The Insecurity of Esign in Practical Implementations

Pierre-Alain Fouque[1], Nick Howgrave-Graham[2], Gwenaëlle Martinet[3], and Guillaume Poupard[3]

[1] École Normale Supérieure, Département d'Informatique
45 rue d'Ulm, 75230 Paris Cedex 05, France
Pierre-Alain.Fouque@ens.fr

[2] NTRU Cryptosystems, 5 Burlington Woods,
Burlington, MA 02144 USA
nhowgravegraham@ntru.com

[3] DCSSI Crypto Lab, 51, Boulevard de Latour-Maubourg
75700 Paris 07 SP, France
Gwenaelle.Martinet@worldonline.fr, Guillaume.Poupard@m4x.org

Abstract. Provable security usually makes the assumption that a source of perfectly random and secret data is available. However, in practical applications, and especially when smart cards are used, random generators are often far from being perfect or may be monitored using probing or electromagnetic analysis. The consequence is the need of a careful evaluation of actual security when idealized random generators are implemented.

In this paper, we show that Esign signature scheme, like many cryptosystems, is highly vulnerable to so called partially known nonces attacks. Using a 1152-bit modulus, the generation of an Esign signature requires to draw at random a 768-bit integer. We show that the exposure of only 8 bits out of those 768 bits, for 57 signatures, is enough to recover the whole secret signature key in a few minutes.

It should be clear that we do not cryptanalyze a good implementation of Esign nor do we find a theoretical flaw. However, our results show that random data used to generate signatures must be very carefully produced and protected against any kind of exposure, even partial.

As an independent result, we show that the factorization problem is equivalent to the existence of an oracle returning the most or least significant bits of $S \bmod p$, on input S randomly chosen in $\mathbb{Z}_{pq}$.

Keywords: Esign signature scheme, Lattice reduction, LLL algorithm, Factorization problem.

1 Introduction

Most cryptographic systems make use of random sources for a range of applications. Random data may, for example, be transformed into secret or private keys for encryption or signature. From a provable security point of view, it is common to assume one has access to a source of perfect randomness. However, such an assumption is far from being totally realistic in many practical applications. The first problem is that a true random number generator must be based on some kind of physical noise source. Such a generator is not commonly accessible on standard computers. When smart cards are used, the situation is even worse since such devices only have access to a very poor and constrained environment. The consequence is that random data is often simulated using a pseudo-random number generator.

In practical applications, there is a danger of adding weaknesses by using a biased generator or a weak pseudo-random number generator. Furthermore, with devices such as smart cards, the risk of secret data exposure by the way of probing or electromagnetic analysis may be increased if the random number generator is separated from the rest of the chip. As a consequence, a crucial question, when we consider practical security, is the impact of partial exposure of this random data for systems which have been proved secure under the assumption that a source of perfectly random and secret data is available.

The answer strongly depends on the application one considers. Usually, key generation is viewed as a crucial issue and people agree that a lot of care must be applied to the production of key material. However, does the exposure of one third of a 128-bit AES key have any real practical implication in usual applications? Such a question is of course rather controversial, but the complexity of an exhaustive search on the remaining secret bits, about 2^{85} block encryptions, might still be thought prohibitive. The same reasoning may be applied to other applications such as the choice of nonces or initial vectors. However, in some cases, partial exposure of secret information can have a far more dramatic consequence on the security of the system. Our first example is related to RSA with short public exponent. Boneh, Durfee and Frankel [6] have shown that the exposure of a quarter of the secret exponent enables one to factor the modulus in polynomial time. Similar results on DSA signature scheme are even more impressive. This scheme uses 160 bits of fresh random data, often called on-time key or ephemeral key, for each signature generation. It is well known that the exposure of those data enables to recover the secret signature key very easily. Howgrave-Graham and Smart [16] applied lattice reduction techniques to prove that the knowledge of only 8 bits out of the 160 bits of ephemeral keys for 30 signed messages enables to recover the secret key in a few seconds! In the same vein, following Boneh and Venkatesan [7], Nguyen and Shparlinski [19] have shown that indeed only 3 bits out of the 160 bits of each one-time key, for 100 messages, are enough to make the attack feasible. Finally, Bleichenbacher [3] has shown that if just one bit out of the 160 bits is biased, as was the case with the pseudo-random generator initially proposed by NIST [21], it is possible to mount an attack with time complexity 2^{64}, memory complexity 2^{40} and 2^{22} signatures.

Another analysis of the security of DSA in practical implementations, was done by Bellare, Goldwasser and Micciancio [2]. They did not assume partial exposure of ephemeral keys but their randomness was generated by a weak pseudo-random number generator, namely the linear congruential generator. In this case, DSA is totally insecure and the knowledge of a few signatures leads to the computation of the secret signature key.

All these results show that in some applications, such as DSA, data must be perfectly random and must remain completely secret. This does not mean that DSA is not secure but it points out a potential source of weakness. In actual implementations, the mechanism used to generate random data must be carefully chosen and evaluated, both from an algorithmic point of view and from a technological point of view. For example, electromagnetic analysis or probing techniques may enable one to learn a few random bits, even if it is not possible to recover the whole secret by these means. The above mentioned results show that the knowledge of a very small part of those bits is enough to totally break systems such as DSA.

Our results

In this paper, we focus on the practical security of the Esign signature scheme [11]. Of course, in practical applications, this scheme is much less used than DSA. However, Esign could be preferred in many scenarios from a computational efficiency point of view. This is important when the signature device has low computing resources, which is the case with smart cards for instance. For applications such as on the fly signature with a contactless card (typical for fast and secure payment in the subway), Esign may be a very good candidate. Its practical security must consequently be carefully analyzed.

The technique we develop, and apply to careless Esign implementations, is of independent interest. It may be applied to other factorization based cryptosystems. Assuming partial exposure of a very small part of some secret data, our lattice reduction based technique allows one to factor the modulus very efficiently. Typically, this may be applied to the optimization of some SPA/DPA attacks on RSA systems [22, 10].

In this paper, we describe an efficient technique based on the partial exposure of a few bits of Esign ephemeral keys. More precisely, using a 1152-bit modulus, the generation of an Esign signature requires to draw at random a 768-bit integer. We show that the exposure of only 8 bits out of those 768 bits for 57 signatures is enough to recover the all secret signature key in a few minutes.

It should be clear that we do not propose neither a cryptanalysis of Esign nor a theoretical flaw. However, our results show that random data used to generate signatures must be very carefully produced and protected against any kind of exposure, even partial.

Previous works

The *hidden number problem* (HNP) has been described by Boneh and Venkatesan in [7] in order to prove the hardness of the most significant bits of secret keys in Diffie-Hellman and related schemes in prime fields. The HNP can be defined as follows: given $s_1, \ldots, s_d$ chosen uniformly and independently at random in $\mathbb{Z}_q{}^\star$ and $\mathrm{MSB}_\ell(\alpha s_i \bmod q)$ for all i, the problem is to recover $\alpha \in \mathbb{Z}_q{}^\star$. Here, $\mathrm{MSB}_\ell(x)$ for $x \in \mathbb{Z}_q$ denotes any integer z satisfying $|x - z| < q/2^{\ell+1}$. In [7], the authors present a simple solution to this problem by reducing HNP to a lattice closest vector problem (CVP). In particular, they show that the HNP can be solved if $\ell \geq \sqrt{\log(q)} + \log(\log(q))$ and $d = 2\sqrt{\log(q)}$. According to [20], using the best known polynomial-time CVP approximation algorithm due to Ajtai *et al.* [1] and Kannan [17], ℓ can be asymptotically improved to $O(\sqrt{\log(q)}\log(\log(\log(q)))/\log(\log(q)))$.

In this paper we consider a problem related to a HNP problem modulo a *secret* value and we propose an algorithm to solve it. In [4], Boneh also mentions the HNP modulo $N = pq$. Now, p and q denote the factors of a modulus N. Our problem can be formulated as follows: given $s_1, \ldots, s_d$ chosen uniformly and independently at random in $\mathbb{Z}_N{}^\star$ and $\mathrm{MSB}_\ell(s_i \bmod p)$ for all i, the problem is to recover p. Our algorithm uses the *orthogonal lattice theory* of [20] to obtain several small lattice vectors. Moreover, we also use the extension of Nguyen and Shparlinski [19] if the distribution of the s_i is not necessarily perfectly uniform using the discrepancy notion. Indeed, if we note $|s|_p = \min_{b\in\mathbb{Z}} |s - bp|$ for any integer s, the values s_i in the lattice are such that $|s_i|_p < p/2^\ell$ and are thus not uniformly distributed in $\mathbb{Z}_p$. If N is a 1024-bit modulus, then the results of the HNP say that with $d = 64$ and $\ell = 9$, N can be factored. We get similar results with our algorithm.

Finally, contrary to the lattice based algorithm used by Boneh, Durfee and Howgrave-Graham [5], our factorization algorithm uses an oracle. In some cases, this oracle can be found in practical implementations. For example, if the pseudorandom generator of the nonces used in Esign implementation is biased such that the MSBs can be learned, then we can break the signature scheme by factoring the modulus. In this application, the secret modulus is a composite number pq and $N = p^2q$. This paper can be seen as an extension of previous attacks on signature schemes, based on the discrete log such as DSA in [19, 16], to some factorization based signature schemes.

The results in this paper were independently discovered, but are of a similar vein to those found in the Esign technical review [15].

2 Description of Esign

Esign is a signature scheme proposed by Okamoto and Shiraishi in 1985 [23]. It is based on modular computations with special form modulus. The main advantage of Esign is its efficiency. Compared to RSA or EC based scheme, Esign is several times faster in terms of signature and verification performance.

Let $N = p^2q$ a $3k$-bit integer, with p and q two primes of roughly the same length. The secret key consists in the two k-bit primes p and q. The public key is (N, e), where e is an integer larger than 4. The scheme uses a cryptographic hash function H to compute $(k-1)$-bit long message digests. The signature of a message M is performed as follows:

1. the message M is first hashed into $H(M)$. We denote by y the integer corresponding to the $3k$-bit string $0\|H(M)\|0^{2k}$, where 0^{2k} denotes the concatenation of $2k$ null bits,
2. An integer r is randomly chosen in $\mathbb{Z}_{pq}^\star$,
3. Compute:
   (a) $z = y - r^e \bmod N$,
   (b) $w_0 = \left\lceil \frac{z}{pq} \right\rceil$,
   (c) $w_1 = w_0 pq - z$. If $w_1 > 2^{2k-1}$, then come back to step 2,
   (d) $u = w_0 (er^{e-1})^{-1} \bmod p$,
   (e) $s = r + upq$,
4. Return s as a signature for M.

Note that in the rest of this paper, we often write signatures as the sum of the random nonce r and a multiple $u \times pq$ of the secret key pq.

To verify if a signature s is valid for the message M, a verifier simply checks if the k most significant bits of $s^e \bmod N$ are equal to $0\|H(M)$. The verification algorithm is consistent since:

$$\begin{aligned} s^e &= (r + upq)^e \bmod N \\ &= r^e + er^{e-1}upq \bmod N \\ &= (y - z) + w_0 pq \bmod N \\ &= y + w_1 \bmod N \end{aligned}$$

Since $w_1 < 2^{2k-1}$, and N is exactly $3k$ bits long, the k most significant bits of $s^e \bmod N$ are those of y, *i.e.* $0\|H(M)$.

The security of Esign is based on a variant of the RSA problem which consists in computing modular e-th roots. More precisely, even the computation of approximations of such roots seems to be difficult. The Approximate e-th Root (AER) problem is formally defined as follows:

> *Given a modulus $N = p^2q$, an exponent $e \geq 4$ and $y \in \mathbb{Z}_N^\star$, find $x \in \mathbb{Z}_N^\star$ such that $x^e \in [y, y + 2^{2k-1}]$.*

The knowledge of the factorization of N gives an efficient solution to this problem. Without p or q, this problem is supposed to be hard. The AER assumption is that the AER problem is intractable.

The initial scheme proposed in [23] was based on the exponent $e = 2$. This version has been cryptanalyzed the same year by Brickell and DeLaurentis in [8]. The cubic scheme has also been broken using lattice reduction (see in particular [9, 13, 26]). However, for $e \geq 4$, no attack has been reported for the moment. A potential way to break the signature scheme is to factor the modulus N and then to recover the secret key. This constitutes a total break of the scheme. Note that if the random value r is compromised for just one signature, the factorization can be easily recovered. Indeed, since $s = r + upq$, if r is known, then the GCD of the modulus $N = p^2q$ and $s - r$ reveals pq.

We also notice that the knowledge of $r^e \bmod N$ allows to recover the prime factors p and q. Indeed, s^e can be written as $r^e - er^{e-1}upq \bmod N$ and the

GCD of N and $s^e - r^e \bmod N$ gives pq. The secrecy of the random values is consequently a crucial issue for Esign.

Moreover, the scheme with $e \geq 4$ and e prime with $\phi(N) = p(p-1)(q-1)$, is provably secure in the random oracle model. More precisely, it is proved secure against existential forgeries in Single Occurrence Chosen Message Attacks scenarios, under the AER assumption (see [25]). An adversary querying a signature oracle for messages of his choice, but with the restriction that a message cannot be submitted twice to the oracle, cannot forge a signature for a message. Otherwise, he can solve the AER problem, supposed to be intractable. Extending the proof to the stronger adaptive chosen message attacks model is an open problem. Thus, two different ways have been proposed to make Esign provably secure in the strong sense [14]. The first method, called Esign-D, is deterministic: the random nonce r is generated from the message to sign and an additional secret string, included in the private key. The second one, called Esign-R, uses another random nonce ρ, given as part of the signature, to generate the hash of the message as $H(M\|\rho)$. In the following, the attacks we present are not chosen message ones, but are based on flawed implementations. Hence, they do not depend on the version used. So without lost of generality, we focus on the first scheme described above.

3 Lattice based attacks

In this section we first recall some basic facts about lattices and reduction algorithms. Then, we detail how to use lattice reduction in order to factor modulus such as N under some assumptions on the random data used in Esign.

3.1 Lattice reduction

Notations. Let $N = p^2q$ an Esign modulus. Then any integer s in $\mathbb{Z}_N$ can be written as $s = r + upq$ with $0 \leq r < pq$ and $0 \leq u < p$.

Definitions. In the following, we denote by $\|\mathbf{x}\|$ the Euclidean norm of the vector $\mathbf{x} = (x_1, \ldots, x_{d+1})$, defined by $\|\mathbf{x}\| = \sqrt{\sum_{i=1}^{d+1} x_i^2}$. Let $\mathbf{v_1}, \ldots, \mathbf{v_d}$, be d linearly independent vectors such that for $1 \leq i \leq d$, $\mathbf{v_i} \in \mathbb{Z}^{d+1}$. We denote by L, the lattice spanned by the matrix V whose rows are $\mathbf{v_1}, \ldots, \mathbf{v_d}$. L is the set of all integer linear combinations of $\mathbf{v_1}, \ldots, \mathbf{v_d}$:

$$L = \left\{ \sum_{i=1}^{d} c_i \mathbf{v_i},\ c_i \in \mathbb{Z} \right\}$$

Geometrically, $\det(L) = \det(V \times V^T)$ is the volume of the parallelepiped spanned by $\mathbf{v_1}, \ldots, \mathbf{v_d}$. The Hadamard's inequality says that $\det(L) \leq \|\mathbf{v_1}\| \times \ldots \times \|\mathbf{v_d}\|$.

Given $\langle \mathbf{v_1}, \ldots, \mathbf{v_d} \rangle$ the LLL algorithm [18] will produce a so called "reduced" basis $\langle \mathbf{b_1}, \ldots, \mathbf{b_d} \rangle$ of L such that

$$\|\mathbf{b_1}\| \leq 2^{(d-1)/2} \det(L)^{1/d} \tag{1}$$

in time $O(d^4 \log(M))$ where $M = \max_{1 \leq i \leq d} \|\mathbf{v_i}\|$. Consequently, given a basis of a lattice, the LLL algorithm finds a short vector $\mathbf{b_1}$ of L satisfying equation (1). Moreover, we assume in the following that the new basis vectors are of the same length and also have all their coordinates of approximatively the same length. Indeed, a basis for a random lattice can be reduced into an almost orthonormal basis. Therefore, $\|b_i\| \approx \|b_1\|$ for $1 \leq i \leq d$, and so $\|b_i\|^d \approx \det(L)$.

3.2 Lattice-based Factoring Algorithm

In this subsection, we present a lattice technique to factor a modulus $N = p^2 q$, where p and q are two k-bit primes, given an oracle $\mathcal{O}_{\ell,pq}$ that, on input $\tilde{s} \in \mathbb{Z}_N$, returns the ℓ MSBs of $\tilde{s} \bmod pq$. We will see in section 4 that in practical applications it is sometimes possible to realize such an oracle. In the following we denote by $n = 3k$ the bit length of N.

Let $\tilde{s} \in \mathbb{Z}_N$ be an integer smaller than N. If an $\mathcal{O}_{\ell,pq}$ oracle is available, let us query the ℓ most significant bits of $\tilde{s} \bmod pq$; we denote by t the answer of the oracle. Then, $s = \tilde{s} - t \times 2^{2k-\ell}$ may always be written as $r + upq$ with $0 \leq r < pq/2^\ell$ and $0 \leq u < p$. Finally, after d queries to the oracle, we may consider that we know d integers $s_i \in \mathbb{Z}_N$ such that $s_i = r_i + u_i pq$ with $0 \leq r_i < pq/2^\ell$ and $0 \leq u_i < p$. However, the r_i and u_i values are unknown. Our objective is to recover pq.

First we note that if we are able to recover one of the u_i, then recovering the factors p and q of N can be efficiently done. Indeed, we suppose first that the recovered u_i value is larger than $p/2^\ell$. This occurs with probability $1 - 1/2^\ell$ and if this is not true, we can recover another u_i until this event occurs. Thus, we have $p/2^\ell < u_i < p$ and we can write $s_i/u_i = r_i/u_i + pq$ where r_i/u_i is at most k bits. Consequently, the k most significant bits of s_i/u_i are those of pq. We denote by A the integer matching pq on its k MSBs and zeroing the k least significant bits. The $2k$-bit value A is known and we can write $pq = A + \alpha$ where $\alpha < 2^k$ is unknown. Finally, since $N = p \times pq = p \times (A + \alpha)$, we have:

$$\frac{N}{A} = p + \frac{p\alpha}{A} \tag{2}$$

where $0 \leq \frac{p\alpha}{A} < 2$, since $p\alpha$ is at most of $2k$ bits and A is exactly $2k$ bits. Thus, p equals either $\lfloor N/A \rfloor$ or $\lfloor N/A \rfloor - 1$.

In the following, we present an algorithm to recover all the u_i. In a first phase, the algorithm looks for small linear integer combinations of the s_i using the LLL algorithm. Then, in a second phase, we solve a linear system to recover the u_i. In the sequel, we describe these two phases.

Finding Small Linear Integer Combinations of the u_i. The following lemma shows that searching a small linear integer combination of the s_i with small coefficients is sufficient to find a null linear combination of the u_i.

Lemma 1. *Let $N = p^2q$ be a n-bit modulus with p and q of roughly the same length. Let $s_1, \dots, s_d$ be d random integers in $\mathbb{Z}_N$, $s_i = r_i + u_i pq$ such that $|r_i| < pq/2^\ell$.*

If there exist d integers c_i, for $1 \le i \le d$, such that $c = \max_{1 \le i \le d} |c_i| < 2^\ell/d$ and $|\sum_{i=1}^d c_i s_i| < pq$, then $\sum_{i=1}^d c_i u_i \in \{-1, 0, 1\}$.

Moreover, if $c < 2^\ell/2d$ and $|\sum_{i=1}^d c_i s_i| < pq/2$, then $\sum_{i=1}^d c_i u_i = 0$.

Proof. By definition, we have $\sum_{i=1}^d c_i s_i = \sum_{i=1}^d c_i r_i + pq \sum_{i=1}^d c_i u_i$. Thus by the triangle inequality, we can write:

$$pq \left| \sum_{i=1}^d c_i u_i \right| \le \left| \sum_{i=1}^d c_i s_i \right| + \left| \sum_{i=1}^d c_i r_i \right| \tag{3}$$

Moreover, since $c < 2^\ell/d$ and $|r_i| < pq/2^\ell$ for $1 \le i \le d$, then

$$\left| \sum_{i=1}^d c_i r_i \right| \le \sum_{i=1}^d |c_i r_i| \le d \times \left(\frac{2^\ell}{d} \times \frac{pq}{2^\ell} \right) \le pq$$

Now we know that $|\sum_{i=1}^d c_i s_i| < pq$. Then from equation (3), $pq|\sum_{i=1}^d c_i u_i| < 2pq$ and $|\sum_{i=1}^d c_i u_i| < 2$. This proves the first part of the lemma. The second part of the lemma can be easily deduced from the previous computations. □

Therefore, we look for small integer linear combination of the s_i, *i.e.* such that $|\sum_{i=1}^d c_i s_i| \le pq$ and $c < 2^\ell/d$. From previous lemma, finding such a combination gives a linear equation in the u_i variables.

Now we present a lattice-based method to recover the coefficients of a small combination of the s_i. Suppose K is an integer less than N, whose exact value will be defined later. We consider the following $d \times (d+1)$-matrix:

$$M = \begin{pmatrix} s_1 & K & 0 & \dots & 0 \\ s_2 & 0 & K & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ s_d & 0 & \dots & 0 & K \end{pmatrix}$$

The size of the original basis vector is approximately N since the s_i are integers in $\mathbb{Z}_N$. In order to estimate the size of a small vector returned by LLL, we upper bound the volume of the lattice L, spanned by the rows of M. In the following, we upperbound the determinant of the lattice L and show that

$$\det(L)^2 = K^{2d-2}(K^2 + \sum_{j=1}^d s_j^2)$$

Since L is not a full lattice, its volume is the square root of the determinant of the Gramian matrix [12], $M \times M^T$. Thus, we have:

$$\det(L)^2 = \det(M \times M^T) = \begin{vmatrix} s_1^2 + K^2 & s_1 \times s_2 & s_1 \times s_3 & \dots & s_1 \times s_d \\ s_2 \times s_1 & s_2^2 + K^2 & s_2 \times s_3 & \dots & s_2 \times s_d \\ s_3 \times s_1 & s_3 \times s_2 & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & s_{d-1} \times s_d \\ s_d \times s_1 & \dots & \dots & s_d \times s_{d-1} & s_d^2 + K^2 \end{vmatrix}$$

We can factor the first row by s_1, the second by s_2, ..., s_d and similarly for the columns. Therefore the determinant can be written as

$$\det(L)^2 = \prod_{i=1}^{d} s_i^2 \times \begin{vmatrix} 1 + K^2/s_1^2 & 1 & 1 & \dots & 1 \\ 1 & 1 + K^2/s_2^2 & 1 & \dots & 1 \\ 1 & 1 & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 1 \\ 1 & \dots & \dots & 1 & 1 + K^2/s_d^2 \end{vmatrix}$$

The last determinant can be computed exactly and is equal to

$$\prod_{i=1}^{d} \frac{K^2}{s_i^2} + \sum_{i=1}^{d} \prod_{j=1, j \neq i}^{d} \frac{K^2}{s_j^2}$$

and consequently,

$$\det(L)^2 = K^{2d-2}(K^2 + \sum_{j=1}^{d} s_j^2)$$

Therefore, since for all i, $|s_i| \leq N$ and $K \leq N$, the size of the small vector returned by LLL on this lattice is less than

$$2^{(d-1)/2} \times (d+1)^{1/2d} \times K^{\frac{d-1}{d}} N^{\frac{1}{d}}$$

For the present discussion we ignore factors like $2^{(d-1)/2}$ dependent only on the size of the matrix. Indeed, in practice, LLL returns a short vector much smaller than theoretical upperbounds. Consequently, we can assume that the shortest vector returned by LLL is of length about $(d+1)^{1/2d} \times K^{\frac{d-1}{d}} N^{\frac{1}{d}}$.

Now we fix $K = \left\lceil N^{\frac{2}{3} - \frac{1}{3(d-1)}}/2 \right\rceil$. As a consequence, a simple computation shows that, in this case, the length of a short vector returned by LLL is less than

$$(d+1)^{1/2d} \times N^{\frac{2}{3} - \frac{1}{3d}}$$

which is less than pq since $\sqrt{d+1} \ll N^{1/3}$. Therefore a short vector has all its coordinates smaller than pq.

In the following, we show how a short vector $\mathbf{b_1}$ returned by LLL allows us to determine the coefficients of a short linear combination of the s_i. Due to the form of the matrix M, $\mathbf{b_1}$ can be written as

$$\mathbf{b_1} = \left(\sum_{i=1}^{d} c_i \cdot s_i, K \cdot c_1, \ldots, K \cdot c_d \right) \tag{4}$$

where the c_i are integers. We denote by c the maximum of the $|c_i|$. If $\mathbf{b_1}$ is a short vector returned by LLL, then all its coordinates are smaller than pq. In particular, we have $Kc < pq$. Consequently, $c < 2pq/N^{\frac{2}{3}-\frac{1}{3(d-1)}}$. Furthermore, if $\ell > \frac{n}{3(d-1)} + \log(d) + 1$, then

$$c < \frac{2N^{2/3}}{N^{\frac{2}{3}-\frac{1}{3(d-1)}}} \leq 2N^{\frac{1}{3(d-1)}} < \frac{2^\ell}{d}$$

Therefore, since $\sum_{i=1}^{d} c_i \cdot s_i < pq$ and $c < 2^\ell/d$, then lemma 1 implies that $\sum_{i=1}^{d} c_i u_i \in \{-1, 0, 1\}$.

As a consequence, if we have d random values $s_i = r_i + u_i pq$, where $|r_i| < pq/2^\ell$ and $\ell > \left\lceil \frac{n}{3(d-1)} + \log(d) + 1 \right\rceil$, then the shortest vector returned by LLL gives us the coefficients of a very small combination of the u_i and we finally have a linear equation in the u_i variables.

However, one equation is not sufficient to recover at least one u_i. In the second phase of our algorithm, we show that in fact we can obtain d very small linear combinations of the u_i.

Recovering the u_i. The vectors of the new lattice basis have the property to be all of about the same length. Consequently, each vector $\mathbf{b_i}$ of the new basis gives a small integer combination of the s_i and so of the u_i. Experimentally, we observe that the linear combination of the u_i is null except for the last one which is equal to ± 1.

Thus, each short vectors returned by LLL gives a small linear combination of the s_i. The matrix returned by the LLL algorithm can be expressed as $C \times M$ where

$$C = \begin{pmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,d} \\ \vdots & \vdots & \ddots & \vdots \\ c_{j,1} & c_{j,2} & \cdots & c_{j,d} \\ \vdots & \vdots & \ddots & \vdots \\ c_{d,1} & c_{d,2} & \cdots & c_{d,d} \end{pmatrix}$$

Each row of C contains the coefficients of a small linear combination of the s_i. The matrix C is invertible since its determinant is ± 1 and thus solving the system $\mathbf{u} \cdot C^T = (0, \ldots, 0, 1)$, where $\mathbf{u} = (u_1, \ldots, u_d)$ allows to recover the u_i.

Once the u_i are obtained, recovering half of the bits of pq is easy by computing $\left\lfloor \frac{s_i}{u_i} \right\rfloor$ for one of the value s_i. Then p is computed according to equation 2. Finally, we have the following theorem:

Theorem 1. *Let $N = p^2 q$ be a n-bit modulus. Given an oracle $\mathcal{O}_{\ell,pq}$ that on input $s \in \mathbb{Z}_N$, returns the ℓ most significant bits of $s \bmod pq$ where $\ell \geq \left\lceil \frac{n}{3(d-1)} + \log(d) + 1 \right\rceil$ and $d < n$, there exists a probabilistic polynomial-time algorithm in n to factor N from d random and independent numbers s in $\mathbb{Z}_N$.*

3.3 Extending the attack to the Least Significant Bits

In this paper, we focus on the importance of MSBs confidentiality. However, such a presentation has been chosen for simplicity reasons since the same analysis can be done with the least significant bits. More precisely the knowledge of the ℓ least significant bits of $S_i \bmod pq$, for d values $S_i \in \mathbb{Z}_N$, also allows us to factor N for the obvious reason that the knowledge of the least significant bits can be reduced to the knowledge of the most significant bits, as explained below.

Consider a $3k$-bit Esign modulus $N = p^2 q$. A value S randomly chosen in $\mathbb{Z}_N$ can always be written as $S = r + upq$ where $0 \leq r < pq$ and $u < p$. Assume now that the ℓ LSBs of r, denoted by r_0, are known. Then, the ℓ LSBs of $S - r_0 \bmod pq = r - r_0$ are zero. We now denote by r_1 the $(2k - \ell)$-bit value $(r - r_0)/2^\ell$. Let a be the inverse of $2^\ell \bmod N$. We can note that a is also the inverse of 2^ℓ modulo pq. Consequently, we can compute

$$\begin{aligned}
a \times (S - r_0) &= a \times (r - r_0) \bmod pq \\
&= a \times 2^\ell \times \frac{(r - r_0)}{2^\ell} \bmod pq \\
&= (1 \bmod pq) \times r_1 \bmod pq \\
&= r_1 \bmod pq
\end{aligned}$$

Therefore, $s = a(S - r_0) \bmod N$ can be written as $r_1 + u_1 pq$ for $u_1 < p$ and $r_1 < pq/2^\ell$. Thus s is a candidate input for the matrix M of the algorithm of the previous subsection.

3.4 Application to RSA modulus

It is worth noticing that this algorithm is independent of the special form $N = p^2 q$ of the modulus. It also works for any RSA modulus $N = pq$ as soon as:

$$\ell \geq \left\lceil \frac{n}{2(d-1)} + \log(d) + 1 \right\rceil$$

If $s_i \in \mathbb{Z}_N$ is written as $s_i = r_i + u_i p$, for $r_i < p/2^\ell$ and $u_i < q$, then we can recover the u_i and computing p from $\lfloor \frac{s_i}{u_i} \rfloor$.

As a consequence, if there exists an oracle $\mathcal{O}_{\ell,p}$ which on input $S \in \mathbb{Z}_N$ returns the ℓ most significant bits of $S \bmod p$ where p is a factor of the modulus N, then we can factor N in polynomial time in $\log(N)$. Therefore the problem of finding the ℓ MSBs of $S \bmod p$ for d different random and independent values $S \in \mathbb{Z}_N$, is equivalent to the factorization problem.

4 Partially known nonces in Esign signature scheme

In the following we describe some potential flaws in practical implementations of Esign. The main idea is to notice that the secrecy and the randomness of all the nonces is a crucial security point: the knowledge of only a few bits of these random values is enough to efficiently recover the secret signature key.

Let $N = p^2 q$ an Esign modulus where p and q are two k-bit primes such that $q < p$. The signature scheme is fully described in section 2.

We first consider an attack on Esign when the random nonces are not full-size. Suppose the random number generator is biased so that the most significant bits of the random values are always zero. We show how to efficiently factor the modulus from a small set of signatures by using the technique described in section 3.2. Precisely, suppose the random number generator produces nonces smaller than $pq/2^\ell$, for an integer $\ell \geq 1$, instead of randomly drawing uniformly distributed integers in the interval $[0, pq[$. We know that all the generated signatures may be written as $s = r + upq$ where r is the random nonce. Thus, a signature is a noisy multiple of the secret factor pq. If the number ℓ of null most significant bits of r is sufficiently large, then we can factor N by recovering p with the technique presented above. The attack goes as follows: suppose we have a set of d Esign signatures s_i, for any messages. Each can be written as $s_i = r_i + u_i pq$, for $1 \leq i \leq d$, and where $r_i \leq pq/2^\ell$ and $u_i < p$. As shown in section 3.2 we can recover the u_i by reducing a lattice with the LLL algorithm. As soon as we have $\ell \geq \left\lceil \frac{n}{3(d-1)} + \log(d) + 1 \right\rceil$, where n is the bit size of the modulus N. Then we can write:

$$\frac{s_i}{u_i} = \frac{r_i}{u_i} + pq$$

We remark that $\left\lfloor \frac{r_i}{u_i} \right\rfloor$ is at most a k-bit integer. Thus, we can finally recover p according to equation 2. Experimented results are provided below. The tests have been run on an Intel Pentium IV, XEON 1.5 GHz, with the Shoup's library NTL ([24]). For each modulus length n, we give the length of pq (that is also the expected length for the random r), the effective length of the nonce r, the experimental and theoretical bounds for ℓ, and the time needed to recover p and q. The number of required signatures is d.

We observe that the experimental bound for ℓ is better than expected. This can be simply explained by the good performances of the LLL algorithm. In

$n = 3k$	$2k$	$\log(r)$	experimental value for ℓ	theoretical bound for ℓ	d	time to factor
512	340	335	5	8	55	2 min 10
768	512	506	6	9	55	2 min 20
1024	682	674	8	11	56	2 min 30
1152	768	760	8	11	57	3 min
1536	1024	1013	11	14	57	4 min 10
2048	1364	1349	15	17	57	5 min 50

Fig. 1. Experimental results on Esign with partially known nonces.

practice, this algorithm works indeed better than expected and the vectors returned are shorter than the provable upper bounds. Another explanation can be made for this fact: in section 3.2, we have used a theoretical bound on the sum $\sum_{i=1}^{d} c_i s_i \leq dcN$. This bound has then been used to find the theoretical bound on ℓ. However, in practice, the sum $\sum_{i=1}^{d} c_i s_i$ is approximately $\sqrt{d} \cdot cN$ on average. Thus, this gives a smaller bound on ℓ: the algorithm works as soon as $\ell \geq \left\lceil \frac{n}{3(d-1)} + \frac{\log(d)}{2} + 1 \right\rceil$. This gives results closer to the experimental results. This bound is given in figure 1.

Hence, if the random number generator produces nonces in an interval smaller than expected, then recovering the secret key can be made from a small set of signatures, for any messages. However, even if the random values are generated in all the interval $[0, pq[$, the difference between two consecutive nonces should not be too small. Indeed, in this case, the same attack applies: considering the differences $s_{i+1} - s_i$ whose most significants bits modulo pq are small, gives the same results.

Thus, the random number generator is a crucial security point and the nonces should be generated uniformly and independently in the range $[0, pq[$. If we now consider physical attacks on probing or electromagnetic analysis, the attack can also be mounted as soon as the observation of the ℓ MSB or LSB of the random nonces is feasible. This may be realistic using smart cards.

5 Other potential weaknesses in Esign implementations

In [2], Bellare, Goldwasser and Miccianco have pointed out that using linear congruential generator in DSS signature scheme is totally insecure. The secret key can be easily recovered in this case, and even if the outputs of the generator are truncated. As for DSS, using a linear congruential generator (LCG) with public parameters leads to insecure implementations of Esign.

Such a generator is parameterized by integers a, b, M and is based on a linear recurrence: $r_{i+1} = ar_i + b \bmod M$. The initial seed r_0 is the secret. We consider the security of Esign in this case and we show that the knowledge of only two signatures allows to recover the secret signature key. Suppose that Esign is used with the pseudo-random generator defined by $r_{i+1} = ar_i + b \bmod M$ where M is a secret multiple of pq, less than N, and a and b are public integers in $\mathbb{Z}_M$. The initial state r_0, that should not be reset, is kept secret as part of the private key. The modulus M is chosen to be a multiple of pq so that after reduction modulo pq in the signature generation, the generated random values are still uniformly distributed in the range $[0, pq[$. Such a choice seems to be the most natural one.

For any positive index i, we have $s_i = r_i + u_i pq$. Thus the following equality holds:

$$s_{i+1} = r_{i+1} + u_{i+1}pq = ((ar_i + b) \bmod M) \bmod pq + u_{i+1}pq$$

Thus, since a and b are public and M is a multiple of pq, one can compute $s_{i+1} - as_i - b$ which is a multiple of pq. Its GCD with the modulus N is pq, and the secret key is found.

Note that this can also be applied even if the parameter b is secret. With only four signatures s_i, s_{i+1}, s_j and s_{j+1}, the secret factor pq can also be recovered. Indeed, it suffices to compute $(s_{i-1} - s_i) - (s_{j+1} - s_j) = (u_{i+1} - u_i + u_{j+1} - u_j + K)pq$ where K is an integer. The GCD of N with this difference reveals pq.

Finally, using a linear congruential generator is insecure in this case.

6 Conclusion

In conclusion we have shown in this paper that Esign must be carefully implemented since like many other public key cryptosystems, security of ephemeral keys is of crucial importance. We also insist on the idea that physical techniques like probing or electromagnetic analysis can be very efficiently combined with more theoretical algorithmic cryptanalysis methods, for example based on LLL.

References

1. M. Ajtai, R. Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *Proceedings of the 33rd Annual Symposium on the Theory of Computing (STOC) 2001*, pages 601 – 610. ACM Press, 2001.
2. M. Bellare, S. Goldwasser, and D. Miccianco. "Pseudo-Random" Number Generation within Cryptographic Algorithms: the DSS Case. In B. Kaliski, editor, *Advances in Cryptology – Crypto'97*, volume 1294 of *LNCS*, pages 277 – 291. Springer-Verlag, 1997.
3. D. Bleichenbacher. On the Generation of DSA One-Time Keys. In *The 6th Workshop on Elliptic Curve Cryptography (ECC 2002)*, 2002.
4. D. Boneh. Simplified OAEP for the RSA and Rabin Functions. In M. Bellare, editor, *Advances in Cryptology – Crypto'01*, volume 2139 of *LNCS*, pages 275 – 291. Springer-Verlag, 2001.

5. D. Boneh, G. Durfee, and N. Howgrave-Graham. Factoring $n = p^r q$ for large r. In M. Wiener, editor, *Advances in Cryptology – Crypto'99*, volume 1666 of *LNCS*, pages 326 – 337. Springer-Verlag, 1999.
6. D. Boneh, G. Durfee, and Y. Frankel. An attack on RSA given a fraction on the private key bits. In K. Ohta and D. Pei, editors, *Advances in Cryptology – Asiacrypt'98*, volume 1514 of *LNCS*, pages 25 – 34. Springer-Verlag, 1998.
7. D. Boneh and R. Venkatesan. Hardness of Computing the Most Significant Bits of Secret Keys in Diffie-Hellman and Related Schemes. In N. Koblitz, editor, *Advances in Cryptology – Crypto'96*, volume 1109 of *LNCS*, pages 129 – 142. Springer-Verlag, 1996.
8. E. F. Brickell and J. M. DeLaurentis. An attack on a signature scheme proposed by Okamoto and Shiraishi. In H. C. Williams, editor, *Advances in Cryptology – Crypto '85*, volume 218 of *LNCS*, pages 28 – 32. Springer Verlag, 1985.
9. E. F. Brickell and A. M. Odlyzko. Cryptanalysis: A Survey of Recent Results. In G. J. Simmons, editor, *Contemporary Cryptology – The Science of Information Integrity*, pages 501 – 540. IEEE Press, 1991.
10. P. A. Fouque, G. Martinet, and G. Poupard. Attacking Unbalanced RSA-CRT using SPA, 2003. To appear in the Proceedings of CHES'03.
11. E. Fujisaki, T. Kobayashi, H. Morita, H. Oguro, T. Okamoto, and S. Okasaki. ESIGN: Efficient Digital Signature Scheme, submission to NESSIE, 2000.
12. J. van zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, 1999.
13. M. Girault, P. Toffin, and B. Vallée. Computation of Approximate l-th Roots Modulo n and Application to Cryptography. In S. Goldwasser, editor, *Advances in Cryptology – Crypto'88*, volume 403 of *LNCS*, pages 100 – 117. Springer Verlag, 1988.
14. L. Granboulan. How to repair Esign. In *SCN'02*. Springer-Verlag, 2002.
15. N. Howgrave-Graham. A Review of the ESIGN digital signature standard, 2001. Available at `http://www.shiba.tao.go.jp/kenkyu/CRYPTREC/fy15/cryptrec20030424_outrep.html`. Report #1007.
16. N. Howgrave-Graham and N. P. Smart. Lattice Attacks on Digital Signature Schemes. *Design, Codes and Cryptography*, 23:283 – 290, 2001.
17. R. Kannan. Algorithmic geometry of numbers. *Annual Review of Computer Science*, 2:231 – 267, 1987.
18. A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring Polynomials with Rational Coeficients. *Mathematische Annalen*, 261(4):515 – 534, 1982.
19. P. Q. Nguyen and I. E. Shparlinski. The Insecurity of the Digital Signature Algorithm with Partially Known Nonces. *Journal of Cryptology*, 15(3):151 – 176, 2002.
20. P. Q. Nguyen and J. Stern. The Two Faces of Lattices in Cryptography. In J. Silverman, editor, *Proc. of Cryptography and Lattices Conference*, volume 2146 of *LNCS*, pages 146 – 180. Springer-Verlag, 2001.
21. NIST. Digital Signature Standard (DSS). Federal Information Processing Standards PUBlication 186, November 1994.
22. R. Novak. SPA-Based Adaptive Chosen-Ciphertext Attack on RSA Implementation. In D. Naccache and P. Paillier, editors, *Public Key Cyptography – PKC'2002*, volume 2274 of *LNCS*, pages 252 – 261. Springer-Verlag, 2002.
23. T. Okamoto and A. Shiraishi. A Digital Signature Scheme Based on Quadratic Inequalities. *Proceedings of Symposium on Security and Privacy*, pages 123 – 132, 1985.

24. V. Shoup. Number Theory C++ Library (NTL), version 5.0b. Available at `http://www.shoup.net`.
25. J. Stern, D. Pointcheval, J. Malone Lee, and P. Smart. Flaws in Applying Proof Methodologies to Signatures Schemes. In M. Yung, editor, *Advances in Cryptology – Crypto'02*, volume 2442 of *LNCS*, pages 93 – 110. Springer-Verlag, 2002.
26. B. Vallée, M. Girault, and P. Toffin. How to Break Okamoto's Cryptosystem by Reducing Lattices Bases. In C. G. Günther, editor, *Advances in Cryptology – Eurocrypt'88*, volume 330 of *LNCS*, pages 281 – 291. Springer Verlag, 1988.