Efficient Construction of (Distributed) Verifiable Random Functions

Yevgeniy Dodis

Department of Computer Science
New York University, USA
dodis@cs.nyu.edu

Abstract. We give the first simple and efficient construction of *verifiable random functions* (VRFs). VRFs, introduced by Micali et al. [13], combine the properties of regular pseudorandom functions (PRFs) (i.e., indistinguishability from a random function) and digital signatures (i.e., one can provide an unforgeable proof that the VRF value is correctly computed). The efficiency of our VRF construction is only slightly worse than that of a regular PRF construction of Naor and Reingold [16]. In contrast to our direct construction, all previous VRF constructions [13, 12] involved an expensive generic transformation from verifiable unpredictable functions (VUFs).

We also provide the first construction of *distributed* VRFs. Our construction is more efficient than the only known construction of distributed (non-verifiable) PRFs [17], but has more applications than the latter. For example, it can be used to distributively implement the random oracle model in a *publicly verifiable* manner, which by itself has many applications.

Our construction is based on a new variant of decisional Diffie-Hellman (DDH) assumption on certain groups where the regular DDH assumption does *not* hold [10, 9]. Nevertheless, this variant of DDH seems to be plausible based on our *current* understanding of these groups. We hope that the demonstrated power of our assumption will serve as a motivation for its closer study.

1 Introduction

As a motivating example for our discussion, consider the problem of implementing the *random oracle model* [2]. Recall that in this model one assumes the existence of a publicly verifiable random function $\mathcal{O}$ (over some suitable domain and range). Each value $\mathcal{O}(x)$ is random and independent from the other values, and evaluating $\mathcal{O}$ on the same input twice yields the same (random) output. This model has found numerous applications in cryptography, which we do not even attempt to enumerate. It was shown by Canetti et al. [5], though, that no fixed public function can generically replace the random oracle, so more elaborate solutions are needed.

Y.G. Desmedt (Ed.): PKC 2003, LNCS 2567, pp. 1–17, 2003.

Pseudorandom Functions. As the first attempt, we may assume the existence of a trusted (but computationally bounded) party T. Since a function is an exponential sized object, T cannot store it explicitly. While maintaining a dynamically growing look-up table is a possibility, it is very inefficient as it requires large storage and growing complexity. A slightly better option is to use a *pseudorandom function* (PRF) $F_{SK}(\cdot)$ [8]. As indicated, this function is fully specified and efficiently computable given its short secret key (or *seed*) SK. However, without the knowledge of SK it looks *computationally* indistinguishable from exponential-sized $\mathcal{O}$.

In terms of constructing PRFs, there are several options. The most relevant to this paper, however, is the number-theoretic construction due to Naor and Reingold [16], which is based on the decisional Diffie-Hellman (DDH) assumption. This assumption in some group $\mathbb{G}$ of prime order q states that given elements g, g^a and g^b of (where g is the generator of $\mathbb{G}$), it is hard to distinguish the value g^{ab} from a truly random value g^c (where a, b, c are random in $\mathbb{Z}_q$). The PRF of [16] is a tree-based construction similar to the PRF construction of [8] from a pseudorandom generator. Namely, the secret key $SK = (g, a_1, \dots a_\ell)$ consists of a random generator g of $\mathbb{G}$ and ℓ random exponents in $\mathbb{Z}_q$ (where ℓ is the length of the input to our PRF $F_{SK} : \{0,1\}^\ell \to \mathbb{G}$). Given $x = x_1 \dots x_\ell \in \{0,1\}^\ell$, the PRF is defined by:

$$F_{g,a_1,\dots,a_\ell}(x_1 \dots x_\ell) \overset{\text{def}}{=} g^{\prod_{\{i|x_i=1\}} a_i \bmod q} \tag{1}$$

Verifiable Random Functions. Coming back to our motivating application, replacing random oracle with a PRF has several problems. The first one is the question of verifiability and transferability. Even if everybody trusts T (which we will revisit soon), T has to be contacted not only when the value of F has to be computed for the first time, but even if one party needs to verify that another party has used the correct value of F. Thus, it would be much nicer if each value of $F_{SK}(x)$ would come with a proof $\pi_{SK}(x)$ of correctness, so that the recipient and everybody else can use this proof without the need to contact T again. As a side product, the ability to give such proof will also ensure that T himself cannot "cheat" by giving inconsistent values of F, or denying a correctly computed value of the function. This leads to the notion of *verifiable (pseudo)random functions*, or VRFs [13]. Intuitively, such functions remain (pseudo)random when restricted to all inputs whose function values were not previously revealed (and proved). Notice, the pseudorandomness and verifiability of a VRF immediately imply that a VRF by itself is an unforgeable signature scheme secure against chosen message attack.

Constructions of VRFs. Unfortunately, VRFs are not very well studied yet. Currently, we have two constructions of VRFs: based on RSA [13], and based on a separation between computational and decisional Diffie-Hellman problems in certain groups [12]. Both of these constructions roughly proceed as follows. First, they construct a relatively simple and efficient verifiable *unpredictable* function (VUF) based on the corresponding assumption. Roughly, a VUF is the same verifiable object as a VRF, except each "new" value $F_{SK}(x)$ is only unpredictable (i.e., hard to compute) rather than pseudorandom. From VUFs, a generic construction to VRFs is given, as introduced by [13]. Unfortunately, this construction is very inefficient and also looses a very large factor in its exact security. Essentially, first one uses the Goldreich-Levin theorem [7] to construct a VRF with very small (slightly super-logarithmic) input size and output size 1 (and pretty dramatic security loss too!).[1] Then one makes enough such computations to amplify the output size to roughly match that of the input. Then one follows another rather inefficient tree-based construction on the resulting VRF to get a VRF with arbitrary input size and small output size. Finally, one evaluates the resulting convoluted VRF several times to increase the output size to the desired level. In some sense, the inefficiency of the above construction is expected given its generality and the fact that it has to convert pure unpredictability into a much stronger property of pseudorandomness. Still, this means that the resulting VRF constructions are very bulky and inelegant. In this work we present the first simple, efficient and "direct" VRF construction.

Distributed PRFs. Returning to our target application of implementing the random oracle, the biggest problem of both PRF/VRF-based solutions is the necessity to fully trust the honest party T holding the secret key for F. Of course, VRFs slightly reduced this trust level, but T still singlehandedly knows all the values of F. Clearly, this approach (1) puts to much trust into T, (2) makes T is bottleneck of all the computations; (3) makes T is "single point of failure": compromising T will break the security of any application which depends on the random oracle assumption.

The natural solution to this problem is to distribute the role of T among n servers. This leads to the notion of *distributed* PRF*s* (DPRFs) and *distributed* VRF*s* (DVRFs). Since the latter concept was not studied prior to our work, we start with DPRFs, thus ignoring the issue of verifiability for now. Intuitively, DPRFs with threshold $1 \leq t < n$ allow any $(t + 1)$ out of n servers to jointly compute the function using their shares, while no coalition of up to t servers to be in a better situation that any outside party. Namely, the function remains pseudorandom to any such coalition.

DPRFs first originate in the work of Micali and Sidney [14]. However, their construction (later improved by [15]) can tolerate only a moderate number of servers or a small threshold, since its complexity is proportional to n^t. The next influential work is that of Naor et al. [15], who give several efficient constructions of certain weak variants of DPRFs. Ironically, one of the constructions (namely, that of distributed *weak* PRF) can be turned into an efficient DPRF by utilizing random oracles. Even though this is non-trivial (since nobody should compute the value of a DPRF without the cooperation of $t+1$ servers), we would certainly prefer a solution in the plain model, since elimination of the random oracle was one of the main motivation for DPRFs!

[1] The latter is the reason for such a small input size. One can make a very strong exponential assumption to increase the input size, like was done in [12], but the construction still loses a lot in security, and still goes through an intermediate VUF.

The first regular DPRF was recently constructed by Nielsen [17] by distributing a slightly modified variant of the Naor-Reingold PRF [16], given in Equation (1) (in the final version of their work, [16] also give essentially the same construction). Unfortunately, the resulting DPRF in highly *interactive* among the servers (while ideally the servers would only talk to the user requesting the function value) and requires a lot of rounds (proportional to the length of the input). In particular, the question of non-interactive DPRF construction remained open prior to this work.

Distributed VRFs. Even though DVRFs were not explicitly studied prior to this work, they seem to provide the most satisfactory solution to our original problem of implementing the random oracle. Indeed, distributing the secret key ensures that no coalition of up to t servers can compromise the security (i.e., pseudorandomness) of the resulting random oracle. On the other hand, verifiability ensures that one does not need to contact the servers again after the random oracle was computed once: the proof can convince any other party of the correctness of the VRF value. For example, DVRFs by themselves provide an ordinary threshold signature scheme, which can be verified without further involvement of the servers. And, of course, using DVRFs are likely to enhance the security, robustness or functionality of many applications originally designed for plain PRFs, VRFs and DPRFs.

Our Contributions. We give the first simple and direct construction of VRFs, based on a new "DDH-like" assumption which seems to be plausible on certain recently proposed elliptic and hyper-elliptic groups (e.g., [10]). We call this assumption *sum-free decisional Diffie-Hellman* (sf-DDH) assumption. While we will discuss this assumption later, we mention that in the proposed groups the regular regular DDH assumption is *false* (in fact, this is what gives us verifiability!), and yet the sf-DDH or some similar assumption seems plausible. Our construction is similar to the Naor-Reingold (NR) construction given by Equation (1), except we utilize some carefully chosen encoding C before applying the NR-construction. Specifically, if $C : \{0,1\}^\ell \to \{0,1\}^L$ is some injective encoding, we consider the function of the form

$$F_{g,a_1,\ldots,a_L}(x_1 \ldots x_\ell) \overset{\text{def}}{=} g^{\prod_{\{i|C(x)_i=1\}} a_i \bmod q} \tag{2}$$

Identifying the properties of the encoding C and constructing C satisfying these properties will be one of the main technical challenges we will have to face. At the end we will achieve $L = O(\ell)$ (specifically, $L = 2\ell$ to get a regular PRF, and $L = 3\ell+2$ to get a VRF), making our efficiency very close to the NR-construction.

Our second main contribution is the first construction of a distributed VRF(DVRF). Namely, we show that our VRF construction can be made distributed and *non-interactive* (although multi-round). This is the first non-interactive construction of a distributed PRF (let alone VRF), since the only previous DPRF construction of [17, 16] is highly interactive among the servers. In fact, our DVRF construction is more efficient than the above mention DPRF construction, despite achieving the extra verifiability. We already mentioned the big saving in communication complexity (roughly, from $n^2\ell k$ to $n\ell k$, where k is the security parameter). Another important advantage, though, is that we dispense with the need to perform somewhat expensive (concurrently composable) zero knowledge proofs for the equality of discrete logs. This is because in our groups the DDH problem is easy, so it can be locally checked by each party without the need for the proof. In particular, even though we need to apply the encoding C to the message, while the construction of [17, 16] does not, the lack of ZK-proofs makes our round complexity again slightly better. Finally, we remark that the same distributed construction can be applied to distribute the VUF of Lysyanskaya [12] (which results in a threshold "unique signature" scheme under a different assumption than the one we propose).

2 Definitions

2.1 Verifiable Random Functions and Friends

Definition 1. *A function family $F_{(\cdot)}(\cdot) : \{0,1\}^{\ell(k)} \to \{0,1\}^{m(k)}$ is a family of* VRF*s, if there exists a probabilistic polynomial time algorithm* Gen *and deterministic algorithms* Prove *and* Verify *such that:* $\mathsf{Gen}(1^k)$ *outputs a pair of keys (PK, SK);* $\mathsf{Prove}_{SK}(x)$ *outputs a pair $\langle F_{SK}(x), \pi_{SK}(x)\rangle$, where $\pi_{SK}(x)$ is the proof of correctness; and* $\mathsf{Verify}_{PK}(x, y, \pi)$ *verifies that $y = F_{SK}(x)$ using the proof π. We require:*

1. *Uniqueness: no values $(PK, x, y_1, y_2, \pi_1, \pi_2)$ can satisfy $\mathsf{Verify}_{PK}(x, y_1, \pi_1) = \mathsf{Verify}_{PK}(x, y_2, \pi_2)$ when $y_1 \neq y_2$.*
2. *Provability: if $(y, \pi) = \mathsf{Prove}_{SK}(x)$, then $\mathsf{Verify}_{PK}(x, y, \pi) = 1$.*
3. *Pseudorandomness: for any PPT $A = (A_1, A_1)$ who did not call its oracle on x (see below), the following probability is at most $\frac{1}{2} + \mathsf{negl}(k)$ (here and everywhere, $\mathsf{negl}()$ stands for some negligible function in the security parameter k):*

$$\Pr\left[\, b = b' \;\middle|\; \begin{array}{c} (PK, SK) \leftarrow \mathsf{Gen}(1^k);\; (x, st) \leftarrow A_1^{\mathsf{Prove}(\cdot)}(PK);\; y_0 = F_{SK}(x); \\ y_1 \leftarrow \{0,1\}^{m(k)};\; b \leftarrow \{0,1\};\; b' \leftarrow A_2^{\mathsf{Prove}(\cdot)}(y_b, st) \end{array} \right]$$

Intuitively, the definition states that no "new" value of the function can be distinguished from a random string, even after seeing any other function values together with the corresponding proofs. Regular PRFs form the non-verifiable analogs of VRFs. Namely, $PK = \emptyset$, $\pi_{SK}(\cdot) = \emptyset$, there is no algortihm Verify, no uniqueness and provability properties, and pseudorandomness is the only remaining property. We notice that the resulting definition is not the typical definition for PRFs [8]: namely, that no adversary can tell having oracle access to a truly random function from having oracle access to a pseudorandom function. However, it is easy to see that our definition is equivalent to that usual one, so will we use it as the more convenient in the context of VRFs.

2.2 Diffie-Hellman Assumptions

Assume $\mathsf{Setup}(1^k)$ outputs the description of some cyclic group $\mathbb{G}$ of prime order q together with its random generator g. Let $L = L(k)$ be some integer and $a_1 \dots a_L$ be random elements of $\mathbb{Z}_q$. Let $[L]$ denote $\{1 \dots L\}$, and given a subset $I \subseteq [L]$, we denote $a_I = \prod_{i \in I} a_i \bmod q$ (where $a_\emptyset = 1$), $G(I) = G_I = g^{a_I}$. Finally, we will often view an element $z \in \{0,1\}^L$ as either a subset $\{i \mid z_i = 1\}$, or an L-dimensional vector over $GF(2)$ (and vice versa).

Generalized Diffie-Hellman Assumptions. The security of ours, as well as the previous related constructions [16, 12], will rely on various assumptions of the following common flavor. The adversary A has oracle access to $G(\cdot)$, and tries to "obtain information" about some value $G(J)$. The meaning of obtaining information depends on whether we are making a computational or a decisional assumption. In the former case, A has to compute $G(J)$, while in the latter case A has to distinguish $G(J)$ from a random element of $\mathbb{G}$. While the decisional assumption is stronger, it has a potential of yielding a (verifiable) *pseudorandom* function, while the computational assumption can yield at best[2] a (verifiable) *unpredictable* function.

In either case, we require that it should be hard to any polynomial time adversary to succeed. Of course, one has to make some non-trivial restrictions on when the adversary is considered suceessful. Formally, given that the adversary called its oracle on subsets $I_1, \dots, I_t$ and "obtained information" about $G(J)$, we can define a predicate $\mathcal{R}(J, I_1, \dots I_t)$ which indicates whether the adversary's actions are "legal". For example, at the very least the predicate should be false if $J \in \{I_1 \dots I_t\}$. We call any such predicate *non-trivial*. We will certainly restrict ourselves to non-trivial predicates, but may sometimes place some more restrictions on $\mathcal{R}$ in order to make a more plausible and weaker assumption (see below).

Definition 2. *Given $L = L(k)$, we say that the group $\mathbb{G}$ satisfies the* generalized decisional Diffie-Hellman *(*gDDH*) assumption of order L* relative to a non-trivial predicate $\mathcal{R}$*, if for any* PPT *adversary $A = (A_1, A_1)$ who called its oracle on subsets $I_1 \dots I_t$ satisfying $\mathcal{R}(J, I_1, \dots, I_t) = 1$, the probability below is at most $\frac{1}{2} + \mathsf{negl}(k)$:*

$$\Pr\left[\, b = b' \;\middle|\; \begin{array}{c} (\mathbb{G}, q, g) \leftarrow \mathsf{Setup}(1^k);\ (a_1 \dots a_L) \leftarrow \mathbb{Z}_q,\ (J, st) \leftarrow A_1^{G(\cdot)}(\mathbb{G}, q); \\ y_0 = G(J);\ y_1 \leftarrow \mathbb{G};\ b \leftarrow \{0,1\};\ b' \leftarrow A_2^{G(\cdot)}(y_b, st) \end{array} \right]$$

Very similarly one can define the *generalized computational Diffie-Hellman* (gCDH) assumption of order L *relative to* $\mathcal{R}$, where the job of A is to *compute* $G(J)$. We notice that the more restrictions $\mathcal{R}$ places on the I_i's and the "target" set J, the harder it is for the adversary to succeed, so the assumption becomes weaker (and more preferable). Thus, the strongest possible assumption of the above type is to put no further restrictions on $\mathcal{R}$ other than non-triviality (i.e., $J \notin \{I_1, \dots I_t\}$). We call the two resulting assumptions simply gDDH and gCDH (without specifying $\mathcal{R}$). A slightly weaker assumption results when we

[2] Unless a generic inefficient conversion is used, or one assumes random oracles.

require that the target set is equal to the full set $J = [L]$, i.e. the adversary has to obtain information about $g^{a_1 \cdots a_L}$. We call the resulting assumptions *full target* gDDH/gCDH (where $L = 2$ yields regular DDH/CDH). Finally, making L larger generally makes the assumption stronger, since the adversary can always choose to concentrate on some subset of L. Thus, it is preferable to base the security of some contsruction on as small L and as restrictive $\mathcal{R}$ as possible.

Before moving to our new sum-free gDDH assumption, let us briefly state some simple facts about gDDH/gCDH. It was already observed by [19] that gDDH assumption of any polynomial order $L(k)$ (with or without full target) follows from the regular DDH assumption (which corresponds to $L = 2$). Unfortunately, we do not know of the same result for the gCDH problem. The best analog of this result was implicitly obtained by [12], who more or less showed that the regular gCDH assumption of logarithmic order $O(\log k)$ (even with full target) implies the gCDH assumption of any polynomial order $L(k)$, *provided* in the latter we restrict the adversary to operate on the codewords of any good error-correcting code.

Sum-Free gDDH. We already saw that the regular DDH assumption is a very strong security assumption in that it implies the gDDH assumption. This useful fact almost immediately implies, for example, that the Naor-Reingold construction in Equation (1) is a PRF under DDH, illustrating the power of DDH for proving pseudorandomness. Unfortunately, groups were DDH is true are not convenient for making *verifiable* random functions, since nobody can verify the equality of discrete logs. On the other hand, we will see shortly that it is very easy to obtain verifiability in groups where DDH is solvable in polynomial time (such as the group suggested by [10]). Unfortunately, such groups certainly do not satisfy the gDDH assumption too, which seems to imply that we have to settle for the computational assumption (like gCDH) in such groups, which in turn implies that we settle only for the VUF construction rather than the desired VRF. Indeed, obtaining such a VUF is exactly what was recently done by Lysyanskaya [12] in groups where DDH is easy but gCDH is hard.

However, we make the crucial observation that the easiness of regular DDH does *not* mean that no version of gDDH assumption can be true: it only means *we might have to put more restrictions on the predicate* $\mathcal{R}$ in order to make it hard for the adversary to break the gDDH assumption relative to $\mathcal{R}$. Indeed, for the current elliptic groups for which we believe in a separation between DDH and CDH, we only know how to test if (h, u, v, w) is of the form $u = h^a, v = h^b, w = h^{ab}$ (this is called a DDH-tuple). This is done by means of a certain bilinear mapping (details are not important), for which we do not know a multi-linear variant. In fact, Boneh and Silverberg [4] observe that a multi-linear variant of such mapping seems unlikely to exist in the currently proposed groups, and pose as a major open problem to exhibit groups where such mappings exist. This suggests that many natural, but more restrictive flavors of DDH seem to hold in the currently proposed groups (where regular DDH is easy). For example, as was mentioned by Boneh and Franklin [3], it seems reasonable to assume that it is hard to distinguidh a tuple $(h, h^a, h^b, h^c, h^{abc})$ from a random tuple

(h, h^a, h^b, h^c, h^d). Put differently, when $a_1 \ldots a_L$ are chosen at random and given a sample $g = G(\emptyset), G(I_1) \ldots G(I_t)$, the only way we know how to distinguish $G(J)$ from a random element of such groups is by exhibiting three sets I_m, I_p, I_s (where $0 \le m, p, s \le t$, and I_0 denotes the empty set) such that $a_J \cdot a_{I_m} \equiv a_{I_p} \cdot a_{I_s} \bmod q$.[3] The last equation implies that "$J + I_m = I_p + I_s$", where we view the sets as L-bit 0/1-vectors, and the addition is bitwise over the integers. In other words, one has to explicitly find a DDH-tuple among the samples $G(I_i)$'s and the target $G(J)$.

We formalize this intuition into the following predicate $\mathcal{R}(J, I_1, \ldots, I_t)$. Let us denote $I_0 = \emptyset$. We say that J is DDH-*dependent* on $I_1 \ldots I_t$ if there are indices $0 \le m, p, s \le t$ satisfying $J + I_m = I_p + I_s$ (see explanation above). For example, 10101 is DDH-dependent on 01010, 00001 and 11111, since $10101 + 01011 = 11111 + 00001 = 11112$. Then we define the DDH-*free* relation $\mathcal{R}$ to be true if and only if J is DDH-independent from $I_1 \ldots I_t$.

Definition 3. *Given $L = L(k)$, we say that the group $\mathbb{G}$ (where regular* DDH *is easy) satisfies the* sum-free decisional Diffie-Hellman *(*sf-DDH*) assumption of order L if it satisfies the* gDDH *assumption of order L relative to the* DDH*-free relation $\mathcal{R}$ above.*

For our purposes we notice that DDH-dependence also implies that $J \oplus I_m = I_p \oplus I_s$, where $\oplus$ indicates the bitwise addition moduo 2 (i.e., we make "$2 = 0$"), or $J \oplus I_m \oplus I_p \oplus I_s = 0$. Let us call J 4*-wise independent* from $I_1 \ldots I_t$ if no three sets I_m, I_p, I_s yield $J \oplus I_m \oplus I_p \oplus I_s = 0$. Hence, if we let $\mathcal{R}'(J, I_1, \ldots, I_t) = 1$ if and only if J is 4-wise independent from the I_i's, we get that $\mathcal{R}'$ is a stricter relation than our DDH-free $\mathcal{R}$. But this means that gDDH assumption relative to $\mathcal{R}'$ is a *weaker* assumption than sf-DDH, so we call it *weak* sf-DDH. Our actual construction will in fact be based on weak sf-DDH.

To summarize, sf-DDH is the strongest assumption possible in groups were regular DDH is false. We chose this assumption to get the simplest and most efficient VRF construction possible when DDH is false. However, even if the ambitious sf-DDH assumption we propose turns out to be false in the current groups where DDH is easy — which we currently have no indication of — it seems plausible that some reasonable weaker gDDH assumptions (relative to more restrictive $\mathcal{R}$) might still hold. And our approach seems to be general enough to allow some easy modification to our construction (at slight efficiency loss) meet many such weaker gDDH assumptions.

3 Constructions

Assume $\mathbb{G}$ is the group where DDH is easy while some version of sf-DDH holds. We consider the natural the type of functions given by Equation (2); in our new

[3] One can also try to find the additive relations, but since the a_i's are all random, it seems that the only such relations one can find would trivially follow from some multiplicative relations.

notation, $F_{g,a_1,\ldots,a_L}(x_1 \ldots x_\ell) = G(C(x))$,[4] where C is some currently unspecified (but efficiently computable) injective mapping from $\{0,1\}^\ell$ to $\{0,1\}^L$. To emphasize this dependence on C, we will sometimes denote the above function by $NR_C(\cdot)$.

3.1 Building PRFs

As a warm-up towards VRFs, we first determine the conditions on C and the kind of gDDH assumption we need in order to get a regular PRF.

Lemma 1. *Assume $\mathcal{R}$ and C are such that $\mathcal{R}(C(w), C(x_1), \ldots, C(x_t)) = 1$ for any $w \notin \{x_1, \ldots, x_t\}$. Then $NR_C(\cdot)$ is a* PRF *under the* gDDH *assumption of order L relative to $\mathcal{R}$.*

Proof. The proof follows almost immediately by comparing the deginition of gDDH relative to $\mathcal{R}$ (Definition 2) and the definition of PRF given in Section 2.1. Indeed, the adversary can query $NR_C(\cdot)$ at any points $x_1, \ldots x_t$, which corresponds to querying $G(\cdot)$ on $C(x_1) \ldots C(x_t)$, and has to distinguish $NR_C(w) = G(C(w))$ for some $w \notin \{x_1 \ldots x_t\}$. Since our assumption implies that $\mathcal{R}(C(w), C(x_1), \ldots, C(x_t)) = 1$, this adversary is legal for breaking gDDH (of order L) relative to $\mathcal{R}$.

As an immediate corollary, usual gDDH assumption implies that $NR_C(\cdot)$ is a PRF for any (injective) C, including the identity. This in turn gives the result of [16], since we mentioned that regular DDH implies gDDH [19].

More interestingly, we will now determine the properties of C which suffice to show that NR_C is a PRF under the much weaker sf-DDH assumption (for now, of the same large order L; we will reduce the order later). In the following, view every subset of $[L]$ (or element of $\{0,1\}^L$) as an L-dimensional vector over $GF(2)$. We say that the collection of vectors $I_1 \ldots I_t$ is 4-*wise independent*, if no 4 or fewer vectors are linearly dependent. The proof of the theorem below is now obvious from Lemma 1.

Theorem 1. *Assume C is such that the collection $\left\{C(x) \mid x \in \{0,1\}^\ell\right\}$ is 4-wise independent. Then $NR_C(\cdot)$ is a* PRF *under the (weak)* sf-DDH *assumption of order L.*

Constructing 4-wise Independent Encodings. To get our PRF under the sf-DDH assumption, it thus suffices to construct a 4-wise independent encoding C. Naturally, the goal is to make L as close to ℓ as possible. Such encodings come up quite often in the theory of derandomization (see [1]), and are closely related to coding theory.[5] In our case, the well known construction is very simple

[4] Notice, we output a (pseudo)random element of $\mathbb{G}$ instead of a (pseudo)random m-bit string. However, standard hashing techniques imply we can extract an almost uniform string of length close to $\log |\mathbb{G}|$ from such an output. See [16].

[5] In particular, obtaining the 4-wise independent encoding C we need is equivalent to designing a parity check matrix of any linear code of distance 5. Our specific code gives such matrix for the famous (and optimal) BCH code of designed distance 5.

and efficient, so we present it in a self-contained manner. It will achieve (easily seen to be optimal) $L = 2\ell$.

Let us view any non-zero element $x \in \{0,1\}^\ell$ as an element of the field $GF(2^\ell)$, which can also be represented as an ℓ-dimensional vector over $GF(2)$. This gives us the same bitwise addition operation $\oplus$, but now we also have a multiplication operation. Then we set $L = 2\ell$ and define $C(x) = (x^3 \| x)$, which is interpreted as follows. We first cube x, which gives us another ℓ-dimentional vector x^3, and then we append x to it. Notice, the code C is explicit and extremely efficient to evaluate. Now, assume there are some *non-zero* distinct $x_1, x_2, x_3, x_4 \in GF(2^\ell)$ and constants $\alpha_1, \alpha_2, \alpha_3, \alpha_4 \in \{0,1\}$ such that $\sum_{i=1}^4 \alpha_i C(x_i) = 0$. We will show that $\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = 0$, which yields 4-wise independence.

Since our bitwise addition is the same as in the field, we get $\sum_{i=1}^4 \alpha_i x_i = 0$ and $\sum_{i=1}^4 \alpha_i x_i^3 = 0$ over $GF(2^\ell)$. Next, we square the first equation. Since $GF(2^\ell)$ has characteristic 2 and $\alpha_i^2 = \alpha_i$, the only surviving terms are $\alpha_i x_i^2$, which gives us $\sum_{i=1}^4 \alpha_i x_i^2 = 0$. Similarly, raising the first equation to the power 4 gives $\sum_{i=1}^4 \alpha_i x_i^4 = 0$. We get a linear system (with unknowns $\alpha_1, \alpha_2, \alpha_3, \alpha_4$) saying that $\sum_{i=1}^4 \alpha_i x_i^j = 0$ for $j = 1, 2, 3, 4$. The system corresponds to the famous Vandermonde matrix whose determinant is $x_1 x_2 x_3 x_4 \cdot \prod_{i<j}(x_i - x_j) \neq 0$, since all the x_i's are distinct and non-zero. Thus, the only solution to the system is the trivial all-zero solution.

As a small technicality, we get the 4-wise independent encoding $C : \{0,1\}^\ell \backslash \{0^\ell\} \to \{0,1\}^{2\ell}$, i.e. we exclude the all-zero vector. This implies that we get the PRF whose input domain excludes the all-zero vector too. To summarize,

Theorem 2. *The encoding C above defines a* PRF *mapping ℓ bits (except 0^ℓ) to an element of $\mathbb{G}$, which is secure under the (weak)* sf-DDH *assumption of order 2ℓ.*

Reducing the Order. While Theorem 2 gives a simple PRF construction, it is based on the sf-DDH assumption of high polynomial order $2\ell(k)$. While this assumption is reasonable, we now show how to reduce the order to $O(\log k)$ at only a marginal efficiency loss. So let $C : \{0,1\}^\ell \to \{0,1\}^L$ be any 4-wise independet encoding satisfying Theorem 1 (like the one we constructied above). The idea, similar to that of [12], is to use an error-correcting code $E : \{0,1\}^L \to \{0,1\}^N$ on top of our encoding C. However, since we are dealing with linear dependence, we will have to restrict ourselves to *linear* codes (which was not needed in [12]), and the analysis will be slightly more involved. Thus, let E be a linear error correcting code of distance δN (where $\delta > 0$ and $N = O(L)$), and define $\tilde{C} = E \circ C : \{0,1\}^\ell \to \{0,1\}^N$. We get the following result, whose proof can be found in the full version [6].

Theorem 3. *Assume (weak)* sf-DDH *assumption holds for any order $p = O(\log k)$. Then $NR_{\tilde{C}}(\cdot)$ is a* PRF.

We remark that since error-correcting code can in principle approach a rate of 1, using Theorem 3 we can get a PRF construction with final expansion $N = (2+\varepsilon)\ell$.

3.2 Building VRFs

We now show how extend our ideas to get a VRF based of sf-DDH. As before, the construction is parameterized by some encoding $C : \{0,1\}^\ell \to \{0,1\}^L$. Recall also that we assume that testing regular DDH can be done in polynomial time.

- $\mathsf{Gen}(1^k)$: runs $(G, q, g) \leftarrow \mathsf{Setup}(1^k)$, picks random $a_1, \ldots, a_{L+1} \in \mathbb{Z}_q$, sets $h = g^{a_{L+1}}$, $y_1 = h^{a_1}$,..., $y_L = h^{a_L}$. Outputs public key $PK = (\mathbb{G}, q, g, h, y_1 = h^{a_1}, \ldots, y_L = h^{a_L})$, secret key $SK = (g, a_1, \ldots, a_L)$.
- $\mathsf{Prove}_{SK}(x)$: outputs $(\sigma_1, \ldots \sigma_L)$, where $\sigma_j = g^{\prod_{\{i \leq j | C(x)=1\}} a_i}$ for $j = 1 \ldots L$. In particular, the value σ_L is $F_{SK}(x)$, while $(\sigma_1, \ldots, \sigma_{L-1})$ is the proof $\pi_{SK}(x)$.
- $\mathsf{Verify}_{PK}(\sigma_1, \ldots, \sigma_L)$: sets $\sigma_0 = g$ and checks, for $i = 1 \ldots L$, that $(\sigma_{i-1}, \sigma_i, h, y_i)$ form a DDH-tuple (recall, DDH is easy!) when $C(x) = 1$, or that $\sigma_{i-1} = \sigma_i$ is $C(x)_i = 0$. Accept if all the tests pass.

To satisfy the definition of VRFs (Definition 1), we need to examine uniqueness, provability and pseudorandomness. The first two properties are very easy. Uniqueness follows from the fact that discrete logs are unique in $\mathbb{G}$ (and that our assumed algorithm for DDH will never accept an invalid tuple), while provability is obvious by construction.

Thus, we only need to examine pseudorandomness. Luckily, a lot of machinery has been already developed in Section 3.1. Essentially, the main difference we have is that when the adversary asks $\mathsf{Prove}(x)$, not only does he get $F(x) = G(C(x))$, but he also gets the proof values $G(I)$ for all $I \in \mathsf{Prefixes}(C(x))$, where for a set $J \subseteq [L]$ we define $\mathsf{Prefixes}(J) \overset{\text{def}}{=} \{\emptyset, J \cap [1], J \cap [2], \ldots, J \cap [L-1], J\}$. Additionally, the public key gives the adversary the values $G(\{L+1\})$, $G(\{L+1, 1\}), \ldots, G(\{L+1, L\})$. We denote the latter $L+1$ subsets of $[L+1]$ involving element $L+1$ by $Pub(L+1)$. With these in mind, we easily get the following analog of Lemma 1.

Lemma 2. *Assume $\mathcal{R}$ and C are such that that for any $w \notin \{x_1, \ldots, x_t\}$ we have $\mathcal{R}(C(w), \mathsf{Prefixes}(C(x_1)), \ldots, \mathsf{Prefixes}(C(x_t)), Pub(L+1)) = 1$. Then our construction is a* VRF*, under the* gDDH *assumption of order $L+1$ relative to $\mathcal{R}$.*

Next, we can generalize the notion of 4-wise independence to that of 4-wise *prefix-independence*. Namely, a vector J is 4-wise prefix independent from vectors $I_1 \ldots I_t$ if there exist no $1 \leq p, r, s, \leq t$ and $I'_p \in \mathsf{Prefixes}(I_p)$, $I'_r \in \mathsf{Prefixes}(I_r)$, $I'_s \in \mathsf{Prefixes}(I_s)$ such that $J \oplus I'_p \oplus I'_r \oplus I'_s = 0$. A collection $\{I_1 \ldots I_t\}$ is said to be 4-wise prefix independent if every vector I_i is 4-wise prefix independent from the remaining vectors. Finally, we will say that the above collection has *prefix-distance* at least 3, if for any $i \neq j$ and $I'_j \in \mathsf{Prefixes}(I_j)$, we have that I_i and I'_j differ in at least 3 positions when viewed as binary vectors of length L. Then, we get the following analog of Theorem 1.

Theorem 4. *Assume C is such that the collection $\left\{C(x) \mid x \in \{0,1\}^\ell\right\}$ is 4-wise prefix-independent and has prefix-distance at least 3. Then our construction is a* VRF *under the weak (and thus regular)* sf-DDH *assumption of order $L+1$.*

Proof. By Lemma 2, we only need to show that no vector $C(w)$ is linearly dependent on 3 vectors z_1, z_2, z_3 inside the sets $\mathsf{Prefixes}(C(x_1)), \ldots, \mathsf{Prefixes}(C(x_t))$, $Pub(L+1)$. Assuming the contrary, if none of z_1, z_2, z_3 comes from $Pub(L+1)$, we would exactly get that the collection $\left\{C(x) \mid x \in \{0,1\}^\ell\right\}$ is 4-wise prefix-dependent, a contradiction. Otherwise, some z_i's (say, z_1) is one of $\{\{L+1\}, \{L+1, 1\}, \ldots, \{L+1, L\}\}$. Since these are the only sets containing element $(L+1)$, in order to "cancel" $(L+1)$ one other z_i (say, z_2) also comes from this collection, which means that $z_1 \oplus z_2$ is some subset of I of $[L]$ or cardinality *at most* 2. The only way we can now have $C(w) \oplus I \oplus z_3 = 0$, is if some z_3 was a prefix of some $C(x_j)$ (where $x_j \neq w$) which differs from $C(w)$ in at most 2 coordinates. But this is exactly what is ruled out by the fact the collection $\left\{C(x) \mid x \in \{0,1\}^\ell\right\}$ has prefix-distance at least 3.

Constructing the Encoding. It remains to construct a 4-wise prefix-independent encoding of prefix distance at least 3. We do it by giving a simple generic transformation from any regular 4-wise independent encoding $C : \{0,1\}^\ell \rightarrow \{0,1\}^L$, such as the encoding $(x^3 \| x)$ considered in the previous section. We will assume without loss of generality that every two distinct elements $C(x)$ and $C(w)$ differ in at least two positions. For example, this is true with the 4-wise independent encoding $(x^3 \| x)$ constructed in the previous section. However, even if originally false in C, one can always increase L by 1 by adding a "parity" bit to C (i.e., the XOR of all the bits of $C(x)$) and get the required distance at least 2 between distinct codewords. Also, for a technical reason we will exclude the zero vector 0^ℓ from the domain of our new encoding.

Lemma 3. *If C is 4-wise independent (and has distance at least 2), then $C'(x) = (C(x) \| 1 \| x \| 1)$ is 4-wise prefix-independent and has prefix-distance at least 3.*

Proof. Below we will refer to the two 1's in the definition of C' as "middle" and "last". We start with showing the prefix distance. Take any $x \neq w$ and consider any prefix I of $C'(w)$. This prefix either "crosses" both the middle and the last 1, only the middle 1, or none of them. In the first case (i.e., we look at $C'(w)$ itself), we get distance three between $C'(x)$ and $C'(w)$ since $C(x)$ differs from $C(w)$ in at least two locations, and x differs from w in at least one more location. In the second case, $C(x)$ still differs from $C(w)$ in at least two locations, and now also I does not have the last 1 which $C'(x)$ has. Finally, in the last case (no 1's are crossed), I does not have both 1's that $C'(x)$ has, and also in between the 1's x is non-zero (this is where we exclude 0^ℓ) while the prefix I is zero, giving distance at least 3 again.

Next, we show the 4-wise prefix independence. Take any x, w_1, w_2, w_3 where $x \notin \{w_1, w_2, w_3\}$, and let z_1, z_2, z_3 be some prefixes of $C'(w_1), C'(w_2), C'(w_2)$ such that $(C(x) \| 1 \| x \| 1) \oplus z_1 \oplus z_2 \oplus z_3 = 0$. Notice, in order to cancel the last 1 of $C'(x)$, at least one of the prefixes, say z_1, has to be full; i.e., $z_1 = C'(w_1) = C(w_1) \| 1 \| w_1 \| 1$. Since the middle 1's cancel out in $C'(x) \oplus C'(w_1)$, we have two possibilities for them to cancel in the full sum $C'(x) \oplus C'(w_1) \oplus z_2 \oplus z_3$. Either both prefixes z_2 and z_3 cross the middle 1, or none does. In the first case, taking the "C-prefixes" we get that $C(x) \oplus C(w_1) \oplus C(w_2) \oplus C(w_3) = 0$, which contradicts the fact that C is 4-wise independent. In the second case, we get that the identity parts between the 1's yield $x \oplus w_1 = 0$, i.e. $x = w_1$, which is again a contradiction.

Applying this Lemma to the encoding $C(x) = (x^3 \| x)$ used in Theorem 2, we get:

Theorem 5. *The encoding $C'(x) = (x^3 \| x \| 1 \| x \| 1)$ defines a* VRF *mapping ℓ bits (except 0^ℓ) to an element of $\mathbb{G}$, under the (weak)* $\mathsf{sf\text{-}DDH}$ *assumption of order $3\ell + 3$.*

Reducing the Order. Similarly to Theorem 3, we apply an "outer" error-correcting code to reduce the order of the $\mathsf{sf\text{-}DDH}$ assumption we need for Theorem 5. However, we need to be sure that our construction preserves prefix-independence. Here is one direct way of doing it if we start — as in Lemma 3 — from any regular 4-wise independent (but perhaps not prefix-independent) $C : \{0,1\}^\ell \to \{0,1\}^L$ with minimum distance 2. Let $E_1 : \{0,1\}^L \to \{0,1\}^{N_1}$ and $E_2 : \{0,1\}^\ell \to \{0,1\}^{N_2}$ be two linear error correcting codes, both correcting some constant fraction of errors. We define the final encoding $\tilde{C}(x) = (E_1(C(x)) \| 1 \| E_2(x) \| 1)$ which maps ℓ non-zero bits to $N_1 + N_2 + 2 = O(\ell)$ bits. By carefully combining the arguments in Theorem 3 with the technique in Lemma 3, we get the following corollary:

Theorem 6. *Assume (weak)* $\mathsf{sf\text{-}DDH}$ *assumption holds for any order $p = O(\log k)$. Then the code $\tilde{C}$ above defines a* VRF*.*

As earlier, using a very good code we can in principle construct a VRF with final expansion $N = (3 + \varepsilon)\ell$ based of the $\mathsf{sf\text{-}DDH}$ assumption of order $O(\log k)$.

Finally, we remark that with an extra overhead of 2 in the expansion of $\tilde{C}$ (and a large polynomial loss in exact security), we can reduce our PRF and VRF constructions in both Theorem 3 and Theorem 6 to using the *full target* $\mathsf{sf\text{-}DDH}$ assumption of order $O(\log k)$. We omit the details due to space constraints.

4 Distributed VRF

In this section we show that our VRF construction can be easily made distributed, which results in the first DVRF construction. Our construction is extremely simple and reminds DPRF construction of Nielsen [17] based on regular DDH. However, the fact that DDH is easy implies we can make our construction non-interactive (i.e., servers do not need to know about each other) and more efficient than that of Nielsen. We start by presenting our model, and then show our simple construction.

The Model. We assume there are n servers $S_1, \ldots, S_n$ and that we have a regular VRF $V = (\mathsf{Gen}, \mathsf{Prove}, \mathsf{Setup})$ which we want to distribute. First, we define the syntax of the new generation algorithm $\mathsf{Gen}'(\cdot)$ run by the trusted party. $\mathsf{Gen}'(1^k)$ not only outputs the public/secret keys PK and SK for V, but also a pair of public/secret key (PK_i, SK_i) for each server S_i. The global secret key SK is then erased, each server S_i gets SK_i, and the values $(PK, PK_1, \ldots, PK_n)$ are published. When a user U approaches the server S_i with input x, the server determines if the user is qualified to learn the value/proof of $F(x)$. How this is done is specifed by the application at hand and is unimportant to us. If U is successful, though, we say that S_i was *initiated* on input x, and U and S_i engage in a possibly interactive protocol. To successfully complete this protocol, the user might have to simultaneously interact with several servers in some possibly predefined order (see below), but the servers do not need to interact to each other or know each other's state. Given a threshold t of the systems, the *robustness* property states that if U contacts s servers on input x, and at least at least $(t+1)$ of these servers are honest, then at the end of the protocol the user learns the unique correct output of $\mathsf{Prove}(x)$; i.e., the value $F(x)$ and the proof $\pi(x)$. This should hold even if the remaining $(s-t-1)$ of the contacted servers are malicious. We notice also that while the user U needs to know the "local' public key PK_i of server i in order to interact with server S_i, any outside party only needs to know the "global" public key PK in order to verify the consistency of $F(x)$ and $\pi(x)$. In other words, the verification algortihm Verify does not have to be changed from the non-distributed setting.

The *security* property of the DVRF protocol states that for any t indices $i_1, \ldots, i_t$ and for any adversary $A = (A_1, A_2)$ who "breaks" the security of DVRF by "corrupting" servers $S_{i_1}, \ldots, S_{i_t}$ (see below), there exists an adversary $B = (B_1, B_2)$ which breaks the pseudorandomness property of our original VRF, as given by Definition 1. We now define what it means to break the security of DVRF. In addition to the public key $(PK, PK_1, \ldots, PK_n)$, A learns the values $SK_{i_1}, \ldots, SK_{i_t}$ of the corrupted servers. Then, A_1 runs in the first stage, in which it is given the ability to interact with any honest servers S_j on arbitrary inputs and in any manner that A_1 desires. However, we keep track of the set of inputs I which were initiated by A_1. At the end of the phase, A_1 outputs the challenge input x (and the state information for A_2). Then A_2 is given back a challenge y_b (for random b), which is either the value $y_0 = F(x)$ or a random element y_1 in the range of F. A_2 can then again interact with honest servers, just like A_1 did. At the end, A_2 outputs the guess $\tilde{b}$ and succeeds if $\tilde{b} = b$ and neither A_1 nor A_2 initiated the input x with any of the servers. A breaks the scheme if it succeeds with non-negligible advantage over $1/2$.

Construction. In Section 3.2 we defined a general candidate for VRF parametrized by any encoding C. We now show how to make such construction distributed for any C for which the basic construction is a VRF. The construction is quite simple, but it shows how convenient it is to have verifiability (given by the easiness of DDH) "for free". Recall that we had $SK = (g, a_1, \ldots, a_L)$;

$PK = (\mathbb{G}, q, g, h, y_1 = h^{a_1}, \dots, y_L = h^{a_L})$; and $\mathsf{Prove}_{SK}(x) = (\sigma_1, \dots \sigma_L)$, where $\sigma_0 = g$, $\sigma_j = \sigma_{j-1}^{a_j}$ if $C(x)_j = 1$ and $\sigma_j = \sigma_{j-1}$ otherwise.

To distribute this process, for every $j = 1 \dots L$ we use Shamir's $(t+1, n)$-secret sharing [18] over $\mathbb{Z}_q$ to split each a_j into n shares $(a_{j,1}, \dots, a_{j,n})$, so that any $t+1$ of these shares suffice to recover a_j, while t or fewer shares give no information about a_j. We set the secret key SK_i of server i to $(a_{1,i}, \dots, a_{L,i})$, and its public key PK_i to $(y_{1,i} = h^{a_{1,i}}, \dots, y_{L,i} = h^{a_{L,i}})$. To compute $\mathsf{Prove}(x)$, the user U needs to contact at least $(t+1)$ honest servers. The protocol with the contacted S_i's proceeds in rounds. Assuming inductively that the value σ_{j-1} is known to both the user and the servers (with the base being $\sigma_0 = g$ which is known to everybody), we show how to compute σ_j. If $C(x)_j = 0$, $\sigma_j = \sigma_{j-1}$, so we are done. Otherwise, each server S_i sends the value $\sigma_{j,i} = \sigma_{j-1}^{a_{j,i}}$ to the user. The user locally checks that $(\sigma_{j-1}, \sigma_{j,i}, h, y_{j,i})$ form a proper DDH-tuple. If they do not, U discards the share and stops interacting with S_i. Upon receiving at least $(t+1)$ correct shares, U uses the corresponding Lagrange interpolation in the exponent to compute the (necessarily correct) value σ_j, and sends σ_j to all the servers it is communicating with. Each server S_i, upon receiving σ_j, checks if $(\sigma_{j-1}, \sigma_j, h, y_j)$ form a valid DDH-tuple. If they do not, the server stops the interaction with U. Then the protocol proceeds to the next round until the entire output is computed.

Security. The security of the above scheme is quite straightforward. Robustness is immediate since every share is checked for consistency. As for pseudorandomness, consider any successful distributed adversary $A = (A_1, A_2)$ who corrupts servers $i_1 \dots i_t$. We build $B = (B_1, B_2)$ for our original VRF as follows. B picks random values $a_{j,i_s} \in \mathbb{Z}_q$ for every $j \in [L]$ and $s \in [t]$, and gives the resulting secret keys $SK_{i_1}, \dots, SK_{i_t}$ to A. It then computes the induced public keys $PK_{i_1}, \dots, PK_{i_t}$ and uses its own public key $h^{a_1}, \dots, h^{a_L}$ to compute the remaining public keys PK_i for all non-corrupted users. This is done by performing the appropriate Lagrange interpolation in the exponent which computes the value $y_{j,i}$ from $y_j, y_{j,i_1}, \dots, y_{j,i_t}$. It hands all these public keys to A, after which B_1 starts running A_1. When A_1 initiates any server on input x, B_1 asks for the value $\mathsf{Prove}(x)$, and uses the response $(\sigma_1, \dots, \sigma_L)$, together with the knowledge of $SK_{i_1}, \dots, SK_{i_t}$, to compute all the relevant shares $\sigma_{j,i}$ (by again doing straightforward Lagrange interpolation in the exponent; details are obvious and omitted). This allows B_1 to simulate all the responses to A_1. After B_1 outputs the same challenge x' as A_1, B_2 gets the output challenge y', which it forwards to A_2 as well. Then B_2 simulates A_2's interaction with the servers in exactly the same way B_1 did it for A_1. Finally, B_2 outputs the same guess $\tilde{b}$ as A_2, which completes the reduction and the proof of security.

Efficiency. The above protocol is quite efficient. The communication complexity is $O(t\ell k)$, and the round complexity is $L = O(\ell)$.

Acknowledgments

I would like to thank Alexander Barg and Venkatesan Guruswami for useful discussions about BCH codes. I would also like to thank Dan Boneh and Don

Coppersmith for their preliminary (positive) evaluation of the sf-DDH assumption. Finally, I would like to thank Anna Lysyanskaya for inspiring this work.

References

[1] Noga Alon and Joel Spencer. *Probabilistic Method*. Wiley, John and Sons, 2000. 9

[2] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communication Security*, pages 62–73, November 1993. Revised version appears in http://www-cse.ucsd.edu/users/mihir/papers/crypto-papers.html. 1

[3] Dan Boneh and Matthew Franklin. Identity based encryption from the weil pairing. In Kilian [11], pages 213–229. 7

[4] Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. IACR E-print Archive. Available from http://eprint.iacr.org/2002/080/, 2002. 7

[5] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, pages 209–218, Dallas, Texas, 23–26 May 1998. 1

[6] Yevgeniy Dodis. Efficient construction of (distributed) verifiable random functions. IACR E-print Archive. Available from http://eprint.iacr.org/2002/133/, 2002. 10

[7] O. Goldreich and L. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing*, pages 25–32, Seattle, Washington, 15–17 May 1989. 3

[8] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, October 1986. 2, 5

[9] Antoine Joux. A one-round protocol for tripartite diffie-hellman. In *ANTS-IV Conference*, volume 1838 of *Lecture Notes in Computer Science*, pages 385–394. Spring-Verlag, 2000. 1

[10] Antoine Joux and Kim Nguyen. Separating decision Diffie-Hellman from Diffie-Hellman in cryptographic groups. IACR E-print Archive. Available from http://eprint.iacr.org/2001/003/, 2001. 1, 4, 7

[11] Joe Kilian, editor. *Advances in Cryptology—CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*. Springer-Verlag, 19–23 August 2001. 16

[12] Anna Lysyanskaya. Unique signatures and verifiable random functions from the dh-ddh separation. In Yung [21]. 1, 2, 3, 5, 6, 7, 10

[13] Silvio Micali, Michael Rabin, and Salil Vadhan. Verifiable random functions. In *40th Annual Symposium on Foundations of Computer Science*, pages 120–130, New York, October 1999. IEEE. 1, 2, 3

[14] Silvio Micali and Ray Sidney. A simple method for generating and sharing pseudo-random functions. In Don Coppersmith, editor, *Advances in Cryptology—CRYPTO '95*, volume 963 of *Lecture Notes in Computer Science*, pages 185–196. Springer-Verlag, 27–31 August 1995. 3

[15] Moni Naor, Benny Pinkas, and Omer Reingold. Distributed pseudo-random functions and KDCs. In Stern [20], pages 327–346. 3

[16] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *38th Annual Symposium on Foundations of Computer Science*, pages 458–467, Miami Beach, Florida, 20–22 October 1997. IEEE. 1, 2, 4, 5, 6, 9

[17] Jesper Nielsen. Threshold pseudorandom function construction and its applications. In Yung [21]. 1, 4, 5, 13

[18] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979. 15

[19] Michael Steiner, Gene Tsudik, and Michael Waidner. Diffie-hellman key distribution extended to group communicatio. In *Third ACM Conference on Computer and Communication Security*, pages 31–37. ACM, March 14–16 1996. 7, 9

[20] Jacques Stern, editor. *Advances in Cryptology—EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*. Springer-Verlag, 2–6 May 1999. 16

[21] Moti Yung, editor. *Advances in Cryptology—CRYPTO 2002*, Lecture Notes in Computer Science. Springer-Verlag, 18–22 August 2002. 16, 17