Simple Power Analysis on Exponentiation Revisited

Jean-Christophe Courrège[1], Benoit Feix[2], and Mylène Roussellet[2]

[1] CEACI-THALES
18 Avenue Edouard BELIN
31401 Toulouse, France
Jean-Christophe.courrege@thalesgroup.fr
[2] INSIDE CONTACTLESS
41 Parc Club du Golf
13856 Aix-en-Provence, Cedex 3, France
{bfeix, mroussellet}@insidefr.com

Abstract. Power Analysis has been studied since 1998 when P. Kocher *et al.* presented the first attack. From the initial Simple Power Analysis more complex techniques have been designed and studied during the previous decade such as Differential and Correlation Power Analysis. In this paper we revisit Simple Power Analysis which is at the heart of side channel techniques. We aim at showing its true efficiency when studied rigorously. Based on existing Chosen Message attacks we explain in this paper how particular message values can reveal the secret exponent manipulated during a modular exponentiation with a single power consumption curve. We detail the different ways to achieve this and then show that some blinded exponentiations can still be threatened by Simple Power Analysis depending on the implementation. Finally we will give advice on countermeasures to prevent such enhanced Simple Power Analysis techniques.

Keywords: Public key cryptography, long integer arithmetic, modular exponentiation, power analysis.

1 Introduction

The appearance of public key cryptography [DH76] and of the RSA cryptosystem [RSA78] was the beginning of modern cryptography. The use of these schemes, and especially RSA, has become very popular and more and more systems have based their security on it. Thus, in order to implement these systems efficiently, various modular multiplication algorithms have been designed to be embedded in constrained hardware resources devices such as Trusted Platform Modules (TPM) and smart cards.

Another consideration has become a key point for developers is the tamper resistance topic. For years smart cards had been considered as tamper resistant devices until Kocher *et al.* introduced in 1996 the Timing Attacks [Koc96] and

few years later the Power Analysis Attacks [KJJ99]. Their techniques, named Simple Power Analysis (SPA) and Differential Power Analysis (DPA), threaten any naive cryptographic algorithm implementation. As electronic devices are composed of thousands of logical gates that switch differently depending on the executed operations, the power consumption depends on the executed instructions and the manipulated data. Thus by analyzing the power consumption of the device on an oscilloscope it is possible to observe its behavior and then to deduce from this power curve the secret data manipulated.

From the initial SPA and DPA of Kocher *et al.* many studies were presented to introduce new attack techniques on different popular cryptographic schemes and to improve the power curve processing in order to recover secrets with fewer curves than the classical DPA. Others presented some countermeasures to these attacks. In this paper we focus on SPA and show it is more powerful than what can be inferred from reading current side channel papers. To illustrate our paper and assertions some practical results are presented on some secure implementations.

The paper is organized as follows. First we recall in sections 2 and 3 the fundamentals notations and techniques on which our work is based. Section 2 gives an overview of long integer arithmetic for public key embedded implementations. Section 3 describes the Side Channel Analysis techniques related to this paper. We present in Section 4 some chosen message SPA techniques and explain the reasons of observed power leakages. We also explain how these power leakages can be exploited to mount enhanced SPA on non-chosen or blinded message exponentiations. In Section 5 we analyze the efficiency of the classical countermeasures and give some advice on their use for preventing enhanced SPA. We conclude our research in Section 7.

2 Embedded Implementations of Exponentiation

We recall here the mathematical principles and the arithmetic algorithms that are used to implement public key algorithms in embedded devices.

2.1 Long Integer Multiplication

In this paper we use the following notation: $x = (x_{k-1} \dots x_1 x_0)_b$ corresponds to integer x decomposition in base b, i.e. the x decomposition in t-bit words with $b = 2^t$ and $k = \lceil log_b(x) \rceil$.

Algorithm 2.1 presents to the classical long integer multiplication algorithm used to compute $x \times y$.

2.2 Long Integer Modular Multiplication

Chip manufacturers usually embed arithmetic coprocessors to compute modular multiplications $x \times y \mod n$ for long integers x, y and n. In this paper we choose to illustrate our analysis on the Barrett and the Montgomery [Mon85]

reductions. But other techniques exist such as the interleaved multiplication-reduction with Knuth, Sedlack or Quisquater methods [Dhe98]. Our analysis can also be adapted to these methods.

Algorithm 2.1 Long Integer Multiplication

INPUT: $x = (x_{k-1}x_{k-2}\dots x_1x_0)_b, y = (y_{k-1}y_{k-2}\dots y_1y_0)_b$
OUTPUT: $\mathsf{LIM}(x, y) = x \times y$

Step 1. for i from 0 to $2k-1$ **do** $w_i = 0$
Step 2. for i from 0 to $k-1$ **do**
  $c \leftarrow 0$
  for j from 0 to $k-1$ **do**
    $(uv)_b \leftarrow w_{i+j} + x_j \times y_i + c$
    $w_{i+j} \leftarrow v$ and $c \leftarrow u$
  $w_{i+k} \leftarrow v$
Step 3. Return(w)

Multiplication with Barrett Reduction. Here a modular multiplication $x \times y \mod n$ is the combination of a long integer multiplication $\mathsf{LIM}(x,y)$ followed by a Barrett reduction by the modulus value n. We use the notation $\mathsf{BarrettRed}(a,n)$ for this reduction, thus $\mathsf{BarrettRed}(\mathsf{LIM}(a,m),n)$ corresponds to the computation of $a \times m \mod n$. We do not detail the Barrett reduction algorithm here, for more details the reader can refer to [MOV96] or [ACD+06].

Montgomery Modular Multiplication. Given a modulus n and two integers x and y, of size v in base b, with $\gcd(n, b) = 1$ and $r = b^{\lceil log_b(n) \rceil}$, MontMul algorithm computes:

$$\mathsf{MontMul}(x,\ y,\ n) = x \times y \times r^{-1} \mod n$$

Refer to papers [Mon85] and [KAK96] for details of MontMul implementation.

We denote by $\mathsf{ModMul}(x,y,n)$ the operation $x \times y \mod n$, it can be done using Barrett or Montgomery processing.
Then the Square and Multiply algorithm used for an exponentiation becomes:

Algorithm 2.2 Exponentiation

INPUT: integers m and n with $m < n$, k-bit exponent $d = (d_{k-1}d_{k-2}\dots d_1d_0)_2$
OUTPUT: $\mathsf{Exp}(m,d,n) = m^d \mod n$

Step 1. $a = 1$
Step 2. Process ModMul precomputations
Step 3. for i from $k-1$ to 0 **do**
  $a = \mathsf{ModMul}(a,a,n)$
  if $d_i = 1$ **then** $a = \mathsf{ModMul}(a,m,n)$
Step 4. Return(a)

2.3 The RSA cryptosystem

Let p and q be two secret prime integers and $n = p \times q$ be the public modulus used in the RSA cryptosystem. Let e be the public exponent and d the corresponding private exponent such that $e \cdot d = 1 \mod \phi(n)$ where $\phi(n) = (p-1)(q-1)$. Signing with RSA a message m consists of computing the value $s = m^d \mod n$. Signature s is then verified by checking that $s^e \mod n$ is equal to m.

3 Simple Power Analysis

Power Analysis has been studied for years since it was introduced by Kocher, Jaffe and Jun in [KJJ99]. Many attacks on the most frequently used cryptosystems (DES, AES, RSA, ECC ...) have been published and improvements on power analysis techniques have been done during the last decade. For example Correlation Power Analysis publication from Brier, Clavier and Olivier [BCO04] requires far fewer curves for recovering the key than the original DPA. More recently many studies have been published to improve the Side Channel methodology [GBTP08], [SGV08], [PR09].

The initial publication [KJJ99] on SPA showed how to recover the secret exponent during a modular exponentiation from a single power consumption curve. Impressive results are obtained when the squaring and the multiplying operations have different recognizable and sizeable patterns. If so the bits of the secret exponent can directly be *read* on the power curve of a classical Square and Multiply algorithm. Indeed two consecutives squares on the curve imply the exponent bit is 0 while when a squaring is followed by a multiplication the exponent bit is 1.

This SPA corresponds to the power leakage related to differences in the executed code. The leakage is caused by the fact that the executed code is different for a squaring than for a multiplication. An efficient countermeasure against this SPA is the Side-Channel Atomicity introduced by Chevalier-Mames, Ciet and Joye [CCJ04]. In their implementation the code executed during the whole exponentiation loop is the same for a squaring and a multiplication step. Consequently, it is no more possible to distinguish which operation is performed. Their method improves resistance to classical SPA without adding supplementary multiplications contrary to the Square and Multiply Always algorithm.

Yen *et al.* introduced in [YLMH05] a new type of SPA attack based on the use of particular message values. It defeats previous countermeasures considered as resistant against SPA. In their paper Yen *et al.* use as an input message $m = n-1$ for modular exponentiation. The only two values involved during the exponentiation $m^d \mod n$ are 1 and $n-1$. The Hamming weights of these data are so different by simply observing the power trace, it is very easy to determine the moments when 1 is involved in a multiplication. Only three different operation cases can be processed during the exponentiation: 1×1, $1 \times n-1$ or $n-1 \times n-1$ with three different and recognizable signal patterns. It is then simple to deduce

the sequence of squarings and multiplications and to recover the secret exponent.

Other attacks named Doubling and Collision attacks have been presented in [FV03] and [YLMH05] but they need at least two executions of an exponentiation with the same exponent to mount the attack. However in this paper we choose to only consider attacks recovering the secret from a single power consumption curve and thus counterfeiting the exponent blinding countermeasure.Therefore we do not consider Doubling and Collision attacks here.

4 Enhanced Simple Power Analysis on Exponentiation

In Yen *et al.* attack and the other techniques introduced in this paper, SPA does not aim at distinguishing differences in code execution but rather to detect when specific data are manipulated through their specific power signature. Indeed power signatures during an operation $(x \times y)$ or $(x \times y \mod n)$ will depend on values x and y. If x and/or y have very particular Hamming weights then it will lead to a very characteristic power trace for the multiplication. We present here many values which can generate a recognizable pattern and thus lead to the exponent being recovered from a single power trace.

We illustrate our analysis on the ModMul operation using the Barrett reduction and especially during the computation of LIM(x,y). The same analysis can be done in other kind of modular multiplication methods, for instance in the modular Montgomery multiplication method MontMul(x,y).

4.1 Origin of Power Leakage

The power leakage appears during the operation $x_i \times y_j$ of the long integer multiplication LIM(x,y). Any operation $x_i \times y_j$ has a power consumption related to the number of bit flips of the bit lines manipulated. When one of the operands is null or has very few bits set, for instance is equal to 0, or 2^i with i in $0 \dots t-1$, the t-bit multiplication has a lower power consumption than the average one. We can then distinguish in a long integer multiplication when such a value is manipulated.

If the value of the multiplicand m contains one (or more) of the t-bit word(s) set to 0 or 2^i with i in $0 \dots t-1$, during an atomic exponentiation loop we can recognize each time this value m is manipulated, i.e. each time the exponent bit is 1.

The condition for this SPA to succeed is that one (or more) of the t-bit word(s) of x or y is set to 0 (or in some cases it could be also to 2^i with i in $0 \dots t-1$). We consider here that the leakage appears only for zero values.

We can then quantify the power consumed by the device for computing $x_i \times y_j$. We denote by $C(x_i, y_j)$ this power consumption . As illustrated in Table 1 we can distinguish three categories depending on whether x_i and y_j values are 0 or not.

x_i	y_j	$C(x_i, y_j)$
$x_i \neq 0$	$y_j \neq 0$	C_{High}
$x_i \neq 0$	$y_j = 0$	C_{Medium}
$x_i = 0$	$y_j = 0$	C_{Low}

Table 1. Power Signal quantity for $x_i \times y_j$

When $x_i = 0$ and $y_j = 0$ the device only manipulates zero bits. Thus the amount of power consumed by the multiplication is *Low*, we denote it as C_{Low}. A multiplication with non zero values ($x_i \neq 0$ and $y_j \neq 0$) yields a higher power consumption: we consider this as *High* and denote it as C_{High}. Finally when $x_i \neq 0$ and $y_j = 0$ the amount of power consumed by a multiplication is considered as *Medium*: we denote it as C_{Medium}.

In the operation $\mathsf{LIM}(x,y)$ we can graphically estimate the power curve by $C_{LIM(x,y)} = \sum_{i=0}^{k-1} \sum_{j=0}^{k-1} C(x_i, y_j) \cdot T(k, i, j)$ with $C(x_i, y_j)$ being the power consumption of the device for computing $x_i \times y_j$ and T a function which represents the number of cycles executed for a set (k, i, j). This corresponds to the schematic power curve of Figure 1.
A graphical estimation of power consumption expected depending on whether we have C_{High}, C_{Medium} or C_{Low} is given in Figure 2.

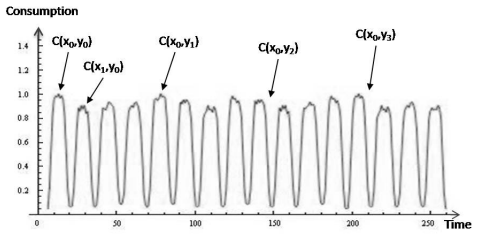

Fig. 1. $C_{LIM(x,y)}$: power curve representation of operation $\mathsf{LIM}(x,y)$ with $k = 4$

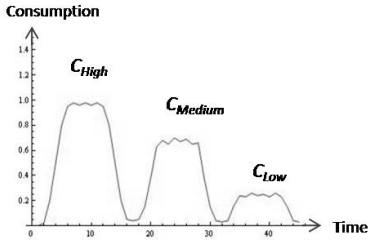

Fig. 2. Three cases of estimated power curves for $C(x_i, y_j)$

When an operation $x_i \times y_j$ leads to $C(x_i, y_j)$ being C_{Low} or C_{Medium} we can identify this operation in the curve. This explains why the SPA introduced by Yen *et al.* allows the secret exponent recovering with a single curve for a well chosen message. Indeed when comparing the three possible operations occurring in an exponentiation with the input chosen message $m = n - 1$ we obtain for $k = 3$ the Table 2. In this table we observe that $C_{LIM(x,y)}$ has different recognizable patterns for each long integer multiplication.

$x \times y$	x in base b	y in base b	$C_{LIM(x,y)}$
$(n-1) \times (n-1)$	$a = (a_2, a_1, a_0)_b$	$a = (a_2, a_1, a_0)_b$	$C_H \vert C_H \vert C_H \vert C_H \vert C_H \vert C_H \vert C_H \vert C_H \vert C_H$
$1 \times (n-1)$	$a = (0, 0, 1)_b$	$m = (m_2, m_1, m_0)_b$	$C_H \vert C_M \vert C_M \vert C_H \vert C_M \vert C_M \vert C_H \vert C_M \vert C_M$
1×1	$a = (0, 0, 1)_b$	$a = (0, 0, 1)_b$	$C_H \vert C_M \vert C_M \vert C_M \vert C_L \vert C_L \vert C_M \vert C_L \vert C_L$

Table 2. The three possible power traces for $\mathsf{LIM}(x,y)$ in Yen *et al.*'s attack for $k = 3$

More Chosen Messages. From this analysis we can enumerate other chosen messages leading to successful SPA on atomic exponentiations such as messages with one or many t-bit word equal to 0 or 2^i with i in $0 \dots t-1$. Messages with a globally low Hamming weight can also lead to a medium or low power consumption and allow to recover the secret exponent in a single power curve.

4.2 Experiments and Practical Results

We experimented this attack on many different multipliers processors to confirm our theoretical analysis. In this section we present some results we obtained on two different devices.

First Device. We implemented a Montgomery Modular exponentiation on a 32×32-bit multiplier, in this case we have t equal to 32. We chose as input messages for exponentiations the following values with $k = 4$: $m_1 = (\alpha, \alpha, 0, 0)$, $m_2 = (\alpha, 0, 0, 0)$ where α is 32-bit random value.

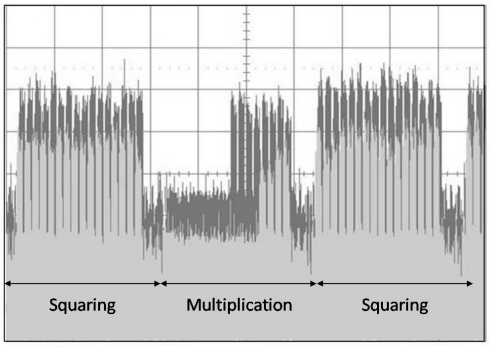

Fig. 3. Part of exponentiation power curves with messages m_1 and m_2 and $k = 4$, zoom on $\mathsf{LIM}(m_1,a)$ (black) and $\mathsf{LIM}(m_2,a)$ (grey)

Figure 3 represents a part of the measured exponentiation curves of these two messages. The black curve corresponds to the exponentiation with message m_1 and grey curve with m_2. The multiplication is clearly identifiable by a lower power consumption compared to the squaring. Message m_2 takes one more 32-bit word equal to 0 than m_1. This results in Figure 3 in a low power consumption longer for grey curve than for black curve during the multiplication.

In this case we also observed that C_{Medium} is close to C_{High} but the two can be distinguished.

Second Device. We designed an 8×64-bit hardware multiplier with the associated long integer exponentiation. In the multiplication $x \times y$ the operand x is manipulated by 64-bit words when the operand y is taken by 8-bit words. The message is placed in the second operand y for the multiplications during the exponentiation.

We chose several messages containing one or more zero 8-bit words and executed the corresponding long integer exponentiations. We then simulated the power consumption of the synthesized multiplier we have designed. By analyzing these power curves we can observe that a zero byte y_j in operand y produces a lower power consumption curve in the cycles where y_j is manipulated. We are then able to recover the whole secret exponent in an exponentiation when a zero byte is present in the message value.

We have explained here the potential power leakages related to multiplication and exponentiation computations and confirmed our analysis with some practical results. In the next paragraph we study the probability of leakage depending on the multiplier and modulus bit lengths.

4.3 Leakage Probability

In this paragraph letters p and q design probabilities.

Probability of leakage during a multiplication. Let x_i be a t-bit word, and p be the probability for x_i to be null, then we have $P(x_i = 0) = \frac{1}{2^t} = p$ and $P(x_i \neq 0) = 1 - p$.
If Y is the event {None of the t-bit word is null in a k-word integer} with $P(Y) = (1-p)^k$, then we have $\overline{Y}$ which corresponds to the event {at least one of the t-bit words is null in a k-word integer} with probability:

$$q = P(\overline{Y}) = 1 - P(Y) = 1 - (1-p)^k = 1 - (1 - \frac{1}{2^t})^k$$

During a long integer modular multiplication $x \times y$ the leakage appears only if at least one of the k t-bit words of x or/and y is null. The probability for this leakage to appear corresponds to $1 - (1-p)^{2k}$.

Probability of leakage during an exponentiation. During an exponentiation we focus on the probability of having a leakage in a t-bit multiplication $x \times y$ when only y takes part in the leakage and not x (or the opposite). Indeed during an exponentiation $m^d \mod n$ the message m is used during each multiplication at step 3 of Algorithm 2.2, when $d_i = 1$. Thus if the value m contains a t-bit word m_i leading to leakage in the operations $m_i \times a_j$ and/or $m_i \times a$ then each multiplication by m_i and thus by m could be identified and the secret exponent d can be recovered from a single power curve.
In this case the probability of having one or many of the t-bit words of m leading to a signing pattern is:

$$q = 1 - (1-p)^k = 1 - (1 - \frac{1}{2^t})^k$$

This is also the probability of having an SPA leaking curve for a single execution of the exponentiation.

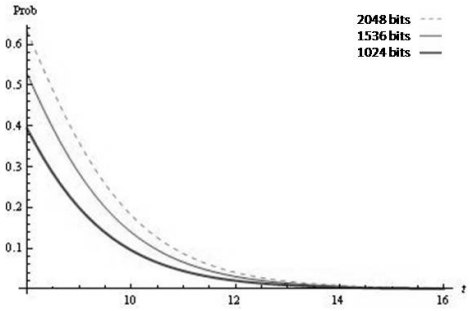

Fig. 4. Probability of having a message with a signing pattern depending of multiplier size (t) and modulus size (1024, 1536, 2048)

In the case of an 8-bit multiplier Figure 4 shows that the probability of having a message with a signing pattern is about 0.394 for a 1024-bit modulus, 0.528 for a 1536-bit modulus and 0.633 for a 2048-bit modulus. When the multiplier is greater than 16 bits this probability decreases for all modulus sizes.
It also obvious that bigger the key length is and smaller the multiplier size (t) is, the higher the probability of recovering the secret exponent d in a single curve is.

Using Poisson law as an approximation of binomial law we have the property that with $1/q$ exponentiation power curves the probability for recovering the secret exponent is $(1 - \frac{1}{exp(1)})$. Thus the probability P_{leak} of recovering the secret exponent using one of the h/q acquired curves is approximated by $P_{leak} = P(h/q) = 1 - (\frac{1}{exp(1)})^h$.

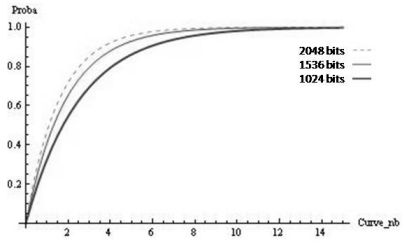

Fig. 5. Probability P_{leak} for an 8-bit multiplier depending on the number of curves acquired and modulus size (1024, 1536, 2048)

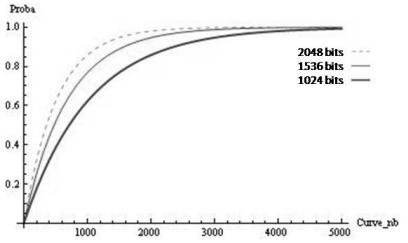

Fig. 6. Probability P_{leak} for a 16-bit multiplier depending on the number of curves acquired and modulus size (1024, 1536, 2048)

Figures 5 and 6 show how many curves would be needed to have, with a probability close to 1, a message leading to a signing pattern which can be used for our SPA attack. With an 8-bit multiplier (Figure 5), a very few messages (5 to 10) are necessary to obtain an exploitable leakage with high probability. For a 16-bit multiplier (Figure 6) between 3000 and 5000 curves are needed for a success probability close to 1. But for a multiplier size greater than 16 the number of curves needed for recovering the secret exponent makes the attack not practical; example is given for $t = 32$ in Appendix A.
The bigger the multiplier, the greater the number of collected curves needed. Examples are given in Table 3.

Modulus size	**Multiplier size**		
	8	16	32
1024	12	4720	$\approx 2^{29}$
1536	9	3150	$\approx 2^{29}$
2048	8	2360	$\approx 2^{28}$

Table 3. Number of messages needed to have $P_{leak} \geq 0.99$

4.4 Enhanced Simple Power Analysis on Blinded Exponentiations

In this section we consider that the exponentiation is secured using message and exponent blinding.

Exponent Blinding. This common countermeasure consists in randomizing the secret exponent d by $d^\star = d + r_1 \cdot n \mod \phi(n)$ with r_1 being a random value. However here the exponent blinding has no effect on our analysis since a single curve is used to recover the private exponent and recovering $d^\star$ is equivalent to recovering d.

Randomized Chosen Message. Now we consider that the message is randomized additively by the classical countermeasure: $m^\star = m + r_1 \cdot n \mod r_2 \cdot n$, with r_1 and r_2 being two l-bit random values. In this case we have $m^\star$ equal to $m + u \cdot n$ with u being a l-bit value equal to $r_1 \mod r_2$. In this case an attack could consist of choosing a message $m^\star$ being 1 or 2^i, guessing a random value u_{guess}, computing message m from guessed randomized message $m^\star$, i.e. $m = m^\star - u_{guess} \cdot n$, and executing at least 2^l exponentiations with input message m. One of the 2^l exponentiation power curves should present leakages and should allow the secret exponent to be recovered with SPA.

However if r_1 and r_2 are effectively chosen in a pure random way we observe that this attack could be done faster. Indeed if we analyze the distribution of values $u = r_1 \mod r_2$ we observe that values do not appear with same probability and that the more frequent are the smallest ones. The most frequent one being $u = 0$. It is illustrated in Figure 7 for $l = 8$. While less pronouced the same phenomenon can also be observed for bigger l values. The best attack method in this case would consist of choosing $u_{guess} = 0$ (or 1 or 2) and executing the exponentiation many times until a leaking power curve is obtained.

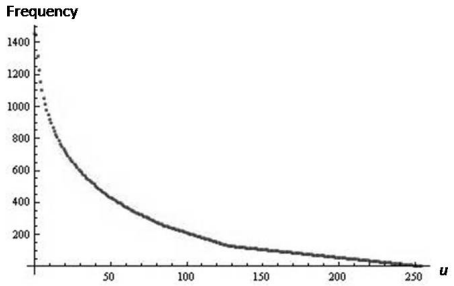

Fig. 7. Distribution of u for $l = 8$

Unknown Message. When analyzing the leakage probabilities of Section 4.3 it appears that the number of curves needed to recover the secret exponent for a fixed multiplier size only depends on the modulus length, even if the message is unknown to the attacker. For instance for a 1024-bit modulus and a 16-bit multiplier by collecting 5000 power consumption curves of exponentiations done with unknown different messages the probability of recovering the secret exponent is close to 1.

Synthesis. As the additive randomization of the message does not significantly increase message length, the amount of messages needed does not increase either. Thus if the attacker can choose input messages of the blinded exponentiation,

he will choose the attack which requires less effort comparing number of chosen message acquisitions needed (when guessing the random) with the number of curves to collect to have $P_{Leak} = 1$.

5 Countermeasures and Recommendations

5.1 Balancing the Power Consumption

The attack presented in this paper is based on the fact that manipulating zero t-bit values results in low power consumption cycles. Thus a method to prevent this attack would consist in using balanced power consumption technology such as dual rail technique. In this case the manipulation of a value with a low Hamming weight (for instance 0) will no longer have a different power consumption than the one due to the manipulation of other values.

5.2 Random Choice for Blinding

As we showed previously the values $r_1 \mod r_2$ are not uniformly distributed when r_1 and r_2 are random. A better solution consists of choosing a fixed value for r_2 and a random value for r_1. From our analysis the best choice for r_2 is to take the biggest l-bit prime number. In that case r_2 will never divide r_1, thus u cannot be null and u values are uniformly distributed as it is showed in Figure 8.

Another consideration is that the random length choice is also directly related to the multiplier size. In section 4.4 we have seen that while the number of possible random values u is smaller than the number of messages to test given by the leakage probability analysis, it is easier to test all random values u. Regarding this statistical properties we showed that when the multiplier is small (8 or 16 bits) the quantity of curves needed for a successful Simple Power Analysis is reasonable.

Thus by combining a multiplier with a size of at least equal to 32 bits, with big random number r_1 (longer than 32 or 64 bits) and the biggest prime integer r_2, the feasibility of the attack explained in this paper is significantly reduced.

6 Remarks on RSA CRT and ECC

We presented our analysis on exponentiation computations. It corresponds directly to straightforward implementations of RSA signature and decryption algorithms as they simply consist of an exponentiation with the secret exponent. In case of RSA CRT the analysis is a little bit different since the input message is reduced modulo p and q before the exponentiations. Even if data manipulated into the multiplications are twice shorter than the modulus n, similar analysis can be conducted on reduced messages. However the countermeasure which consists in fixing the random value r_2, must not be used in RSA CRT implementations as it would not be protected against the correlation analysis on the CRT recombination presented in [AFV07].

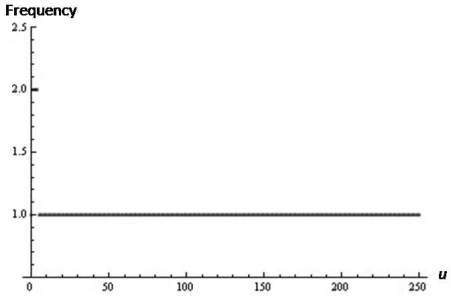

Fig. 8. Distribution of u for $l = 8$ and $r_2 = 251$

ECC are also concerned. The analysis depends on the kind of coordinates and algorithm chosen for the scalar multiplication, anyway implementations using small multipliers and/or small random numbers for coordinates randomization [Cor99] have to be avoided.

7 Conclusion

In this paper we have explained the origin of the power leakages during multiplications and presented other ways to mount Simple Power Analysis attacks. Indeed by observing differences in data power signatures instead of differences in code execution, using some well chosen messages allows the whole secret exponent of RSA cryptosystem to be recovered from a single curve. Moreover we have shown that some improvements in SPA attacks lead to the recovery of the secret exponent on secured exponentiations using blinding countermeasures and with non chosen messages. We analyzed the blinding countermeasures and gave advice to developers to protect their implementations against this enhanced SPA. Judicious choice and large random numbers in blinding countermeasures combined with large size multipliers, especially greater than 32 bits, are recommended for SPA resistance.

Acknowledgments

The authors would like to thank Christophe Clavier for the fruitful discussions we had and the improvements he suggested to us. Thanks also to Sean Commercial for his valuable comments and advice on this manuscrit.

References

[ACD+06] R-M. Avanzi, H. Cohen, C. Doche, G. Frey, T. Lange, K. Nguyen, and F. Verkauteren. Handbook of Elliptic and Hyperelliptic Curve Cryptography, 2006.

[AFV07] F. Amiel, B. Feix, and K. Villegas. Power Analysis for Secret Recovering and Reverse Engineering of Public Key Algorithms. In *Selected Areas in Cryptography*, volume 4876 of *Lecture Notes in Computer Science*, pages 110–125. Springer, 2007.

[BCO04] E. Brier, C. Clavier, and F. Olivier. Correlation Power Analysis with a Leakage Model. In M. Joye and J-J. Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2004.

[CCJ04] B. Chevallier-Mames, M. Ciet, and M. Joye. Low-cost Solutions for Preventing Simple Side-Channel Analysis: side-channel atomicity. *IEEE Transactions on Computers*, 53(6):760–768, 2004.

[Cor99] J-S Coron. Resistance against differential power analysis for elliptic curve cryptosystems. In Ç. K. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 1999*, volume 1717 of *Lecture Notes in Computer Science*, pages 292–302. Springer, 1999.

[DH76] W. Diffie and M. E. Hellman. New Directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[Dhe98] J-F. Dhem. *Design of an efficient public-key cryptographic library for RISC-based smart cards*. PhD thesis, Université catholique de Louvain, Louvain, 1998.

[FV03] P-A. Fouque and F. Valette. The Doubling Attack - *why upwards is better than downwards*. In C. D. Walter, Ç. K. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2003*, volume 2779 of *Lecture Notes in Computer Science*, pages 269–280. Springer, 2003.

[GBTP08] B. Gierlichs, L. Batina, P. Tuyls, and B. Preneel. Mutual Information Analysis. In E. Oswald and P. Rohatgi, editors, *Cryptographic Hardware and Embedded Systems CHES 2008*, volume 5154 of *Lecture Notes in Computer Science*, pages 426–442. Springer, 2008.

[KAK96] Ç. K. Koç, T. Acar, and B-S. Kaliski. Analysing and comparing Montgomery multiplication algorithms. *IEEE Micro*, 16(3):26–33, 1996.

[KJJ99] P. C. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In M. J. Wiener, editor, *Advances in Cryptology - CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[Koc96] P. C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996.

[Mon85] P.L. Montgomery. Modular multiplication without trial division. *Mathematics of Computation, 44(170)*, pages 519–521, April 1985.

[MOV96] Alfred Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[PR09] E. Prouff and M. Rivain. Theoretical and practical aspects of mutual information based side channel analysis. In M. Abdalla, D. Pointcheval, P-A. Fouque, and D. Vergnaud, editors, *Applied Cryptography and Network Security, ACNS 2009*, volume 5536 of *Lecture Notes in Computer Science*, pages 499–518, 2009.

[RSA78] R. L. Rivest, A Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM 21*, pages 120–126, 1978.

[SGV08] F-X. Standaert, B. Gierlichs, and I. Verbauwhede. Partition vs. comparison side-channel distinguishers: An empirical evaluation of statistical tests for univariate side-channel attacks against two unprotected cmos devices. In Pil Joong Lee and Jung Hee Cheon, editors, *Information Security and Cryptology ICISC 2008*, volume 5461 of *Lecture Notes in Computer Science*, pages 253–267. Springer, 2008.

[YLMH05] S-M. Yen, W-C. Lien, S. Moon, and J. Ha. Power Analysis by Exploiting Chosen Message and Internal Collisions - Vulnerability of Checking Mechanism for RSA-decryption. In E. Dawson and S. Vaudenay, editors, *Mycrypt 2005*, volume 3715 of *Lecture Notes in Computer Science*, pages 183–1956. Springer, February 2005.

A Leakage Probability for t up to 32

The following figures give the probability of leakage for a 32-bit multiplier and the leakage probability increasement regarding the number of exponentiation executions.

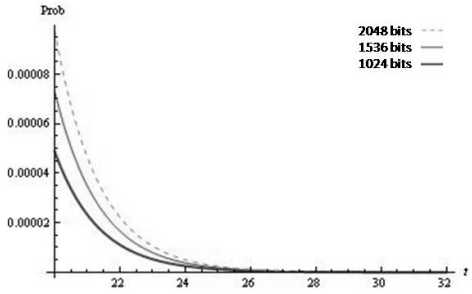

Fig. 9. Probability of having a message with a signing pattern for 32-bit multiplier depending on modulus size (1024, 1536, 2048)

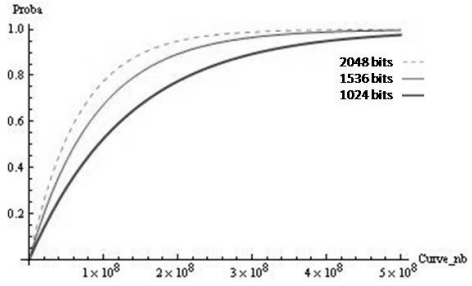

Fig. 10. Probability P_{leak} for a 32-bit multiplier depending on the number of curves acquired and modulus size (1024, 1536, 2048)