geq w-u\big]$ is small. As a result, all the terms in the sum in (19) are small, making the total sum small, as required.

Concretely, we show that every term in the sum (19) is at most $1/2Qw$, from which (16) follows immediately. The standard tail bound for a hypergeometric distribution shows that

$$\Pr\big[X=u\big] \leq e^{-2\alpha^2 w} \qquad \text{for all } \alpha > 0 \text{ and } u \geq wk/n + \alpha w.$$

Therefore, if $u \geq (wk/n) + \sqrt{w\ln(2Qw)/2}$ then $\Pr\big[X=u\big] < 1/2Qw$. Hence, terms in the sum (19) with u greater than this bound are smaller than $1/2Qw$, as required. We know that w from (15) satisfies $w \leq 2n/\ell$, for a sufficiently large n. Therefore, for convenience, we use the slightly worse bound $u \geq wk/n + \sqrt{n\ln(2Qn)/\ell}$.

Let $c_1 := \sqrt{n\ln(2Qn)/\ell}$. Then we know that terms in (19) with $u \geq wk/n + c_1$ are less than $1/2Qw$. Now, consider terms with $u < wk/n + c_1$. The standard tail bound for binomials shows that

$$\Pr\big[\mathrm{Bin}(t(u), \tfrac{1}{\ell-1}) \geq w-u\big] \leq e^{-\alpha^2/3} \quad \text{for all } \alpha \in \Big[0,\ \sqrt{\tfrac{t(u)}{\ell-1}}\ \Big] \text{ and } w-u \geq \tfrac{t(u)}{\ell-1} + \alpha\sqrt{\tfrac{t(u)}{\ell-1}}\ .$$

By the assumption on n in Theorem 12 we know that $t(u) \geq \ell(3\ln(2Qw))$. This implies that $\alpha := \sqrt{3\ln(2Qw)} \leq \sqrt{t(u)/(\ell-1)}$. Therefore,

$$\Pr\big[\mathrm{Bin}(t(u), \tfrac{1}{\ell-1}) \geq w-u\big] \leq 1/2Qw \quad \text{whenever} \quad w-u \geq \tfrac{t(u)}{\ell-1} + \sqrt{3\ln(2Qw)\tfrac{t(u)}{\ell-1}},$$

or more simply, whenever

$$w-u \geq \frac{t(u)}{\ell-1} + c_2 \quad \text{where } c_2 := \sqrt{3n\ln(2Qn)/\ell}.$$

Plugging in the value of $t(u) = n-k-(w-u)$, we see that

$$\Pr\big[\mathrm{Bin}(t(u), \tfrac{1}{\ell-1}) \geq w-u\big] \leq 1/2Qw \quad \text{whenever} \quad u \leq w - \tfrac{n-k}{\ell} - c_2.$$

To cover the set of u where $u < wk/n + c_1$, we want

$$wk/n + c_1 \leq w - (n-k)/\ell - c_2$$

which implies

$$w \geq (n/\ell) + \frac{c_1+c_2}{1-k/n}.$$

Since $k/n < \epsilon$, this bound is satisfied whenever

$$w \geq \frac{n}{\ell} + \frac{c_1+c_2}{1-\epsilon}$$

or, after plugging in the values for c_1 and c_2, whenever

$$w \geq \frac{n}{\ell} + \frac{3}{1-\epsilon}\sqrt{\frac{n}{\ell}\cdot\ln(2Qn)}.$$

For such w we know that all the terms in (19) are less than $1/2Qw$ from which (16) follows immediately. This completes the proof (16) and of the lemma. $\square$