Proceedings of Selected Areas in Cryptography '98 (August 17–18, 1998, Kingston, Ontario, Canada)
S. Tavares and H. Meijer Eds. Springer-Verlag, LNCS 1556, pages 72–80.

Computational Alternatives to Random Number Generators

David M'Raïhi[1], David Naccache[2], David Pointcheval[3], and Serge Vaudenay[3]

[1] Gemplus Corporation, 3 Lagoon Drive, Suite 300, Redwood City, CA 94065, USA
[2] Gemplus Card International, 34 rue Guynemer, 92447 Issy-les-Moulineaux, France
[3] LIENS – CNRS, École Normale Supérieure, 45 rue d'Ulm, 75230 Paris, France.
E-mail: {david.pointcheval,serge.vaudenay}@ens.fr.
URL: http://www.dmi.ens.fr/~{pointche, vaudenay}

Abstract. In this paper, we present a simple method for generating random-based signatures when random number generators are either unavailable or of suspected quality (malicious or accidental).
By opposition to all past state-machine models, we assume that the signer is a memoryless automaton that starts from some internal state, receives a message, outputs its signature and returns *precisely* to the same initial state; therefore, the new technique *formally* converts randomized signatures into deterministic ones.
Finally, we show how to translate the random oracle concept required in security proofs into a realistic set of tamper-resistance assumptions.

1 Introduction

Most digital signature algorithms rely on random sources which stability and quality crucially influence security: a typical example is El-Gamal's scheme [9] where the secret key is protected by the collision-freedom of the source.

Although biasing tamper-resistant generators is difficult[1], discrete components can be easily short-circuited or replaced by fraudulent emulators.

Unfortunately, for pure technological reasons, combining a micro-controller and a noise generator on the same die is not a trivial engineering exercise and most of today's smart-cards do not have real random number generators (traditional substitutes to random sources are keyed state-machines that receive a query, output a pseudo-random number, update their internal state and halt until the next query: a typical example is the BBS generator presented in [4]).

In this paper, we present an alternative approach that converts randomized signature schemes into deterministic ones: in our construction, the signer is a memoryless automaton that starts from some internal state, receives a message, outputs its signature and returns *precisely* to the same initial state.

Being very broad, we will illustrate our approach with Schnorr's signature scheme [22] before extending the idea to other randomized cryptosystems.

2 Digital signatures

In EUROCRYPT'96, Pointcheval and Stern [20] proved the security of an El-Gamal variant where the hash-function has been replaced by a random oracle. However,

[1] such designs are usually buried in the lowest silicon layers and protected by a continuous scanning for sudden statistical defects, extreme temperatures, unusual voltage levels, clock bursts and physical exposure.

since hash functions are fully specified (non-random) objects, the factual significance of this result was somewhat unclear. The following sections will show how to put this concept to work in practice.

In short, we follow Pointcheval and Stern's idea of using random oracles[2] but distinguish two fundamental implementations of such oracles (private and public), depending on their use.

Recall, *pro memoria*, that a digital signature scheme is defined by a distribution $\mathsf{generate}$ over a key-space, a (possibly probabilistic) signature algorithm sign depending on a secret key and a verification algorithm verify depending on the public key (see Goldwasser *et al.* [11]).

We also assume that sign has access to a private oracle f (which is a part of its private key) while verify has access to the public oracle h that commonly formalizes the hash function transforming the signed message into a digest.

Definition 1. Let $\Sigma^h = (\mathsf{generate}, \mathsf{sign}^h, \mathsf{verify}^h)$ denote a signature scheme depending on a uniformly-distributed random oracle h. Σ is (n, t, ϵ)-secure against existential-forgery adaptive-attacks if no probabilistic Turing machine, allowed to make up to n queries to h and sign can forge, with probability greater than ϵ and within t state-transitions (time), a pair $\{m, \sigma\}$, accepted by verify.

More formally, for any (n, t)-limited probabilistic Turing machine $\mathcal{A}$ that outputs valid signatures or fails, we have:

$$\Pr_{\omega,h}\left[\mathcal{A}^{h,\mathsf{sign}}(\omega) \text{ succeeds}\right] \leq \epsilon$$

where ω is the random tape.

Figure 1 presents such a bi-oracle variant of Schnorr's scheme: h is a public (common) oracle while f is a secret oracle (looked upon as a part of the signer's private key); note that this variant's verify is strictly identical to Schnorr's original one.

Definition 2. Let $\mathcal{H} = (h_K)_{K \in \mathcal{K}} \; : A \to B$ be a family of hash-functions, from a finite set A to a finite set B, where the key K follows a distribution $\mathcal{K}$. $\mathcal{H}$ is an (n, ϵ)-pseudo-random hash-family if no probabilistic Turing machine $\mathcal{A}$ can distinguish h_K from a random oracle in less than t state-transitions and n queries, with an advantage greater than ϵ.

In other words, we require that for all n-limited $\mathcal{A}$:

$$\left|\Pr_{\omega,K}\left[\mathcal{A}^{h_K}(\omega) \text{ accepts}\right] - \Pr_{\omega,h}\left[\mathcal{A}^{h}(\omega) \text{ accepts}\right]\right| \leq \epsilon$$

where ω is the random tape and h is a random mapping from A to B.

So far, this criterion has been used in block-cipher design but never in conjunction with hash functions. Actually, Luby and Rackoff [16] proved that a truly random 3-round, ℓ-bit message Feistel-cipher is $(n, n^2/2^{\ell/2})$-pseudo-random and

[2] although, as showed recently, there is no guarantee that a provably secure scheme in the random oracle model will still be secure in reality [5].

System parameters:	k, security parameter
	p and q primes, $q \mid (p-1)$
	$g \in \mathbb{Z}_p^\star$ of order q
	$h : \{0,1\}^* \to \mathbb{Z}_q$
Key generation:	$\mathsf{generate}(1^k)$
	secret: $x \in_R \mathbb{Z}_q$ and $f : \{0,1\}^* \to \mathbb{Z}_q$
	public: $y = g^x \bmod p$
Signature generation:	$\mathsf{sign}(m) := \{e, s\}$
	$u = f(m, p, q, g, y)$
	$r = g^u \bmod p$
	$e = h(m, r)$
	$s = u - xe \bmod q$
Signature verification:	$\mathsf{verify}(m; e, s)$
	$r = g^s y^e \bmod p$
	check that $e = h(m, r)$

Fig. 1. A deterministic variant of Schnorr's scheme.

safe until $n \cong 2^{\ell/4}$ messages have been encrypted (this argument was brought as an evidence for DES' security).

Note that (n, ϵ)-pseudo-randomness was recently shown to be close to the notion of n-wise decorrelation bias, investigated by Vaudenay in [24].

This construction can be adapted to pseudo-random hash-functions as follows: we first show how to construct a pseudo-random hash-function from a huge random string and then simplify the model by de-randomizing the string and shrinking it to what is strictly necessary for providing provable security. Further reduction will still be possible, at the cost of additional pseudo-randomness assumptions.

Theorem 3. *Let B be the set of ℓ-bit strings and $A = B^2$. Let us define two B-to-B functions, denoted F and G, from an $\ell \times 2^{\ell+1}$-bit key $K = \{F, G\}$. Let $h_K(x, y) = y \oplus G(x \oplus F(y))$. The family $(h_K)_K$ is $(n, n^2/2^{\ell+1})$-pseudo-random.*

Proof. The considered family is nothing but a truncated two-round Feistel construction and the proof is adapted from [16, 19] and [17]. The core of the proof consists in finding a meaningful lower bound for the probability that n different $\{x_i, y_i\}$'s produce n given z_i's. More precisely, the *ratio* between this probability and its value for a truly random function needs to be greater than $1 - \epsilon$. Letting $T = x \oplus F(y)$, we have:

$$\begin{aligned} \Pr[h_K(x_i y_i) = z_i; i = 1, \ldots, n] &\geq \Pr[h_K(x_i y_i) = z_i \text{ and } T_i \text{ pairwise different}] \\ &\geq \left(\frac{1}{2^\ell}\right)^n \left(1 - \frac{n(n-1)}{2} \min_{i,j} \Pr[T_i = T_j]\right) \end{aligned}$$

and for any $i \neq j$ (since $x_i y_i \neq x_j y_j$), we either have $y_i \neq y_j \Rightarrow \Pr[T_i = T_j] = 1/2^\ell$, or $y_i = y_j$ and $x_i \neq x_j$ which implies $\Pr[T_i = T_j] = 1$.

Consequently:

$$\Pr[h_K(x_iy_i) = z_i; i = 1, \ldots, n] \geq \left(\frac{1}{2^\ell}\right)^n \left(1 - \frac{n(n-1)}{2}\frac{1}{2^\ell}\right) \Rightarrow \epsilon = \frac{n^2}{2^{\ell-1}}.$$

Considering a probabilistic distinguisher $\mathcal{A}^O$ using a random tape ω, we get:

$$\begin{aligned}
\Pr_{\omega,K}[\mathcal{A}^{h_K}(\omega) \text{ accepts}] &= \sum_{\substack{\text{accepting} \\ x_1y_1z_1\ldots x_ny_nz_n}} \Pr_{\omega,K}[x_1y_1z_1\ldots x_ny_nz_n] \\
&= \sum_{x_iy_iz_i} \Pr_{\omega}[x_iy_iz_i/x_iy_i \xrightarrow{O} z_i] \Pr_K[h_K(x_iy_i) = z_i] \\
&\geq (1-\epsilon) \sum_{x_iy_iz_i} \Pr_{\omega}[x_iy_iz_i/x_iy_i \xrightarrow{O} z_i] \Pr_O[O(x_iy_i) = z_i] \\
&= (1-\epsilon) \Pr_{\omega,O}[\mathcal{A}^O(\omega) \text{ accepts}]
\end{aligned}$$

and

$$\Pr_{\omega,K}[\mathcal{A}^{h_K}(\omega) \text{ accepts}] - \Pr_{\omega,O}[\mathcal{A}^O(\omega) \text{ accepts}] \geq -\epsilon$$

which yields an advantage smaller than ϵ by symmetry (*i.e.* by considering another distinguisher that accepts if and only if $\mathcal{A}$ rejects). □

Note that this construction can be improved by replacing F by a random linear function: if $K = \{a, G\}$ where a is an ℓ-bit string and G an $n\ell$-bit string defining a random polynomial of degree $n-1$, we define $h_K(x) = y \oplus G(x \oplus a \times y)$ where $a \times y$ is the product in $\mathrm{GF}(2^\ell)$ (this uses Carter-Wegman's xor-universal hash function [6]).

More practically, we can use standard hash-functions such as:

$$h_K(x) = \text{HMAC-SHA}(K, x)$$

at the cost of adding the function's pseudo-randomness hypothesis [2, 3] to the (already assumed) hardness of the discrete logarithm problem.

To adapt random oracle-secure signatures to everyday's life, we regard $(h_K)_K$ as a pseudo-random keyed hash-family and require an indistinguishability between elements of this family and random functions. In engineering terms, this *precisely* corresponds to encapsulating the hash function in a tamper-resistant device.

Theorem 4. *Let $\mathcal{H}$ be a (n, ϵ_1)-pseudo-random hash-family. If the signature scheme Σ^h is (n, t, ϵ_2)-secure against adaptive-attacks for existential-forgery, where h is a uniformly-distributed random-oracle, then $\Sigma^{\mathcal{H}}$ is $(n, t, \epsilon_1+\epsilon_2)$-secure as well.*

Proof. Let $\mathcal{A}^{\mathcal{H},\mathsf{sign}}$ be a Turing machine capable of forging signatures for h_K with a probability greater than $\epsilon_1 + \epsilon_2$. h_K is distinguished from h by applying $\mathcal{A}$ and considering whether it succeeds or fails. Since $\mathcal{A}^{h,\mathsf{sign}}$ can not forge signatures with a probability greater than ϵ_2, the advantage is greater than ϵ_1, which contradicts the hypothesis. □

3 Implementation

An interesting corollary of theorem 4 is that if n hashings take more than t seconds, then K can be chosen randomly by a trusted authority, with some temporal validity. In this setting, long-term signatures become very similar to time-stamping [13, 1].

Another consequence is that random oracle security-proofs are no longer theoretical arguments with no practical justification as they become, *de facto*, a step towards practical and provably-secure schemes using pseudo-random hash families; however, the key has to remain secret, which forces the implementer to distinguish two types of oracles:

- A public random oracle h, that could be implemented as keyed pseudo-random hash function protected in a all tamper-resistant devices (signers and verifiers).
- A private random oracle f, which in practice could also be any pseudo-random hash-function keyed with a secret (unique to each signature device) generated by generate.

An efficient variant of Schnorr's scheme, provably-secure in the standard model under the tamper-resistance assumption, the existence of one-way functions and the DLP's hardness is depicted in figure 2.

System parameters:	k, security parameter
	p and q primes, $q\|(p-1)$
	$g \in \mathbb{Z}_p^\star$ of order q
	$(h_v : \{0,1\}^* \to \mathbb{Z}_q)_{v \in \mathcal{K}}$ pseudo-random hash-family
	$v \in_R \mathcal{K}$ secret key
	(same in all tamper-resistant devices)
Key generation:	$\mathsf{generate}(1^k)$
	secret: $x \in_R \mathbb{Z}_q$ and $z \in_R \mathcal{K}$
	public: $y = g^x \bmod p$
Signature generation:	$\mathsf{sign}(m) := \{e, s\}$
	$u = h_z(m, p, q, g, y)$
	$r = g^u \bmod p$
	$e = h_v(m, r)$
	$s = u - xe \bmod q$
Signature verification:	$\mathsf{verify}(m; e, s)$
	$r = g^s y^e \bmod p$
	check that $e = h_v(m, r)$

Fig. 2. A provably-secure deterministic Schnorr variant.

The main motivation behind our design is to provide a memoryless pseudo-random generator, making the dynamic information related to the state of the generator avoidable. In essence, the advocated methodology is very cheap in terms of entropy as one can re-use the already existing key-material for generating randomness.

Surprisingly, the security of realistic random-oracle implementations is enhanced by using *intentionally* slow devices:

- use a slow implementation (*e.g.* 0.1 seconds per query) of a $(2^{40}, 1/2000)$-pseudo-random hash-family.
- consider an attacker having access to 1000 such devices during 2 years ($\cong 2^{26}$ seconds).
- consider Schnorr's scheme, which is $(n, t, 2^{20}nt/T_{\text{DL}})$-secure in the random oracle model, where T_{DL} denotes the inherent complexity of the DLP [21].

For example, $\{|p| = 512, |q| = 256\}$-discrete logarithms can not be computed in less than 2^{98} seconds ($\cong$ a 10,000-processor machine performing 1,000 modular multiplications per processor per second, executing Shank's baby-step giant-step algorithm [23]) and theorem 4 guarantees that within two years, no attacker can succeed an existential-forgery under an adaptive-attack with probability greater than 1/1000.

This proves that realistic low-cost implementation and provable security can survive in harmony. Should a card be compromised, the overall system security will simply become equivalent to Schnorr's original scheme.

Finally, we would like to put forward a variant (see figure 3) which is not provably-secure but presents the attractive property of being *fully* deterministic (a given message m, will always yield the same signature):

Lemma 5. *Let $\{r_1, s_1\}$ and $\{r_2, s_2\}$ be two Schnorr signatures, generated by the same signer using algorithm 2 then $\{r_1, s_1\} = \{r_2, s_2\} \Leftrightarrow m_1 = m_2$.*

Proof. If $m_1 = m_2 = m$ then $r_1 = r_2 = g^{h(x,m,p,q,g,y)} = r \bmod p$, $e_1 = e_2 = h(m, r) = e \bmod q$ and $s_1 = h(x, m, p, q, g, y) - xe \bmod q = s_2 = s$, therefore $\{r_1, s_1\} = \{r_2, s_2\}$.

To prove the converse, observe that if $r_1 = r_2 = r$ then $g^{u_1} = g^{u_2} \bmod p$ meaning that $u_1 = u_2 = u$. Furthermore, $s_1 = u - xe_1 = u - xe_2 = s_2 \bmod q$ implies that $e_1 = h(m_1, r) = h(m_2, r) = e_2 \bmod q$; consequently, unless we found a collision, $m_1 = m_2$. □

Industrial motivation: This feature is a cheap protection against direct physical attacks on the signer's noise-generator (corrupting the source to obtain twice an identical u).

4 Deterministic versions of other schemes

The idea described in the previous sections can be trivially applied to other signature schemes such as [10] or [12]. Suffice it to say that one should replace each session's random number by a digest of the keys (secret and public) and the signed message.

System parameters:	k, security parameter
	p and q prime numbers such that $q \mid (p-1)$
	$g \in \mathbb{Z}_p^\star$ of order q
	h, hash function
Key generation:	$\mathsf{generate}(1^k)$
	secret: $x \in_R \mathbb{Z}_q$
	public: $y = g^x \bmod p$
Signature generation:	$\mathsf{sign}(m) := \{e, s\}$
	$u = h(x, m, p, q, g, y) \bmod q$
	$r = g^u \bmod p$
	$e = h(m, r) \bmod q$
	$s = u - xe \bmod q$
Signature verification:	$\mathsf{verify}(m; e, s)$
	$r = g^s y^e \bmod p$
	check that $e = h(m, r) \bmod q$

Fig. 3. A practical deterministic Schnorr variant.

Blind signatures [8] (a popular building-block of most e-cash schemes) can be easily transformed as well: in the usual RSA setting the user computes $w = h(k, m, e, n)$ (where k is a short secret-key) and sends $m' = w^e m \bmod n$ to the authority who replies with $s' = w^{ed} m^d \bmod n$ that the user un-blinds by a modular division ($s = s'/w = m^d \bmod n$).

The "blinding" technique can also be used to prevent timing-attacks [15], but it requires again a random blinding factor [14].

More fundamentally, our technique completely *eliminates* a well-known attack on Mc Eleice's cryptosystem [18] where, by asking the sender to re-encrypt logarithmically many messages, one can filter-out the error vectors (e, chosen randomly by the sender at each encryption) through simple majority votes.

We refer the reader to section III.1.4.A.C of [7] for more detailed description of this attack (that disappears by replacing e by a hash-value of m and the receiver's public-keys).

References

1. D. Bayer, S. Haber, and W. S. Stornetta. Improving the Efficiency and Reliability of Digital Time-Stamping. *Sequences II, Methods in Communication, Security and Computer Science*, pages 329–334, 1993.
2. M. Bellare, R. Canetti, and H. Krawczyk. Keying Hash Functions for Message Authentication. In *Crypto '96*, LNCS 1109. Springer-Verlag, 1996.
3. M. Bellare, R. Canetti, and H. Krawczyk. Message Authentication using Hash Functions: The HMAC construction. *RSA Laboratories' Cryptobytes*, 2(1), Spring 1996.
4. L. Blum, M. Blum, and M. Shub. A Simple Unpredictable Random Number Generator. *SIAM Journal on computing*, 15:364–383, 1986.
5. R. Canetti, O. Goldreich, and S. Halevi. The Random Oracles Methodology, Revisited. In *Proc. of the 30th STOC*. ACM Press, 1998.
6. L. Carter and M. Wegman. Universal Hash Functions. *Journal of Computer and System Sciences*, 18:143–154, 1979.

7. F. Chabaud. *Recherche de Performance dans l'Algorithmique des Corps Finis, Applications à la Cryptographie*. PhD thesis, École Polytechnique, 1996.
8. D. Chaum. Blind Signatures for Untraceable Payments. In *Crypto '82*, pages 199–203. Plenum, NY, 1983.
9. T. El Gamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In *IEEE Transactions on Information Theory*, volume IT–31, no. 4, pages 469–472, July 1985.
10. A. Fiat and A. Shamir. How to Prove Yourself: practical solutions of identification and signature problems. In *Crypto '86*, LNCS 263, pages 186–194. Springer-Verlag, 1987.
11. S. Goldwasser, S. Micali, and R. Rivest. A Digital Signature Scheme Secure Against Adaptative Chosen-Message Attacks. *SIAM Journal of Computing*, 17(2):281–308, April 1988.
12. L. C. Guillou and J.-J. Quisquater. A Practical Zero-Knowledge Protocol Fitted to Security Microprocessor Minimizing Both Transmission and Memory. In *Eurocrypt '88*, LNCS 330, pages 123–128. Springer-Verlag, 1988.
13. S. Haber and W. S. Stornetta. How to Timestamp a Digital Document. *Journal of Cryptology*, 3:99–111, 1991.
14. B. Kaliski. Timing Attacks on Cryptosystems. *RSA Laboratories' Bulletin*, 2, January 1996.
15. P. C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *Crypto '96*, LNCS 1109, pages 104–113. Springer-Verlag, 1996.
16. M. Luby and Ch. Rackoff. How to Construct Pseudorandom Permutations from Pseudorandom Functions. *SIAM Journal of Computing*, 17(2):373–386, 1988.
17. U. M. Maurer. A Simplified and Generalized Treatment of Luby-Rackoff Pseudorandom Permutation Generators. In *Eurocrypt '92*, LNCS 658, pages 239–255. Springer-Verlag, 1993.
18. R. J. McEliece. A Public-Key Cryptosystem Based on Algebraic Coding Theory. *DSN progress report*, 42-44:114–116, 1978. Jet Propulsion Laboratories, CALTECH.
19. J. Patarin. *Étude des Générateurs de Permutations Pseudo-aléatoires Basés sur le Schéma du DES*. PhD thesis, Université de Paris VI, November 1991.
20. D. Pointcheval and J. Stern. Security Proofs for Signature Schemes. In *Eurocrypt '96*, LNCS 1070, pages 387–398. Springer-Verlag, 1996.
21. D. Pointcheval and J. Stern. Security Arguments for Digital Signatures and Blind Signatures. *Journal of Cryptology*, 1998. To appear.
22. C. P. Schnorr. Efficient Identification and Signatures for Smart Cards. In *Crypto '89*, LNCS 435, pages 235–251. Springer-Verlag, 1990.
23. D. Shanks. Class number, a theory of factorization, and genera. In *Proceedings of the symposium on Pure Mathematics*, volume 20, pages 415–440. AMS, 1971.
24. S. Vaudenay. Provable Security for Block Ciphers by Decorrelation. In *STACS '98*, LNCS 1373, pages 249–275. Springer-Verlag, 1998.