Updated Recommendations for Blinded Exponentiation vs. Single Trace Analysis

Christophe Clavier[1] and Benoit Feix[1,2,⋆]

[1] XLIM-CNRS, Université de Limoges,
Limoges, France
christophe.clavier@unilim.fr

[2] UL Transactions, UK Security Lab
benoit.feix@ul.com

Abstract. Side-channel analysis has become a very powerful tool helpful for attackers trying to recover the secrets embedded in microprocessors such as smartcards. Since the initial publications from Kocher et al. many improvements on side-channel techniques have been proposed. At the same time developers have designed countermeasures to counterfeit those threats. The challenge for securing smart devices remains rough. The most complex techniques like Differential, Correlation and Mutual-information analysis are more studied today than simple side-channel analysis which seems less considered as said less powerful. We revisit in this paper the simple side-channel analysis attacks previously published. Relying on previous leakage models we design two new methods to build chosen message which allows more efficient analysis on blinded exponentiation. We also show that, contrarily to common belief, with our chosen message method simple side-channel analysis can be successful also in some hashed message models. In a second step we introduce a more precise but realistic leakage model for hardware multipliers which leads us to new results on simple side-channel efficiency. Relying on these models we show that even with big base multipliers leakages can be exploited to recover the secret exponent on blinded exponentiations.

Keywords: side-channel analysis, arithmetic coprocessor, long integer algorithms, exponentiation, padding.

1 Introduction

Today secure smart devices such as smartcards or other form factors are widely spread in many different applications. Payment, Pay-TV, health or cell phone SIM cards, etc. Each device contains sensitive issuer and user secret data which must not be disclosed. Many techniques threatening smartcard security exist. The most famous still remains the *Side-Channel Analysis* (SCA). Introduced by Kocher et al. [19,20] in the previous decade, SCA includes the *Simple Side-Channel Analysis* (SSCA) as well as the *Differential Side-Channel Analysis*

⋆ This work was carried out when the author was with Inside Secure.

© Springer-Verlag Berlin Heidelberg 2013

(DSCA). Many studies on these subjects have been published either for improving the attack methods or to present some new countermeasures. The task consisting in developing secure products which must resist many years has then become a difficult challenge when every year new attacks are published.

More precisely public key implementations like RSA [24] and DH [11] essentially consists in modular exponentiations. They are often used nowadays respectively for signature and key exchange schemes. Exponentiation operates on long integers and then requires much more computations than in any symmetric algorithm. Therefore in order to implement it efficiently, various modular multiplication algorithms have been designed to be embedded in constrained hardware resource devices such as smartcards. Although efficiency is a key concern, since the initial publication from Kocher side-channel resistance must also be taken into consideration when developing the code for this operation. Countermeasures must not only resist each and every SCA known so far but must also have the smallest impact in terms of performance and memory consumption.

SSCA on exponentiation has been introduced by Kocher et al. in [20], and one year later improved by Mayer-Sommer in [21]. However even if more complex side-channel techniques like differential and correlation side-channel analysis, the collision attacks (for instance the doubling attack [13]) have been applied on exponentiation, very few publications have dealt with simple side-channel analysis on exponentiation. One of them is the zero value side-channel attack from Goubin [15]. It was originally presented as a differential analysis but works on a single execution trace of an elliptic curve scalar multiplication. Later Yen et al. [26] presented a chosen message SSCA defeating some of the common exponentiation algorithms. Later Courrege et al. [9] improved these results; they showed that random message SSCA can be very powerful even on blinded exponentiation for some cases and gave recommendations for safe developments.

Many countermeasures have been proposed. Common countermeasures to prevent up-to-date SSCA on exponentiation consists in using an exponentiation algorithm where the sequence of modular operations leaks no information on the secret exponent. Examples of such exponentiation are the Montgomery ladder [16], the Joye ladder [17] and the so-called atomic exponentiation [5]. Additionally to prevent leakage on data manipulation, common countermeasure consists in blinding the modulus and the message as well as the exponent [20,7]. Their effect is to randomize the intermediate values manipulated during the exponentiation as well as the sequence of squaring and multiplication operations.

In this paper we denote by *blinded exponentiation* an atomic exponentiation where modulus, message and exponent are blinded and then focus our study on this algorithm. Considering this state-of-the-art implementation we present how to build chosen messages leading to more efficient SSCA when attacking blinded exponentiation on devices and show that, contrarily to common belief, our SSCA can also be successful in some hashed message models. Moreover we introduce a more precise new leakage model for hardware multipliers than [9] we know realistic for practical measurements. We then obtain new results when explaining simple side-channel efficiency for different key length, multipliers and the size of the random used for blinding. We highlight then that, even if the hardware multiplier architecture is a 32-bit one, SSCA can be very efficient with a reasonable number of executions to recover the secret exponent manipulated. We discuss then the need for a deep side-channel characterization of hardware multipliers in order to establish the best recommendations for any hardware multiplier being used for secure products. It would allow developers to select with strong assurance the right countermeasures (and algorithm) when implementing for a selected device any public key algorithm.

Roadmap. The paper is organized as follows. Section 2 reminds basics on long integer arithmetic and exponentiation. We also give the reader the necessary knowledge and background on simple side-channel analysis to understand the attack improvements we are presenting. In Section 3 we describe our new methods to build chosen message attacks and analyze its efficiency in blinded exponentiation use cases. We also show that chosen message SSCA can be efficient on hashed message models. Section 4 introduces our new leakage models for hardware multipliers, we detail the probabilities of leakage for blinded exponentiation depending on long integer bit-lengths. We discuss the need for a deep characterization of hardware multiplier and related countermeasures in Section 5 to finally conclude our paper in Section 6.

2 Simple Side Channel Analysis and Embedded Exponentiation

In this section we give the notations we use in the paper, remind the reader the classical algorithms used to calculate multiplication and exponentiation on long integers. We also remind the necessary knowledge and background on simple side-channel analysis to understand the attack improvements we are presenting.

Definitions and Notations

- $x = (x_{\ell-1} \ldots x_1 x_0)_b$ corresponds to integer x decomposition in base b, i.e. the x decomposition in t-bit words with $b = 2^t$ and $\ell = \lceil log_b(x) \rceil$.
- $\mathsf{LIM}(x,y) = x \cdot y$ long-integer multiplication operation is detailed in the following. Algorithm 2.1 presents the classical long integer multiplication algorithm.
- $\mathsf{BarrettRed}(x,n) = x \bmod n$ using the Barrett reduction method. In this paper we consider reduction operations are done using this algorithm.
- $\mathsf{ModMul}(x,y,n) = x{\cdot}y \bmod n = \mathsf{BarrettRed}(\mathsf{LIM}(x,y),n)$. It is the combination of a long integer multiplication $\mathsf{LIM}(x,y)$ followed by a Barrett reduction by the modulus value n.
- $\mathsf{Exp}(m,d,n) = m^d \bmod n$. Algorithm 2.2 gives more detail on this exponentiation algorithm.

Algorithm 2.1. Long Integer Multiplication

Require: $x = (x_{\ell-1}x_{\ell-2}\dots x_1x_0)_b, y = (y_{\ell-1}y_{\ell-2}\dots y_1y_0)_b$
Ensure: multiplication result $\mathsf{LIM}(x, y) = x \cdot y$

```
1: for $i = 0$ to $2\ell - 1$ do
2:     $w_i \leftarrow 0$
3: for $i = 0$ to $\ell - 1$ do
4:     $c \leftarrow 0$
5:     for $j = 0$ to $\ell - 1$ do
6:         $(uv)_b \leftarrow w_{i+j} + x_j \cdot y_i + c$
7:         $w_{i+j} \leftarrow v$ and $c \leftarrow u$
8:     $w_{i+\ell} \leftarrow c$
9: return $w$
```

2.1 Embedded Exponentiation

We do not detail the Barrett reduction algorithm here, for more details the reader can refer to [1] or [22]. Other techniques can be chosen for processing modular multiplications such as the interleaved multiplication-reduction with Knuth, Sedlak, Quisquater or Montgomery methods [10]. Although we have chosen the Barrett reduction our results can also be adapted to these other methods.

Algorithm 2.2. Exponentiation

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

```
1: $R_0 \leftarrow 1$; $R_1 \leftarrow m$
2: for $i = k - 1$ down to $0$ do
3:     $R_0 \leftarrow \mathsf{ModMul}(R_0, R_0, n)$
4:     if $d_i = 1$ then $R_0 \leftarrow \mathsf{ModMul}(R_0, R_1, n)$
5: return $R_0$
```

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

2.2 Simple Side-Channel Analysis

Side-channel analysis has been studied for years since it was introduced by Kocher et al. [20]. It has then be applied to the most frequently used cryptosystems (DES, AES, RSA, ECDSA,...) and many improvements on those

[1] It can be replaced, as in some standards, by $e \cdot d \equiv 1 \bmod \psi(n)$ where $\psi(n) = \mathrm{lcm}(p-1, q-1)$ is the Carmichael function applied to n.

attack techniques have been done and published during the last decade. We can mention the Correlation Side-Channel Analysis (CSCA), introduced by Brier et al. [4], which requires far fewer traces for recovering the key than the original DPA from Kocher et al. More recently many other studies have been published to improve the side-channel methodology [14,23,25].

The original simple side-channel analysis [20] recovered the secret exponent manipulated in an RSA exponentiation from a single consumption trace. Indeed, when the squaring and the multiplying operations have different recognizable and sizeable patterns the recovery can be done easily because the bits of the secret exponent are directly *read* on the side-channel trace for a classical *Square and Multiply* algorithm. Indeed two consecutive squares on the trace imply the exponent bit is 0 while when a squaring is followed by a multiplication the exponent bit is 1.

The side-channel leakage appears due to differences in the executed code. Indeed it happens when the executed code is different for a squaring than for a multiplication operation. An efficient countermeasure against this SPA is the *side-channel atomicity* introduced by Chevallier-Mames et al. [5] as mentioned previously. In an atomic implementation the code executed during the whole exponentiation loop is the same for a squaring and a multiplication step rendering the attack no more possible.

Yen et al. introduced in [26] a new type of SPA attack defeating the atomicity countermeasure by using as input of exponentiation particular message value $m = n-1$. However classical blinding countermeasure counterfeits this technique. Later Courrege et al. in [9] gave an explanation for the coprocessor leakages when computing an exponentiation, especially for the long integer multiplication operation, and they illustrated their analysis with practical results. They discussed the success of simple side-channel analysis on secure exponentiation with regard to the size of the multiplier, the length of the modulus and the choice done on random values used for blinding. Indeed authors explained that the side-channel leakage appears during the operation $x_i \cdot y_j$ of the long integer multiplication $\mathsf{LIM}(x,y)$. Any operation $x_i \cdot y_j$ has a side-channel consumption related to the number of bit flips of the bit lines manipulated. When one of the operands is null the t-bit multiplication has a lower side-channel consumption than the average one. It is then possible to distinguish in a long integer multiplication when such a value is manipulated.

Blinded Exponentiation. As we said, in this paper our analysis targets a secured state of the art blinded exponentiation. Therefore we include into the previous exponentiation algorithm 2.2 the following countermeasures:

- exponent blinding: the secret exponent d is randomized by $d^\star = d + r \cdot \phi(n)$, with r being a random value and $\phi()$ the Euler totient function[2]. More detail for such exponent blinding when $\psi(n)$ and/or $\phi(n)$ are unknown can be found in [18]. However here the exponent blinding will not have any effect

[2] or $d^\star = d + r \cdot \psi(n)$.

on our analysis since a single trace is used to recover the private exponent and recovering $d^\star$ is equivalent to recovering d.

- additive message blinding: message is randomized additively by the classical countermeasure: $m^\star = m + r_1 \cdot n \bmod r_2 \cdot n$, with r_1 and r_2 being two λ-bit random values. In this case we have $m^\star$ equal to $m + u \cdot n$ with u being a λ-bit value equal to $r_1 \bmod r_2$.
- atomicity principle: the code is implemented following the atomicity principle [5]; it consists in using the same code during the exponentiation whether the operation executed is a squaring or a multiplying operation. Thus it allows protecting from the classical SSCA which consists in distinguishing both types of operations on the side-channel trace.

We obtain the blinded exponentiation detailed in algorithm 2.3. This algorithm implementation is the target of the new simple side-channel analysis we detail in the rest of this paper.

Algorithm 2.3. Blinded exponentiation

```
Require: integers $m$ and $n$ with $m < n$, $\ell \cdot t$-bit exponent $d = (d_{\ell\cdot t-1} d_{\ell\cdot t-2} \ldots d_1 d_0)_2$,
    a security parameter $\lambda$
Ensure: $\mathsf{Exp}(m,d,n) = m^d \bmod n$
 1: $r_1 \leftarrow random(1, 2^\lambda - 1)$
 2: $r_2 \leftarrow random(1, 2^\lambda - 1)$
 3: $R_0 \leftarrow 1 + r_1 \cdot n \bmod r_2 \cdot n$
 4: $R_1 \leftarrow m + r_1 \cdot n \bmod r_2 \cdot n$
 5: $i \leftarrow \ell \cdot t - 1$; $\alpha \leftarrow 0$
 6: while $i \geqslant 0$ do
 7:     $R_0 \leftarrow \mathsf{ModMul}(R_0, R_\alpha, n)$
 8:     $\alpha \leftarrow \alpha \oplus d_i$;
 9:     $i \leftarrow i - 1 + \alpha$
10: return $R_0$
```

3 Improving the Previous Simple Side Channel Analysis on Exponentiation

In [9] authors considered that during a long integer multiplication $R_0 \cdot R_1$, if the multiplicand $R_1 = m$ contains one (or more) of the t-bit words set to 0, it is possible to recognize each time this value m is manipulated all along the exponentiation, i.e. each time the exponent bit is 1.

In that case we say in the following a message m or an operand x are tagged because their manipulation can be distinguished.

Authors considered for their leakage statistical analysis during exponentiation scheme the following side-channel tag model:

$[A_0]$ *Side-channel tag originates when a whole t bit word equals zero in the operand m.*

Notations: We denote by $tag(m^\star)$ the event "the operand m has a t-bit word equal to zero" and by $tag^i(m^\star)$ the event "the operand m has its i^{th} t-bit word equal to zero". For a given ℓ-word operand $x = (x_{\ell-1} \ldots x_1 x_0)_b$ we introduce the following notations:

$$\begin{aligned}\overline{x_i} &= x \bmod b^{i+1} = (x_i \ldots x_1 x_0)_b \\ \underline{x_i} &= x \bmod b^i = (x_{i-1} \ldots x_1 x_0)_b \quad \text{with} \quad \underline{x_0} = 0\end{aligned}$$

The general principle of the attack is based on the fact that whenever the randomized message $m^\star$ is tagged, this easily detectable event points the attacker to all LIM operations corresponding to multiplications by the message, which thus reveals the private exponent $d^\star$. The probability for a tag to occur is usually quite small so that the attacker has to acquire and analyze many side-channel traces until one of them eventually happens to be tagged.

3.1 Improving the Analysis

A first contribution of this paper is to observe that an attacker who has control of the non randomized message m is able to further reduce the attack complexity – measured as the number of required side-channel traces – by causing tags on the randomized message $m^\star$ to happen more frequently than by pure chance. More precisely, for any word index $0 \leqslant i < \ell$, and for any integer $0 \leqslant u^{(i)} < 2^\lambda - 1$ which denotes a targeted value for the random $u = r_1 \bmod r_2$ involved in the randomization of m, it is possible to find a message m such that $m^\star = m + u^{(i)} \cdot n$ is tagged on word i. This chosen message gives access to the private exponent whenever $u = u^{(i)}$ which may be more probable than would naturally arise, particularly when $\lambda < t$.

We can even do better since we will show that it is possible to build a message which simultaneously verifies such kind of conditional tag property on each of its words. Then in a second study we consider the scenario where the attacker does not have full control on the message which is to be randomized since we assume that this message is the output of a deterministic hash function whose input is chosen by the attacker.

3.2 Known Message Scenario

We assume here a known message scenario where the message value to be exponentiated is uniformly distributed over the set of all integers that can be represented on ℓt bits.

Theorem 1. *Given a message m uniformly distributed over $\{0, \ldots, 2^{\ell t} - 1\}$, the probability that the randomized message $m^\star = m + u \cdot n$ is tagged on any of its ℓ least significant words is:*

$$\begin{aligned}\text{Proba}\big(tag(m^\star)\big) &= 1 - (1 - 2^{-t})^\ell \\ &\simeq \ell \cdot 2^{-t}\end{aligned}$$

Proof. For any $0 \leqslant i < \ell$, and any arbitrary integer $0 \leqslant u < 2^\lambda - 1$, letting $s = u \cdot n$, we have:

$$\begin{aligned}\mathrm{Proba}_m\big(tag^{(i)}(m^\star)\big) &= \mathrm{Proba}_m\left(m_i^\star = 0\right)\\ &= \mathrm{Proba}_m\left(m_i = -\left\lfloor \frac{\overline{s_i} + \underline{\underline{m_i}}}{b^i} \right\rfloor \bmod b\right)\\ &= 2^{-t}\end{aligned}$$

Now, considering also u as random:

$$\begin{aligned}\mathrm{Proba}_{u,m}\big(tag^{(i)}(m^\star)\big) &= \sum_u \mathrm{Proba}(u)\mathrm{Proba}_m\big(tag^{(i)}(m^\star)\big)\\ &= 2^{-t}\end{aligned}$$

The proof follows immediately from the independence of the tag on each word. □

In the known message only setting, the probability for a side-channel trace to be tagged is close to $\ell\, 2^{-t}$. This result holds whatever the probability distribution of u. In particular it makes no difference whether u is biased – which is the case when r_1 and r_2 are both random – or uniformly distributed.

3.3 Chosen Message Scenario

Theorem 2 and Algorithm 3.1 show how an attacker can build a message whose randomization will be tagged whenever u belongs to a set of ℓ prescribed chosen target values.

Theorem 2. *Let $U = \big(u^{(0)}, \dots, u^{(\ell-1)}\big)$ be an arbitrary set of ℓ targets, with $\forall i, 0 \leqslant u^{(i)} < 2^\lambda - 1$. The message m returned by Algorithm 3.1 is such that $m^\star = m + u \cdot n$ is tagged on word i whenever $u = u^{(i)}$.*

Proof. For each i, let $s^{(i)} = u^{(i)} \cdot n$. We have

$$m_i = -\left\lfloor \frac{\overline{s_i^{(i)}} + \underline{\underline{m_i}}}{b^i} \right\rfloor \bmod b$$

so that $(m + s^{(i)})_i = 0$ which implies that $m_i^\star = 0$ if $u = u^{(i)}$. □

We now compute the probability that a randomization of the message returned by Algo. 3.1 is tagged:

$$\begin{aligned}\mathrm{Proba}\big(tag^{(i)}(m^\star)\big) &= \mathrm{Proba}(u = u^{(i)}) \cdot 1 + \mathrm{Proba}(u \neq u^{(i)}) \cdot 2^{-t}\\ &\simeq \mathrm{Proba}(u = u^{(i)}) + 2^{-t}\\ &\simeq \begin{cases} 2^{-t} & \text{if } \lambda > t\\ 2^{-\lambda} & \text{if } \lambda \leqslant t\end{cases} \qquad (1)\\ &\simeq \max(2^{-\lambda}, 2^{-t})\end{aligned}$$

Algorithm 3.1. Chosen message construction

```
Require: a $\ell$-word modulus $n$ and a set $\left(u^{(0)}, \ldots, u^{(\ell-1)}\right)$ of targeted randoms
Ensure: a message $m$ whose randomization is tagged for any specified target
1: $m \leftarrow 0$
2: for $i = 0$ to $\ell - 1$ do
3:     $s^{(i)} \leftarrow u^{(i)}\, n$
4:     $\mu \leftarrow -\left\lfloor \frac{\overline{s_i^{(i)} + m_i}}{b^i} \right\rfloor \bmod b$
5:     $m \leftarrow m + \mu\, b^i$
6: return $m$
```

Equation 1 clearly shows that our chosen message method is particularly interesting when $\lambda \leqslant t$. Indeed, when $\lambda > t$ the randomized message is tagged with same probability than in the known message model. For this reason we consider from now on that $\lambda \leqslant t$. In that case choosing the message according to Algo. 3.1 changes the complexity of tag probability from $\mathcal{O}(2^{-t})$ to $\mathcal{O}(2^{-\lambda})$. Depending on λ, the attack may now be feasible even on large multipliers (e.g. $t \geqslant 64$) as the tag probability does not depend on t any more.

When u has uniform distribution the choice of the $u^{(i)}$s is not relevant provided they are all distinct. In that case we have:

$$\text{Proba}\big(tag(m^\star)\big) \;\simeq\; \ell\, 2^{-\lambda}$$

When u is biased due to the random choice of both r_1 and r_2 the smaller u the more probable it is. The best strategy for an attacker is then to choose $U = (0, \ldots, \ell - 1)$ which has the largest probability. This results in a tag probability that can be expressed as:

$$\begin{aligned}\text{Proba}\big(tag(m^\star)\big) &\;\simeq\; \text{Proba}(u \in U)\\ &\;\simeq\; \omega\, \ell\, 2^{-\lambda}\end{aligned}$$

where $\omega \geqslant 1$ is a multiplicative factor which quantifies the gain related to the biased case compared to the uniform one.

Let's now enumerate the three advantages from which our chosen message attack benefits:

1. Considering some given word of the randomized message, the probability that it is tagged is at least $2^{-\lambda}$ instead of 2^{-t} (for $\lambda \leqslant t$). This is by far the more fundamental advantage provided by our method.
2. As it is possible to simultaneously generate a conditional tag on all words, the probability of a tag on $m^\star$ is l times that of a tag on a single word. Note that this gain by a factor l also holds in the known message model.
3. In case of biased randomization – which is more usually implemented than the uniform randomization – the attacker targets the most probable random values u. This results in another gain by a factor ω which is far from being negligible as shown in Table 1.

Experimental Results. For different sets of parameters t, λ we have simulated our attack on a large number of runs by generating a random 1024-bit modulus n, building a message m according to Algo. 3.1, computing a randomized message $m^\star$ by applying the classical biased masking procedure, and testing whether $m^\star$ is tagged.

We present in Table 1 the experimental averaged tag probabilities, together with the theoretical ones for comparison. We also mention the resulting mean number of side-channel traces needed, the gain factor ω, as well as the number of simulation runs in each case.

Table 1. Simulation results of the chosen message attack for a 1024-bit RSA key with biased randomization

		Tag probability		Number of traces		Gain ω	
		Simu	Theory	Simu	Theory	Simu	Theory
$\lambda = 8$ (10^6 runs)	$t = 16$	$6.50\,10^{-1}$	$6.51\,10^{-1}$	1.54	1.54	2.60	2.60
	$t = 32$	$4.28\,10^{-1}$	$4.28\,10^{-1}$	2.33	2,33	3.43	3.43
	$t = 64$	$2.63\,10^{-1}$	$2.62\,10^{-1}$	3.80	3.81	4.21	4.20
$\lambda = 16$ (10^7 runs)	$t = 16$	$8.30\,10^{-3}$	$8.30\,10^{-3}$	121	121	8.50	8.50
	$t = 32$	$4.49\,10^{-3}$	$4.48\,10^{-3}$	223	223	9.19	9.18
	$t = 64$	$2.42\,10^{-3}$	$2.41\,10^{-3}$	414	415	9.89	9.86
$\lambda = 24$ (10^8 runs)	$t = 16$	—	—	—	—	—	—
	$t = 32$	$2.77\,10^{-5}$	$2.81\,10^{-5}$	36062	35590	14.5	14.7
	$t = 64$	$1.48\,10^{-5}$	$1.47\,10^{-5}$	67476	68049	15.5	15.4
$\lambda = 32$ (10^9 runs)	$t = 16$	—	—	—	—	—	—
	$t = 32$	—	—	—	—	—	—
	$t = 64$	$8.3\,10^{-8}$	$7.78\,10^{-8}$	$12.0\,10^6$	$12.8\,10^6$	22.3	20.9

From a practical point of view, the proposed chosen message method allows our tag-based simple side-channel analysis on randomized exponentiation to be feasible in a much wider range of settings. Definitely, the security against our attack cannot be provided by a large multiplier. Also, Table 1 shows that the mean number of traces required to recover the private exponent is small for $\lambda = 16$ and quite practicable for $\lambda = 24$, while these random bit-length values may be considered providing enough security for message blinding purpose. In light of our method, we can say that message blinding must not use random values smaller than 32 bits.

3.4 Hashed Message Scenario

In this section, we consider a more restricted model where the message is hashed and padded before being randomized and then exponentiated. We still assume that the message m is chosen by the attacker, but the aim is now to obtain a tag on $h^\star$ where:

$$\begin{cases} h &= \mathcal{H}(m) \\ h^\star &= h + u \cdot n \end{cases}$$

We assume that $\mathcal{H}$ is a deterministic hash and pad function – e.g. the full domain hash RSA-FDH [2]. Because we do not have control on the hash output, it is not possible to directly set some word of h to that precise value which would create a tag for some given targeted u. Rather we can try to search for some m whose hash has this property. Suppose we want to tag the least significant word of $h^\star$. In order for that word to be tagged for a prescribed target u, we must find a message m such that $h_0 = -s_0 \bmod b$ with $s = u \cdot n$. This allows the attack to necessitate only $\mathcal{O}(2^\lambda)$ side-channel traces as in the chosen message model, but requires an average of $\mathcal{O}(2^t)$ hash computations.

We can do better if we allow any u value to be targeted. Let $S_0 = \{s_0 = (u \cdot n)_0\}$ where $0 \leqslant u < 2^\lambda - 1$. Then we only have to find a message such that $-h_0 \in S_0$. Provided that $\lambda \leqslant t$, the number of distinct values in S_0 is close to 2^λ and the search for a convenient message requires $\mathcal{O}(2^{t-\lambda})$ hash computations and $\mathcal{O}(t\,2^\lambda)$ space storage. We thus found a (time : memory : data) tradeoff – where *data* means the number of side-channel traces required – which achieves $(2^{t-\lambda} : t\,2^\lambda : 2^\lambda)$ complexity.

A further improvement consists in allowing the tag to appear on any word. Defining

$$S = \bigcup_{i=0}^{\ell-1} S_i \quad \text{where } S_i = \{s_i = (u \cdot n)_i\}$$

we now have about $\ell\,2^\lambda$ elements in S so that the tradeoff complexity becomes $(2^{t-\lambda}/\ell : \ell\, t\,2^\lambda : 2^\lambda)$.

This proposed hashed message attack admits three drawbacks compared to the chosen message one:

1. We do not see any means to simultaneously target different u on different words. As a consequence the number of traces required does not benefit from the division by ℓ.
2. Also it seems impossible to provoke a tag for a prescribed u – except if we accept a time complexity $\mathcal{O}(2^{-t})$ instead of $\mathcal{O}(2^{t-\lambda})$. Thus, the number of traces required is not divided by the gain factor ω.
3. The method requires the pre-computation of $\mathcal{O}(2^{t-\lambda})$ hash values and the storage of $\ell\,t\,2^\lambda$ bits.

Despite these drawbacks we think that there are some settings for which the proposed hashed message method can be practically applied while the known message one would be infeasible. For instance when $t = 32$ and $\lambda = 16$ the attack needs 2^{16} traces and a short pre-computation phase, while it would require 2^{29} traces in the known message model to break a 1024-bit key.

Note that the method described in this section seems restricted to the use of a deterministic padding. It is an open question whether it could be modified to apply also to probabilistic padding schemes such as RSA-PFDH [8] or RSA-PSS [3].

Those analysis exploits the well-known efficient leakage model $[A_0]$ to design an SSCA efficient chosen message technique which improves the previous results and to propose a hashed message attack. In the following we consider now a relaxed model leakage. Indeed it is also realistic to consider less restrictive leakage models for a side-channel tag to appear in a multiplication calculation. With these new leakage models we give new results that highlight SSCA is still more efficient than said previously to defeat state of the art blinded exponentiations.

4 Relaxed Side-Channel Leakage Model

We assume here a tag in a message could be due to two following assumptions that are not independent:

$[A_1]$ *Side-channel tag originates from the fact that at least τ consecutive bits in a t-bit word of m are set to* 0, *with $\tau \leqslant t$.*

$[A_2]$ *Side-channel tag originates from the fact that the Hamming weight h of the t-bit word is lower than a value ν, with $h \leqslant \nu < t$.*

Both assumptions $[A_1]$ and $[A_2]$ are realistic and well suited for hardware implementations of multipliers. The choice of the most relevant model between $[A_1]$ and $[A_2]$ and the best values of parameters τ and ν varies from one integrated circuit to another one, it also depends on t. From our experiments we observed that some integrated circuits are more resistant than others.

In this sequel we separately consider the two leakage models given by both assumptions $[A_1]$ and $[A_2]$[3].

We say that x is A_1-tagged on word i whenever x_i contains at least τ consecutive zero bits. This event will be denoted by $tag^{(i)}_{A_1}(x)$. We also denote by $tag_{A_1}(x)$ the event that x is A_1-tagged on at least one of its words.

In the same way, we say that x is A_2-tagged on word i whenever the Hamming weight of x_i is less than ν, and this event will be denoted by $tag^{(i)}_{A_2}(x)$. We also denote by $tag_{A_2}(x)$ the event that x is A_2-tagged on at least one of its words.

In the following let's denote by p the probability for a t-bit word to be either A_1-tagged or A_2-tagged depending on the considered leakage model.

Theorem 3. *Given a message m uniformly distributed over $\{0, \ldots, 2^{\ell t} - 1\}$, the probability that the randomized message $m^\star = m + u \cdot n$ is tagged on any of its ℓ least significant words is:*

$$\begin{aligned}\text{Proba}\big(tag(m^\star)\big) &= 1 - (1-p)^\ell \\ &\simeq \ell \cdot p\end{aligned}$$

4.1 Tag Probabilities for τ and t Values with $[A_1]$ Leakage Model

Considering the leakage model $[A_1]$ we have computed the different p values for all τ values in the range $[0, \ldots t]$. We have then exhausted the number n_τ of

[3] $[A_0]$ leakage model is a particular case of model $[A_1]$ (resp. $[A_2]$) when τ equals t (resp. when ν is null).

existing words which have their longest consecutive zeros sequence being of exact length τ. Knowing this number we compute $p_1(t,\tau)$ the probability for a t-bit word to have its longest consecutive zero sequence to be exactly τ: $p_1(t,\tau) = n_\tau/(2^t)$. Then we have Proba$(tag_{A_1}^{(i)}(x)) = \sum_{j=\tau}^{t} p_1(t,j)$. Once we obtain these different $tag_{A_1}^{(i)}(x)$ values we compute the $tag_{A_1}(m)$ probabilities for 512, 1024 and 2048 bits long integer messages.

Case $t = 16$. Table 2 gives result examples for a t = 16-bit multiplier architecture.[4]

Table 2. $[A_1]$ Leakage probability examples for some τ values when $t = 16$

τ	t-bit word number	$p_1(t,\tau)$	$P(tag_{A_1}^{(i)}(x))$	$P(tag_{A_1}(m_{512}))$	$P(tag_{A_1}(m_{1024}))$	$P(tag_{A_1}(m_{2048}))$
0	1	$1.53\,10^{-05}$	1	1	1	1
4	13008	$1.98\,10^{-01}$	$3.95\,10^{-01}$	1	1	1
8	704	$1.07\,10^{-02}$	$1.95\,10^{-02}$	$4.68\,10^{-01}$	$7.17\,10^{-01}$	$9.20\,10^{-01}$
12	28	$4.27\,10^{-04}$	$7.32\,10^{-04}$	$2.32\,10^{-02}$	$4.58\,10^{-02}$	$8.95\,10^{-02}$
16	1	$1.53\,10^{-05}$	$1.53\,10^{-05}$	$4.88\,10^{-04}$	$9.76\,10^{-04}$	$1.95\,10^{-03}$

Considering for instance the case $\tau = 12$, we observe there are 28 words which have their longest consecutive zeros sequence being of length 12. The probability for a word to be exactly τ bit A_1 tagged is $p_1(16,12) = 4.27\,10^{-4}$. The probability for a word to have at least $\tau = 12$ consecutive zero bits is then Proba$(tag_{A_1}^{(i)}(x)) = \sum_{i=12}^{16} p_1(16,i) = 7.32\,10^{-4}$.

It is then worth to notice the probability a 1024-bit integer is tagged is reduced from $9.76\,10^{-4}$ to $4.58\,10^{-2}$ from model $[A_0]$ to model $[A_1]$ with $\tau = 12$ which can happen in practice. It means that only 22 ($\approx 1/(4.58\,10^{-2})$) messages would be enough for recovering the secret exponent in a 1024-bit blinded exponentiation with probability $1/e \approx 0.63$ instead of 1020 messages when considering $[A_0]$. Finally to reach a leakage probability equal to 0.999 SSCA would require only 140 messages and not 6700 when considering the previous leakage model $[A_0]$.

Case $t = 32$. We processed the same study for a 32-bit multiplier. Table 3 gives result examples.

In [9] authors considered that using a 32-bit multiplier counterfeited simple side-channel analysis in blinded exponentiation when random used for blinding were big enough (i.e. $\geqslant 32$ bits). We observe here than it is not exact considering the relaxed but realistic model $[A_1]$. Indeed considering τ equal to 16 we obtain Proba$(tag_{A_1}^{(i)}(x)) = 1.37\,10^{-4}$, it signifies Proba$(tag_{A_1}(m)) = 4.39\,10^{-3}$ for m a 1024-bit integer message. It means that 230 messages would be enough for

[4] The complete result tables of our analysis for $[A_1]$ and $[A_2]$ models, considering all possible τ and ν values in the range $[0,\ldots, t]$ are given in the extended version of this paper [6].

Table 3. $[A_1]$ Leakage probability examples for some τ values when $t = 32$

τ	t-bit word number	$p_1(t,\tau)$	$P(tag_{A_1}^{(i)}(x))$	$P(tag_{A_1}(m_{512}))$	$P(tag_{A_1}(m_{1024}))$	$P(tag_{A_1}(m_{2048}))$
0	1	$2.33\,10^{-10}$	1	1	1	1
8	111246728	$2.59\,10^{-02}$	$5.02\,10^{-02}$	$5.61\,10^{-01}$	$8.08\,10^{-01}$	$9.63\,10^{-01}$
16	311296	$7.25\,10^{-05}$	$1.37\,10^{-04}$	$2.20\,10^{-03}$	$4.39\,10^{-03}$	$8.75\,10^{-03}$
24	704	$1.64\,10^{-07}$	$2.98\,10^{-07}$	$4.77\,10^{-06}$	$9.54\,10^{-06}$	$1.91\,10^{-05}$
32	1	$2.33\,10^{-10}$	$2.33\,10^{-10}$	$3.73\,10^{-09}$	$7.45\,10^{-09}$	$1.49\,10^{-08}$

recovering the secret exponent in a 1024-bit blinded exponentiation with probability $1/e \approx 0.63$ instead of $1.34\,10^8$ messages when considering $[A_0]$. Moreover to reach a leakage probability equal to 0.999 only 1480 messages are required instead of $8.73\,10^8$.

We have studied the leakage probabilities for exponentiation with the $[A_1]$ model. Our analysis highlights the risk of SSCA leakage even when the hardware multiplier base size is big, for instance 32-bit contrarily to previous paper results. In the following we reproduce the same study for the second ($[A_2]$) model leakage.

4.2 Tag Probabilities for ν and t Values with $[A_2]$ Leakage Model

The number of t-bit words which have their Hamming weight being μ is $\binom{\mu}{t}$. The probability for a t-bit word to have its Hamming weight being exactly μ is $p_2(t,\mu) = \binom{\mu}{t} \cdot 2^{-t}$. Thus we obtain the probability for a t-bit word to be ν $[A_2]$ tagged is:

$$\text{Proba}(tag_{A_2}^{(i)}(x)) = \frac{\sum_{\mu=0}^{\nu} \binom{\mu}{t}}{2^t}. \tag{2}$$

Using this simple formula we compute in the following the values $\text{Proba}(tag_{A_2}^{(i)}(x))$ and $\text{Proba}(tag_{A_2}(m))$ for t=16 and $t = 32$ bits multipliers and different message bit-length.

Case $t = 16$. Table 4 gives results examples of $\text{Proba}(tag_{A_2}^{(i)}(x))$ and Proba $(tag_{A_2}(m))$ for t=16.

Table 4. $[A_2]$ Leakage probability for some ν values when $t = 16$

ν	t-bit word number	$p_2(t,\nu)$	$P(tag_{A_2}^{(i)}(x))$	$P(tag_{A_2}(m_{512}))$	$P(tag_{A_2}(m_{1024}))$	$P(tag_{A_2}(m_{2048}))$
0	1	$1.53\,10^{-05}$	$1.53\,10^{-05}$	$7.78\,10^{-03}$	$1.55\,10^{-02}$	$3.08\,10^{-02}$
2	120	$1.83\,10^{-03}$	$2.08\,10^{-03}$	$6.43\,10^{-02}$	$1.24\,10^{-01}$	$2.33\,10^{-01}$
4	1820	$2.78\,10^{-02}$	$3.84\,10^{-02}$	$7.14\,10^{-01}$	$9.18\,10^{-01}$	$9.93\,10^{-01}$
8	12870	$1.96\,10^{-01}$	$5.98\,10^{-01}$	1	1	1
12	1820	$2.78\,10^{-02}$	$9.89\,10^{-01}$	1	1	1
16	1	$1.53\,10^{-05}$	1	1	1	1

Considering for instance the case $\nu = 2$, the probability a 1024-bit integer is tagged is $\text{Proba}(tag_{A_2}(m_{1024}) = 1.24\,10^{-1}$. It signifies that only 8 messages would be enough for recovering the secret exponent in a 1024-bit blinded exponentiation with a probability of success equal to $1/e \approx 0.63$. Finally to reach a leakage probability equal to 0.999 SSCA it would require only 49 messages (exponentiation executions).

Case $t = 32$. We processed the same study for a 32-bit multiplier.

Table 5. $[A_2]$ Leakage probability for some ν values when $t = 32$

ν	t-bit word number	$p_2(t,\nu)$	$P(tag^{(i)}_{A_2}(x))$	$P(tag_{A_2}(m_{512}))$	$P(tag_{A_2}(m_{1024}))$	$P(tag_{A_2}(m_{2048}))$
0	1	$2.33\,10^{-10}$	$2.33\,10^{-10}$	$3.73\,10^{-09}$	$7.45\,10^{-09}$	$1.49\,10^{-08}$
4	35960	$8.37\,10^{-06}$	$9.65\,10^{-06}$	$1.54\,10^{-04}$	$3.09\,10^{-04}$	$6.17\,10^{-04}$
8	10518300	$2.45\,10^{-03}$	$3.50\,10^{-03}$	$5.46\,10^{-02}$	$1.06\,10^{-01}$	$2.01\,10^{-01}$
16	601080390	$1.40\,10^{-01}$	$5.70\,10^{-01}$	1	1	1
24	10518300	$2.45\,10^{-03}$	$9.99\,10^{-01}$	1	1	1
32	1	$2.33\,10^{-10}$	1	1	1	1

We consider here a device where the power leakage appears for this $[A_2]$ model when $\nu = 4$, we know by experiments it is a realistic case. The probability a 1024-bit integer is tagged becomes Proba$(tag_{A_2}(m_{1024})) = 3.09\,10^{-4}$. It means that only $3.24\,10^3$ messages would be enough for recovering the secret exponent in a 1024-bit blinded exponentiation with probability $1/e \approx 0.63$ instead of $1.34\,10^8$ when considering the $[A_0]$ model. Moreover to reach a leakage probability equal to 0.999 $2.1\,10^4$ messages are required instead of $8.73\,10^8$.

Synthesis. We have discussed the probability of SSCA leakage for the two relaxed models $[A_1]$ and $[A_2]$ we have introduced. We have shown that the previous model $[A_0]$ is too restrictive and that even for big size multipliers like 32-bit ones it is possible with a reasonable number of executions to recover the private exponent in a blinded exponentiation.

To illustrate our results we gives in Table 6 different leakage probabilities for different models we consider realistic. Of course this table is an example and each integrated circuit will have different leakage characteristic. It is then important to measure the right values τ and ν for each integrated circuit.

Table 6. Leakage probability examples for t=32

$\tau,\ \nu$	t-bit word number	p	$P(tag_{A_i}(m_{512}))$	$P(tag_{A_i}(m_{1024}))$	$P(tag_{A_i}(m_{2048}))$
$[A_2]\ \nu = 4$	$8.37\,10^{-06}$	$9.65\,10^{-06}$	$1.54\,10^{-04}$	$3.09\,10^{-04}$	$6.17\,10^{-04}$
$[A_1]\ \tau = 16$	$7.25\,10^{-05}$	$1.37\,10^{-04}$	$2.20\,10^{-03}$	$4.39\,10^{-03}$	$8.75\,10^{-03}$
$[A_0]$	$2.33\,10^{-10}$	$2.33\,10^{-10}$	$3.73\,10^{-09}$	$7.45\,10^{-09}$	$1.49\,10^{-08}$

It is important to notice that SSCA is much more efficient than previous studies said and particularly can threaten blinded exponentiation implemented with 32-bit cores which are commonly used today. Of course it depends on the

Table 7. Number of messages/executions needed for leakage probability at $0,999$, for t=32

$\tau,\ \nu$	m_{512}	m_{1024}	m_{2048}
$[A_2]\ \nu = 4$	$4.22\,10^4$	$2.11\,10^4$	$1.06\,10^3$
$[A_1]\ \tau = 16$	$3\,10^3$	$1.5\,10^3$	750
$[A_0]$	$1.75\,10^9$	$8.73\,10^8$	$4.37\,10^8$

kind of hardware selected for the implementation, it is then very important to measure the exact side-channel leakage of the multiplier, for instance the exact values τ and ν in our two models.

5 Countermeasures and Recommendations

We have shown previously that some mandatory conditions must be respected to prevent any implementations from the enhanced simple side-channel analysis.

Hardware Multiplier Characterization. The first consideration to take into account consists in precisely characterizing the leakage characteristics of any designed hardware multiplier. Effectively, contrarily to [9], we have shown that for a t-bit hardware multiplier the leakage probability does not depend only of t but more of the values τ and ν we described previously. It is of particular interest when $t = 32$ as previous studies considered using such a hardware multiplier hardware rendered the SSCA not available if the blinded exponentiation was using big enough random values. But we have shown that it is not true. Indeed, whatever the random size used for blinding and base t value are, if the value ν (resp. τ) is much smaller than t (much bigger than 0) then SSCA can defeat a state of the art blinded exponentiation. It is then important to determine for leakage models $[A_1]$ and $[A_2]$ the values τ and ν leading to a power tag of the selected multiplier in order to determine the exact power leakage of an exponentiation. Once these exact values are determined a developer can select the appropriate algorithm and countermeasure(s) he must use (or not use) for his implementation to be secure enough.

Moreover it is obvious that hardware countermeasures such as jitter, clock divider or in best cases balanced consumption circuits should be also present in the embedded product to enforce the resistance to side-channel analysis and render enhanced SSCA more difficult.

The previous study recommendation still applies: "λ (random bit-length) value must be bigger than 32 bits whatever the value of τ and ν still applies. It is also still recommended to use a constant (rather than random) value for r_2. For instance r_2 could be equal to $2^\lambda - 1$."

Exponentiation Algorithm Choice. For better resistance we recommend to select an exponentiation algorithm resistant to this analysis. The best solution to us consists in always using right-to-left blinded exponentiation algorithm instead of left-to-right classical ones. As already highlighted by Fouque et al. in [13] this implementation is much more resistant to the many side-channel attacks than the left-to-right ones. Indeed the square operations being applied on the message value the operands used in multiplications are never the same and it is not possible any more to observe tags on any message multiplication in a same trace. Developers can also decide to apply a new message randomization on the message operand m used in exponentiation after each multiplying (squaring and multiplying) operation, for instance by using for message the new value $m = m + n \mod r_2 \cdot n$. It is also interesting to notice than in case of Barrett or Montgomery reduction methods, the resistant reduction algorithms given in [12] offers a good protection.

6 Conclusion

We have presented some SSCA improvements enhancing simple side-channel analysis to recover of the secret exponent manipulated during state of the art blinded embedded exponentiations, when all the other side-channel techniques are inefficient. We have also demonstrated how to build a chosen message more significantly to reduce the number of needed execution for SSCA attack to succeed with a higher probability. Moreover we have shown that, contrarily to a common belief, simple side-channel analysis can be successful also in some hashed message models. Our results depend on the size of the random values used for blinding and the way they are generated, as well as on the hardware multiplier leakage properties. We have also presented two new side-channel leakage models we consider realistic and well suited for long integer multiplications and exponentiation side-channel analysis. We observe that SSCA remains a very powerful side-channel analysis to defeat blinded exponentiation even when using big random values and big multipliers. Indeed it requires a deep characterization of the hardware multiplier used.

Our new analysis strengthens again the advice previously given by Fouque and Valette at CHES 2003: *"Upwards is better than downwards!"*. Although less often used than left-to-right exponentiation, right-to-left methods appear to be much more resistant against the numerous side-channel attacks.

References

1. Avanzi, R.-M., Cohen, H., Doche, C., Frey, G., Lange, T., Nguyen, K., Verkauteren, F.: Handbook of Elliptic and Hyperelliptic Curve Cryptography (2006)
2. Bellare, M., Rogaway, P.: Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In: ACM Conference on Computer and Communications Security, pp. 62–73 (1993)
3. Bellare, M., Rogaway, P.: The Exact Security of Digital Signatures - How to Sign with RSA and Rabin. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 399–416. Springer, Heidelberg (1996)

4. 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)
5. 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)
6. Clavier, C., Feix, B.: Updated recommendations for blinded exponentiation vs. single trace analysis - extended version. IACR Cryptology ePrint Archive (2013)
7. 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)
8. Coron, J.-S.: Optimal Security Proofs for PSS and Other Signature Schemes. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 272–287. Springer, Heidelberg (2002)
9. Courrège, J.-C., Feix, B., Roussellet, M.: Simple Power Analysis on Exponentiation Revisited. In: Gollmann, D., Lanet, J.-L., Iguchi-Cartigny, J. (eds.) CARDIS 2010. LNCS, vol. 6035, pp. 65–79. Springer, Heidelberg (2010)
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. Dupaquis, V., Venelli, A.: Redundant modular reduction algorithms. In: Prouff, E. (ed.) CARDIS 2011. LNCS, vol. 7079, pp. 102–114. Springer, Heidelberg (2011)
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. Goubin, L.: A refined power-analysis attack on elliptic curve cryptosystems. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 199–210. Springer, Heidelberg (2002)
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. Joye, M.: Highly regular m-ary powering ladders, pp. 135–147
18. Joye, M.: Protecting RSA against fault attacks: The embedding method. In: Breveglieri, L., Koren, I., Naccache, D., Oswald, E., Seifert, J.-P. (eds.) Sixth International Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2009, pp. 41–45. IEEE Computer Society (2009)
19. 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)
20. Kocher, P.C., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
21. Mayer Sommer, R.: Smartly Analyzing the Simplicity and the Power of Simple Power Analysis on Smartcards, pp. 78–92
22. Menezes, A., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press (1996)

23. 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)
24. Rivest, R.L., Shamir, A., Adleman, L.: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM 21, 120–126 (1978)
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. Yen, S.-M., Lien, W.-C., Moon, S.-J., Ha, J.C.: 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)