Attacking Unbalanced RSA-CRT Using SPA

Pierre-Alain Fouque, Gwenaëlle Martinet, and Guillaume Poupard

DCSSI Crypto Lab
51, Boulevard de Latour-Maubourg
75700 Paris 07 SP, France
Pierre-Alain.Fouque@ens.fr
Gwenaelle.Martinet@worldonline.fr
Guillaume.Poupard@m4x.org

Abstract. Efficient implementations of RSA on computationally limited devices, such as smartcards, often use the CRT technique in combination with Garner's algorithm in order to make the computation of modular exponentiation as fast as possible. At PKC 2001, Novak has proposed to use some information that may be obtained by simple power analysis on the execution of Garner's algorithm to recover the factorization of the RSA modulus. The drawback of this approach is that it requires chosen messages; in the context of RSA decryption it can be realistic but if we consider RSA signature, standardized padding schemes make impossible adaptive choice of message representative.
In this paper, we use the same basic idea than Novak but we focus on the use of known messages. Consequently, our attack applies to RSA signature scheme, whatever the padding may be. However, our new technique based on SPA and lattice reduction, requires a small difference, say 10 bits, between the bit lengths of modulus prime factors.

Keywords: Simple Power Analysis, RSA signature, factorization, LLL algorithm.

1 Introduction

Since the introduction in 1996 of the timing attacks by Kocher [5], many papers have considered various side channel attacks and the potential countermeasures. Side channels attacks allow to extract some information on the manipulated data which can be used to recover secret data. This general kind of attacks can be divided into several different techniques: timings attacks [5], or Simple Power Analysis (SPA) and Differential Power Analysis (DPA), both introduced in 1999 by Kocher, Jaffe and Jun [6]. Lots of countermeasures have been proposed against such attacks but addressing all weaknesses when implementing an algorithm is a hard task. Power attacks are very difficult to prevent, and thus, most of the time, countermeasures do not suffice to thwart all of them.

In the public key setting, many papers have focused on the security of cryptosystems against such side channel attacks [5,11,10]. In particular, the RSA signature and encryption schemes have been mainly studied. To sign a message

C.D. Walter et al. (Eds.): CHES 2003, LNCS 2779, pp. 254–268, 2003.

M with RSA, M is first transformed using appropriate padding scheme and hash function into a representative $m \in \mathbb{Z}_N$, where N is the product of two primes p and q. Then $m^d \bmod N$ is computed, where d is the secret key of the signer. This yields the signature for the message M.

Side channel attacks on RSA extract information from the exponentiation step. For example, by precisely measuring the time it takes for the cryptographic device to perform the RSA signature, an attacker can recover the secret key, as shown by Kocher in [5]. This timing attack can be mounted against naive implementations of RSA using the repeated square and multiply algorithm. The attack recovers the secret exponent d, one bit at a time. Indeed, for each non zero bit on the secret exponent d, an additional multiplication is performed. Analyzing differences between running time for various input values reveals the secret key. Another classical way to attack basic RSA implementation is to use SPA technique that consists in measuring the power consumption during exponentiation [3]. Since power consumption also allows to determine if the additional multiplication is done, all bits of the secret exponent can be recovered by monitoring only one exponentiation.

Optimized implementations are also subject to attacks. The Chinese Remainder Theorem is a well-known technique to optimize RSA exponentiation. In a CRT implementation, the signer first computes separately the signature modulo each prime factors p and q. He then uses the Chinese Remainder Theorem to compute the signature $S \bmod N$. Since the size of p and q is about half the size of N, CRT exponentiation is about four times faster than direct exponentiation. The first attack on RSA-CRT has been presented in 1997 by Boneh, DeMillo and Lipton [1]. It is based on fault injection during computation. By using a valid signature for a message and a faulty one, the modulus N can be efficiently factored. A timing attacks against RSA with the Chinese Remainder Theorem (CRT) is also possible [11], when the Montgomery algorithm is used for squaring and multiplication operations. Recently, Novak [10] has described an adaptive chosen message attack against smart cards implementations of RSA decryption when the Garner's algorithm implements the CRT. This attack is based on a simple power analysis (SPA). The power consumption of the card leaks information on the secret manipulated data. The cryptanalyst goal is to relate such information to the bits of the secret key. Although this attack can be mounted against RSA decryption scheme, it is not realistic in practice against the RSA signature scheme. Indeed, a padding scheme is used in practical implementations and then chosen inputs attacks cannot be made.

In this paper, we show how to extend this attack to the case of RSA signature based on any encoding scheme, such as PKCS#1 [7]. In particular we show that with a simple power analysis, if the RSA modulus $N = pq$ is such that $q < p/2^\ell$, the RSA factors p and q can be recovered by performing $60 \times 2^\ell$ signatures on average. The value ℓ should be larger than an explicit bound we precise in this paper. These signatures can be computed on any messages, not necessarily chosen by the adversary.

- Input: A message M to sign, the private key (p, q, d), with $p > q$, the pre-calculated values $d_p = d \bmod p - 1$, $d_q = d \bmod q - 1$, and $u = q^{-1} \bmod p$.
- Output: a valid signature S for the message M

1. Encode the message M in $m \in \mathbb{Z}_N$ (with PKCS#1)
2. Compute $s_p = m^{d_p} \bmod p$
3. Compute $s_q = m^{d_q} \bmod q$
4. Set $t = s_p - s_q$
5. If $t < 0$ then $t \leftarrow t + p$
6. Compute $S = s_q + ((t \cdot u) \bmod p) \cdot q$
7. Return S as a signature for the message M

Fig. 1. The RSA-CRT signature generation with Garner's algorithm

In the next section, we briefly describe the RSA-CRT signature scheme implemented with the Garner's algorithm. Then, we develop a new technique to factor N when having access to a set of special form integers modulo N. In section 4, we precisely describe the attack and how to collect such special form RSA signature. We also give practical results on experiments. Finally, we propose classical countermeasures to thwart this attack.

2 RSA Signature Scheme

Let $N = pq$ an n-bit RSA modulus. The public key of the signer is denoted by (N, e) and the private key by (p, q, d), where e and d are such that $e \cdot d = 1 \bmod (p-1)(q-1)$. Let M be a message to sign. A signature for M is $S = m^d \bmod N$, where m is deduced from M by an encoding scheme, randomized or not, such as PKCS#1 for example [7]. To check if a signature S is valid for M, a verifier simply computes m and checks if the equality $m = S^e \bmod N$ holds. Note that the encoding step is mainly used in practice to avoid some basic attacks on RSA. The requirement on the encoding scheme is that the outputs are uniformly distributed in $\mathbb{Z}_N$.

Smart cards implementations of RSA frequently use the Chinese Remainder Theorem to speed up the computation of $S = m^d \bmod N$. The Garner's algorithm is an efficient method to determine the signature S from $s_p = S \bmod p$ and $s_q = S \bmod q$. This algorithm does not require any reduction modulo N but uses instead reductions modulo the factors p and q. It is thus more efficient than the classical implementation of the CRT. A detailed description of this algorithm can be found in [9] and in [4]. In figure 1, we describe the RSA signature generation using the Garner's algorithm.

Step 5 of this algorithm needs some explanation: we first remark that the value t computed at the previous step may be negative, since, as we assume

$q < p$, it lies in the range $[-q, p]$. However, the modular multiplication $tu \bmod p$ has to be performed in the next step. Since inputs for modular multiplications have to be already reduced in the range $[0, p-1]$, if t is negative, p should be added so that $t > 0$. Therefore step 5 consists in computing $t \bmod p$ before the modular multiplication with u.

In [10], Novak has described a method to factor an RSA modulus when the Garner's algorithm is used for CRT. This attack applies to the RSA encryption schemes. It is based on the observation that, for a message m encrypted into c, if $m_p = c^{d_p} \bmod p$ is smaller than $m_q = c^{d_q} \bmod q$, step 5 of the decryption algorithm (similar to 5 of the signature generation in 1) is performed. Otherwise, no addition is made in this step. The analysis of the power trace gives the information on the execution of such a conditional step. In other words, using a SPA analysis, an adversary is able to detect whether the addition $t \leftarrow t + p$ is performed, and then to deduce if $m_p < m_q$. This information allows a binary search to recover the factor p: an attacker searches for a plaintext m such that $m \bmod p < m \bmod q$ and $(m-1) \bmod p \geq (m-1) \bmod q$. Such a plaintext can be efficiently found with a binary search combined with a simple power analysis. Once m is found, Novak has remarked that m is in fact a multiple of the factor p that can then be deduced as the GCD of m and the modulus N. However, this attack is only possible in a chosen-plaintext scenario. Thus it cannot be made in practical implementations of the RSA signature scheme due to the encoding step. Indeed, an adversary is still supposed to choose the message M to sign but does not have enough control over the encoding m of M, particularly when randomization techniques are used.

In the following we show how to recover the factor q using this leaked information even if a padding scheme is used. However, we require the prime factors of the modulus to be slightly unbalanced.

3 Lattice Based Techniques

3.1 Preliminaries on Lattices

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, $\det(L)$ 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 [8] 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 Factoring Using LLL

In the following we describe a new method, based on lattice reduction, to factor a modulus N given some special form integers s_i in $\mathbb{Z}_N$.

Let $N = p \times q$ be an RSA modulus such that p and q are two prime integers. Let $s_1, s_2, \ldots, s_d$ be d integers from $\mathbb{Z}_N$. For each s_i, we consider its euclidian division by p. $\forall i \in [1, d]$, we can write $s_i = r_i + u_i \times p$ with $r_i \in \mathbb{Z}_p$ and $u_i \in \mathbb{Z}_q$. Let us assume that, instead of being distributed all over the set $\mathbb{Z}_p$, the r_is values are smaller than a bound $A < p/2$. We further consider the lattice L spanned by the $d + 1$ rows of the following matrix:

$$\begin{pmatrix} N & 0 & \ldots & \ldots & 0 \\ 0 & N & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ldots & 0 & N & 0 \\ -s_1 & -s_2 & \ldots & -s_d & A \end{pmatrix}$$

Theorem 1. *Assuming the LLL algorithm returns the shortest vector of a lattice, the reduction of lattice L computes the factorization of modulus N with probability $> 1 - \varepsilon_0$ if the bit-length difference between p and A is such that*

$$\log p - \log A > \max\left(\frac{\log q - \log \varepsilon_0 - 0.105}{d} + 2.047, \log\left(2\sqrt{d+1}\right)\right)$$

As an example, for a 512-bit prime factor q and a probability of success of the algorithm $> 1 - 2^{-10}$, we obtain a minimum $\log p - \log A \approx 10.4$ for $d = 60$. Note that this minimum does not strongly depend on the size of q and is about 5 bits for any cryptographic size of this factor.

Sketch of proof. [A complete proof is proposed in appendix A]
By definition of the lattice L, a vector of L is an integer combination of the rows of the matrix. In other words, we may define the lattice in the following way :

$$L = \left\{(c_1 N - c s_1, c_2 N - c s_2, \ldots, c_d N - c s_d, Ac) \; ; \; (c_1, c_2, \ldots, c_d, c) \in \mathbb{Z}^{d+1}\right\}$$

For a fixed choice of the integer coefficients $(c_1, c_2, \ldots, c_d, c)$, we note

$$b(c_1, c_2, \ldots, c_d, c) = (c_1 N - c s_1, c_2 N - c s_2, \ldots, c_d N - c s_d, Ac) \in L$$

In other words, the lattice L is the set of all the vectors $b(c_1, c_2, \ldots, c_d, c)$ for $(c_1, c_2, \ldots, c_d, c) \in \mathbb{Z}^{d+1}$.

A special vector of the lattice, strongly related with the q prime factor of N, is "abnormally" short. The consequence is that we can expect the LLL lattice reduction algorithm to compute this short vector.

This special vector is $b^* = b(u_1, u_2, \ldots, u_d, q)$, where u_i is defined by division of the s_i by p, and q is the other factor of the modulus $N = p \times q$. Note that the knowledge of b^* immediately reveals q since its last coordinate is Aq and A is known. The size of b^*, i.e. its euclidian norm, can be easily estimated and we obtain $||b^*|| < \sqrt{d+1}Aq$.

Then, in order to prove that the vector b^* is the shortest one, we study the Euclidean norm of the vectors $b(c_1, c_2, \ldots, c_n, c)$, and we prove that, if $c \neq q$, those vectors are larger that b^*, whatever the c_is may be, for all the s_i but only a very small fraction. A precise analysis, described in appendix A, leads to the result of theorem 1.

□

Therefore the knowledge of d values s in $\mathbb{Z}_N$ such that $|s \bmod p| < A$ allows us to factor N. In the following, we describe how SPA can be used to find such d values in the context of RSA signature generation, with "slightly" unbalanced modulus.

4 Application to RSA-CRT Signature Scheme

In this section, we use the results presented above to extend the chosen ciphertext attack described by Novak in [10]. In particular, we show that if the factors p and q of the modulus are such that $|p| - |q| > \ell$, for a given bound ℓ, then they can be recovered with a known message attack combined with a simple power analysis. In the following we suppose that a SPA attack allows us to detect if the addition of step 5 is performed during a signature generation. Such an assumption is realistic in practice if no countermeasure is implemented, and a detailed way to extract this information can be found in [10] and in [5].

Attack. In the previous section we have shown how the prime factors p and q of a modulus N can be recovered, given a set of integers s in $\mathbb{Z}_N$ such that $s \bmod p$ is less than a given bound. We apply this result by using a simple power analysis on the RSA signature scheme in order to find these integers.

We assume that the prime factors p and q are such that $|p| - |q| > \ell$, for ℓ a small integer. Such an assumption is realistic in many actual implementations since, in many descriptions of the RSA algorithm, we can find that p and q have to be of "roughly" the same length, or "about the same bit-length". Here, we consider that p and q have a very small bit-length difference, about 10 bits for an 1024-bit modulus. This does not constitute a contradiction with the usual description of RSA, that can be interpreted in many different ways.

Let S be a signature for a random message M, computed by using the algorithm described in figure 1. We suppose that the step 5 of the algorithm has been

performed to generate the signature S. Otherwise, we choose another random message until this step is executed. Since the optional addition has been made, we know that $s_p - s_q < 0$. Thus, since we assume that $q < p$, we simply have that $s_p < s_q < q < p$. By definition of $s_p = S \bmod p$, S can always be written as $S = s_p + u \times p$ for an integer $u < q$. Consequently S is a candidate input to the factoring algorithm, given in section 3.2, for the upperbound $A = q$ on $s \bmod p$. The problem here is that clearly, this bound is not known to the attacker. Thus, the last entry A of the matrix cannot be explicitely given. However, A is an upperbound on the $s_i \bmod p$ where s_i is the ith input of the last row. Thus choosing an integer $A > q$ is a correct choice and we choose in practice A to be the largest integer such that $|A| = |q|$.

To run the lattice reduction described in the previous section, we have to find d signatures s_i such that $s_i \bmod p$ is less than the bound A we choose. We thus query the signature of messages and we perform a SPA attack on each generation. Each signature verifying $s_p < s_q$ is kept as input to the matrix. We query the signing card until d valid candidate signatures have been found.

The number d of required signatures has to be sufficiently large, according to the bit-length difference ℓ between the factors p and q, and according to the modulus length. To estimate the average number of queries made to the signing card, we compute the probability that $s_p = m^{d_p} \bmod p$ is less than $s_q = m^{d_q} \bmod q$, for $q < p/2^\ell$, and for random integers $s \in \mathbb{Z}_N$. We suppose that the values s are uniformly and independently distributed in $\mathbb{Z}_N$. This is verified in practice when an appropriate hash function, such as SHA-1, is used. In this case we assume that the output m of the encoding scheme is uniformly distributed in $\mathbb{Z}_N$ so that $s = m^d \bmod N$ is uniformly distributed. Let s_p and s_q the values computed during the signature generation in steps 2 and 3 respectively. We have:

$$\begin{aligned}\Pr\{s_p < s_q\} &= \sum_{B=0}^{q-1} \Pr\{s_p < B | s_q = B\} \cdot \Pr\{s_q = B\} \\ &= \sum_{B=0}^{q-1} \frac{B}{p} \cdot \frac{1}{q} = \frac{q(q-1)}{2pq} < \frac{1}{2^{\ell+1}}\end{aligned}$$

where the last inequality comes from the fact that $q < p/2^\ell$.

Thus in a set of 2^ℓ signatures, there is a probability of one half that at least one of them is such that $s_p < s_q$. Detecting such a signature is possible with a SPA attack during the signature generation: when the step 5 is performed, we know that the signature s_i is a good candidate input for our algorithm, if we write it as $s_i = (s_i \bmod p) + u_i p$ where $s_i \bmod p < p/2^\ell$. Otherwise, we query another signature until we find a good candidate. On average, 2^ℓ trials are needed. To have d such signatures, a set of $d \cdot 2^\ell$ signatures is required. The algorithm described in section 3.2 to factor a modulus N can then be used with the candidate signatures as inputs. Following the analysis made above, the number d of required signatures must be such that $\ell \geq \frac{\log q - 10}{d} + 2$ so that the algorithm successfully ends with probability greater than $1 - 2^{-10}$. However, we

modulus length in bits	$\lvert p\rvert - \lvert q\rvert$	lattice dimension $d+1$	average number of required signatures	time to factor
512	8	41	2^{13}	30 s
	7	51	2^{13}	2 min
	6	61	2^{12}	14 min
768	10	43	2^{15}	50 min
	9	51	2^{15}	2 min
	8	61	2^{14}	15 min
1024	12	46	2^{17}	2 min
	11	56	2^{17}	6 min
	10	61	2^{16}	16 min
1536	16	53	2^{22}	7 min
	15	56	2^{21}	10 min
	14	61	2^{20}	32 min
2048	20	53	2^{26}	11 min
	19	56	2^{25}	20 min
	18	61	2^{24}	32 min

Fig. 2. Experimental results on the RSA-CRT with unbalanced modulus.

assume here that $q < p/2^{\ell}$, and thus, $\log q = \frac{\log N - \ell}{2}$. Thus, after some simple computations we obtain:

$$\ell \geq \frac{\log N - 20 + 4d}{2d+1}$$

Thus, if p and q are such that $q < p/2^{\ell}$ for ℓ greater than $\frac{\log N - 20 + 4d}{2d+1}$, we can factor N from $d \times 2^{\ell}$ signatures on average and with probability at least $1 - 2^{-10}$.

Experimental results. In practice, the lattice dimension depends on the value ℓ, and on the number of available signatures. However, if the lattice dimension is too large, then the LLL algorithm fails. Particularly, under a reasonable time, it is not possible to run the LLL algorithm on a 100×100 dimensional matrix where the entries are 1024-bit numbers.

For each modulus length, we give in figure 2 the integer ℓ such that $q < p/2^{\ell}$, the dimension $d+1$ of the lattice, the average number of required signatures and the time needed to recover p and q. The number of required signatures, equal to $d \times 2^{|p|-|q|}$, is upperbounded by $2^{6+|p|-|q|}$ since we always have $d < 2^6$. The tests have been run on an Intel Pentium IV, XEON 1.5 GHz, with the Victor Shoup's library NTL ([12]).

From this previous table, we show that the LLL algorithm works better than the theoretical results indicated in section 3.2. For 1024-bit modulus, the expected result gives $\ell > 10$. These values are realistic since a difference of 10 bits between p and q for a 1024-bit modulus means that p is a 517-bit prime and q is a 507-bit prime number. This distance between p and q is not ruled out by the specifications of the key generation of the RSA algorithm. It is worth

noticing that this attack can be extended to more "secure" moduli, say 2048 bits long. Moreover, this attack can be mounted in practice since the number of required signatures for known messages is not large, one million if $n = 1536$.

5 Countermeasures

Such an attack proves that an implementation of the Garner's algorithm should be carefully checked so that SPA or other side channels attacks should not be possible. We now propose some countermeasures to avoid this attack.

In order to balance time execution and power consumption, dummy operations can be added. This can be done by modifying the step 4 of the algorithm as follows: first, t is computed as $s_p - s_q$. Then, new variables t' and t'' are respectively set as $t+p$ and t. The step 5 is then defined as: `5. If` $t < 0$ `then` $t \leftarrow t'$`, else,` $t \leftarrow t''$. In this case, the implementation does not leak any information about the difference $s_p - s_q$ since the addition is always performed. The crucial remark here is that this implementation should use a probabilistic encoding step (Step 1 of figure 1). If not, another attack is possible: suppose for example that PKCS#1 v1.5 is used. Thus the encoding of a message is always the same. In this case, this countermeasure can be broken by using safe errors attacks [13]. Such attacks use fault injection at particular computational step to produce an error during operations, possibly unused depending on some secret data. Here a fault can be performed during the computation of t'. If the resulting signature is not valid (the card outputs a failed error), we learn that for this signature, t' has been used, that is $t < 0$. In this case, we ask again the card with the same message without producing any error during the generation. We know that the resulting signature is such that $t < 0$, since it has been computed on the same input m. We can then use this signature as input for our algorithm.

Another classical countermeasure [2] is based on the randomization of m: a signature s is computed on $r^e \times m$. The signature for m is given as $s/r \bmod N$. In this case, since r is kept secret, there is no relationship between the information leaked by the card on the value t and the output signature $s/r \bmod N$. Thus, in this case, our attack is no longer feasible. Note that it is also possible to fully randomize all the parameters of the signature generation, as done by many actual implementations of RSA signatures: the factors p and q are randomized as $p' = r_1 \times p$ and $q' = r_2 \times q$ and the signature s is deduced from $s \bmod p'$ and $s \bmod q'$ respectively computed as $((m \bmod p) + r'_1 \times p)^{d_p + r''_1 \times (p-1)}$ and $((m \bmod q) + r'_2 \times q)^{d_q + r''_2 \times (q-1)}$. The signature is finally given by $s \bmod N$. Due to a full randomization of each step of the signature generation, such an implementation is not subject to our attack.

Acknowledgments. We wish to thank anonymous referees for their constructive remarks and suggestions of countermeasures.

References

1. D. Boneh, R. A. DeMillo, and R. J. Lipton. On the Importance of Checking Cryptographic Protocols for Faults. In W. Fumy, editor, *Advances in Cryptology – Eurocrypt'97*, volume 1233 of *LNCS*, pages 37–51. Springer-Verlag, Berlin, 1997.
2. D. Chaum. Blind signatures for untraceable payments. In D. Chaum, R.L. Rivest, and A.T. Sherman, editors, *Advances in Cryptology – Crypto'82*, pages 199–204. Plenum Publishing, 1982.
3. B. den Boer, K. Lemke, and G. Wicke. A DPA Attack Against the Modular Reduction within a CRT Implemenataion of RSA. In B. S. Kaliski Jr, Ç. K. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2002*, volume 2523 of *LNCS*, pages 228–243. Springer-Verlag, 2002.
4. D. E. Knuth. *The Art of Computer Programming, Vol 2: Semi Numerical Algorithms*. Addison Wesley, 1969.
5. P. C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Others Systems. In N. Koblitz, editor, *Advances in Cryptology – Crypto '96*, volume 1109 of *LNCS*, pages 104–113. Springer-Verlag, 1996.
6. P. C. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In M. Wiener, editor, *Advances in Cryptology – Crypto '99*, volume 1666 of *LNCS*, pages 388–397. Springer-Verlag, 1999.
7. RSA Laboratories. PKCS #1 v2.1 : RSA Cryptography Standard, 2002. Available at `http://www.rsalabs.com/pkcs/pkcs-1`.
8. A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring Polynomials with Rational Coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.
9. A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.
10. R. Novak. SPA-Based Adaptive Chosen-Ciphertext Attack on RSA Implementation. In D. Naccache and P. Paillier, editors, *PKC 2002*, volume 2274 of *LNCS*, pages 252–261. Springer-Verlag, 2002.
11. W. Schindler. A Timing Attack against RSA with the Chinese Remainder Theorem. In Ç. K. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems 2000*, volume 1965 of *LNCS*, pages 109–124. Springer-Verlag, 2000.
12. V. Shoup. Number Theory C++ Library (NTL), version 5.0b. Available at `http://www.shoup.net`.
13. S.M. Yen and M. Joye. Checking before output may not be enough against fault-based cryptanalysis. *IEEE Trans. on Computers*, 49:967–970, 2000.

A Proof of Theorem 1

By definition of the lattice L, a vector of L is an integer combination of the rows of the matrix. In other words, we may define the lattice in the following way:

$$L = \left\{(c_1 N - cs_1, c_2 N - cs_2, \ldots, c_d N - cs_d, Ac)\ ;\ (c_1, c_2, \ldots, c_d, c) \in \mathbb{Z}^{d+1}\right\}$$

For a fixed choice of the integer coefficients $(c_1, c_2, \ldots, c_d, c)$, we note

$$b(c_1, c_2, \ldots, c_d, c) = (c_1 N - cs_1, c_2 N - cs_2, \ldots, c_d N - cs_d, Ac) \in L$$

In other words, the lattice L is the set of all the vectors $b(c_1, c_2, \ldots, c_d, c)$ for $(c_1, c_2, \ldots, c_d, c) \in \mathbb{Z}^{d+1}$.

We now show that a special vector of the lattice, strongly related with the q prime factor of N, is "abnormally" short. The consequence is that we can expect the LLL lattice reduction algorithm to compute this short vector.

This special vector is $b^* = b(u_1, u_2, \dots, u_d, q)$, where u_i is defined by division of the s_i by p, and q is the other factor of the modulus $N = p \times q$. Note that the knowledge of b^* immediately reveals q since its last coordinate is Aq and A is known. Let us evaluate the size of b^*, i.e. its euclidian norm

$$\begin{aligned} ||b^*||^2 &= ||b(u_1, u_2, \dots, u_d, q)||^2 \\ &= ||(u_1 N - qs_1, u_2 N - qs_2, \dots, u_d N - qs_d, Aq)||^2 \\ &= \sum_{i=1}^{d} (u_i N - q(r_i + u_i p))^2 + A^2 q^2 \\ &= \sum_{i=1}^{d} q^2 r_i^2 + A^2 q^2 \end{aligned}$$

Since $0 \leq r_i < A$, we immediately obtain that $||b^*||^2 < dq^2 A^2 + A^2 q^2$, and so $||b^*|| < \sqrt{d+1}Aq$.

In order to prove that the vector b^* is abnormally short, we now show that the other elements of the lattice L are, with overwhelming probability, larger. With this aim in view, we define, for any integer c, the function

$$\mathcal{F}(c) = \min_{(c_1, c_2, \dots, c_d) \in \mathbb{Z}^d} ||b(c_1, c_2, \dots, c_d, c)||$$

i.e., $\mathcal{F}(c)$ is the size of the shortest vector in L whose last coordinate is equal to Ac. From the definition of $\mathcal{F}(c)$, we can derive the following expression

$$\mathcal{F}(c) = \min_{(c_1, c_2, \dots, c_d) \in \mathbb{Z}^d} \sqrt{\sum_{i=1}^{d} (c_i N - cs_i)^2 + A^2 c^2}$$

Then, it is easy, for a fixed c, to find the c_is that reach the minimum since $\mathcal{F}(c)$ is the minimum of a sum of independent squares. As a consequence, the minimum is reached when each term is as small as possible. This means that c_i is the nearest integer of cs_i/N; we note $\left\lfloor \frac{cs_i}{N} \right\rceil$ this integer. We finally obtain

$$\mathcal{F}(c) = \sqrt{\sum_{i=1}^{d} \left(\left\lfloor \frac{cs_i}{N} \right\rceil N - cs_i \right)^2 + A^2 c^2}$$

We can now notice that, by definition of $\mathcal{F}(c)$ and $b^* = b(u_1, u_2, \dots, u_d, q)$, we have $\mathcal{F}(q) \leq ||b^*|| < \sqrt{d+1}Aq$. In fact, b^* is exactly the smallest vector with last coordinate equal to Aq because

$$\begin{aligned}\mathcal{F}(q)^2 &= \sum_{i=1}^{d} \left(\left\lfloor \frac{qs_i}{N} \right\rceil N - qs_i \right)^2 + A^2q^2 \\ &= q^2 \times \left(\sum_{i=1}^{d} \left(\left\lfloor \frac{r_i + u_i \times p}{p} \right\rceil p - r_i - u_i \times p \right)^2 + A^2 \right) \\ &= q^2 \times \left(\sum_{i=1}^{d} \left(\left\lfloor \frac{r_i}{p} \right\rceil p - r_i \right)^2 + A^2 \right) = q^2 \times \left(\sum_{i=1}^{d} r_i^2 + A^2 \right)\end{aligned}$$

since, for $0 \leq r_i < A < \frac{p}{2}$, $\left\lfloor \frac{r_i}{p} \right\rceil = 0$. This finally proves that $\mathcal{F}(q) = ||b^*||$.

Then, we can further notice that if $c > \sqrt{d+1} \times q$, $\mathcal{F}(c)$ is obviously greater than Ac so $\mathcal{F}(c) > \sqrt{d+1} \times q \times A > \mathcal{F}(q)$. We finally need to evaluate, for a fixed $c \neq q$, the probability that $\mathcal{F}(c) < \mathcal{F}(q)$ where the probabilities are computed when the s_i are uniformally distributed in

$$\mathcal{S} = \{r + u \times p \,;\, 0 \leq r < A \,,\, 0 \leq u < q\} \subset \mathbb{Z}_N$$

If this probability is negligible, we can conclude that b^* is, with overwhelming probability, the shortest vector of the lattice.

We first notice that the distribution of the $\left\lfloor \frac{c \times s_i}{N} \right\rceil \times N - c \times s_i$, when s_i is uniformally distributed in $\mathbb{Z}_N$, is uniform between $\frac{-(N-1)}{2}$ and $\frac{N-1}{2}$. This is an obvious consequence of the fact that $\left\lfloor \frac{c \times s_i}{N} \right\rceil - \frac{c \times s_i}{N} \in [-1/2; 1/2]$ and that if $\left\lfloor \frac{c \times s_i}{N} \right\rceil \times N - c \times s_i = \alpha$ for an integer α, we obtain by modular reduction that $-c \times s_i = \alpha \bmod N$ and thus that $s_i = -\alpha \times c^{-1} \bmod N$ if c is prime with N.

If we restrict the possible values of s_i to the set $\mathcal{S}$, the previous result implies

$$\mathcal{D}_c = \left\{ \left\lfloor \frac{cs_i}{N} \right\rceil \times N - cs_i \,;\, s_i = r_i + u_i \times p \in \mathcal{S} \right\} \subset \left[\frac{-(N-1)}{2}, \frac{N-1}{2} \right]$$

We can further write

$$\begin{aligned}\mathcal{D}_c &= \left\{ \left\lfloor \frac{cs_i}{N} \right\rceil \times N - cs_i \,;\, s_i = r_i + u_i p \,,\, 0 \leq r_i < A \,,\, 0 \leq u_i < q \right\} \\ &= \left\{ \left\lfloor \frac{cr_i}{N} + \frac{cu_i}{q} \right\rceil \times N - cr_i - cu_i p \,;\, 0 \leq r_i < A \,,\, 0 \leq u_i < q \right\} \\ &= \left\{ \left\lfloor \frac{cr_i}{N} + \frac{cu_i \bmod q}{q} \right\rceil \times N - cr_i - (cu_i \bmod q) \times p;\; 0 \leq r_i < A,\; u_i \in \mathbb{Z}_q \right\} \\ &= \left\{ \left\lfloor \frac{cr_i}{N} + \frac{v_i}{q} \right\rceil \times N - cr_i - v_i p \,;\, 0 \leq r_i < A \,,\, 0 \leq v_i < q \right\}\end{aligned}$$

Then, for any $\alpha \in [-(N-1)/2, (N-1)/2]$, if $\left\lfloor \frac{c \times r_i}{N} + \frac{v_i}{q} \right\rceil \times N - c \times r_i - v_i \times p = \alpha$ we obtain that $\alpha = -cr_i - v_i p \bmod N$. Since v_i is uniformally distributed in $\mathbb{Z}_q$ and $\alpha + p = -cr_i - (v_i - 1 \bmod q) \times p \bmod N$, we conclude that, if α is an element of $\mathcal{D}_c$, $\alpha + p$ (or $\alpha + p - N$ if $\alpha + p > N/2$) is also an element of $\mathcal{D}_c$. So, if $\gcd(c, N) = 1$, the set $\mathcal{D}_c$ is a subset of $\left[\frac{-(N-1)}{2}, \frac{N-1}{2} \right]$ that is invariant by (circular) translation of length p.

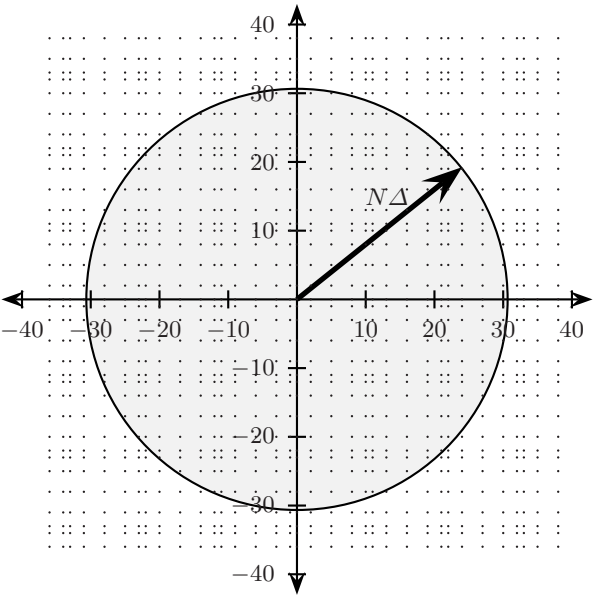

Fig. 3. Graphical representation of the pairs (s_1, s_2) for the parameters $d = 2$, $p = 11$, $q = 7$, $N = 77$ and $A = 5$. The disk of radius $N\Delta$ covers a ratio of those points that is illustrated in figure 4.

In other words, this formalizes the idea that, if c is prime with N, the elements of $\mathcal{D}_c$ are well distributed in $\left[\frac{-(N-1)}{2}, \frac{N-1}{2}\right]$. As a toy example, figure 3 represents the elements of $\mathcal{D}_3 \times \mathcal{D}_3$ for $N = 7 \times 11$ and $A = 5$.

Always with the aim of computing the probability for $\mathcal{F}(c)$ to be less that $\mathcal{F}(q)$, if $\gcd(c, N) = 1$, we can state that

$$\Pr_{(s_1,s_2,\dots,s_d)\in\mathcal{S}^d}\{\mathcal{F}(c) < \mathcal{F}(q)\} < \Pr_{(s_1,s_2,\dots,s_d)\in\mathcal{S}^d}\left\{\mathcal{F}(c) < \sqrt{d+1}Aq\right\}$$

$$< \Pr_{(s_1,s_2,\dots,s_d)\in\mathcal{S}^d}\left\{\sum_{i=1}^{d}\left(\left\lfloor\frac{cs_i}{N}\right\rceil N - cs_i\right)^2 < (d+1)A^2q^2 - A^2c^2\right\}$$

$$< \Pr_{(s_1,s_2,\dots,s_d)\in\mathcal{S}^d}\left\{\sum_{i=1}^{d}\left(\left\lfloor\frac{cs_i}{N}\right\rceil N - cs_i\right)^2 < (d+1)A^2q^2\right\}$$

Let $\Delta = \sqrt{d+1}A/p$; we need to evaluate the probability

$$\Pr_{(s_1,s_2,\dots,s_d)\in\mathcal{S}^d}\left\{\sum_{i=1}^{d}\left(\left\lfloor\frac{cs_i}{N}\right\rceil N - cs_i\right)^2 < N^2\Delta^2\right\}$$

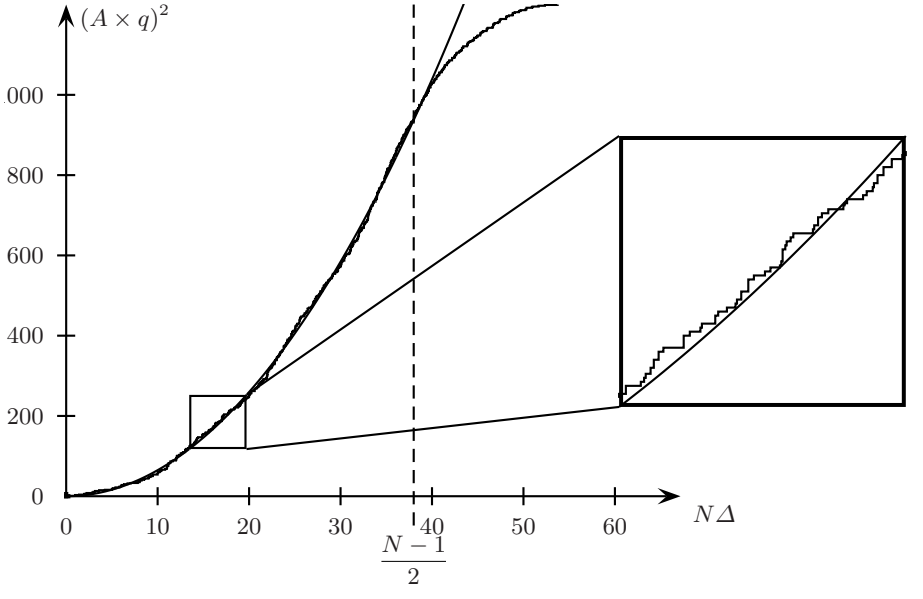

Fig. 4. Ration of points in figure 3 covered by a disk of radius $N\Delta$. The irregular experimental curve represents the number of points covered by the disk and the smooth curve is based on the approximation by the surface of this disk.

From the previous result on the distribution of $X_i = \left\lfloor \frac{cs_i}{N} \right\rceil - \frac{cs_i}{N}$ in $\left[-\frac{1}{2}, \frac{1}{2}\right]$, this probability can be approximated, for large N, by

$$\Pr_{(X_1, X_2, \ldots, X_d) \in_u \left[-\frac{1}{2}, \frac{1}{2}\right]} \left\{ \sum_{i=1}^{d} X_i^2 < \Delta^2 \right\}$$

where the X_is are independent and uniformally distributed over $\left[-\frac{1}{2}, \frac{1}{2}\right]$. If $0 \leq \Delta < \frac{1}{2}$, this probability is equal to the volume of the d-dimensional ball of radius Δ. Figure 4 provides a graphical illustration of this fact, using the toy example of figure 3. If d is even, this volume is equal to $\pi^{d/2}\Delta^d/(d/2)!$.

Note that this approximation of the number of points in the ball of radius $N\Delta$ (see figure 3) using the volume of this ball is very good for values of c that are relatively prime with q, even if the repartition of the points is not perfectly uniform. This is mainly due to the compensation of local errors of estimation. When $c = q$, such a compensation does not apply and the approximation can no longer be used. Then, using the well-known Stirling formula, we obtain the upper-bound

$$\Pr_{(s_1, s_2, \ldots, s_d) \in \mathcal{S}^d} \left\{ \sum_{i=1}^{d} \left(\left\lfloor \frac{cs_i}{N} \right\rceil N - cs_i \right)^2 < N^2 \Delta^2 \right\} < \left(\frac{2e\pi\Delta^2}{d} \right)^{\frac{d}{2}} \times \frac{1}{\sqrt{d\pi}}$$

We finally obtain

$$\Pr_{(s_1,s_2,\dots,s_d)\in\mathcal{S}^d}\{\mathcal{F}(c)<\mathcal{F}(q)\}<\left(\frac{2e\pi\Delta^2}{d}\right)^{\frac{d}{2}}\times\frac{1}{\sqrt{d\pi}}$$

Note that this is true only if $\Delta<\frac{1}{2}$, i.e., if $A<\frac{p}{2\sqrt{d+1}}$. In other words, in order to make the proof correct, this means that the difference $-\log(A/p)$ of bit-length of p and A must fulfill the inequality $-\log\left(\frac{A}{p}\right)>\log\left(2\sqrt{d+1}\right)$

Using the fact that $\left(\frac{d+1}{d}\right)^d<e$, we finally obtain an upper bound of the probability for $\mathcal{F}(c)$ to be smaller than $\mathcal{F}(q)$

$$\begin{aligned}\Pr_{(s_1,s_2,\dots,s_d)\in\mathcal{S}^d}\{\mathcal{F}(c)<\mathcal{F}(q)\}&<\left(\frac{2e\pi A^2}{p^2}\times\frac{d+1}{d}\right)^{\frac{d}{2}}\times\frac{1}{\sqrt{d\pi}}\\&<\sqrt{\frac{e}{d\pi}}\left(\sqrt{2\pi e}\right)^d\left(\frac{A}{p}\right)^d\end{aligned}$$

The last step is to estimate for which values of the parameters we can consider that $\mathcal{F}(c)>\mathcal{F}(q)=||b^*||$ for any $c\neq q$. Let $P_0=1-\varepsilon_0$ be a lower bound of the probability for b^* to be the smallest vector of the lattice. From the previous results, using an approximative argument of independence of the probabilities for different values of c, we deduce

$$\Pr_{(s_1,s_2,\dots,s_d)\in\mathcal{S}^d}\{\forall c\neq q\ \mathcal{F}(c)>\mathcal{F}(q)\}>\left(1-\sqrt{\frac{e}{d\pi}}\left(\sqrt{2\pi e}\right)^d\left(\frac{A}{p}\right)^d\right)^{\sqrt{d+1}\times q}$$

If $\varepsilon_0>\left(\sqrt{d+1}\times q\right)\times\sqrt{\frac{e}{d\pi}}\left(\sqrt{2\pi e}\right)^d\left(\frac{A}{p}\right)^d$, the last expression is greater than P_0. Thus, the inequality can be reworded as follows

$$\begin{aligned}-\log\left(\frac{A}{p}\right)&>\frac{1}{d}\left(\frac{1}{2}\log\left(\frac{d+1}{d}\right)-\log\varepsilon_0+\log q+\frac{1}{2}\log\left(\frac{e}{\pi}\right)+d\log\left(\sqrt{2\pi e}\right)\right)\\&>\frac{\log q-\log\varepsilon_0+\frac{1}{2}\log\left(\frac{e}{\pi}\right)}{d}+\log\left(\sqrt{2\pi e}\right)\end{aligned}$$

In other words, this means that, if the difference $-\log(A/p)$ of bit-length of p and A is larger than $\frac{\log q-\log\varepsilon_0-0.105}{d}+2.047$, the algorithm finds q with probability $>1-\varepsilon_0$, assuming that LLL returns the shortest vector of the lattice.

□