Evaluation Report on the **ESIGN** signature scheme

Jacques Stern

1 Introduction

This document is an evaluation of the ESIGN signature scheme. Our work is based on the analysis of various documents [12, 13, 14, 23], which provide the specification of the scheme, as well as on various research papers related to the scheme. Among these research papers are the original work [24, 22], proposing the design and the subsequent cryptanalytic work [28, 16] of Vallée and *al.*

The present report is organized as follows: firstly, we review the ESIGN primitive, and we discuss its relation to the so-called approximate e-th root problem (AERP). Next, we recall the strongest security notion that is now mandatory for signature schemes: security against existential forgery under adaptive chosen-message attacks. This allows us to provide a proof of the security of ESIGN against a slight variant of adaptive chosen-message attacks, in the random oracle model. This proof is based on the assumption that AERP is intractable. We then analyze the security of the proposed ESIGN signature in view of the previous proofs. In particular, we discuss whether one can derive practical implications, notably in terms of key sizes. This is as requested by IPA.

2 The ESIGN primitive and its relation to AERP

In this section, we review the RSA primitive, mainly for notational purposes, and introduce the specific properties of RSA moduli of the form p^2q. This leads us to discuss the approximate e-th root problem.

2.1 The RSA primitive

The famous RSA primitive has been proposed by Rivest, Shamir and Adleman [26]. On input a security parameter k, the key generation algorithm of RSA chooses two large primes p, q of equal size and issues the so-called modulus $n = pq$. The sizes of p, q are set in such a way that the binary length $|n|$ of n equals $2k$. Additionally, an exponent e, relatively prime to $\varphi(n) = (p-1)(q-1)$ is chosen, so that the public key is the pair (n, e).

The basic security assumption on which all uses of the RSA primitive rely is its *one-wayness* (OW): using only public data, an attacker cannot invert the RSA function

$$x :\longrightarrow (x^e) \bmod n$$

More precisely, denote by $\mathsf{Succ}^{\mathsf{rsa}}(\tau, k)$ the probability for an adversary to find the preimage of a given element within time τ, in symbols:

$$\mathsf{Succ}^{\mathsf{rsa}}(\tau, k) = \Pr[((n,e),d) \leftarrow \mathcal{K}(1^k), y \leftarrow \mathbb{Z}_N, x \leftarrow \mathcal{A}(n,e,y) : y = x^e \bmod n],$$

then, for large enough moduli, this probability is extremely small. An asymptotic version states that, when $\mathcal{A}$ has running time bounded by a polynomial function τ of the security parameter, $\mathsf{Succ}^{\mathsf{rsa}}(\tau, k)$ eventually becomes smaller than the inverse of any polynomial in k.

It is well known that, if d is the inverse of e modulo $\varphi(n)$, $y^d \bmod n$ computes the inverse of the RSA function. Thus, the factorization of n allows to invert the RSA function, since d can be computed from p and q. It is unknown whether the converse is true, i.e. whether factoring and inverting RSA are computationally equivalent. There are indications that it might not be true (see [4]). Thus, the assumption that RSA is one-way might be a stronger assumption than the hardness of factoring. Still, it is a widely believed assumption and the only method to assess the strength of RSA is to check whether the size of the modulus n outreaches the current performances of the various factoring algorithms.

2.2 RSA moduli of the form p^2q

Variants of RSA allow the use of more than two prime factors. Of particular interest in the present report is the case of a modulus $n = p^2q$, where p, q are two prime numbers of equal size. Key generation of such moduli involves a straightforward modification: on input a security parameter k, the key generation algorithm chooses two large primes p, q of equal size k and issues the so-called modulus $n = p^2q$. The sizes of p, q are set in such a way that the binary length $|n|$ of n equals $3k$. Additionally, an exponent e, relatively prime to $\varphi(n) = (p-1)(q-1)$ is chosen, so that the public key is the pair (n, e).

The following lemma discloses an algebraic property of the RSA function, which is the basis of the design of ESIGN.

Lemma 1 *Let r be an integer in $\mathbb{Z}_n^\star$. Let α be the inverse of er^{e-1} in $\mathbb{Z}_n^\star$. Then, for any t in $\mathbb{Z}_n$*

$$(r + \alpha tpq)^e = r^e + tpq \bmod n$$

Proof. The proof relies on the easy equality

$$\begin{aligned}(r+\alpha tpq)^e &= \sum_{i=0}^{e}\binom{e}{i} r^{e-i}(\alpha tpq)^i \bmod n \\ &= r^e + er^{e-1}\alpha tpq \bmod n \\ &= r^e + tpq \bmod n\end{aligned}$$

□

Thus, given the trapdoor pq and the power $r^e \bmod n$ of a randomly chosen r, one can obtain an arithmetical progression consisting of e-th powers of known integers. This offers an extremely efficient way to find an element of $\mathbb{Z}_n^\star$ whose e-th power lies in a prescribed interval of length at least pq. One simply adjusts parameter t in the above. Even if the interval is of length slightly $< pq$, but of the same order of magnitude, the method is successful after a few trials. Thus interval lengths of 2^{2k}, 2^{2k-1}, $n^{2/3}$ are appropriate. Of course, it is also possible to invert the RSA function by computing the inverse of e modulo $\varphi(n) = p(p-1)(q-1)$. However, in settings where e is small, this is much less efficient.

2.3 The approximate e-th root problem

As explained in the previous section, RSA moduli of the from p^2q offer a very efficient way to solve the following problem, having knowledge of the factorization of n: given n and y in $\mathbb{Z}_n^\star$, find x such that $x^e \bmod n$ lies in the interval $[y, y+2^{2k-1})$, where the bit-size of n is $3k$ and $[y, y+2^{2k-1})$ denotes $\{u | y \leq u < y+2^{2k-1}\}$.

It is conjectured that the above problem, called the approximate e-th root problem (AERP) in [23], is hard to solve. More precisely, denote by $\mathsf{Succ}^{\mathsf{aerp}}(\tau, k)$ the probability for an adversary to find an element whose e-th power lies in the prescribed interval, within time τ, in symbols:

$$\mathsf{Succ}^{\mathsf{aerp}}(\tau,k) = \Pr[(n,e) \leftarrow \mathcal{K}(1^k), y \leftarrow \mathbb{Z}_N, x \leftarrow \mathcal{A}(N,e,y) : (x^e \bmod n) \in [y, y+2^{2k-1})],$$

then, for large enough moduli, this probability is extremely small. As usual, the asymptotic version states that, when $\mathcal{A}$ has running time bounded by a polynomial function τ of the security parameter, $\mathsf{Succ}^{\mathsf{aerp}}(\tau, k)$ eventually becomes smaller than the inverse of any polynomial in k. Variants of the above can be considered, where the length of the interval is replaced by 2^{2k} or 2^{2k+1}.

Of course, the factorization of n allows to solve the AERP problem. It is unknown whether the converse is true, i.e. whether AERP and inverting RSA are computationally equivalent. This is conjectured in [14], as soon as $e \geq 4$.

2.4 Small exponent AERP

2.4.1 The basic attack

Although it is not reported in clear terms in the literature, there is a straightforward attack against AERP, for $e \leq 3$. It is based on the following easy lemma.

Lemma 2 *Let δ be a fixed positive real. For any large enough integer y, there is an e-th power in the interval $[y, y + e(1+\delta)y^{\frac{e-1}{e}})$.*

Proof. Computing over the reals, we let $a = y^{1/e}$. Using the derivative of the function x^e, we get

$$(a+1)^e \leq a^e + e(a+1)^{e-1} \leq y + e(y^{1/e}+1)^{e-1} \leq e(y^{\frac{e-1}{e}})(1+\frac{1}{y^{1/e}})^{e-1} \leq e(1+\delta)y^{\frac{e-1}{e}}$$

Thus $\lceil a+1 \rceil$ has e-th power in the prescribed interval. $\square$

When $e = 2$, and y is in $\mathbb{Z}_n^\star$, the interval length is much smaller than 2^{2k-1}. When $e = 3$, the interval's length is smaller than 2^{2k-1} as long as $y \leq (\frac{2^{2k-1}}{e(1+\delta)})^{3/2}$, which is a significant proportion of all possible y's. Thus, in both cases, extracting eth root over the reals provides an easy forgery.

For all practical purposes, we will set $\delta = 0$ in the above lemma. This is a way to use the lemma in a probabilistic manner.

2.4.2 Improved attacks

The naive attack described in the previous section for $e = 2, 3$ can be avoided by putting a lower bound on the e-th root to be found: it is indeed the case that the attack produces a root of small value $\leq n^{1/e}$. Note that there is no such requirement in the proposed version of ESIGN.

We will show that such protection can actually be bypassed, as demonstrated in [5]. We first consider the case $e = 2$. In this case, we can search for a solution to AERP of the form $x+tu$, for suitably chosen x and u, so that the product $c = 2xu \bmod n$ is small. For example, we can choose $u = n^\beta$, $\beta < 1/12$, and pick x so that $n \leq 2xu < n + 2n^\beta$.

$$(x+tu)^2 = x^2 + 2txu + t^2u^2 = x^2 + 2tc + t^2 \bmod n$$

We set $z = (y - x^2) \bmod n$. Next, we adjust $a > 0$ so that au^2 belongs to the interval $[z, z + n^{2\beta})$. We then use lemma 2 in order to choose an integer t, such that t^2 is in the interval $[a, a + 2\sqrt{a})$. It follows that

$$\begin{aligned} z \leq au^2 &\leq t^2u^2 \\ &\leq u^2(a + 2\sqrt{a}) \\ &\leq z + 2n^{2\beta} + 2n^{2\beta}\sqrt{a} \\ &\leq z + 2n^{2\beta}\sqrt{n}. \end{aligned}$$

Adding x^2 mod n encloses

$$(x^2 + t^2u^2) \bmod n$$

in an interval with lower end point y and length $\leq 2n^{2\beta+1/2}$. Further adding $2txu$ yields a square in an interval with lower end point y and length $\leq 2n^{2\beta+1/2} + 2\sqrt{n}n^{\beta} < n^{2/3}$.

The attack does not extend to the case $e = 3$. However, there is a way to bypass the protection offered by bounding the square root from below, in this case as well. Let x be such that $3x = n \pm 1$. Write:

$$(x + 3u)^3 = x^3 + 9x^2u + 27xu^2 + 27u^3 \bmod n$$

Since $3x$ mod n is ± 1, then, provided u is small, the above is

$$(x^3 \bmod n) + u \pm 9u^2 + 27u^3$$

let $z = (y - x^3) \bmod n$. Using lemma 2 three times, we can find the cubes of three consecutive integers in the interval $[z, z + 9n^{2/3})$. One of them is divisible by 3 and we interpret it as $3u$ in the above formula. Note that $3u$ is bounded by $n^{1/3}$. Adding $x^3 \bmod n + u \pm 9u^2$, we enclose

$$(x^3 \bmod n) + u \pm 9u^2 + 27u^3$$

in an interval with lower end point y and length $\leq 9n^{2/3} + n^{2/3} + n^{1/3}$. Thus, we have found a cube in an interval, whose prescribed lower end is y, and whose length is of the same order of magnitude as $n^{2/3}$. This provides a forgery attack, which is successful with significant probability.

2.4.3 Lattice attacks

In [28], Vallée, Girault and Toffin extended the attack for the case $e = 2$, to show that it was possible to constrain the square root in a prescribed interval of length $n^{1/3+\varepsilon}$, centered at x, while still having the square in a prescribed interval of length $n^{2/3+\varepsilon}$, centered at y. The algorithm is deterministic, once x and y have been chosen. However, it may fail for certain values of x. The set of exceptional values are in proportion $\simeq n^{-\varepsilon}$. In the above, ε is any fixed small constant, but it is easily seen that n^{ε} can essentially be set to any increasing function of the security parameter. Thus, the result easily translates into a forgery: picking ε small enough, one ends up with a square in a prescribed interval of length $n^{1/3}$, with significant probability $\simeq \frac{1}{n^{\varepsilon}}$.

The attack uses two dimensional lattices. We let $\alpha = 2x \bmod n$ and we consider the lattice $L(\alpha)$ generated by the columns of the following matrix:

$$M = \begin{pmatrix} \lfloor n^{1/3-\varepsilon} \rfloor & 0 \\ \alpha & n \end{pmatrix}$$

We first claim that, for all values of $\alpha < n$, except a fraction $\frac{4}{n^{\varepsilon}}$ of them, $L(\alpha)$ has no non zero element of euclidean length $\leq n^{2/3-\varepsilon}$. To check the claim, observe that such an element corresponds to a linear combination of the columns of M, with coefficients u, v such that $\lfloor n^{1/3-\varepsilon} \rfloor u \leq n^{2/3-\varepsilon}$. This allows about $2n^{1/3}$ values of u. We further have $|\alpha u \bmod n| \leq n^{2/3-\varepsilon}$, which allows $2n^{2/3-\varepsilon}$ values for α, for each fixed u. Altogether, this makes $4n^{1-\varepsilon}$ exceptional values of α.

We restrict our attention to the case where $L(\alpha)$ does not have a non-zero vector of euclidean length $\leq n^{2/3-\varepsilon}$. We use the wording *good values* for the corresponding αs. Given a good α, one applies the Gaussian reduction algorithm. One gets within time $\mathcal{O}((\log n)^3)$ a basis of $L(\alpha)$ consisting of two non-zero vectors U and V such that

$$\|U\| \leq \|V\| \text{ and } |(U, V)| \leq \|U\|^2/2.$$

where (U, V) denotes the usual inner product. If we define θ by $\cos\theta = \frac{(U,V)}{\|U\|.\|V\|}$, $-\pi/2 \leq \theta \leq \pi/2$, we can compute the absolute value of the determinant of $L(\alpha)$ as $\|U\|.\|V\| \sin\theta$. Since we have $|\cos\theta| \leq 1/2$, we get $\sin\theta \geq \sqrt{3}/2$. Observing that the determinant is precisely $n\lfloor n^{1/3-\varepsilon} \rfloor$, we get the approximate upper bound $n^{4/3-\varepsilon}$ for $(\sqrt{3}/2)\|U\|.\|V\|$. Since $\|U\|$ is at least $n^{2/3-\varepsilon}$, it follows that $\|V\|$ is bounded by $\frac{2}{\sqrt{3}} n^{2/3}$.

We now consider $T = (0, (y - x^2) \bmod n)$, where y is the center of the prescribed interval where a square should be found.

$$T = \lambda U + \mu V, \text{ for some real } \lambda, \mu.$$

and define λ', μ' as the closest integers to λ, μ respectively. Let $S = \lambda' U + \mu' V$. Observe that $T - S$ is such that

$$\|T - S\|^2 \quad = \quad (1/2)^2\|U\|^2 + (1/2)^2\|V\|^2 + (1/2)|(U, V)| \leq \|V\|^2$$

Thus, $\|T - S\|$ is bounded by $\frac{2}{\sqrt{3}} n^{2/3}$. Now, S is a lattice element and can be written as a combination of the column vectors of M. Let u, v be the corresponding coefficients. We can write the coordinates of $T - S$ as $(-\lfloor n^{1/3-\varepsilon} \rfloor u)$ and $((y - x^2) \bmod n - u\alpha - vn)$. We get

$$|u|^2 \leq \frac{4}{3} n^{2/3+2\varepsilon}$$

$$|(y - x^2 - u\alpha \bmod n)|^2 \leq \frac{4}{3} n^{4/3}$$

Since α is $2x \bmod n$, this encloses

$$(x^2 + 2xu + u^2) \bmod n$$

in an interval centered at y, of length $\simeq \frac{4}{3} n^{2/3+\varepsilon}$. This is the square of $(x+u)$, an integer lying in the interval centered at x, of length $\simeq \frac{2}{\sqrt{3}} n^{2/3+\varepsilon}$. Note that the multiplicative constants are irrelevant since they can easily be absorbed into n^{ϵ}.

2.5 Conclusion

It is fair to say that there is no known attack against AERP when e is ≥ 4. In particular, it should be emphasized that the attack of Vallée and *al.*, only applies to $e = 2$. There happens to be some kind of common misunderstanding about this early work, maybe because a confusion arises between the work in [28] and subsequent work [29] by the same authors, where they applied lattice reduction to the problem of solving modular polynomial equations of low degree. This pioneering work was spectacularly improved by Coppersmith [7, 9]. However, as far as we know, it has not proved useful in the context of ESIGN.

3 The provable security of ESIGN

In this section, we review the strongest security notion that is currently required for signature schemes: security against existential forgery under adaptive chosen-message attacks. Based on the assumption that AERP is intractable, we provide a proof, in the random oracle model, that ESIGN almost meets this security level. More accurately, we introduce a slight variation of adaptive chosen-message attacks, which we call single-occurrence chosen message attacks, which appears needed to carry the proof. Besides being based on a slightly weaker security model, our proof is different in spirit from [23] and is therefore of independent interest. Contrary to what is claimed in [23], we believe that the security result cannot be extended to the stronger security model.

3.1 Digital signatures and their security

In modern terms (see [17]), a digital signature scheme consists of three algorithms $(\mathcal{K}, \Sigma, V)$:

- A *key generation algorithm* $\mathcal{K}$, which, on input 1^k, where k is the security parameter, outputs a pair $(\mathsf{pk}, \mathsf{sk})$ of matching public and private keys. Algorithm $\mathcal{K}$ is probabilistic.
- A *signing algorithm* Σ, which receives a message m and the private key sk, and outputs a signature $\sigma = \Sigma_{\mathsf{sk}}(m)$. The signing algorithm might be probabilistic.
- A *verification algorithm* V, which receives a candidate signature σ, a message m and a public key pk, and returns an answer $V_{\mathsf{pk}}(m, \sigma)$ testing whether σ is a valid signature of m with respect to pk. In general, the verification algorithm need not be probabilistic.

Attacks against signature schemes can be classified according to the goals of the adversary and to the resources that it can use. The goals are diverse:

- Disclosing the private key of the signer. It is the most drastic attack. It is termed *total break.*
- Constructing an efficient algorithm which is able to sign any message with significant probability of success. This is called *universal forgery.*
- Providing a single message/signature pair. This is called *existential forgery.*

In many cases the latter does not appear dangerous because the output message is likely to be meaningless. Nevertheless, a signature scheme, which is not existentially unforgeable, does not guarantee by itself the identity of the signer. For example, it cannot be used to certify randomly looking elements, such as keys or compressed data. Furthermore, it cannot formally guarantee the so-called non-repudiation property, since anyone may be able to produce a message with a valid signature.

In terms of resources, the setting can also vary. We focus on two specific attacks against signature schemes: the *no-message attacks* and the *known-message attacks.* In the first scenario, the attacker only knows the public key of the signer. In the second, the attacker has access to a list of valid message/signature pairs. Again, many subcases appear, depending on how the adversary gains knowledge. The strongest is the *adaptive chosen-message attack* (CMA), where the attacker can require the signer to sign any message of its choice, where the queries are based upon previously obtained answers. When signature generation is not dterministic, there may be several signatures corresponding to a given message. A slightly weaker security model, which we call *single-occurrence adaptive chosen-message attack* ($\mathsf{SO\text{-}CMA}$), allows the adversary at most one signature query for each message. In other words the adversary cannot submit the same message twice for signature.

In chosen-message attacks, one should point out that existential forgery becomes the ability to forge a fresh message/signature pair that has not been obtained from queries asked during the attack. Again there is a subtle point here, related to the context where several signatures may correspond to a given message. We actually adopt the stronger rule that the attacker needs to forge the signature of message, whose signature was not queried.

When designing a signature scheme, one wishes to rule out existential forgeries, even under adaptive chosen-message attacks. More formally, one requires that the success probability of any adversary $\mathcal{A}$, whose running time remains below some security bound t, is negligible, where the success probability is defined by:

$$\mathsf{Succ}^{\mathsf{cma}}(\mathcal{A}) = \Pr\left[(\mathsf{pk},\mathsf{sk}) \leftarrow \mathcal{K}(1^k), (m,\sigma) \leftarrow \mathcal{A}^{\Sigma_{\mathsf{sk}}}(\mathsf{pk}) : V_{\mathsf{pk}}(m,\sigma) = 1\right].$$

In the above, note the superscript Σ_{sk}, indicating adaptive calls to the signing algorithm: this is consistent with the framework of relativized complexity theory, where oracle calls are allowed and, accordingly, we will use the wording *signing oracle* in this setting. When dealing with single-occurrence attacks, $\mathsf{Succ}^{\mathsf{cma}}(\mathcal{A})$ is replaced by an appropriately defined variant $\mathsf{Succ}^{\mathsf{so-cma}}(\mathcal{A})$.

3.2 The random oracle model

Ideally, one would like to provide appropriate formatting rules for ESIGN, such that the resulting signature scheme has provable security, based on the sole assumption that AERP is hard. Unfortunately, as for many other schemes, no formatting rule is currently known that allows such a proof.

Thus, the best one can hope for is a proof carried in a non-standard computational model, as proposed by Bellare and Rogaway [1], following an earlier suggestion by Fiat and Shamir [15]. In this model, called the random oracle model, concrete objects such that hash functions are treated as random objects. This allows to carry through the usual reduction arguments to the context of relativized computations, where the hash function is treated as an oracle returning a random answer for each new query. A reduction still uses an adversary as a subroutine of a program that contradicts a mathematical assumption, such as the assumption that RSA is one-way. However, probabilities are taken not only over coin tosses but also over the random oracle.

Of course, the significance of proofs carried in the random oracle is debatable. Firstly, hash functions are deterministic and therefore do not return random answers. Along those lines, Canetti *et al.* [6] gave an example of a signature scheme which is secure in the random oracle model, but insecure under any instantiation of the random oracle. Secondly, proofs in the random oracle model cannot easily receive a quantitative interpretation. One would like to derive concrete estimates - in terms of key sizes - from the proof: if a reduction is efficient, the security "loss" is small and the existence of an efficient adversary leads to an algorithm for solving the underlying mathematical problem, which is almost as efficient. Thus, key sizes that outreach the performances of the known algorithms to break the underlying problem, can be used for the scheme as well.

Despite these restrictions, the random oracle model has proved extremely useful to analyze many encryption and signature schemes. It clearly provides an overall guarantee that a scheme is not flawed, based on the intuition that an attacker would be forced to use the hash function in a non generic way.

3.3 The provable version of ESIGN

Specifications of ESIGN, which appear in various documents, slightly differ from one version to another. Document [13] sums up the changes. In this section, we will describe the encoding rule from [23], and provide the corresponding security proof. In a subsequent section, we will discuss whether any of the proposed modifications degrades security.

3.3.1 Description

In document [23], the security parameter k is also denoted by pLen. The key generation algorithm chooses two large primes p, q of equal size k and computes the modulus $n = p^2 q$. The sizes of p, q are set in such a way that the binary length $|n|$ of n equals $3k$. Additionally, an exponent $e > 4$ is chosen. It is recommended that $e = 2^\ell$. Key generation is successful when n has $3k$ bits. Otherwise, it is performed anew.

Signature generation is performed as follows, using a hash function $\mathtt{H}$, outputting strings of length $\mathsf{pLen} - 1$.

1. Pick at random r in $\mathbb{Z}^\star_{pq}$.
2. Convert $(0\|\mathtt{H}(m)\|0^{2k})$ into an integer y and compute $z = (y - r^e) \bmod n$.
3. Compute
$$w_0 = \lceil \frac{z}{pq} \rceil$$
$$w_1 = w_0.pq - z$$
If $w_1 \geq 2^{2k-1}$, return to step 1.
4. Set $u = (w_0.er^{e-1}) \bmod p$ and $s = r + upq$.
5. Output s as the signature of m.

Note that the algorithm improves on the basic paradigm of ESIGN. For example, it picks r modulo pq and not modulo n. Also, it computes the inverse of er^{e-1} modulo p and not modulo n. It can be seen that these changes do not affect the basic method: the arithmetical progression $r^e \bmod n + tpq$ consists of e-th powers of easily computed integers, and one adjusts t so as to fall into a prescribed interval. Note that the test at step 3 actually sets the length of this prescribed interval to 2^{2k-1}.

Signature verification converts integer $s^e \bmod n$ into a bit string S of length $3k$ and checks that $[S]^k = 0\|\mathtt{H}(m)$, where $[S]^k$ denotes the k leading bits of S.

3.3.2 Security proof

For this signature scheme, one can prove, in the random oracle model, the following security result, where $T_{exp}(k)$ denotes the computing time of modular exponentiation modulo a $3k$-bit integer.

Theorem 1 *Let $\mathcal{A}$ be a SO-CMA-adversary against the ESIGN signature scheme that produces an existential forgery, with success probability ε, within time τ, making q_H queries to the hash function and q_s distinct requests to the signing oracle respectively. Then, AERP can be solved with probability ε', and within time τ', where*

$$\varepsilon' \geq \frac{\varepsilon}{q_H} - (q_H + q_s) \times (3/4)^k - \frac{1}{2^{k-1}} \quad \textit{and} \quad \tau' \leq \tau + k(q_s + q_H) \cdot T_{exp}(k).$$

Before turning to the proof, we clarify our notion of existential forgery: a forgery, that provides a second signature of a message for which the adversary has already obtained one from the signing oracle, is not accepted. We do not know how to extend the proof to deal with such forgeries. We also mention that, contrary to what is claimed in [23], the result only applies to single-occurrence adaptive chosen message attacks. We do not know how to extend the proof to deal with the stronger CMA model. We finally note that our method of proof is inspired by Shoup [27] and differs from [23]: we define a sequence of Game_1, Game_2, etc of modified attack games starting from the actual game Game_0. Each of the games operates on the same underlying probability space: the public and private key of the signature scheme, the coin tosses of the adversary $\mathcal{A}$, the values of the random oracles. Only the rules defining how the view is computed differ from game to game. To go from one game to another, we repeatedly use the following lemma from [27]:

Lemma 3 *Let E_1, E_2 and F be events defined on a probabilistic space*

$$\Pr[\mathsf{E}_1 \wedge \neg\mathsf{F}] = \Pr[\mathsf{E}_2 \wedge \neg\mathsf{F}] \implies |\Pr[\mathsf{E}_1] - \Pr[\mathsf{E}_2]| \leq \Pr[\mathsf{F}].$$

Proof. The proof follows from easy computations:

$$\begin{aligned}
|\Pr[\mathsf{E}_1] - \Pr[\mathsf{E}_2]| &= |\Pr[\mathsf{E}_1 \wedge \neg\mathsf{F}] + \Pr[\mathsf{E}_1 \wedge \mathsf{F}] - \Pr[\mathsf{E}_2 \wedge \neg\mathsf{F}] - \Pr[\mathsf{E}_2 \wedge \mathsf{F}]| \\
&= |\Pr[\mathsf{E}_1 \wedge \mathsf{F}] - \Pr[\mathsf{E}_2 \wedge \mathsf{F}]| = |\Pr[\mathsf{E}_1 \,|\, \mathsf{F}] \cdot \Pr[\mathsf{F}] - \Pr[\mathsf{E}_2 \,|\, \mathsf{F}] \cdot \Pr[\mathsf{F}]| \\
&\leq |\Pr[\mathsf{E}_1 \,|\, \mathsf{F}] - \Pr[\mathsf{E}_2 \,|\, \mathsf{F}]| \cdot \Pr[\mathsf{F}] \leq \Pr[\mathsf{F}]
\end{aligned}$$

□

Proof.(of theorem 1). We consider an adversary $\mathcal{A}$ outputting an existential forgery (m, s), with probability ε, within time τ. We denote by q_H and q_s respectively the number of queries from the random oracle $\mathtt{H}$ and from the signing oracle. As explained, we start by playing the game coming from the actual adversary, and modify it step by step, until we reach a final game, whose success probability has an upper-bound obviously related to solving AERP.

Game_0**:** The key generation algorithm $\mathcal{K}(1^k)$ is run and produces a pair of keys $(\mathsf{pk}, \mathsf{sk})$. The adversary $\mathcal{A}$ is fed with pk and, querying the random oracle $\mathtt{H}$ and the signing oracle Σ_{sk}, it outputs a pair (m, s). We denote by S_0 the event that $V_{\mathsf{pk}}(m, s) = 1$. We use a similar notation S_i in any Game_i below. By definition, we have $\Pr[S_0] = \varepsilon$.

Game_1**:** In this game, we discard executions, which end up outputting a valid message/signature pair (m, s), such that m has not been queried from $\mathtt{H}$. This means restricting to the event AskH that m has been queried from $\mathtt{H}$. Unwinding the ESIGN format, we write:

$$s^e = 0 \,\|\, w \,\|\, \star \bmod n.$$

If AskH does not hold, $\mathtt{H}(m)$ is undefined, and the probability that $\mathtt{H}(m) = w$ holds is $1/2^{k-1}$: $\Pr[S_0 \mid \neg\mathsf{AskH}] \leq 2^{-k+1}$. Thus, $\Pr[S_1] = \Pr[S_0 \wedge \mathsf{AskH}] \geq \Pr[S_0] - 2^{-k+1}$.

Game_2: In this game, we choose at random an index κ between 1 and q_H. We let m_κ be the κ-th message queried to $\mathtt{H}$. We then discard executions which output a valid message/signature pair (m, s), such that $m \neq m_\kappa$. Since the additional random value κ is chosen independently of the execution of Game_1, $\Pr[S_2] = \frac{\Pr[S_1]}{q_H}$.

Game_3: In this game, we immediately abort if a signing query involves message m_κ. By the definition of existential forgery, this only eliminates executions outside S_2. Thus: $\Pr[S_3] = \Pr[S_2]$.

Game_4: We now simulate the random oracle $\mathtt{H}$, by maintaining an appropriate list, which we denote by $\mathsf{H\text{-}List}$. For any fresh query m, we pick at random $u \in \mathbb{Z}_n$ and compute $z = u^e \bmod N$, until the most significant bit of z is 0. We next parse z as $0 \,\|\, w \,\|\, \star$, where w is of length $k-1$ and ckeck whether $z - w.2^{2k}$ is $< 2^{2k-1}$. If this is true, we store (m, u, w) in $\mathsf{H\text{-}List}$ and returns w as the answer to the oracle call. Otherwise we restart the simulation of the current query. However, we stop and abort the game after k trials. This game differs from the previous one if z remains undefined after k attempts.

$$|\Pr[S_4] - \Pr[S_3]| \leq (q_H + q_s) \times (3/4)^k.$$

Game_5: We modify the simulation by replacing the $\mathtt{H}(m_\kappa)$ by v, where v is a bit string of length $k-1$, which serves as an additional input. The distribution of $\mathtt{H}$-outputs is unchanged: $\Pr[S_5] = \Pr[S_4]$.

Game_6: We finally simulate the signing oracle: for any m, whose signature is queried, we know that $m \neq m_\kappa$ cannot hold, since corresponding executions have been aborted. Thus $\mathsf{H\text{-}List}$ includes a triple (m, u, w), such that $u^e \bmod N$ has its k leading bits of the form $0\|\mathtt{H}(m)$. Accordingly, u provides a valid signature of m. Therefore, $\Pr[S_6] = \Pr[S_5]$.

Summing up the above inequalities, we obtain

$$\Pr[S_6] \geq \Pr[S_3] - (q_H + q_s) \times (3/4)^k \geq \frac{\varepsilon}{q_H} - (q_H + q_s) \times (3/4)^k - \frac{1}{2^{k-1}}.$$

When Game_6 terminates outputting a valid message/signature pair (m, s), we unwind the ESIGN format and get:

$$s^e = 0 \,\|\, w \,\|\, \star \bmod n \text{ with } w = \mathtt{H}(m).$$

If S_6 holds, we know that $m = m_\kappa$ and $\mathtt{H}(m) = v$. This leads to an element whose e-th power lies in the interval $[v2^{2k}, v2^{2k} + 2^{2k})$, thus solving an instance of AERP. We finally, have: $\Pr[S_6] \le \mathsf{Succ}^{\mathsf{aerp}}(\tau', k)$, where τ' denotes the running time of Game_6. This is the requested bound. Observe that τ' is the sum of the time for the original attack, plus the time required for simulations, which amounts to at most $k(q_s + q_H)$ modular exponentiations. We get $\tau' \le \tau + k(q_s + q_H) \cdot T_{exp}(k)$. □

We close the section by several remarks.

Remark 1. Observe that, given an element y in $\mathbb{Z}_n^\star$, we can define v as the binary expansion of $\lceil \frac{y}{2^{2k}} \rceil$. If v does not have its leading bit 0, we stop. Otherwise, we can use v as an input to game Game_6. With probability

$$\ge \frac{\varepsilon}{q_H} - (q_H + q_s) \times (3/4)^k - \frac{1}{2^{k-1}},$$

the game returns an element of the interval $[v2^{2k}, v2^{2k} + 2^{2k})$, which is a subset of $[y, y + 2^{2k+1})$. Thus, referring to section 2.3, we see that we have actually related the security of ESIGN to the variant of AERP, where the interval has length 2^{2k+1}. Relating to other variants would entail further constraints on the formatting, as observed in [23].

Remark 2. It is surprising to note that the verification algorithm accepts as valid signatures which are not produced by the signature generation algorithm. This follows from the fact that it does not ckeck that $s^e \bmod n$ lies in the interval of length 2^{2k-1} whose lower end point has binary expansion $0\|\mathtt{H}(m)\|0^{2k}$ but only that $[S]^k = 0\|\mathtt{H}(m)$, which is not equivalent. However, this does not make the proof incorrect. On the other hand, omitting the check $w_1 < 2^{2k-1}$ at signature generation would definitely make our security proof invalid. More precisely, the proof would collapse at Game_4, since there would be no way to ensure that $z - w.2^{2k}$ follows the distribution generated from message/signature pairs (m, s) by $s^e - (0\|\mathtt{H}(m).2^{2k}) \bmod n$. By lack of such simulation, it would become impossible to simulate the signing oracle in Game_6.

Remark 3. We definitely had to use the $\mathsf{SO\text{-}CMA}$ model. If the adversary was allowed to submit the same message twice to the signing oracle, the simulation would fail at the second call, since there is a single signature available.

3.3.3 Jonsson's trick

The security proof that appears in [23] replaces the k multiplicative factor in the running time by 4. This is intuitively related to the fact that, on average, it takes at most 4 steps to perform the simulation of each call to $\mathtt{H}$ in Game_4. However, we believe that the proof in [23] is not convincing, since it does not bound the corresponding error probability. The correct approach has been suggested by Jonsson [20] in connexion with PSS. It modifies the strategy for the simulation of $\mathtt{H}$, in Game_4: instead of limiting the number of trials allowed, at each execution, to find a value of z in the correct range, it sets a counter that bounds the overall number of retries, during the entire algorithm. We now compare both strategies. To remain at a general level, we denote by θ the probability that a satisfactory element z is found at each attempt. If one allows K trials altogether, the game will pass q simulations with error probability

$$\begin{aligned}
p &\leq \sum_{i=0}^{i=q-1} \binom{K}{i} \theta^i (1-\theta)^{K-i} \\
&\leq \sum_{i=0}^{i=K} \binom{K}{i} \theta^i (1-\theta)^{K-i} t^{i-(q-1)} \text{ for any } 0 < t < 1 \\
&\leq \sum_{i=0}^{i=K} \binom{K}{i} (t\theta)^i (1-\theta)^{K-i} t^{-(q-1)} \\
&\leq t^{-(q-1)} (t\theta + 1 - \theta)^K \leq \exp(-(q-1)\ln t + K \ln(1-\theta(1-t))) \\
&\leq \exp((q-1)(1/t-1) - K\theta(1-t)) \leq \exp((t-1)(K\theta - (q-1)/t)).
\end{aligned}$$

In order to bound this probability by some prescribed error bound $2^{-\ell}$, one sets

$$K \geq \frac{1}{\theta} \times \left(\frac{\ell \cdot \ln 2}{1-t} + \frac{q-1}{t} \right).$$

The right hand side has its minimum at a value of t defined by

$$\left(\frac{1-t}{t} \right)^2 = \frac{\ell \ln 2}{q-1}, \text{ and thus } t = \frac{\sqrt{q-1}}{\sqrt{\ell \ln 2} + \sqrt{q-1}}.$$

The lower estimate becomes

$$K \geq \frac{1}{\theta} \times \left(\sqrt{\ell \ln 2} + \sqrt{q-1} \right)^2 \approx \frac{q}{\theta}.$$

Provided $q_H + q_s$ is large, the above estimate duly provides the bound

$$\tau' \leq \tau + 4(q_s + q_H) \cdot T_{exp}(k).$$

4 The practical security of ESIGN

In this section, we analyze the security of the proposed ESIGN signature in view of the previous proofs. In particular, we discuss whether one can derive practical implications, notably in terms of key sizes.

4.1 Modifications of the encoding

As already mentioned, there are several specifications of ESIGN, slightly different from each other. Document [13] sums up the changes. This is certainly an unpleasant feature for a standard, but one may cope with it. However, apparently minor changes may degrade the security or, at least, make the security proof invalid. Accordingly, we review the arguments of the previous sections in the context of the version submitted to CRYPTREC [13].

4.1.1 EMSA–ESIGN

Document [13] defines an encoding method, called EMSA–ESIGN, that encodes messages into strings of length pLen, where pLen is another notation for k. However, it uses a hash function, which outputs strings of length $\mathsf{hLen} \leq \mathsf{pLen} - 16$. Encoding appends a bit string of length $\mathsf{pLen} - \mathsf{hLen}$ to the left of $\mathtt{H}(m)$. More precisely, the encoded message reads

$$\mathsf{00}\|PS\|\mathsf{FF}\|\mathtt{H}(m),$$

where PS is a padding string consisting of bytes different from FF, and where hexadecimal notation is used for the two remaining bytes.

The additional two bytes have a limited effect on the security proof. In Game_4, the simulation needs to try more random elements $u \in \mathbb{Z}_n$ to meet the appropriate redundancy. A random integer $< n$ is the output of an EMSA–ESIGN encoding with probability

$$\theta \simeq \frac{1}{2^{16}}(1 - \frac{1}{2^8})^{\frac{\mathsf{pLen}-\mathsf{hLen}-16}{8}},$$

whereas the original encoding had $\theta = 1/4$. Note that this probability is rougly $2^{-16.6}$ when $\mathsf{hLen} = 160$ and $|n| = 1024$, which is small but manageable.

However, the padding string PS has a quite negative effect on the security: in Game_5, it is only possible to input a string v of length hLen, and accordingly, the forgery does not guarantee an output in a prescribed interval of length $\simeq n^{2/3}$. Thus, the security is not in term of AERP but in terms of the following variant of AERP: given n of bit-size $3k$ and a bit string v of length hLen, find x such that the binary expansion of $x^e \bmod n$ has a window of bits which coincide with v at positions $2k+1, \cdots, 2k+\mathsf{hLen}$.

The problem is definitely easier to solve. Furthermore, when e is small enough, there is a forgery based on the attack of section 2.4.1: to sign a message m, one sets $PS = 0$ and considers the integer y whose binary expansion is $\mathsf{FF}\|\mathtt{H}(m)\|0^{2k}$. This integer has $\ell = 2k + \mathsf{hLen} + 8$ bits, and applying lemma 2, one finds an element x whose e-th power lies in the interval $[y, y + ey^{\frac{e-1}{e}})$. The length of the interval has bit-size $\log e + \frac{e-1}{e}\ell$. In case it remains $< 2k$, the binary expansion of x^e is of the form

$\mathsf{FF}\|\mathtt{H}(m)\|\star$ and x is a valid signature. the condition reads

$$\log e + (1 - \frac{1}{e})(2k + \mathsf{hLen} + 8) < 2k$$

or, equivalently,

$$2k \geq e(\mathsf{hLen} + \log e + 8).$$

When $\mathsf{hLen} = 160$, which corresponds to the mandatory use of SHA1, the forgery is successful when $|n| = 1024$ and $e = 4$. It is equally successful for $|n| = 2048$ and $e = 7$. When $|n| = 2048$ and $e = 8$, the forgery still has a significant success probability: the right-hand side of the inequality exceeds the left-hand side by less than 3 and, therefore, the forgery might overspill by 3 bits. With probability 1/8, these bits are zero. Note that our observations definitely contradict the security analysis that appears in [13], even though our attack does not endanger the value $e = 1024$, recommended for implementations.

4.1.2 SP–ESIGN

Document [13] departs from earlier version by a further modification at signature generation: referring to the desciption from section 3.3.1, we note that it omits the check $w_1 < 2^{2k-1}$ at step 3. As observed insection 3.3.2, the security proof collapses at Game_4, since there is no way to ensure that $z - w.2^{2k}$ follows the distribution generated from message/signature pairs (m, s) by $s^e - f.2^{2k} \bmod n$, where f is the integer produced from m by EMSA–ESIGN. By lack of such simulation, it becomes impossible to simulate the signing oracle in Game_6.

Besides turning down provable security, the change has highly undesirable consequences. It is easily seen that $s^e - f.2^{2k} \bmod n$ is uniformly distributed in the interval $[0, pq)$. By the law of large numbers, averaging over a large number N of signatures yields an approximation of the secret trapdoor pq with roughly its $\log \sqrt{N}$ leading bits correct. This means 10 bits from one million signatures. Even if it does not seriously endanger the scheme, this is clearly unpleasant.

4.2 Qualitative level of assurance

As observed in the previous section, the version of ESIGN appearing in the submission to CRYPTREC [13] does not benefit from the security proof of section 3.3.2. Furthermore, small values of e allow a forgery attack. Several other standards appear to use the same encoding (see [23]), and are thus at risk.

The submission to P1363a (document [23]) has provable security in the random oracle model. However, contrary to what is claimed in [23], this holds only in the weaker context of single-occurrence adaptive chosen-message attacks. The proof does appear to extend to the usual CMA scenario. Also, the security is based on AERP. It is questionable whether this problem has received enough attention from the research community to form the basis of a cryptosystem. As already noted, the early work of Vallée and *al.* has been somehow misunderstood. Also, even if no attack is known when $e \geq 4$, there is no indication either on how e should be chosen. This is actually reflected into the variety of statements made by the designers of ESIGN: the following recommendations are quoted in [23]

- $e \geq 4$ in the ATM Forum/BTD-SECURITY, and ISO 14888-3,
- $e \geq 8$ in [12],
- $e \geq 8$ in [13], with a recommendation $= 1024$.

Given the lack of information on the exact status of AERP for small values of e, it appears advisable to choose e as large as possible.

Concerning the modulus n, one may ask whether it is possible to speed up factorization of numbers of the form p^2q. There is an indication that the ECM factoring method can be made slightly more efficient for such numbers (see [25]). However, the ECM method currently disloses factors of at most 54 digits and it is highly unlikely that the ECM method can endanger the scheme in a foreseeable future.

In recent work (see [3]), a new factoring method that applies to integers of the form p^dq has been found. The method is based on an earlier result of Coppersmith (see[8]), showing that an RSA modulus $n = pq$, with p, q of the same size, can be factored given half the most significant bits of p. It turns out that, for numbers of the form $n = p^dq$, with p, q of the same size, fewer bits are needed.

Note that disclosing the leading bits of p provides a rough approximation P of p. What remains to be found is the difference $x = p - P$. The new method is based on finding polynomials with short enough integer coefficients, which vanish at x modulo some power p^{dm} of p. Such polynomials are actually zero at x. Thus, factoring is achieved by finding the appropriate root. The polynomial itself is computed by the LLL lattice reduction algorithm from [21]. LLL is run on lattices of dimension d^2 with basis vectors of size $O(d \log n)$. Let γ be the corresponding computing time. Taking into account the workfactor tied with guessing the approximation of p, the total running time is

$$2^{\frac{c+1}{d+c} \cdot \log p} . \gamma$$

where c is such that $q \simeq p^c$.

Comparing the above estimate with the known running time for ECM, one can see that the new method beats ECM for d larger than, approximately $\sqrt{\log p}$. In the case of ESIGN, where $d = 2$, the algorithm is certainly impractical.

bit-size of the modulus	complexity of NFS in $\log_2$
512	63
1024	85
4096	155
6144	182
8192	206

Figure 1: Complexity of factoring

4.3 Quantitative level of assurance

As explained in section 3.2, it is debatable whether or not the security estimates obtained in the random oracle model can be used to derive key sizes. There is a further difficulty in the case of ESIGN, since its security is based on the assumption that AERP is a difficult problem. It is unclear whether one can identify the hardness of AERP to the hardness of factoring. Finally, the reduction to AERP is not tight and the estimates that were provided in the previous section cannot be used to derive key sizes. To see this, observe that the success probability in theorem 1 has the approximate lower bound $\frac{\varepsilon}{q_H}$ and that the running time includes a term $\approx 4q_H\mathcal{O}(k^3)$. We may of course assume that the attack has been repeated $1/\varepsilon$ times to reach success probability close to one. This yields an estimate of the time $T' = \tau'/\varepsilon'$ for solving AERP of the form $\approx 4 \times q_H^2$. We now allow the adversary a number of hash queries $q_H = 2^\ell$, where ℓ is a security parameter. This allows solving AERP with complexity $4 \times 2^{2\ell}$. Taking logarithms yields $\approx 2 + 2\ell$.

We now use a table of time estimates for the best known factoring method NFS (see figure 1). We derive an estimate of the key size corresponding to a given value of the security parameter by searching for a modulus for which the righthand side is $2 + 2\ell$. As a typical case, one should select a modulus over 4000 bits to provide a security level 2^{80}. In the reverse direction, we see that the table indicates that a 1024-bit modulus provides a quite low security level 2^{42}.

We can thus conclude that the security proof only gives assurance that the overall design of ESIGN is not flawed. It does not appear possible to obtain any meaningful indication on key sizes from the estimates derived from the proof.

5 Conclusions

We have investigated the security of the ESIGN primitive and its relation to AERP. No attack against AERP is known when the exponent e is ≥ 4. Despite this fact, It is questionable whether AERP has received enough attention from the research community to form the basis of a cryptosystem.

Based on our analysis, we believe that the version of the ESIGN cryptosystem described in the P1363 submission [23] withstands existential forgery against single-occurrence adaptive chosen-message attacks, based on the hardness of AERP. We have indeed provided a security proof in Shoup's style [27], different from the proof earlier published by Okamoto, Fujisaki and Morita [23]. Both proofs use the random oracle model. However, the proof does not extend to the usual CMA scenario and, furthermore, the estimates that follow from our proof do not provide any conclusive evidence in terms of practical parameter sizes.

Due to several modifications in the specification, the version of ESIGN currently submitted to CRYPTREC and included in document [13] cannot benefit from the above security proof. Furthermore, we have found a forgery attack that is efficient for small values of the parameters. Although it does not apply to the recommended parameter $e = 1024$, the forgery is successful for several combinations of parameters claimed secure in [13]. Also, the version submitted to CRYPTREC allows to compute a few leading bits of the secret trapdoor pq, from the transcript of a large number of signatures. This is certainly undesirable. In conclusion, we regret that we cannot recommend the version of ESIGN currently submitted to CRYPTREC.

References

[1] M. Bellare and P. Rogaway. Random Oracles Are Practical: a Paradigm for Designing Efficient Protocols. In *Proc. of the 1st CCS*, pages 62–73, ACM Press, New York, 1993.

[2] M. Bellare and P. Rogaway. The Exact Security of Digital Signatures – How to Sign with RSA and Rabin. In *Eurocrypt '96*, LNCS 1070, pages 399–416, Springer-Verlag, Berlin, 1996.

[3] D. Boneh, G. Durfee, and N. Howgrave-Graham, Factoring $N = p^r q$ for large r, Crypto'99, LNCS 1666, 1999, pages 326–337, Springer-Verlag, Berlin, 1999.

[4] D. Boneh and R. Venkatesan. Breaking RSA may not be equivalent to factoring. In *Eurocrypt '98*, LNCS 1402, pages 59–71, Springer-Verlag, Berlin, 1998.

[5] E. Brickell and J. M. DeLaurentis. An Attack on a Signature Scheme proposed by Okamoto and Shiraishi. In *Crypto '85*, LNCS 218, pages 28–32, Springer-Verlag, Berlin, 1986.

[6] R. Canetti, O. Goldreich, and S. Halevi. The Random Oracles Methodology, Revisited. In *Proc. of the 30th STOC*, pages 209–218, ACM Press, New York, 1998.

[7] D. Coppersmith. Finding a Small Root of a Univariate Modular Equation. Eurocrypt '96, LNCS 1070, pages 155–165, Springer-Verlag, Berlin, 1996.

[8] D. Coppersmith. Finding a Small Root of a Bivariate Integer Equation; Factoring with High Bits Known. Eurocrypt '96, LNCS 1070, pages 155–165, Springer-Verlag, Berlin, 1996.

[9] D. Coppersmith. Small Solutions to Polynomial Equations, and Low Exponent RSA Vulnerabilities. *J. of Cryptology*, 10, 1997, 233–260.

[10] Y. Desmedt and A. M. Odlyzko. A Chosen text Attack on the RSA Cryptosystem and Some Discrete Logarithm Schemes. In *Crypto '85*, LNCS 218, pages 516–521, Springer-Verlag, Berlin, 1986.

[11] W. Diffie and M. E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT–22(6):644–654, 1976.

[12] Specification of ESIGN Signatures, 2000.

[13] NTT Corporation. ESIGN Specification, submission to CRYPTREC, September 2001.

[14] NTT Corporation. Self evaluation of ESIGN. submission to CRYPTREC, September 2001.

[15] 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, Berlin, 1987.

[16] M. Girault, P. Toffin and B. Vallée. Computation of Approximate L-th Roots Modulo n and Application to Cryptography. In *Crypto '88*, LNCS 403, pages 100-118, Springer-Verlag, Berlin, 1989.

[17] 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.

[18] IEEE Standard 1363–2000. Standard Specifications for Public Key Cryptography. IEEE. Available from `http://grouper.ieee.org/groups/1363`, August 2000.

[19] IEEE P1363a Draft Version 9. Standard Specifications for Public Key Cryptography:Additional Techniques.

[20] J. Jonsson. Security Proofs for RSA–PSS and Its Variants. Cryptology ePrint Archive 2001/053. June 2001. Available from `http://eprint.iacr.org/`.

[21] A. K. Lenstra, H. W. Lenstra and L. Lovász, Factoring polynomials with rational coefficients, *Mathematische Ann.*, 261, (1982), 513–534.

[22] T. Okamoto. A Fast Signature Scheme Based on Congruential Polynomial Operations. *IEEE Transactions on Information Theory*, IT–36 (1), pages 47–53, 1990.

[23] T. Okamoto, E. Fujisaki and H. Morita. TSH-ESIGN: Efficient Digital Signature Scheme Using Trisection Size Hash, Submission to P1363a, 1998.

[24] T. Okamoto and A. Shiraishi. A Fast Signature Scheme Based on Quadratic Inequalities. Proc. of the ACM Symp. Security and Privacy, ACM Press, pages 123–132, 1985.

[25] R. Peralta and E. Okamoto, Faster Factoring of Integers of a Special Form , IEICE Transactions on Fundamentals of Electronics, Communications, and Computer Sciences, v. E79-A, n.4 (1996), 489–493.

[26] R. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[27] V. Shoup. OAEP Reconsidered. In *Crypto '2001*, LNCS 2139, pages 239–259. Springer-Verlag, Berlin, 2001. Also appeared in the Cryptology ePrint Archive 2000/060. November 2000. Available from `http://eprint.iacr.org/`.

[28] B. Vallée, M. Girault and P. Toffin. How to break Okamoto's Cryptosystem by Reducing Lattice Bases. In *Eurocrypt '88*, LNCS 330, pages 281–292, Springer-Verlag, Berlin, 1988.

[29] B. Vallée, M. Girault and P. Toffin. How to Guess ℓth Roots Modulo n by Reducing Lattice Bases. In *AAECC-6*, LNCS 357, pages 427–442, Springer-Verlag, Berlin, 1988.