Horizontal Correlation Analysis on Exponentiation

Christophe Clavier[1], Benoit Feix[2], Georges Gagnerot[2], Mylène Roussellet[2], and Vincent Verneuil[2,3]

[1] XLIM-CNRS, Université de Limoges, Limoges, France
firstname.familyname@unilim.fr
[2] Inside Contactless
41 Parc du Golf, 13856 Aix-en-Provence, Cedex 3, France
firstname-first-letterfamilyname@insidefr.com
[3] Institut de Mathématiques de Bordeaux, 351, cours de la Libération, 33 405 Talence cedex, France
firstname.familyname@math.u-bordeaux1.fr

Abstract. We introduce in this paper a technique in which we apply correlation analysis using only one execution power curve during an exponentiation to recover the whole secret exponent manipulated by the chip. As in the Big Mac attack from Walter, longer keys may facilitate this analysis and success will depend on the arithmetic coprocessor characteristics. We present the theory of the attack with some practical successful results on an embedded device and analyze the efficiency of classical countermeasures with respect to our attack.

Our technique, which uses a single exponentiation curve, cannot be prevented by exponent blinding. Also, contrarily to the Big Mac attack, it applies even in the case of regular implementations such as the *square and multiply always* or the Montgomery ladder. We also point out that DSA and Diffie-Hellman exponentiations are no longer immune against CPA. Then we discuss the efficiency of known countermeasures, and we finally present some new ones.

Keywords: Public Key Cryptography, Side-Channel Analysis, Exponentiation, Arithmetic Coprocessors.

1 Introduction

Securing embedded products from *Side-Channel Analysis* (SCA) has become a difficult challenge for developers who are confronted with more and more analysis techniques as the physical attacks field is studied. Since the original *Simple* Side-Channel Analysis (SSCA) – which include *Timing Attacks*, SPA, and SEMA – and *Differential* Side-Channel Analysis (DSCA) – including DPA and DEMA – have been introduced by Kocher et al. [18,19] many improvements and new SCA techniques have been published. Messerges et al. were the first to apply these techniques to public key implementations [21]. Later on, original DSCA has been

M. Soriano, S. Qing, and J. López (Eds.): ICICS 2010, LNCS 6476, pp. 46–61, 2010.

improved by more efficient techniques such as the one based on the *likelihood test* proposed by Bevan et al. [4], the *Correlation* Power Analysis (CPA) introduced by Brier et al. [5], and more recent techniques like the *Mutual Information Analysis* (MIA) [14,24,25]. A common principle of all these techniques is that they require many power consumption or electromagnetic radiation curves to recover the secret manipulated. Hardware protections and software blinding [9,18] countermeasures are generally used and when correctly implemented they counteract these attacks.

Among all those studies the so-called *Big Mac attack* is a refined approach introduced by Walter [26,27] from which our contribution is inspired. This technique aims at distinguishing squarings from multiplications and thus recovering the secret exponent of an RSA exponentiation with a single execution curve.

We present in this paper another analysis which uses a single curve. We named this technique *horizontal* correlation analysis, which consists of computing classical statistical treatments such as the correlation factor on several segments extracted from a single execution curve of a known message RSA encryption. Since this analysis method requires only one execution of the exponentiation as the Big Mac attack, it is then not prevented by the usual exponent blinding countermeasure.

The paper is organized as follows. Section 2 gives an overview of asymmetric algorithms and the way to compute long integer multiplication in embedded implementations. Section 3 reminds the reader of previous studies on power analysis techniques discussed in this article. The horizontal correlation analysis is presented in Section 4 with some practical results and a comparison between our technique and the Big Mac attack. Known and new countermeasures are discussed in Section 5. In Section 6 we deal with horizontal side channel analysis in the most common cryptosystems. Finally we conclude this paper in Section 7.

2 Public Key Embedded Implementations

Most of the public key cryptosystems embedded in smart devices, RSA, DSA [12], Diffie-Hellman key exchange [11] and their equivalent in Elliptic Curve Cryptography (ECC) – namely ECDSA and ECDH [12], are based on the modular exponentiation or the scalar multiplication. In both cases the underlying operation is the modular long integer multiplication. Many methods such as the Montgomery multiplication [23] and interleaved multiplication-reduction with Knuth, Barrett, Sedlack or Quisquater methods [10] can be applied to perform efficient modular multiplications. Most of them have in common that the long integer multiplication is internally done with a loop of one (or more) smaller multiplier(s) operating on t-bit words. An example is given in Alg. 2.1 which performs the schoolbook long integer multiplication using a t-bit internal multiplier giving a $2t$-bit result. The decomposition of an integer x in t-bit words is given by $x = (x_{l-1}x_{l-2}\dots x_0)_b$ with $b = 2^t$ and $l = \lceil log_b(x) \rceil$. Other long integer multiplication algorithms may also be used such as Comba [8] and Karatsuba [17] methods.

Algorithm 2.1. Long Integer Multiplication
INPUT: $x = (x_{l-1}x_{l-2}\dots x_0)_b, y = (y_{l-1}y_{l-2}\dots y_0)_b$
OUTPUT: $\mathrm{LIM}(x, y) = x \times y$

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

We consider in this paper that a modular multiplication $x \times y \bmod n$ is performed using a long integer multiplication followed by a Barrett reduction denoted by BarrettRed(LIM(x,y),n).

Algorithm 2.2. Square and Multiply Exponentiation
INPUT: integers m and n such that $m < n$, v-bit exponent $d = (d_{v-1}d_{v-2}\dots d_0)_2$
OUTPUT: $\mathrm{Exp}(m, d, n) = m^d \bmod n$

Step 1. $a \leftarrow 1$
Step 2. Process Barrett reduction precomputations
Step 3. for i from $v-1$ to 0 **do**
  $a \leftarrow$ BarrettRed(LIM(a,a), n)
  if $d_i = 1$ **then** $a \leftarrow$ BarrettRed(LIM(a,m), n)
Step 4. Return(a)

Alg. 2.2 presents the classical *square and multiply* modular exponentiation algorithm using Barrett reduction. More details on Barrett reduction can be found in [3,20] and other methods can be used to perform the exponentiation such as Montgomery ladder [22] and *sliding window* techniques [6].

We assume in the following of this paper that Alg. 2.2 is implemented in an SPA resistant way, for instance using the *atomicity* principle [7].

While we have chosen to consider modular multiplication using Barrett reduction, and square and multiply exponentiation, the results we present in this paper also apply to the other modular multiplication methods, long integer multiplication techniques and exponentiation algorithms mentioned above.

3 Side-Channel Analysis

We have chosen to introduce in this paper the terms of *vertical* and *horizontal* side-channel analysis to classify the different known attacks. The present section deals with known vertical and horizontal power analysis techniques. Our contribution, the horizontal correlation analysis on exponentiation is detailed in Section 4.

3.1 Background

Side-channel attacks rely on the following physical property: a microprocessor is physically made of thousands of logical gates switching differently depending on the executed operations and on the manipulated data. Therefore the power consumption and the electromagnetic radiation, which depend on those gates switches, reflect and may leak information on the executed instructions and the manipulated data. Consequently, by monitoring the power consumption or radiation of a device performing cryptographic operations, an observer may recover information on the implementation of the program executed and on the secret data involved.

Simple Side-Channel Analysis. In the case of an exponentiation, original SSCA consists in observing that, if the squaring operation has a different pattern from the one of the multiplication, the secret exponent can be read on the curve. Classical countermeasures consist of using so-called *regular* algorithms like the *square and multiply always* or Montgomery ladder algorithms [22,16], *atomicity* principle which leads to regular power curves.

Differential Side-Channel Analysis. Deeper analysis such as DSCA [21] can be used to recover the private key of an SSCA protected implementation. These analyses make use of the relationship between the manipulated data and the power consumption/radiation. Since this leakage is very small, hundreds to thousands of curves and statistical treatment are generally required to learn a single bit of the exponent. Usual countermeasures consist of randomizing the modulus, the message, and/or the exponent.

Correlation Power Analysis. This technique is essentially an improvement of the Differential Power Analysis. Initially published by Brier et al. [5] to recover secrets on symmetric implementations, CPA is also successful in attacking asymmetric algorithms [2] with much fewer curves than classical DPA.

In [2], Amiel et al. apply the CPA to recover the secret exponent of public key implementations. Their practical results show that the number of curves necessary to an attack is much lower compared to DPA: less than one hundred of curves is sufficient. It is worth noticing that the correlation is the highest when computed on t bits, t being the bit length of the device multiplier.

The authors shows the details [2, Fig. 8] of the correlation factor obtained for every multiplicand t-bit word A_i during the squaring operation $A \times A$ using a hardware multiplier. Interestingly a correlation peak occurs for $\mathrm{H}(A_i)$ each time a word A_i is involved in a multiplication $A_i \times A_j$.

We present in the next section our horizontal correlation analysis which takes advantage of this observation.

Collision Power Analysis. The *Doubling attack* from Fouque and Valette [13] is the first collision technique published on public key implementations. It recovers the whole secret scalar (exponent) with only a couple of curves. Other collision attacks have been presented in [1,15,28]. They all require at least two power execution curves, therefore the classical exponent randomization (blinding) countermeasure counterfeits those techniques.

Notations. Let C^k denote the portion of an exponentiation curve C corresponding to the k-th long integer multiplication, and $C^k_{i,j}$ denote the curve segment corresponding to the internal multiplication $x_i \times y_j$ in C^k.

Big Mac Attack. Walter's attack needs, as our technique, a single exponentiation power curve to recover the secret exponent. For each long integer multiplication, the Big Mac attack detects if the operation processed is either $a \times a$ or $a \times m$. The operations $x_i \times y_j$ – and thus curves $C^k_{i,j}$ – can be easily identified on the power curve from their specific pattern which is repeated l^2 times in the long integer multiplication loop. A template power trace T^1_m is computed (either from the precomputations or from the first squaring operation) to characterize the message value m manipulation during the long integer multiplication. The Euclidean distance between T^1_m and each long integer multplication template power trace is then computed. If it exceeds a threshold the multiplication trace is supposed to be a squaring, and a multiplication by m otherwise.

Cross-Correlation. Cross correlation technique has been introduced in [21] to try to recover the secret exponent in a single curve, however no successful practical result using a single exponentiation power curve has been yet published.

3.2 Vertical and Horizontal Attacks Classification

We refer to the techniques analyzing a same time sample in many execution curves – see Fig. 1 – as *vertical* side-channel analysis. The classical DPA and CPA techniques thus fall into this category. We also include in the vertical analysis class the collision attacks mentioned above. Indeed even if many points on a same curve are used by those techniques, they require at least two power execution curves and manipulate them together. All those attacks are avoided with the exponent blinding countermeasure presented by Kocher [18, Section 10].

We propose the *horizontal* side-channel analysis denomination for the attacks using a single curve. First known horizontal power analysis is the classical SPA. Single curve Cross-correlation and Big Mac attacks are also horizontal techniques.

Our attack, we present in the next section, computes the correlation factor on many curve segments extracted from a single consumption/radiation curve as depicted in Fig. 2. It thus contrasts with vertical attacks which target a particular instant of the execution in several curves.

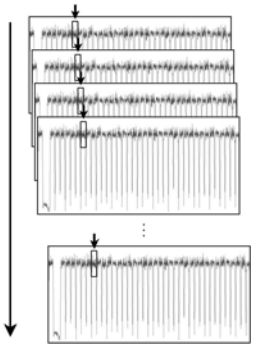


Fig. 1. Vertical Side Channel Analysis

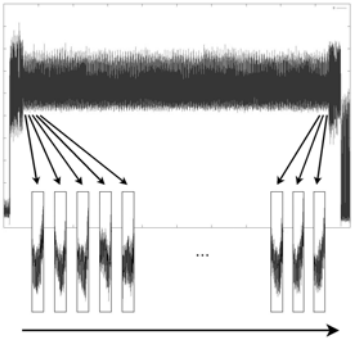


Fig. 2. Horizontal side-channel analysis

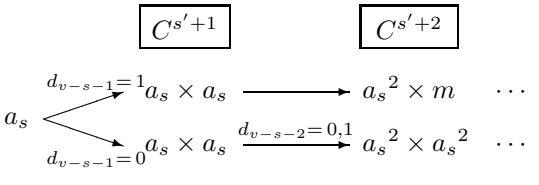
4 Horizontal Correlation Analysis

We present hereafter our attack on an atomically protected RSA exponentiation using Barrett reduction.

4.1 Recovering the Secret Exponent with One Known Message Encryption

As in vertical DPA and CPA on modular exponentiation, the horizontal correlation analysis reveals the bits of the private exponent d one after another. Each exponent bit is recovered by determining whether the processing of this bit involves a multiplication by m or not (cf. Alg. 2.2). The difference with classical vertical analysis lies in the way to build such hypothesis test. Computing the long integer multiplication $x \times y$ using Alg. 2.1 requires l^2 t-bit multiplier calls. The multiplication side-channel curve thus yields l^2 curve segments $C^k_{i,j}$ available to an attacker.

Assuming that the first s bits $d_{v-1}d_{v-2}\dots d_{v-s}$ of the exponent are already known, an attacker is able to compute the value a_s of the accumulator in Alg. 2.2 after processing the s-th bit. The processing of the first s bits corresponds to the first s' long integer multiplications with $s' = s + \mathrm{H}(d_{v-1}d_{v-2}\dots d_{v-s})$ known from the attacker. The value of the unknown $(s+1)$-th exponent bit is then equal to 1 if and only if the $(s'+2)$-th long integer multiplication is ${a_s}^2 \times m$.



At this point there are several ways of determining whether the multiplication by m is performed or not.

First, one may show that the series of consumptions in the set of l^2 curve segments is consistent with the series of operand values m_j presumably involved in each of these segments. To this purpose the attacker simply computes the correlation factor between the series of Hamming weights $\mathrm{H}(m_j)$ and the series of curve segments $C_{i,j}^{s'+2}$ – i.e. taking $D = m_j$ and $R = 0$ in the correlation factor formula. In other words we use the curve segments as they would be in a vertical analysis if they were independent aligned curves. A correlation peak reveals that $d_{v-s-1} = 1$ since it occurs if and only if m is actually handled in this long multiplication.

Alternatively one may correlate the curves segments with the intermediate results of each t-bit multiplication $x_i \times y_j$, cf. Alg. 2.1, with $x = a_s$ and $y = m$, or in other words take $D = a_i \times m_j$. This method may also be appropriate since the words of the result are written in registers at the end of the operation. Moreover in that case l^2 different values are available for correlating the curve segments instead of l previously. This diversity of data may be necessary for the success of the attack when l is small. Note that other intermediate values may also lead to better results depending on the hardware leakages.

Another method consists of using the curve segments $C_{i,j}^{s'+3}$ of the next long integer multiplication and correlating them with the Hamming weight of the words of the result ${a_s}^2 \times m$. If the $(s'+2)$-th operation is a multiplication by m then the $(s'+3)$-th operation is a squaring ${a_{s+1}}^2$, manipulating the words of the integer ${a_s}^2 \times m$ in the t-bit multiplier. As pointed out by Walter in [27] for the Big Mac attack, the longer the integer manipulated and the smaller the size t of the multiplier, the larger the number l^2 of curve segments. Thus longer keys are more at risk with respect to horizontal analysis. For instance in an RSA 2048 bit encryption, if the long integer multiplication is implemented using a 32-bit multiplier we obtain $(2048/32)^2 = 4096$ segments $C_{i,j}^k$ per curve C^k.

Remark: The series of Hamming weights $\mathrm{H}(m_j)$ is not only correlated with the series of curve segments in $C^{s'+2}$ (provided that $d_{v-s-1} = 1$), but also with the series of curve segments in each and any C^k corresponding to a multiplication by m. Defining a *wide segment* $C_{i,j}^*$ as the set of segments $C_{i,j}^k$ for all k on the curve C and correlating the series of $\mathrm{H}(m_j)$ with the series of wide segments $C_{i,j}^*$ (instead of the series of segments $C_{i,j}^{s'+2}$) will produce a wide segment correlation curve with a peak occurring for each k corresponding to a multiplication by the message. It is thus possible to determine in one shot the exact sequence of squarings and multiplications by m, revealing the whole private exponent with only one curve and only one correlation computation.

4.2 Practical Results

This section presents the successful experiments we conducted to demonstrate the efficiency of the horizontal correlation analysis technique. We used a 16-bit RISC microprocessor on which we implemented a software 16×16 bits long integer multiplication to simulate the behavior of a coprocessor. We aim at

correlating a single long integer multiplication with one or both operands manipulated – i.e. y_j or $x_i \times y_j$.

The measurement bench is composed of a Lecroy Wavepro oscilloscope, and homemade softwares and electronic cards were used to acquire the power curves and process the attacks.

Firstly we performed a classical vertical correlation analysis to characterize our implementation and measurement bench, and to validate the correlation model; then we processed with the horizontal correlation analysis previously described.

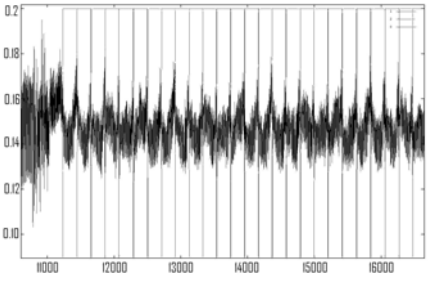


Fig. 3. Beginning of a long integer multiplication power curve, lines delimitate each $C_{i,j}^k$

Vertical Correlation Analysis. This analysis succeeded in two cases during the operation $x \times y$. We obtained correlation peaks by correlating power curves with values x_i and y_j and also by correlating the power curves with the result value of operation $x_i \times y_j$. Fig. 4 and Fig. 5 show the correlation traces we obtained for both cases with 500 power curves.

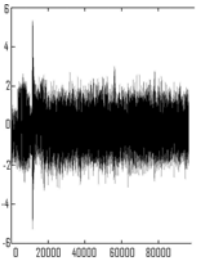


Fig. 4. Vertical CPA on value y_j

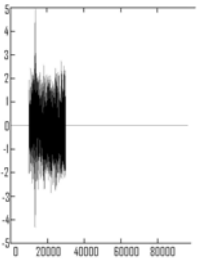


Fig. 5. Vertical CPA on value $x_i \times y_j$

This suggests that one can perform horizontal correlation as explained previously either using y_i values or using result values $x_i \times y_j$ for correlating with segment curves of the long integer multiplication.

Horizontal Correlation Analysis. We have chosen to test our technique within a 512-bit multiplication LIM(x, y). This allows us to obtain 1024 curve segments $C^k_{i,j}$ of 16-bit multiplications to mount the analysis, which should be enough for the success of our attack regarding the vertical analysis results. From the single power curve we acquired, we processed the signal in order to detect each set of cycles corresponding to each t-bit multiplication $x_i \times y_j$ and divide the single power curve in 1024 segments $C^k_{i,j}$ as depicted in Fig. 3.

We performed horizontal correlation analysis as explained in Section 4 for the two cases $D = a_i \times m_j$ and $D = m_j$ and recovered the operation executed as shown in Fig. 6 and Fig. 7. In each figure, the grey trace shows a greater correlation than the black one and thus corresponds to the correct guess on the operation.

Since our attack actually enabled us to distinguish one operation from another, it is then possible to identify a squaring $a \times a$ from a multiplication $a \times m$ in the Step 3 of Alg. 2.2. The secret exponent d used in an exponentiation can thus be recovered by using a single power trace, even when the exponentiation is protected by an atomic implementation.

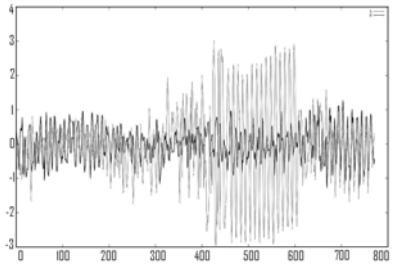


Fig. 6. Horizontal CPA on value $a_i \times m_j$

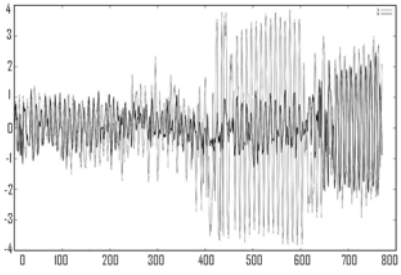


Fig. 7. Horizontal CPA on value m_j

We have presented here a technique to recover the secret exponent using a single curve when the input message is known and have proven this attack to be practically successful. Although the attack is tested on a software implementation, results obtained by Amiel et al. [2, Fig. 8] prove that correlation techniques are efficient on hardware coprocessors (with multiplier size larger than 16 bits), and enable to locate each little multiplication involved in a long integer multiplication. We thus consider that our attack can also threaten hardware coprocessors.

4.3 Comparing Our Technique with the Big Mac Attack

We now compare our proposed horizontal CPA on exponentiation with the Big Mac attack which is the most powerful known horizontal analysis to recover a private exponent. A common property is that both techniques counteract the randomization of the exponent.

A first difference between both methods is that the Big Mac templates are generated by averaging the leakage dependency from a not targeted argument. It is thus implicitly accepted to lose the information brought by this auxiliary data. On the other hand, horizontal correlation exploits the knowledge of both multiplication operands a and m (under assumption on the exponent bit) to correlate it with all l^2 segments $C^k_{i,j}$. This full exploitation of the available information included in the l^2 curve segments tends us to expect a better efficiency of the correlation method particularly when processing noisy observations.

But the main difference is not there. What fundamentally separates the Big Mac and correlation methods is that the former deals with templates – which the attacker tries to identify – while the later rather consider intermediate results – whose manipulation validates a secret-dependent guess. With the Big Mac technique an attacker is able to answer the question *Is this operation of that particular kind?* (squaring, multiplication by m or a power thereof) while the correlation with intermediate data not only brings the same information but also answers the more important question *Is the result of that operation involved in the sequel of the computation?* The main consequence is that horizontal CPA is effective even when the exponentiation implementation is *regular* with respect to the operation performed. This is notably the case of the *square and multiply always*[1] and the Montgomery ladder exponentiations which are not threaten by the Big Mac attack. In this respect we can say that our horizontal CPA combines both the advantage of classical CPA which is able to validate guesses based on the manipulation of intermediate results (but which is defeated by the randomization of the exponent) and that of horizontal techniques which are immune to exponent blinding.

On the other hand the limitation of the Big Mac attack – its ignorance of the intermediate results – is precisely the cause of its noticeable property to be applicable also when the base of the exponentiation is not known from the attacker. The Big Mac attack thus applies when the message is randomized and/or in the case of a Chinese Remainder Theorem (CRT) implementation of RSA. While the horizontal correlation technique does not intrinsically deals with message randomization, we give in the next section some hints that allow breaking those protected implementations when the random bit-length is not sufficiently large.

4.4 Horizontal Analysis on Blinded Exponentiation

To protect public key implementations from SCA developers usually include blinding countermeasures in their cryptographic codes. The most popular ones on RSA exponentiation are:

[1] Referring to the description given in 4.1 the method using the curve segments $C^{s'+3}_{i,j}$ validates that the value produced by the multiplication by m is involved or not in the next squaring operation. A similar technique also applies to the Montgomery ladder.

- Additive randomization of the message and the modulus: $m^\star = m + r_1 \cdot n \bmod r_2 \cdot n = m + u \cdot n$ with r_1, r_2 being λ-bit random values different each time the computation is executed, and $u = r_1 \bmod r_2$.
- Multiplicative randomization of the message: $m^\star = r^e \cdot m \bmod n$ with r a random value and e the public exponent,
- Additive randomization of the exponent: $d^\star = d + r \cdot \phi(n)$ with r a random value.

All these countermeasures prevent from the classical vertical side-channel analysis but the efficiency of the implementations is penalized as the exponent and modulus are extended of the random used bit lengths.

Guessing the randomized message $m^\star$. In this paragraph we consider that the message has been randomized by an additive (or multiplicative) method, the secret exponent has also been randomized and the message is encrypted by an atomic multiply always exponentiation. We analyze the security of such implementation against horizontal CPA. The major difference with vertical side-channel analysis is that the exponent blinding has no effect since we analyze a single curve and recovering $d^\star$ is equivalent to recovering d.

Assuming that the entropy of u is λ bits, there are 2^λ possible values for the message $m^\star$ knowing m and n. The first step of an attack is to deduce the value of the random u. This is achieved by performing one horizontal CPA for each possible value of u on the very first multiplication which computes $(m^\star)^2$. Since this multiplication is necessarily computed, the value of u should be retrieved as the one showing a correlation peak. Once u is recovered, the randomized message $m^\star$ is known and recovering the bits of the exponent d is similar to the non blinded case using $m^\star$ instead of m. Consequently, the entropy of u must be large enough (e.g. $\lambda \geq 32$) to make the number of guess unaffordable and prevent from horizontal correlation analysis.

The actual entropy of the randomization. In the case of additive randomization of the message, $m^\star$ depends on two λ-bit random values r_1 and r_2. Obviously, the actual entropy of this randomization is not 2λ bits, and interestingly it is even strictly less than λ bits. The reason is that $m^\star = m + u \cdot n$ with $u = r_1 \bmod r_2$, and thus smaller u values are more probable than larger ones.

Assuming that r_1 and r_2 are uniformly drawn at random in the ranges $[0, \ldots, 2^\lambda - 1]$ and $[1, \ldots, 2^\lambda - 1]$ respectively, statistical experiments show that the actual entropy of u is about $\lambda - 0.75$ bits[2].

A consequence of this bias on the random u is that an attacker can exhaust only a subset of the smaller guesses about u. If the attack does not succeed, then he can try again on another exponentiation curve. For $\lambda = 8$ guessing only the 41 smaller u will succeed with probability $\frac{1}{2}$.

An extreme case, which optimizes the average number of correlation curve computations, is to guess only the value $u = 0$ [3]. This way, only 38 and 5352

[2] The loss of 0.75 bits of entropy is nearly independent of λ for typical values ($\lambda \leq 64$).
[3] Or $u = 1$ if the implementation does not allow $u = 0$.

correlation curve computations are needed in the mean when λ is equal to 8 and 16 respectively.

These observations demonstrate that the guessing attack described in the previous paragraph is more efficient than may be trivially expected. This confirms the need to use a large random bit length λ.

5 Countermeasures

We now study the real efficiency of the classical side channel countermeasures and propose new countermeasures.

5.1 Blinding

As said previously the blinding of the exponent is not an efficient countermeasure here, it is thus highly recommended to implement a resistant and efficient blinding method on the data manipulated.

5.2 New Countermeasures

We suggest protecting sensitive implementations from this analysis by introducing blinding into the t-bit multiplications, by randomizing their execution order or by mixing both solutions.

Blind Operands in LIM. A full blinding countermeasure on the words x_i and y_j consists in replacing in Alg. 2.1 the operation $(w_{i+j} + x_i \times y_j) + c$ by $(w_{i+j} + (x_i - r_1) \times (y_j - r_2)) + r_1 \times y_j + r_2 \times x_i - r_1 \times r_2 + c$ with r_1 and r_2 two t-bit random values. For efficiency purposes, the values $r_1 \times x_i$, $r_2 \times y_j$, $r_1 \times r_2$ should be computed once and stored. Moreover, these precomputations must also be protected from correlation analysis. For example, performing them in a random order yields $(2l + 1)!$ different possibilities. In this case the LIM operation requires $l^2 + 2l + 1$ t-bit multiplications and necessitates $2(n + 2t)$ bits of additional storage.

In the following we improve this countermeasure by mixing the data blinding with a randomization of the order of the internal loops of the long integer multiplication.

Randomize One Loop in LIM and Blind. This countermeasure consists in randomizing the way the words x_i are taken by the long integer multiplication algorithm. In other words it randomizes the order of the lines of the schoolbook multiplication. Then computing correlation between x_i and $C^k_{i,j}$ does not yield the expected result anymore. On the other hand it remains necessary to blind the words of y. An example of implementation is given in Alg. 5.3.

The random permutation provides $l!$ different possibilities for the execution order of the first loop. For example, using a 32-bit multiplier, a 1024-bit long

integer multiplication has about 2^{117} possible execution orders of the first loop and with 2048-bit operands it comes to about 2^{296} possibilities.

Algorithm 5.3. LIM with lines randomization and blinding
INPUT: $x = (x_{l-1}x_{l-2}\dots x_1x_0)_b, y = (y_{l-1}y_{l-2}\dots y_1y_0)_b$
OUTPUT: LinesRandLIM(x,y) $= x \times y$

Step 1. Draw a random permutation vector $\alpha = (\alpha_{l-1}\dots\alpha_0)$ in $[0, l-1]$
Step 2. Draw a random value r in $\left[1, 2^t - 1\right]$
Step 3. for i from 0 to $2l-1$ **do** $w_i = 0$
Step 4. for h from 0 to $l-1$ **do**
  $i \leftarrow \alpha_h$, $r_i \leftarrow r \times x_i$ and $c \leftarrow 0$
  for j from 0 to $l-1$ **do**
    $(uv)_b \leftarrow (w_{i+j} + x_i \times (y_j - r) + c) + r_i$
    $w_{i+j} \leftarrow v$ and $c \leftarrow u$
  while $c \neq 0$ **do**
    $uv \leftarrow w_{i+j} + c$
    $w_{i+j} \leftarrow v$, $c \leftarrow u$ and $j \leftarrow j + 1$
Step 5. Return(w)

Compared to the previous countermeasure, Alg. 5.3 requires only $l^2 + l$ t-bit multiplications and $2t$ bits of additional storage.

Remark. One may argue that in the case of very small l values such a countermeasure might not be efficient. Remember here that if l is very small, the horizontal correlation analysis is not efficient either because of the small number of curve segments.

Algorithm 5.4. LIM with lines and columns randomization
INPUT: $x = (x_{l-1}x_{l-2}\dots x_1x_0)_b, y = (y_{l-1}y_{l-2}\dots y_1y_0)_b$
OUTPUT: MatrixRandLIM(x,y) $= x \times y$

Step 1. Draw two random permutation vectors α, β in $[0, l-1]$
Step 2. for i from 0 to $2l-1$ **do** $w_i = 0$
Step 3. for h from 0 to $l-1$ **do**
  $i \leftarrow \alpha_h$
  for j from 0 to $2l-1$ **do** $c_j = 0$
  for k from 0 to $l-1$ **do**
    $j \leftarrow \beta_k$
    $(uv)_b \leftarrow w_{i+j} + x_i \times y_j$
    $w_{i+j} \leftarrow v$ and $c_{i+j+1} \leftarrow u$
  $u \leftarrow 0$
  for s from $i+1$ to $2l-1$ **do**
    $(uv)_b \leftarrow w_s + c_s + u$
    $w_s \leftarrow v$
Step 4. Return(w)

Randomize the Two Loops in LIM. We propose a variant of the previous countermeasure in which the execution order of the both internal loops of the long integer multiplication are randomized. This means randomizing both lines and columns of the schoolbook multiplication. The main advantage is that none of the operands x_i or y_j needs to be blinded anymore. The number of possibilities for the order of the l^2 internal multiplication is increased to $(l!)^2$. An example of implementation is given in Alg. 5.4.

Unlike the two previous countermeasures, Alg. 5.4 requires no extra t-bit multiplication compared to LIM. It is then an efficient and interesting countermeasure, while the remaining difficulty for designers consists in implementing it in hardware.

6 Concerns for Common Cryptosystems

In the case of an RSA exponentiation using the CRT method our technique cannot be applied since the operations are performed modulo p and q which are unknown to the attacker. On the other hand DSA and Diffie-Hellman exponentiations were until now considered immune to DPA and CPA because the exponents are chosen at random for each execution. Indeed it naturally protects these cryptosystems from vertical analysis. However, as horizontal CPA requires a single execution power trace to recover the secret exponent, DSA and Diffie-Hellman exponentiations are prone to this attack and other countermeasures must be used in embedded implementations. It is worth noticing that ECC cryptosystems are theoretically also concerned by the horizontal side-channel analysis. However since key lengths are considerably shorter very few curves per scalar multiplication will be available for the attack.

7 Conclusion

We presented in this paper a way to apply classical power analysis techniques such as CPA on a single curve to recover the secret key in some public key implementations – e.g. non CRT RSA, DSA or Diffie-Hellman – protected or not by exponent randomization. We also applied our technique in practice and presented some successful results obtained on a 16-bit RISC microprocessor. However even with bigger multiplier sizes (32 or 64 bits) this attack can be envisaged depending on the key size, cf. Section 4.1. We discussed the resistance of some countermeasures to our analysis and introduced three secure multiplication algorithms.

Our contribution enforces the necessity of using sufficiently large random numbers for blinding in secure implementations and highlights the fact that increasing the key lengths in the next years could improve the efficiency of some side-channel attacks. The attack we presented threatens implementations which may have been considered secure up to now. This new potential risk should then be taken into account when developing embedded products.

Acknowledgments

The authors would like to thank Christophe Giraud and Sean Commercial for their valuable comments and advices on this manuscrit.

References

1. Amiel, F., Feix, B.: On the BRIP Algorithms Security for RSA. In: Onieva, J.A., Sauveron, D., Chaumette, S., Gollmann, D., Markantonakis, C. (eds.) WISTP 2008. LNCS, vol. 5019, pp. 136–149. Springer, Heidelberg (2008)
2. Amiel, F., Feix, B., Villegas, K.: Power Analysis for Secret Recovering and Reverse Engineering of Public Key Algorithms. In: Adams, C., Miri, A., Wiener, M. (eds.) SAC 2007. LNCS, vol. 4876, pp. 110–125. Springer, Heidelberg (2007)
3. Avanzi, R.-M., Cohen, H., Doche, C., Frey, G., Lange, T., Nguyen, K., Verkauteren, F.: Handbook of Elliptic and Hyperelliptic Curve Cryptography (2006)
4. Bevan, R., Knudsen, E.: Ways to Enhance Differential Power Analysis. In: Lee, P.J., Lim, C.H. (eds.) ICISC 2002. LNCS, vol. 2587, pp. 327–342. Springer, Heidelberg (2003)
5. Brier, E., Clavier, C., Olivier, F.: Correlation Power Analysis with a Leakage Model. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 16–29. Springer, Heidelberg (2004)
6. Koç, Ç.K.: Analysis of sliding window techniques for exponentiation. Computers and Mathematics with Applications 30(10), 17–24 (1995)
7. Chevallier-Mames, B., Ciet, M., Joye, M.: Low-cost solutions for preventing simple side-channel analysis: side-channel atomicity. IEEE Transactions on Computers 53(6), 760–768 (2004)
8. Comba, P.G.: Exponentiation cryptosystems on the ibm pc. IBM Syst. J. 29(4), 526–538 (1990)
9. Coron, J.-S.: Resistance against differential power analysis for elliptic curve cryptosystems. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 292–302. Springer, Heidelberg (1999)
10. Dhem, J.-F.: Design of an efficient public-key cryptographic library for RISC-based smart cards. PhD thesis, Université catholique de Louvain, Louvain (1998)
11. Diffie, W., Hellman, M.E.: New Directions in cryptography. IEEE Transactions on Information Theory 22(6), 644–654 (1976)
12. FIPS PUB 186-3. Digital Signature Standard. National Institute of Standards and Technology (October 2009)
13. Fouque, P.-A., Valette, F.: The Doubling Attack - why upwards is better than downwards. In: Walter, C.D., Koç, Ç.K., Paar, C. (eds.) CHES 2003. LNCS, vol. 2779, pp. 269–280. Springer, Heidelberg (2003)
14. Gierlichs, B., Batina, L., Tuyls, P., Preneel, B.: Mutual Information Analysis. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 426–442. Springer, Heidelberg (2008)
15. Homma, N., Miyamoto, A., Aoki, T., Satoh, A., Shamir, A.: Collision-based power analysis of modular exponentiation using chosen-message pairs. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 15–29. Springer, Heidelberg (2008)
16. Joye, M., Yen, S.-M.: The Montgomery Powering Ladder. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 291–302. Springer, Heidelberg (2003)

17. Karatsuba, A.A., Ofman, Y.P.: Multiplication of multidigit numbers on automata. Doklady Akademii Nauk SSSR 45(2), 293–294 (1962)
18. Kocher, P.C.: Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996)
19. Kocher, P.C., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M.J. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
20. Menezes, A., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1996)
21. Messerges, T.S., Dabbish, E.A., Sloan, R.H.: Power analysis attacks of modular exponentiation in smartcards. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 144–157. Springer, Heidelberg (1999)
22. Montgomery, P.L.: Speeding the Pollard and elliptic curve methods of factorization. MC 48, 243–264 (1987)
23. Montgomery, P.L.: Modular multiplication without trial division. Mathematics of Computation 44(170), 519–521 (1985)
24. Prouff, E., Rivain, M.: Theoretical and Practical Aspects of Mutual Information Based Side Channel Analysis. In: Abdalla, M., Pointcheval, D., Fouque, P.-A., Vergnaud, D. (eds.) ACNS 2009. LNCS, vol. 5536, pp. 499–518. Springer, Heidelberg (2009)
25. Standaert, F.-X., Gierlichs, B., Verbauwhede, I.: Partition vs. Comparison Side-Channel Distinguishers: An Empirical Evaluation of Statistical Tests for Univariate Side-Channel Attacks against Two Unprotected CMOS Devices. In: Lee, P.J., Cheon, J.H. (eds.) ICISC 2008. LNCS, vol. 5461, pp. 253–267. Springer, Heidelberg (2009)
26. Walter, C.D.: Sliding Windows Succumbs to Big Mac Attack. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 286–299. Springer, Heidelberg (2001)
27. Walter, C.D.: Longer keys may facilitate side channel attacks. In: Matsui, M., Zuccherato, R.J. (eds.) SAC 2003. LNCS, vol. 3006, pp. 42–57. Springer, Heidelberg (2004)
28. Yen, S.-M., Lien, W.-C., Moon, S., Ha, J.: Power Analysis by Exploiting Chosen Message and Internal Collisions - Vulnerability of Checking Mechanism for RSA-decryption. In: Dawson, E., Vaudenay, S. (eds.) Mycrypt 2005. LNCS, vol. 3715, pp. 183–195. Springer, Heidelberg (2005)