

# Computational Alternatives to Random Number Generators

[Published in S. Tavares and H. Meijer, Eds., *Selected Areas in Cryptography*, vol. 1556 of *Lecture Notes in Computer Science*, pp. 72–80, Springer-Verlag, 1999.]

David M'Raihi, David Naccache, David Pointcheval, and Serge Vaudenay

<sup>1</sup> Gemplus Corporation

3 Lagoon Drive, Suite 300, Redwood City, CA 94065, USA

david.mraihi@gemplus.com

<sup>2</sup> Gemplus Card International

34 rue Guynemer, 92447 Issy-les-Moulineaux, France

david.naccache@gemplus.com

<sup>3</sup> LIENS – CNRS, École Normale Supérieure

45 rue d'Ulm, 75230, Paris, France

{david.pointcheval, serge.vaudenay@ens.fr}@ens.fr

<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 randomised 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,<sup>1</sup> discrete components can be easily short-circuited or replaced by fraudulent emulators.

Unfortunately, for pure technological reasons, combining a microcontroller and a noise generator on the same die is not a trivial engineering exercise and

---

<sup>1</sup> 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.

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 randomised 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 [20] before extending the idea to other randomised cryptosystems.

## 2 Digital Signatures

In EUROCRYPT'96, Pointcheval and Stern [18] proved the security of an El-Gamal variant where the hash-function has been replaced by a random oracle. However, 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<sup>2</sup> 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 `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 formalises the hash function transforming the signed message into a digest.

**Definition 1.** Let  $\Sigma^h = (\text{generate}, \text{sign}^h, \text{verify}^h)$  denote a signature scheme depending on a uniformly-distributed random oracle  $h$ .  $\Sigma$  is  $(n, t, \epsilon)$ -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  $\epsilon$  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:

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

where  $\omega$  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.

<sup>2</sup> 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 $\alpha \in (\mathbb{Z}/p\mathbb{Z})^*$ of order $q$ $p$ and $q$ primes, $q (p-1)$ $h : \{0, 1\}^* \rightarrow \mathbb{Z}/q\mathbb{Z}$
Key generation:	$\text{generate}(1^k)$ secret: $x \in_R \mathbb{Z}/q\mathbb{Z}$ and $f : \{0, 1\}^* \rightarrow \mathbb{Z}/q\mathbb{Z}$ public: $y = \alpha^x \bmod p$
Signature generation:	$\text{sign}(m) := \{e, s\}$ $u = f(m, p, q, g, y) \bmod q$ $r = g^u \bmod p$ $e = h(m, r) \bmod q$ $s = u - xe \bmod q$
Signature verification:	$\text{verify}(m; e, s)$ $r = g^s y^e \bmod p$ check that $e = h(m, r) \bmod q$

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

**Definition 2.** Let  $\mathcal{H} = (h_K)_{K \in \mathcal{K}} : A \rightarrow B$  be a family of hash-functions, where the key  $K$  follows a distribution  $\mathcal{K}$ .  $\mathcal{H}$  is an  $(n, \epsilon)$ -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  $\epsilon$ .

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

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

where  $\omega$  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 [14] proved that a truly-random 3-round,  $\ell$ -bit message Feistel-cipher is  $(n, n^2/2^{\ell/2})$ -pseudo-random and safe until  $n \cong 2^{\ell/4}$  messages have been encrypted (this argument was brought as an evidence for DES' security).

Note that  $(n, \epsilon)$ -pseudo-randomness was recently shown to be close to the notion of  $n$ -wise decorrelation bias, investigated by Vaudenay in [22].

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-randomising 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 1.** Let  $B$  be the set of  $\ell$ -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 [14, 17] and [16]. 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:

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

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:

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

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

$$\begin{aligned} \Pr_{\omega, K}[\mathcal{A}^{h_K}(\omega) \text{ accepts}] &= \sum_{\substack{\text{accepting} \\ x_1 y_1 z_1 \dots x_n y_n z_n}} \Pr_{\omega, K}[x_1 y_1 z_1 \dots x_n y_n z_n] \\ &= \sum_{x_i y_i z_i} \Pr_{\omega} [x_i y_i z_i / x_i y_i \xrightarrow{O} z_i] \Pr_K [h_K(x_i y_i) = z_i] \\ &\geq (1 - \epsilon) \sum_{x_i y_i z_i} \Pr_{\omega} [x_i y_i z_i / x_i y_i \xrightarrow{O} z_i] \Pr_O [O(x_i y_i) = z_i] \\ &= (1 - \epsilon) \Pr_{\omega, O} [\mathcal{A}^O(\omega) \text{ accepts}] \end{aligned}$$

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  $\epsilon$  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  $\ell$ -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  $\text{GF}(2^\ell)$  (this uses Carter-Wegman's xor-universal hash function [6]).  $\square$

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 every-day’s life, we regard  $(h_K)_K$  as a pseudo-random keyed hash-family and require an undistinguishability 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 2.** *Let  $\mathcal{H}$  be a  $(n, \epsilon_1)$ -pseudo-random hash-family. If the signature scheme  $\Sigma^h$  is  $(n, t, \epsilon_2)$ -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  $\mathcal{A}^{h, \text{sign}}$  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  $\mathcal{A}^{h, \text{sign}}$  can not forge signatures with a probability greater than  $\epsilon_2$ , the advantage is greater than  $\epsilon_1$ , which contradicts the hypothesis.  $\square$

### 3 Implementation

An interesting corollary of theorem 2 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.

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:

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

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

- 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  $T_{\text{DL}}$  denotes the inherent complexity of the DLP [19].

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 [21]) and theorem 1 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 (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 1.** *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  $r_1 = r_2 = g^{h_z(m)} = r \bmod p$ ,  $e_1 = e_2 = h_v(m, r) = e \bmod q$  and  $s_1 = h_v(m, r) - xe \bmod q = s_2 = s$ , therefore  $\{r_1, s_1\} = \{r_2, s_2\}$ .

To prove the converse, observe that if  $r_1 = r_2 = r$  then  $g^{k_1} = g^{k_2} \pmod p$  meaning that  $k_1 = k_2 = k$ . Furthermore,  $s_1 = k - xe_1 = k - xe_2 = s_2 \pmod q$  implies that  $e_1 = h_v(m_1, r) = h_v(m_2, r) = e_2 \pmod q$ ; consequently, unless we found a collision,  $m_1 = m_2$ .  $\square$

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

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

**Fig. 3.** A practical deterministic Schnorr variant.

## 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.

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  $m' = w^e m \pmod n$  to the authority who replies with  $s' = w^{ed} m^d \pmod n$  that the user un-blinds by a modular division ( $s = s'/w = m^d \pmod n$ ).

More fundamentally, our technique completely *eliminates* a well-known attack on McEliece's cryptosystem [15] 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 & W. Stornetta, *Improving the efficiency and reliability of digital time-stamping*, Sequences II, Methods in communication, Security and computer science, 1993, pp. 329–334.
2. M. Bellare, R. Canetti & H. Krawczyk, *Keying hash functions for message authentication*, Advances in cryptology - CRYPTO'96, Lecture Notes in Computer Science 1109, Springer-Verlag, 1996.
3. M. Bellare, R. Canetti & H. Krawczyk, *Message authentication using hash functions: The HMAC construction*, RSA Laboratories' CRYPTOBYTES vol. 2, no. 1, Spring 1996.
4. L. Blum, M. Blum & M. Shub, *A simple unpredictable random number generator*, SIAM Journal on computing, vol. 15, 1986, pp. 364–383.
5. R. Canetti, O. Goldreich & S. Halevi, *The random oracle methodology, revisited*, Proceedings of the 30-th STOC, 1998, ACM Press.
6. L. Carter & M. Wegman, *Universal hash functions*, Journal of computer and system sciences, 1979, pp. 143–154.
7. F. Chabaud, *Recherche de performance dans l'algorithmique des corps finis, applications à la cryptographie*, Ph.D. thesis, École polytechnique, 1996.
8. D. Chaum, *Blind signatures for untraceable payments*, Advances in cryptology - Proceedings of CRYPTO'82, Plenum, New-York, pp. 199–203.
9. T. El-Gamal, *A public-key cryptosystem and a signature scheme based on discrete logarithms*, IEEE Transactions on information theory, IT-31, 1985, pp. 469–472.
10. A. Fiat & A. Shamir, *How to prove yourself: practical solutions to identification and signature problems*, Advances in cryptology - EUROCRYPT'86, Lecture Notes in Computer Science 263, Springer-Verlag, pp. 186–194.
11. S. Goldwasser, S. Micali & R. Rivest, *A digital signature scheme secure against adaptive chosen-message attacks*, SIAM journal of computing, vol. 17, no. 2, April 1988, pp. 281–308.
12. L. Guillou & J.-J. Quisquater, *A practical zero-knowledge protocol fitted to security microprocessor minimising both transmission and memory*, Advances in Cryptology - EUROCRYPT'88, Lecture Notes in Computer Science 330, Springer-Verlag, pp. 123–128.
13. S. Haber & W. Stornetta, *How to time-stamp a digital document*, Journal of Cryptology, vol. 3, pp. 99–111, 1991.
14. M. Luby & C. Rackoff, *How to construct pseudo-random permutations from pseud-orandom functions*, SIAM journal of computing, vol. 17, 1988, pp. 373–386.
15. R. McEliece, *A public-key cryptosystem based on algebraic coding theory*, DSN progress report 42-44, Jet propulsion laboratories, CALTECH, pp. 114-116, 1978.
16. U. Maurer, *A simplified and generalised treatment of Luby-Rackoff pseudo-random permutation generators*, Advances in cryptology – EUROCRYPT'92, Lecture Notes in Computer Science 658, Springer-Verlag, pp. 239–255.
17. J. Patarin, *Étude des générateurs de permutations pseudo-aléatoires basés sur le schéma du DES*, Ph.D. thesis, Université de Paris VI, november 1991.

18. D. Pointcheval & J. Stern, *Security proofs for signature schemes*, Advances in cryptology - EUROCRYPT'96, Lecture Notes in Computer Science 1070, Springer-Verlag, pp. 387–398.
19. D. Pointcheval & J. Stern, *Security arguments for digital signatures and blind signatures*, Journal of cryptology, to appear.
20. C. Schnorr, *Efficient identification and signatures for smart-cards*, Advances in cryptology - EUROCRYPT'89, Lecture Notes in Computer Science 765, Springer-Verlag, pp. 435–439.
21. D. Shanks, *Class number, a theory of factorisation, and genera*, Proceedings of the symposium on pure mathematics, vol. 20, AMS, 1971, pp. 415–440.
22. S. Vaudenay, *Provable security for block ciphers by decorrelation*, In STACS'98, Paris, France, Lectures Note in Computer Science 1373, Springer-Verlag, pp. 249–275, 1998.