Unique Signatures and Verifiable Random Functions from the DH-DDH Separation

Anna Lysyanskaya
MIT LCS
200 Technology Square
Cambridge, MA 02139 USA
anna@theory.lcs.mit.edu

Abstract. A *unique* signature scheme has the property that a signature $\sigma_{PK}(m)$ is a (hard-to-compute) function of the public key PK and message m, for all, even adversarially chosen, PK. Unique signatures, introduced by Goldwasser and Ostrovsky, have been shown to be a building block for constructing verifiable random functions. Another useful property of unique signatures is that they are stateless: the signer does not need to update his secret key after an invocation.

The only previously known construction of a unique signature in the plain model was based on the RSA assumption. The only other previously known provably secure constructions of stateless signatures were based on the Strong RSA assumption. Here, we give a construction of a unique signature scheme based on a generalization of the Diffie-Hellman assumption in groups where decisional Diffie-Hellman is easy. Several recent results suggest plausibility of such groups.

We also give a few related constructions of verifiable random functions (VRFs). VRFs, introduced by Micali, Rabin, and Vadhan, are objects that combine the properties of pseudorandom functions (i.e. indistinguishability from random even after querying) with the verifiability property. Prior to our work, VRFs were only known to exist under the RSA assumption.

Keywords. Unique signatures, verifiable random functions, application of groups with DH-DDH separation.

1 Introduction

Signature schemes are one of the most important cryptographic objects. There were invented, together with the entire field of public-key cryptography, by Diffie and Hellman, and Rivest, Shamir and Adleman followed up with the first candidate construction. Goldwasser, Micali and Rivest [GMR88] gave the first signature scheme that is secure even if the adversary is allowed to obtain signatures on messages of its choice. This notion of security for signature schemes is also due to Goldwasser, Micali, and Rivest (GMR).

Since the GMR seminal work, it has become clear that the first requirement from a signature scheme is that it should satisfy the GMR definition of security. However, to be most useful, two additional properties of signature schemes are desirable: (1) that the scheme be secure in the plain model, i.e., without a random oracle or common parameters; and (2) that the scheme be stateless, i.e., not require the signer to update the secret key after each invocation.

The only signature schemes satisfying both of these additional properties are the Strong-RSA-based schemes of Gennaro et al. [GHR99] and of Cramer and Shoup [CS99], and the scheme implied by the verifiable random function due to Micali et al. [MRV99], based on RSA. An open question was to come up with a signature scheme satisfying the two additional properties, such that it would be secure under a different type of assumption. Here, we give such a signature scheme, based on a generalization of the Diffie-Hellman assumption for groups where decisional Diffie-Hellman is easy.

Unique signatures. The signature scheme we propose is a unique signature scheme. Unique signature schemes are GMR-secure signature schemes where the signature is a hard-to-compute function of the public key and the message. They were introduced by Goldwasser and Ostrovsky [GO92][1].

Intuitively, unique signatures are the "right" notion of signatures. This is because if one has verified a signature on a message once, then why should it be necessary to verify the signature on the same message again? Yet, even if indeed the given message has been accepted before, it is a bad idea to just accept it again—what if this time it came from an unauthorized party? Hence, one must verify the signature again, if it happens to be a different signature. If a signature scheme allows the signer to easily (that is to say, more efficiently than the cost of verifying a signature) generate many signatures on the same message, this leads to a simple denial-of-service attack on a verifier who is forced to verify many signatures on the same message. Although this is not a cryptographic attack, it still illustrates that intuitively unique signatures are more desirable.

In the random-oracle model, a realization of unique signatures is well-known. For example, some RSA signatures are unique: $\sigma_{n,e,H}(m) = (H(m))^{1/e} \bmod n$ is a function of (n, e, H, m) so long as e is a prime number greater than n (this is because for a prime e, $e > n$ implies that e is relatively prime to $\phi(n)$). Goldwasser and Ostrovsky give a solution in the common-random-string model. However, in the standard model, the only known construction of unique signatures was the one due to Micali, Rabin, Vadhan [MRV99].

On the negative side, Goldwasser and Ostrovsky have also shown that even in the common random string model, unique signatures require assumptions of the same strength as needed for non-interactive zero knowledge proofs with polynomial-time provers. The weakest assumption known that is required for non-interactive zero knowledge proofs with polynomial-time provers, is existence of trapdoor permutations [FLS99,BY96]. However, we do not know whether this is a necessary assumption; neither do we know whether this assumption is sufficient for constructing unique signatures in the plain model.

Verifiable random functions. Another reason why unique signatures are valuable is that they are closely related to verifiable random functions. Verifiable random functions (VRFs) were introduced by Micali, Rabin, and Vadhan [MRV99]. They are similar to pseudorandom functions [GGM86], except that they are also

[1] They call it an *invariant* signature

verifiable. That is to say, associated with a secret seed SK, there is a public key PK and a function $F_{PK}(\cdot) : \{0,1\}^k \mapsto \{0,1\}^u$ such that (1) $y = F_{PK}(x)$ is efficiently computable given the corresponding SK; (2) a proof $\pi_{PK}(x)$ that this value y corresponds to the public key PK is also efficiently computable given SK; (3) based purely on PK and oracle calls to $F_{PK}(\cdot)$ and the corresponding proof oracle, no adversary can distinguish the value $F_{PK}(x)$ from a random value without explicitly querying for the value x.

VRFs [MRV99] are useful for protocol design. They can be viewed as a commitment to an exponential number of random-looking bits, which can be of use in protocols. For example, using verifiable random functions, one can reduce the number of rounds for resettable zero knowledge proofs to 3 in the bare model [MR01]. Another example application, due to Micali and Rivest [MR02], is a non-interactive lottery system used in micropayments. Here, the lottery organizer holds a public key PK of a VRF. A participant creates his lottery ticket t himself and sends it to the organizer. The organizer computes the value $y = F_{PK}(t)$ on the lottery ticket, and the corresponding proof $\pi = \pi_{PK}(t)$. The value y determines whether the user wins, while the proof π guarantees that the organizer cannot cheat. Since a VRF is hard to predict, the user has no idea how to bias the lottery in his favor.

These objects are not well-studied; in fact, only one construction, based on the RSA assumption, was previously known [MRV99]. Micali, Rabin and Vadhan showed that, for the purposes of constructing a VRF, it is sufficient to construct a unique signature scheme. More precisely, from a unique signature scheme with small (but super-polynomial) message space and security against adversaries that run in super-polynomial time, they constructed a VRF with an arbitrary input size that tolerates a polynomial-time adversary[2] They then gave an RSA-based unique signature scheme for a small message space.

Constructing VRFs from pseudorandom functions (PRFs) is a good problem that we don't know how to solve in the standard model. This is because by its definition, the output of a PRF should be indistinguishable from random. However, if we extend the model to allow interaction, it is possible to commit to the secret seed of a PRF and let that serve as a public key, and then prove that a given output corresponds to the committed seed using a zero-knowledge proof. This solution is unattractive because of the expense of communication rounds. In the so-called *common random string* model where non-interactive zero-knowledge proofs are possible, a construction is possible using commitments and non-interactive zero knowledge [BFM88,BDMP91,FLS99]. However, this is unattractive as well because this model is unusual and non-interactive ZK is expensive. Thus, construction of verifiable random functions from general assumptions remains an interesting open problem.

DH–DDH separation. Recently, Joux and Nguyen [JN01] demonstrated that one can encounter groups in which decisional Diffie-Hellman is easy, and yet

[2] Intuitively, it would seem that the connection ought to be tighter: a unique signature secure against polynomial-time adversaries should imply a secure verifiable random function. This is an interesting open problem, posed by Micali et al.

computational Diffie-Hellman seems hard. This is an elegant result that, among other things, sheds light on how reasonable it is to assume decisional Diffie-Hellman.

Joux [Jou00] proposed using the DH-DDH separation to a good end, by exhibiting a one-round key exchange protocol for three parties. Subsequently, insight into such groups has proved relevant for the recent construction of identity-based encryption due to Boneh and Franklin [BF01] which resolved a long standing open problem [Sha85]. Other interesting consequences of the study of these groups are a construction of a short signature scheme in the random-oracle model [BLS01] and of a simple credential system [Ver01].

Our results. We give a simple construction of a unique signature scheme based on groups where DH is conjectured hard and DDH is easy. Ours is a tree-like construction. The message space consists of codewords of an error-correcting code that can correct a constant fraction of errors. For n-bit codewords, the depth of the tree is n. The root of the tree is labelled with g, a generator of a group where DH is hard and DDH is easy. The 2^n leaves of the tree correspond all the possible n-bit strings. The 2^i nodes of depth i, $1 \leq i < n$ correspond to all the possible i-bit prefixes of an n-bit string.

A pair of group elements $(A_{i,0} = g^{a_{i,0}}, A_{i,1} = g^{a_{i,1}})$ is associated with each depth of the tree as part of the public key. The label of a node of depth i is derived from the label of its parent by raising the parent's label to the exponent $a_{i,0}$ if this node is its parent's left child, or $a_{i,1}$ if it is the parent's right child.

Computing the signature on each codeword m amounts to computing the labels of the nodes on the path from the root of the tree all the way down to the leaf corresponding to m.

At first glance, this may seem very similar to the Naor-Reingold [NR97] pseudorandom function. However the proof is not immediate. The major difference from the cited result is that here we must give a proof that the function was evaluated correctly. That makes it harder to prove security. For example, proof by a hybrid argument, as done by Naor and Reingold [NR97] is ruled out immediately: there is no way we can answer some of the adversary's queries with truly random bits, since for such bits there will be no proof.

Our proof of security for the unique signature relies on a generalization of the Diffie-Hellman assumption. We call it the "Many-DH" assumption. However, for directly converting this simple US to VRF, we need to make a very strong assumption; however the resulting VRF is very simple. We also suggest a more involved but also more secure construction of a VRF based on the Many-DH assumption. This last construction is closely related to that due to Micali et al. [MRV99].

Outline of the rest of the paper. In Section 2 we introduce our notation. In Section 3 we give definitions of verifiable random functions and unique signatures. In Section 4 we state our complexity assumptions for the unique signatures. In Section 5 we give our unique signature and prove it secure in Section 6. We then provide a simple construction for a VRF under a somewhat stronger assumption, in Section 7. We conclude in Section 8 with a construction of a VRF based on the weaker assumption alone, but whose complexity (both in terms of computation and in terms of conceptual simplicity) is the same as that of Micali et al. [MRV99].

2 Notation

The notation in this paper is based on the Cryptography class taught by Silvio Micali at MIT [Mic].

Let $A(\cdot)$ be an algorithm. $y \leftarrow A(x)$ denotes that y was obtained by running A on input x. In case A is deterministic, then this y is unique; if A is probabilistic, then y is a random variable.

Let b be a boolean function. The notation $(y \leftarrow A(x) : b(y))$ denotes the event that $b(y)$ is true after y was generated by running A on input x.

Finally, the statement such as $\Pr[y \leftarrow A(x); z \leftarrow B(y) \ : \ b(z)] = \alpha$ means that the probability that $b(z)$ is TRUE after the value z was obtained by first obtaining y by running algorithm A on input x, and then running algorithm B on input y.

By $A^{O(\cdot)}(\cdot)$, we denote a Turing machine that makes oracle queries to machine O. I.e., this machine will have an additional (read/write-once) query tape, on which it will write its queries in binary; once it is done writing a query, it inserts a special symbol "#". By external means, once the symbol "#" appears on the query tape, an oracle O is invoked on the value just written down, and the oracle's answer appears on the query tape adjacent to the "#" symbol. By $Q = Q(A^{O(\cdot)}(x)) \leftarrow A^{O(\cdot)}(x)$ we denote the contents of the query tape once A terminates, with oracle O and input x. By $(q, a) \in Q$ we denote the event that q was a query issued by A, and a was the answer received from oracle O. Sometimes, we will write, for example, $A^{O(x,\cdot)}$, to denote the fact that the first input to O is fixed, and A's query supplies only the second input to O.

We say that $\nu(k)$ is a negligible function, if for all polynomials $p(k)$, for all sufficiently large k, $\nu(k) < 1/p(k)$.

3 Definitions

3.1 Unique Signatures

Unique signatures are simply secure signatures where the signature is a function as opposed to a distribution.

Definition 1. *A function family $\sigma_{(\cdot)}(\cdot) \ : \ \{0,1\}^k \mapsto \{0,1\}^{\ell(k)}$ is a unique signature scheme (US) if there exists probabilistic algorithm G, efficient deterministic algorithm* `Sign`*, and probabilistic algorithm* `Verify` *such that $G(1^k)$ generates the key pair PK, SK,* `Sign`(SK, x) *computes the value $\sigma = \sigma_{PK}(x)$ and* `Verify`(PK, x, σ) *verifies that $\sigma = \sigma_{PK}(x)$. More formally (but assuming, for simplicity, that* `Verify` *is a deterministic algorithm; in case it is not, the adjustment to the definition is straightforward):*

1. *(Uniqueness of $\sigma_{PK}(m)$) There do not exist values $(PK, m, \sigma_1, \sigma_2)$ such that $\sigma_1 \neq \sigma_2$ and $\texttt{Verify}(PK, m, \sigma_1) = \texttt{Verify}(PK, m, \sigma_2) = 1$.*
2. *(Security) For all families of probabilistic polynomial-time oracle Turing machines $\{A_k^{(\cdot)}\}$, there exists a negligible function $\nu(k)$ such that*

$$\Pr[(PK, SK) \leftarrow G(1^k); \\ (Q, x, \sigma) \leftarrow A_k^{\texttt{Sign}(SK,\cdot)}(1^k); : \texttt{Verify}(PK, x, \sigma) = 1 \wedge (x, \sigma) \notin Q] \leq \nu(k)$$

On a relaxed definition. Goldwasser and Ostrovsky [GO92] give a relaxed definition. In their definition, even though the signature is unique, the verification procedure may require, as an additional input, a proof that the signature is correct. This proof is output by the signing algorithm together with the signature, and it might not be unique. Here we give the stronger definition because we can satisfy it (Goldwasser and Ostrovsky do not, even though they work in the common random string model). However, note that the relaxed definition is sufficient for constructing VRFs [MRV99].

Unique signatures are stateless. Since a unique signature is a function of the public key and the message, a signature on a given message will be the same whether this was the first message signed by the signer, or the n'th message. As a result, it is easy to see that the signer does not need to remember anything about past transactions, i.e., a unique signature must be stateless.

3.2 Verifiable Random Functions

The definition below is due to Micali et al. [MRV99]. We use somewhat different and more compact notation, however.

The intuition of this definition is that a function is a verifiable random function if it is like a pseudorandom function with a public key and proofs.

Definition 2. *A function family $F_{(\cdot)}(\cdot) : \{0,1\}^k \mapsto \{0,1\}^{\ell(k)}$ is a verifiable random function (VRF) if there exist probabilistic algorithm G, and deterministic algorithms* `Eval`*, and* `Prove`*, and algorithm* `Verify` *such that: $G(1^k)$ generates the key pair PK, SK;* $\texttt{Eval}(SK, x)$ *computes the value $y = F_{PK}(x)$;* $\texttt{Prove}(SK, x)$ *computes the proof π that $y = F_{PK}(x)$; and* $\texttt{Verify}(PK, x, y, \pi)$ *verifies that $y = F_{PK}(x)$ using the proof π. More formally (but assuming, for simplicity, that* `Verify` *is a deterministic algorithm; in case it is not, the adjustment to the definition is straightforward):*

1. *(Uniqueness of $F_{PK}(x)$) There do not exist values $(PK, SK, x, y_1, y_2, \pi_1, \pi_2)$ such that $y_1 \neq y_2$ and $\texttt{Verify}(PK, x, y_1, \pi_1) = \texttt{Verify}(PK, x, y_2, \pi_2) = 1$.*
2. *(Computability of $F_{PK}(x)$) $F_{PK}(x) = \texttt{Eval}(SK, x)$.*
3. *(Provability of $F_{PK}(x)$) If $(y, \pi) = \texttt{Prove}(SK, x)$, then*

$$\texttt{Verify}(PK, x, y, \pi) = 1$$

.

4. *(Pseudorandomness of $F_{PK}(x)$) For all families of probabilistic polynomial-time Turing machines $\{A_k^{(\cdot)}, B_k\}$, there exists a negligible function $\nu(k)$ such that*

$$\begin{aligned}
\Pr[(PK, SK) &\leftarrow G(1^k);\\
(Q_A, x, \mathtt{state}) &\leftarrow A_k^{\mathtt{Prove}(SK,\cdot)}(1^k);\\
y_0 &= \mathtt{Eval}(SK, x);\\
y_1 &\leftarrow \{0,1\}^{\ell(k)};\\
b &\leftarrow \{0,1\};\\
(Q_B, b') &\leftarrow B_k^{\mathtt{Prove}(SK,\cdot)}(\mathtt{state}, y_b) : b = b' \wedge (x, \mathtt{Prove}(SK, x)) \notin Q_A \cup Q_B]\\
&\leq 1/2 + \nu(k)
\end{aligned}$$

(The purpose of the state *random variable is so that A_k can save some useful information that B_k will then need.)*
In other words, the only way that an adversary could tell $F_{PK}(x)$ from a random value, for x of its own choice, is by querying it directly.

It is easy to see [MRV99] that given a VRF $F_{(\cdot)} : \{0,1\}^{\ell(k)} \mapsto \{0,1\}$, one can construct a VRF $F'_{(\cdot)} : \{0,1\}^{\ell'(k)} \mapsto \{0,1\}^{m(k)}$, where $\ell'(k) = \ell(k) - \lceil \log m(k) \rceil$, as follows: $F'_S(x_1 \circ \ldots \circ x_{\ell'(k)}) = F_S(x_1 \circ \ldots \circ x_{\ell'(k)} \circ u_0) \circ F_S(x_1 \circ \ldots \circ x_{\ell'(k)} \circ u_1) \circ \ldots \circ F_S(x_1 \circ \ldots \circ x_{\ell'(k)} \circ u_{m(k)})$ where u_i denotes the $\lceil \log m(k) \rceil$-bit representation of the integer i, and "$\circ$" denotes concatenation.

Thus in the sequel we will focus on constructing VRFs with binary outputs.

Unique signatures vs. VRFs. Micali et al. showed how to construct VRFs from unique signatures. The converse, namely construction of a unique signature from a VRF, holds immediately if the proofs in the VRFs are unique. If the proofs are not unique, then no construction satisfying our strong definition of unique signatures in known. However, constructing relaxed unique signatures in the sense of Goldwasser and Ostrovsky (see the end of Section 3.1) is immediate.

4 Assumptions

Let S be the algorithm that, on input 1^k, generates a group $G = (*, q, g)$ with efficiently computable group operation $*$, of prime order q, with generator g. We require that g is written down in binary using $O(\log q)$ bits, and that every element of the group has a unique binary representation.

We require that the decisional Diffie-Hellman problem be easy in G. More precisely, we require that there is an efficient algorithm D for deciding the following language $L_{DDH}(G)$:

$$L_{DDH}(G) = \{(*, q, g, X, Y, Z) \mid \exists x, y \in \mathbb{Z}_q \text{ such that } X = g^x, Y = g^y, Z = g^{xy}\}$$

One the other hand, we will need to make the following assumption which is somewhat stronger than computational Diffie-Hellman. It is essentially like computational Diffie-Hellman, except that instead of taking two inputs, g^x and g^y, and the challenge is computing g^{xy}, we have a logarithmic number of bases $g^{y_1}, \ldots, g^{y_\ell}$, as well as all the products $g^{\prod_{j \in J} y_j}$ for all proper subsets J of the naturals up to and including ℓ, and the challenge is to come up with the value $g^{\prod_{j=1}^{\ell} y_j}$.

Assumption 1 (Many-DH Assumption) *For all $\ell = O(\log k)$, for all probabilistic polynomial-time families of Turing machines $\{A_k\}$,*

$$\begin{aligned}\Pr[(*, q, g) \leftarrow S(1^k); \{y_i \leftarrow \mathbb{Z}_q \ : \ 1 \leq i \leq \ell\};\\ \{z_J = \prod_{j \in J} y_j; Z_J = g^{z_J} \ : \ J \subset [\ell]\};\\ Z \leftarrow A_k(*, q, g, \{Z_J \ : \ J \subset [\ell]\}) : Z = g^{\prod y_i}] \leq \nu(k)\end{aligned}$$

For an exposition of groups of this flavor, where the decisional Diffie-Hellman problem is easy, and yet generalizations of the computational Diffie-Hellman problem are conjectured hard, we refer the reader, for example, to the recent papers by Joux and Nguyen [JN01], Joux [Jou00] and Boneh and Franklin [BF01]. In the sequel, we will not address the number-theoretic aspects of the subject.

5 Construction of a Unique Signature

Suppose the algorithms S, D, as in Section 4, are given. Let k be the security parameter. Let the message space consist of strings of length n_0. Our only assumption on the size of the message space is that $n_0 = \omega(\log k)$.

Let $C \ : \ \{0,1\}^{n_0} \mapsto \{0,1\}^n$ be an error-correcting code of distance cn, where $c > 0$ is a constant. In other words, C is a function such that if $M \neq M'$ are strings of length n_0, then $C(M)$ differs from $C(M')$ in at least cn places. For an overview of error-correcting codes, see the lecture notes of Madhu Sudan [Sud]. Here, we note that since we will not need the decoding operation, only the encoding operation, we can easily achieve $n = O(n_0)$.

We need to construct algorithms G, `Sign`, `Verify` as specified in Definition 1.

Algorithm G Run $S(1^k)$ to obtain$G = (*, q, g)$. Choose n pairs of random elements in $\mathbb{Z}_q$: $(a_{1,0}, a_{1,1}), \ldots, (a_{n,0}, a_{n,1})$. Let $A_{i,b} = g^{a_{i,b}}$, for $1 \leq i \leq k$, $b \in \{0,1\}$. Output the following key pair:

$$SK = \begin{array}{|c|c|c|c|}\hline a_{1,0} & a_{2,0} & \cdots & a_{n,0} \\ \hline a_{1,1} & a_{2,1} & \cdots & a_{n,1} \\ \hline\end{array} \qquad PK = \begin{array}{|c|c|c|c|}\hline A_{1,0} & A_{2,0} & \cdots & A_{n,0} \\ \hline A_{1,1} & A_{2,1} & \cdots & A_{n,1} \\ \hline\end{array}$$

Algorithm `Sign` On input a message M of length n_0, compute the encoding of M using code C: $m = C(M)$. To sign the n-bit codeword $m = m_1 \circ \ldots \circ m_n$, output $\sigma_{PK}(m) = (s_1, \ldots, s_n)$, where $s_0 = g$ and $s_i = (s_{i-1})^{a_{i,m_i}}$ for $1 \leq i \leq n$.

Algorithm `Verify` Let $s_{m,0} = 1$. Verify that, for all $1 \leq i \leq n$,

$$D(*, q, g, s_{m,i-1}, A_{i,m_i}, s_{m,i}) = ACCEPT$$

Graphically, we view the message space as the leaves of a balanced binary tree of depth n. Each internal node of the tree is assigned a label, as follows: the label of the root is g. The label of a child, denoted l_c is obtained from the label of its parent, denoted l_p, as follows: if the depth of the child is i, and it is the left child, then its label is $l_c = l_p^{a_{i,0}}$, while if it is the right child, its label will be $l_c = l_p^{a_{i,1}}$. The signature on an n-bit message consists of all the labels on the path from the leaf corresponding to this message all the way to the root. To verify correctness of a signature, one uses the algorithm D that solves the DDH for the group over which this is done.

Efficiency. In terms of efficiency, this signature scheme is a factor of $O(n)$ worse than the Cramer-Shoup scheme, in all parameters such as the key and signature lengths and the complexity of relevant operations. This means that it is still rather practical, and yet has two benefits: uniqueness, and security based on a different assumption. In comparison to the unique signature of Micali et al., our construction is preferable as far as efficiency is concerned. This is because our construction is direct, while they first give a unique signature for short messages, then show how to construct a VRF and a unique signature of arbitrary length from that.

Reducing the length of signatures. Boneh and Silverberg [BS02] point out that, if the language $L(*, q, g) = \{g^{y_1}, \ldots, g^{y_n}, g^{\prod_{i=1}^{n} y_i}\}$ is efficiently decidable, then the signature does not need to contain the labels of the intermediate nodes. I.e., to sign a message M, $m = C(M)$, it is sufficient to give $s = g^{\prod_{i=1}^{n} a_{i,m_i}}$. This reduces the length of a signature by a factor of n. However, finding groups where $L(*, q, g)$ is efficiently decidable, and yet the Many-DH Assumption is still reasonable, is an open question [BS02].

6 Proof of Security for the Unique Signature

In this section, we show how to reduce breaking the Many-DH problem to forging a signature of the construction in Section 5.

First, we show the following lemma:

Lemma 1. *Suppose* `Verify`$(((*, q, g), \{A_{i,b}\}_{1 \leq i \leq n, b \in \{0,1\}}), m, (s_1, \ldots, s_n)) = 1$. *Then* $s_j = g^{\prod_{i=1}^{j} a_{i,m_i}}$, *where* a_{i,m_i} *denotes the unique value* $\mathbb{Z}_q$ *such that* $g^{a_{i,m_i}} = A_{i,m_i}$.

Proof. We show the lemma by induction on j. For $j = 1$, the verification algorithm will accept only if $s_1 = A_{1,m_i}$.

Let the lemma hold for $j-1$. The verification algorithm will accept s_j only if $D(*, q, g, s_{j-1}, A_{j,m_j}, s_j) = 1$. But by definition of D, this is the case only if $s_j = g^{\sigma a_{j,m_j}}$, where σ is the unique value in $\mathbb{Z}_q$ such that $g^\sigma = s_{j-1}$. By the induction hypothesis, $\sigma = \prod_{i=1}^{j-1} a_{i,m_i}$. Therefore, $s_j = g^{\sigma a_{j,m_j}} = g^{\prod_{i=1}^{j} a_{i,m_i}}$. $\square$

6.1 Description of the Reduction

In the following, "we" refers to the reduction, and the forger for the signature scheme is "our adversary."

Recall that k is the security parameter, and that $n_0 = \omega(\log k)$, $n = O(n_0)$.

Suppose that our adversary's running time is $t(k)$. Recall that c is the distance of the error-correcting code we use. Let $\ell = \frac{\lceil \log t(k) \rceil + 2}{\log(1/(1-.99c))} + 1$. (Note that $\ell = O(\log k)$). Assume G is as in Section 4.

Input to the reduction: As in Assumption 1, we are given a group $G = (*, q, g)$ and ℓ elements $Y_1, Y_2, \ldots, Y_\ell$ of G, where $Y_u = g^{y_u}$. We are also given the values $Z_I = g^{\prod_{u \in I} y_u}$, for $I \subset [\ell]$. The goal is to break Assumption 1, i.e., compute $g^{\prod_{u=1}^{\ell} y_u}$.

Key generation: Pick a random ℓ-bit string $B = b_1 \circ \ldots \circ b_\ell$ and a random ℓ-bit subset $J \subset [n]$. (For simpler notation, assume that the indices are ordered such that $j_u < j_{u+1}$ for all $j_u \in J$.)

Set up the public key as follows: $A_{j_u, b_u} = Y_u$. To set up $A_{i,b}$ where $i \notin J$ or $i = j_u \in J$ but $b \neq b_u$, choose value $a_{i,b} \leftarrow \mathbb{Z}_q$ and set $A_{i,b} = g^{a_{i,b}}$. Figure 1 gives an example of what the reduction does in this step.

$$PK = \begin{array}{|c|c|c|c|c|c|c|c|c|c|} \hline g^{a_{1,0}} & g^{a_{2,0}} & g^{a_{3,0}} & Y_2 & g^{a_{5,0}} & g^{a_{6,0}} & g^{a_{7,0}} & g^{a_{8,0}} & g^{a_{9,0}} & g^{a_{10,0}} \\ \hline g^{a_{1,1}} & g^{a_{2,1}} & Y_1 & g^{a_{4,1}} & g^{a_{5,1}} & g^{a_{6,1}} & g^{a_{7,1}} & g^{a_{8,1}} & Y_3 & g^{a_{10,1}} \\ \hline \end{array}$$

Fig. 1. Toy example of a public key produced by the reduction. In this example, $n = 10$, $\ell = 3$. The reduction randomly picked $J = \{3, 4, 9\}$, $B = 101$.

Responding to signature queries: We will respond to at most $2t$ signature queries from the adversary, as follows.

Suppose the adversary's query is M where $C(M) = m = m_1 \circ \ldots \circ m_n$. Let $J(m)$ denote the string $m_{j_1} \circ \ldots \circ m_{j_\ell}$.

Check if $J(m) = B$. If so, terminate with "Fail." Otherwise, compute the signature as follows: Let $Z_0 = g$. Let $b' = b'(J)$ be an n-bit string such that $b_{j_u} = b_u$ for all $j_u \in J$. By I_u, we denote the ℓ-bit string which has a 1 in position u, and 0's everywhere else.

Compute $(s_1, \ldots, s_n)$ using the following loop:

Initialize $c := 0^\ell$
  For $i = 1$ to n
    (1) If $i = j_u \in J$ and $m_i = b_u$, then $c := c \oplus I_u$
    (2) $s_i = Z_c^{\prod_{j \notin J, j \leq i} a_{j,m_j} \prod_{j \in J, j \leq i, m_j \neq b'_j} a_{j,m_j}}$
    (3) Increment i
  (end of loop)

Processing the forgery: Now suppose that the adversary comes back with a forged signature on message M' for which it has not queried R. Let $m' = C(M')$. This forgery is $\sigma_{PK}(m') = \{s_{m',i}\}$. This forgery is *good* if $J(m') = B$. If the forgery is good, we obtain the value $g^{\prod_{u=1}^{\ell} y_u}$ by Lemma 1, by simply computing $(s_{m',i})^{1/\prod_{i \notin J} a_i^{m'_i}}$.

6.2 Analysis of the Reduction

Let $t = t(k)$ be the expected running time of the adversary. For the purposes of the analysis, consider the following algorithms:

- Algorithm 1 runs the signing algorithm and responds to the adversary's queries as the true signing oracle would. If the adversary outputs a valid forgery, this algorithm outputs "Success."
- Algorithm 2 runs the signing algorithm but only responds to $2t$ queries. If the adversary issues more queries, output "Fail." Otherwise, if the adversary outputs a valid forgery, this algorithm outputs "Success."
- Algorithm 3 is the same as Algorithm 2, except in one case. Namely, it chooses a random $J \subset [n]$, $|J| = \ell$, $J = \{j_1, \ldots, j_\ell\}$, and outputs "Fail" if it so happens that $J(C(M')) = J(C(M))$ where M' is the adversary's forgery, and M is any previously queried message, and notation $J(m)$ denotes $m_{j_1} \circ m_{j_2} \circ \ldots \circ m_{j_\ell}$.
- Algorithm 4 is just like Algorithm 3 except in one case Namely, it chooses a random ℓ-bit string B and outputs "Fail" whenever the forged message M' is such that $C(M') = m'$ where $J(m') \neq B$.
- Algorithm 5 is just like Algorithm 4 except in one case. Namely, it outputs "Fail" whenever the queried message M is such that $C(M) = m$ where $J(m) = B$.
- Algorithm 6 runs our reduction. It outputs "Success" whenever the reduction succeeds in computing its goal.

By p_i, let us denote the success probability of algorithm i.

Lemma 2. *The success probability of Algorithm 1 is the same as the success probability of the forger.*

Proof. By construction, this algorithm outputs "Success" iff the forger succeeds. □

Lemma 3. $p_2 \geq p_1/2$.

Proof. Suppose a successful forgery requires t queries on the average. Then by the Markov inequality, $2t$ queries are sufficient at least half the time. □

Lemma 4. $p_3 \geq p_2/2$.

Proof. Recall that $m' = C(M')$ is a codeword of an error-correcting code of distance cn. That implies that for all M, m' differs from $m = C(M)$ on at least cn locations. So, if we picked ℓ locations at random with replacement, $\Pr_J[J(m) = J(m')] \leq (1-c)^{-\ell}$. Since we are picking without replacement, $\Pr_J[J(m) = J(m')] \leq \prod_{i=1}^{\ell}(n - cn - i)/n \leq (1 - c + \epsilon)^{\ell}$ for any constant $\epsilon > 0$. Let $\epsilon = .01c$ for simplicity. Let Q denote the set of messages queried by the adversary. By the union bound

$$\Pr_J[\exists M \in Q \text{ such that } J(C(M)) = J(m')] \leq 2t(1 - c + \epsilon)^{\ell} < 1/2$$

if $4t < (1/(1 - c + \epsilon))^{\ell}$. Taking the logarithm on both sides of the equation, we get the condition $\log t + 2 < \ell \log(1/(1 - .99c))$, and so since we have set $\ell > \frac{\log t + 2}{\log(1/(1-.99c))}$, this is satisfied. □

Lemma 5. $p_4 \geq p_3/2^{\ell}$.

Proof. Note that Algorithm 3 and Algorithm 4 can be run with the same adversary, but Algorithm 4 may output "Fail" while Algorithm 3 outputs "Success." Consider the case when both of them output "Success." In this case, (1) $J(C(M')) \neq J(C(M))$ for all previously queried M, and (2) $J(C(M)) = B$. Note that, given that (1) is true, the probability of (2) is exactly $2^{-\ell}$. □

Lemma 6. $p_5 = p_4$.

Proof. Note that the only difference between the two algorithms is that Algorithm 5 will sometimes output "Fail" sooner. Algorithm 4 will continue answering queries, but, if Algorithm 5 has output "Fail," then $J(C(M')) \neq J(C(M)) = B$, and so Algorithm 4 will output "Fail" as well. □

Lemma 7. $p_6 = p_5$.

Proof. First, note that whether we are running Algorithm 5 or Algorithm 6, the view of the adversary is the same. Namely: (1) the public key is identically distributed; (2) both algorithms respond to at most $2t$ signature queries; (3) both algorithms pick J and B uniformly at random and refuse to respond to a signature query M if $J(C(M)) = B$. Therefore, the probability that the adversary comes up with a forgery in the two cases is the same. Now, note that the probability that the forgery is good is also the same: in both cases, the forgery is good if $J(C(M')) = B$. □

Putting these together, we have:

Lemma 8. *If the forger's success probability is p, and expected running time is t, then the success probability of the reduction is $p/2^{\ell+2} = p/O(t)$.*

In turn, this implies the following theorem:

Theorem 1. *Under Assumption 1, the construction presented in Section 5 is a unique signature.*

7 A Simple Construction of a VRF

Consider the following, rather strong, complexity assumption:

Assumption 2 *(***Very-Many-DH-Very-Hard Assumption***) There exists a constant $\epsilon > 0$ such that for all probabilistic polynomial-time families of Turing machines $\{A_k\}$ with running time $O(2^{k^\epsilon})$,*

$$\Pr[(*, q, g) \leftarrow S(1^k); \{y_i \leftarrow \mathbb{Z}_q \ : \ 1 \leq i \leq k^\epsilon\};$$
$$Z \leftarrow A_k^{\mathcal{O}[*,q,g,\{y_i\}](\cdot)}(*, q, g, \{g^{y_i} \ : \ 1 \leq i \leq k^\epsilon\}) : Z = g^{\prod y_i}] \leq poly(k) * 2^{-2k^\epsilon}$$

where $\mathcal{O}[*, q, g, \{y_i\}](\cdot)$, on input an k^ϵ-bit string I, outputs $g^{\prod_{i=1}^{k^\epsilon} y_i^{I_i}}$ iff I is not an all-1 string.

Definition 3 (VUF [MRV99]). *A verifiable unpredictable function (VUF) $(G, \texttt{Eval}, \texttt{Prove}, \texttt{Verify})$ with input length $a(k)$, output length $b(k)$, and security $s(k)$ is defined as a VRF except that requirement 4 of Definition 2 is replaced with the following: Let $T(\cdot,\cdot)$ be any oracle algorithm that runs in time $s(k)$ when its first input is 1^k. Then:*

$$\begin{aligned}&\Pr[(PK, SK) \leftarrow G(1^k);\\ &(Q, x, y) \leftarrow T^{\texttt{Prove}(SK,\cdot)}(1^k) : y = \texttt{Eval}(SK, x) \wedge\\ &\qquad\qquad (x, \texttt{Prove}(SK, x)) \notin Q] \leq 1/s(k)\end{aligned}$$

Lemma 9. *The unique signature in Section 5 is a VUF with security 2^{k^ϵ} under Assumption 2.*

Proof. (Sketch) Following the proof in Section 6, let $\ell = k^\epsilon$. Then, by Lemma 8, using an adversary that succeeds in breaking the unique signature in 2^{k^ϵ} steps with probability 2^{-k^ϵ} corresponds to computing $g^{\prod y_i}$ in time 2^{k^ϵ} with probability $\Omega(2^{-2k\epsilon})$, which contradicts the assumption. □

The following proposition completes the picture:

Proposition 1 ([MRV99]). *If there is a VUF $(G, \texttt{Eval}, \texttt{Prove}, \texttt{Verify})$ with input length $a(k)$, output length $b(k)$, and security $s(k)$, then, for any $a'(k) \leq a(k)$, there is a VRF $(G', \texttt{Eval}', \texttt{Prove}', \texttt{Verify}')$ with input length $a'(k)$, output length $b'(k) = 1$, and security $s'(k) = s(k)^{1/3}/(poly(k) \cdot 2^{a'(k)})$. Namely, the following is a VRF with security $s'(k)$:*

– $G'(1^k) = (G(1^k), r)$, *where* r *is a* $b(k)$*-bit string chosen at random.*
– $\texttt{Eval}'(SK, x) = \texttt{Eval}(SK, x).r$, *where "." denotes the inner product.*
– $\texttt{Verify}'(SK, x) = (\texttt{Eval}(SK, x), \texttt{Verify}(SK, x))$.

Corollary 1. *We have obtained a VRF with input length* k*, output length* 1*, and security* $poly(k)$*.*

A natural open problem is to give better security guarantee to a VRF obtained from a unique signature in this fashion, or to show evidence of impossibility.

8 Complicated but More Secure VRF

Here, we construct a VRF and a unique signature based on the weaker assumption alone. This is the same as the Micali et al. [MRV99] construction, except that the underlying verifiable unpredictable function is different.

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

The proof of the proposition gives the VRF construction, which we omit here.

Now let us restate Assumption 1 to make explicit use of security:

Assumption 3 (s(k)-Many-DH assumption) *For some* $s(k) > poly(k)$*, for all* $\{A_k\}$*, if* $\{A_k\}$ *is a family of probabilistic Turing machines with running time* $t(k) = O(s(k))$*, then for all* $\ell > \log t(k)$*,*

$$\begin{aligned}\Pr[(*, q, g) \leftarrow S(1^k); \{y_i \leftarrow \mathbb{Z}_q \ : \ 1 \le i \le \ell\}; \\ \{z_J = \prod_{j \in J} y_j; Z_J = g^{z_J} \ : \ J \subset [\ell]\}; \\ Z \leftarrow A(*, q, g, \{Z_J \ : \ J \subset [\ell]\}) : Z = g^{\prod y_i}] < 1/s^2(k)\end{aligned}$$

We will construct a VUF with input length $\Omega(\log s'(k))$, and security $s'(k) = 2^{\omega(\log k)}$ based on this assumption. By Propositions 1 and 2, this implies a VRF with unrestricted input length and security $\min(s'(k)^{1/5}, 2^{a(k)/5}) = 2^{\omega(\log k)}$.

This is the same as our unique signature construction, only here the public key and the depth of the tree are smaller. The proof works in exactly the same way as in the previous construction.

Theorem 2. *The following construction is a VUF: set* $a_{i,b}$ *at random from* $\mathbb{Z}_q$*, for* $1 \le i \le \ell$*,* $\ell > \log s(k)$*,* $b \in \{0,1\}$*. Let* $PK = \{g^{a_{i,b}} \ : \ 1 \le i \le \ell, b \in \{0,1\}\}$*,* $SK = \{a_{i,b}\}$*. If* J *is a binary string of length* ℓ*, then let* $F_{PK}(g^{\prod_{i=1}^{\ell} a_{i,J_i}})$*. The verification is as in the construction described in Section 5. Its security is at least* $s(k)$ *under Assumption 3.*

Proof. (Sketch) This proof is essentially the same as in Section 6, with $n = \ell = \log s(k)$; except this case is simpler as there is no code. Here, too, we hope that the adversary's forgery will be good, and by Lemma 8, if $s(k)$ is the size of the message space, then $1/s(k)$ of the forgeries will be good. By contrapositive, in $O(s(k))$ running time, the probability of a forgery is $1/s(k)$. Therefore, the probability of a good forgery is $1/s^2(k)$. This contradicts Assumption 3. $\square$

Combining the above, we get a VRF with unrestricted input length, output length 1, and security $\min(s'(k)^{1/5}, 2^{a(k)/5}) = \min(s(k)^{1/10}, 2^{\log(s(k))/5}) = \Theta(s(k)^{1/10}) = 2^{\omega(\log k)}$.

Acknowledgements

I am grateful to Dan Boneh for pointing out an error in a previous version of this paper. I thank Ron Rivest, Silvio Micali, Yevgeniy Dodis, and Leo Reyzin for helpful discussions. This research was supported by an NSF graduate research fellowship, Lucent Technologies GRPW, and NTT grant #6762700.

References

[BDMP91] Manuel Blum, Alfredo De Santis, Silvio Micali, and Guiseppe Persiano. Non-interactive zero-knowledge. *SIAM Journal of Computing*, 20(6):1084–1118, 1991.

[BF01] Dan Boneh and Matthew Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *Advances in Cryptology — CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer Verlag, 2001.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 103–112, Chicago, Illinois, 2–4 May 1988.

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *Advances in Cryptology — ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532. Springer Verlag, 2001.

[BM84] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal on Computing*, 13(4):850–863, November 1984.

[BS02] Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. Manuscript obtained by personal communication, 2002.

[BY96] Mihir Bellare and Moti Yung. Certifying permutations: Non-interactive zero-knowledge based on any trapdoor permutation. *Journal of Cryptology*, 9(1):149–166, 1996.

[CS99] Ronald Cramer and Victor Shoup. Signature schemes based on the strong RSA assumption. In *Proc. 6th ACM Conference on Computer and Communications Security*, pages 46–52. ACM press, nov 1999.

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

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

[GHR99] Rosario Gennaro, Shai Halevi, and Tal Rabin. Secure hash-and-sign signatures without the random oracle. In Jacques Stern, editor, *Advances in Cryptology — EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 123–139. Springer Verlag, 1999.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988.

[GO92] Shafi Goldwasser and Rafail Ostrovsky. Invariant signatures and non-interactive zero-knowledge proofs are equivalent. In Ernest F. Brickell, editor, *Advances in Cryptology — CRYPTO '92*, pages 228–244. Springer-Verlag, 1992. Lecture Notes in Computer Science No. 740.

[JN01] Antoine Joux and Kim Nguyen. Separating decision Diffie-Hellman from Diffie-Hellman in cryptographic groups. Manuscript. Available from `http://eprint.iacr.org`, 2001.

[Jou00] Antoine Joux. A one-round protocol for tripartite Diffie-Hellman. In *Proceedings of the ANTS-IV conference*, volume 1838 of *Lecture Notes in Computer Science*, pages 385–394. Springer-Verlag, 2000.

[Mic] Silvio Micali. 6.875: Introduction to cryptography. MIT course taught in Fall 1997.

[MR01] Silvio Micali and Leonid Reyzin. Soundness in the public-key model. In Joe Kilian, editor, *Advances in Cryptology — CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 542–565. Springer Verlag, 2001.

[MR02] Silvio Micali and Ronald L. Rivest. Micropayments revisited. In Bart Preneel, editor, *Proceedings of the Cryptographer's Track at the RSA Conference*, volume 2271 of *Lecture Notes in Computer Science*, pages 149–163. Springer Verlag, 2002.

[MRV99] Silvio Micali, Michael Rabin, and Salil Vadhan. Verifiable random functions. In *Proc. 40th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 120–130. IEEE Computer Society Press, 1999.

[NR97] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *Proc. 38th IEEE Symposium on Foundations of Computer Science (FOCS)*, 1997.

[Sha85] Adi Shamir. Identity-based cryptosystems and signature schemes. In George Robert Blakley and David Chaum, editors, *Advances in Cryptology — CRYPTO '84*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer Verlag, 1985.

[Sud] Madhu Sudan. Algorithmic introduction to coding theory. MIT course taught in Fall 2001. Lecture notes available from http://theory.lcs.mit.edu/ madhu/FT01/.

[Ver01] Eric Verheul. Self-blindable credential certificates from the weil pairing. In Colin Boyd, editor, *Advances in Cryptology — ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 533–551. Springer Verlag, 2001.