Correlated Extra-Reductions Defeat Blinded Regular Exponentiation

Margaux Dugardin[1,2(✉)], Sylvain Guilley[2,3], Jean-Luc Danger[2,3], Zakaria Najm[4], and Olivier Rioul[2,5]

[1] CESTI, Thales Communications and Security, 31000 Toulouse, France
[2] LTCI, CNRS, Télécom ParisTech, Université Paris-Saclay, 75013 Paris, France
{margaux.dugardin,sylvain.guilley,jean-luc.danger,
olivier.rioul}@telecom-paristech.fr
[3] Secure-IC SAS, 35510 Cesson-Sévigné, France
{sylvain.guilley,jean-luc.danger}@secure-ic.com
[4] ST-Microelectronics, 13790 Rousset, France
zakaria.najm@st.com
[5] CMAP, Ecole Polytechnique, Université Paris-Saclay, 91128 Palaiseau, France
olivier.rioul@polytechnique.edu

Abstract. Walter and Thomson (CT-RSA '01) and Schindler (PKC '02) have shown that extra-reductions allow to break RSA-CRT even with message blinding. Indeed, the extra-reduction probability depends on the type of operation (square, multiply, or multiply with a constant). Regular exponentiation schemes can be regarded as protections since the operation sequence does not depend on the secret.

In this article, we show that there exists a strong negative correlation between extra-reductions of two consecutive operations, provided that the first feeds the second. This allows to mount successful attacks even against blinded asymmetrical computations with a regular exponentiation algorithm, such as Square-and-Multiply Always or Montgomery Ladder. We investigate various attack strategies depending on the context—known or unknown modulus, known or unknown extra-reduction detection probability, etc.—and implement them on two devices: a single core ARM Cortex-M4 and a dual core ARM Cortex M0-M4.

Keywords: Side-channel analysis · Montgomery modular multiplication · Extra-reduction leakage · Message blinding · Regular exponentiation

1 Introduction

State of the Art of Timing Attacks. Any cryptographic algorithm in an embedded system is vulnerable to side-channel attacks. Timing attacks on the RSA Straightforward Method (RSA-SFM) were pioneered by Kocher [12]. The attack consists in building "templates" whose distributions are compared to that of the response. It is required that the cryptographic parameters be known since the attack is profiled.

B. Gierlichs and A.Y. Poschmann (Eds.): CHES 2016, LNCS 9813, pp. 3–22, 2016.
DOI: 10.1007/978-3-662-53140-2_1

Schindler [16] extended timing attacks to RSA with Chinese Remainder Theorem (RSA-CRT) using chosen messages. This attack exploits a conditional extra-reduction at the end of modular multiplications. Schindler and co-authors carried out numerous improvements [1,2,17–20] in the case where the exponentiation uses windows or exponent randomization.

Walter and Thompson [21] remarked that even when data is blinded, the distribution of extra-reductions is different for a square and for a multiply. They assumed that side-channel measurements such as power or timing during exponentiation are sufficiently clean to detect the presence or absence of an extra-reduction at each individual operation. Schindler [17] improved this attack by also distinguishing multiplications by a constant from squarings and multiplications by non-fixed parameters.

Today's Solutions. In order to protect the implementation from the above attacks, a first solution consists in exponent randomization on top of message blinding. Such a protection, however, is sensitive to carry leakage [9] and amenable to other attacks like simple power analysis [7] (SPA). A second solution relies on regular exponentiation like Square-and-Multiply-Always (SMA, see Algorithm 1) or Montgomery Ladder (ML, see Algorithm 2). Both algorithms consist in a square and a multiply operation in each iteration i, yielding no leakage to SPA.

Algorithm 1. Square and Multiply Always Left-to-Right

Input: $m, k = (k_l k_{l-1} \dots k_0)_2, p \quad (k_l = 1)$
Output: $m^k \bmod p$

1: $R_0 \leftarrow 1$
2: $R_1 \leftarrow m$
3: **for** $i = l - 1$ **downto** 0 **do**
4: $\quad R_1 \leftarrow R_1 \times R_1 \bmod p$ ▷ S_i
5: $\quad R_{k_i} \leftarrow R_1 \times m \bmod p$ ▷ M_i
6: **end for**
7: **return** R_1

Algorithm 2. Montgomery Ladder Left-to-Right

Input: $m, k = (k_l k_{l-1} \dots k_0)_2, p \quad (k_l = 1)$
Output: $m^k \bmod p$

1: $R_0 \leftarrow m$
2: $R_1 \leftarrow R_0 \times R_0 \bmod p$ ▷ FS
3: **for** $i = l - 1$ **downto** 0 **do**
4: $\quad R_{\neg k_i} \leftarrow R_0 \times R_1 \bmod p$ ▷ M_i
5: $\quad R_{k_i} \leftarrow R_{k_i} \times R_{k_i} \bmod p$ ▷ S_i
6: **end for**
7: **return** R_0

Contributions of This Paper. We show that despite message blinding and regular exponentiation, it is still possible for an attacker to take advantage of extra-reductions: A new bias is found, namely a strong negative correlation between the extra-reduction of two consecutive operations. As shown in this paper, the bias can be easily leveraged to recover which registers are written to (at line 5 of Algorithm 1 or at lines 4 and 5 of Algorithm 2) which eventually leads to retrieve the secret key. The advantages of this method are the following:

– messages are unknown; this captures general situations such as RSA with OAEP or PSS padding and RSA input blinding [11, Sect. 10];
– RSA parameters can be unknown; hence RSA-CRT is also vulnerable;

– all binary exponentiation algorithms are vulnerable, even the regular ones like Square and Multiply Always, Montgomery Ladder, etc.;
– our attack can also be applied to Elliptic Curve Cryptography (ECC).

From a mathematical viewpoint, we also provide a comprehensive framework for studying the joint probabilities of extra-reductions in a sequence of multiplies and squares.

Related Works. The "horizontal/vertical" side-channel attacks against blinded exponentiation described in [6,10,24] also use the dependency between the input/output of operands in square and multiply algorithms. Such attacks exploit the *vertical* amplitude of the signal during the time duration. Our work is thus complementary to these ideas since it considers a novel *horizontal* exploitable bias.

Outline. The rest of the paper is organized as follows[1]. Section 2 recalls known biases induced by extra-reductions in modular multiplication algorithms such as the Montgomery modular multiplication. Our contribution starts at Sect. 3, where the theoretical rationale for the strong negative correlation between extra-reductions of two chained operations is presented. Section 4 shows how this bias can be turned into a key recovery attack. Experimental validations for synthetic and practical traces are in Sect. 5. Section 6 concludes.

2 State of the Art of Extra-Reductions Probabilities

This section reviews known results about extra-reductions and their probability distributions. The results can be adapted easily to Barrett reduction or multiplication followed by reduction using the extended Euclid algorithm.

2.1 Montgomery Modular Multiplication: Definitions and Notations

Given two integers a and b, the classical modular multiplication $a \times b \bmod p$ computes the multiplication $a \times b$ followed by the modular reduction by p. Montgomery Modular Multiplication (MMM) transforms a and b into special representations known as their Montgomery forms.

Definition 1 (Montgomery Transformation [14]). *For any prime modulus p, the Montgomery form of $a \in \mathbb{F}_p$ is $\phi(a) = a \times R \bmod p$ for some constant R greater than and co-prime with p.*

In order to ease the computation, R is usually chosen as the smallest power of two greater than p, that is $R = 2^{\lceil \log_2(p) \rceil}$. Using the Montgomery form of integers, modular multiplications used in modular exponentiation algorithms (recall Algorithms 1 and 2) can be carried out using the Montgomery Modular Multiplication (MMM):

[1] A complete version containing auxiliary information is available in [8].

Definition 2 Montgomery Modular Multiplication [14]). *Let $\phi(a)$ and $\phi(b)$ two elements of $\mathbb{F}_p$ in Montgomery form. The MMM of $\phi(a)$ and $\phi(b)$ is $\phi(a) \times \phi(b) \times R^{-1} \bmod p$.*

Algorithm 3 below shows that the MMM can be implemented in two steps: (i) compute $D = \phi(a) \times \phi(b)$, then (ii) reduce D using Montgomery reduction which returns $\phi(c)$. In Algorithm 3, the pair (R^{-1}, v) is such that $RR^{-1} - vp = 1$.

Algorithm 3. Montgomery Reduction (Algorithm 14.32 of [13])

Input: $D = \phi(a) \times \phi(b)$
Output: $\phi(c) = \phi(a) \times \phi(b) \times R^{-1} \bmod p$
1: $m \leftarrow (D \bmod R) \times v \bmod R$
2: $U \leftarrow (D + m \times p) \div R$ ▷ Invariant: $0 \leq U < 2p$
3: **if** $U \geq p$ **then**
4: $\quad C \leftarrow U - p$ ▷ Extra-reduction
5: **else**
6: $\quad C \leftarrow U$
7: **end if**
8: **return** C

Definition 3 (Extra-Reduction). *In Algorithm 3, when the intermediate value U is greater than p, a subtraction named eXtra-reduction occurs so as to have a result C of the Montgomery multiplication between 0 and $p - 1$. We set $X = 1$ in the presence of the eXtra-reduction, and $X = 0$ in its absence.*

Most software implementations of modular arithmetic for large numbers (such as OpenSSL and mbedTLS) use the MMM, where there is a final extra-reduction. In mbedTLS, this extra-reduction is compensated. However, as shown below in Sect. 5.2, an attacker is still able in practice to detect using some side-channel which branch has been used (either line 4 or 6 of Algorithm 3).

2.2 A Bias to Differentiate a Multiply from a Square

Proposition 1 (Probability of Extra-Reduction in a Multiply and a Square Operation [16, **Lemma 1**]). *Assuming uniform distribution of operands, the probabilities of an extra-reduction in a multiply X_{M_i} and in a square X_{S_i} at iteration i are*

$$\mathbb{P}(X_{M_i} = 1) = \frac{p}{4R} \qquad \text{and} \qquad \mathbb{P}(X_{S_i} = 1) = \frac{p}{3R}. \tag{1}$$

We note that extra-reductions are 33 % more likely when the operation is a square than when it is a multiply, irrespective of the ratio $\frac{p}{R} \in]\frac{1}{2}, 1[$. This allows one to break unprotected exponentiation algorithms.

3 A Bias to Test the Dependency of Operations

3.1 Principle of Correlated Extra-Reductions

In regular exponentiation algorithms, differentiating a multiply from a square does not allow SPA to distinguish the value of the exponent bits. Indeed, at every iteration i ($l-1 \geq i > 0$ where i is decremented after each iteration), multiply and square operations are carried out unconditionally. However, the input value of each operation depends on the current exponent bit value k_i. Figure 1 illustrates the dependence or independence between the input/output values of multiplication M_i and the input value of the following square S_{i-1} as a function of the bit value k_i during the SMA algorithm (Algorithm 1). Intuitively, when the output of M_i is equal to the input of S_{i-1}, we can expect that the extra-reductions in both operation are strongly correlated.

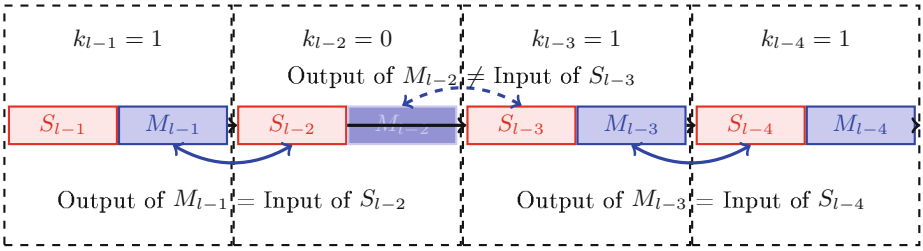


Fig. 1. Comparison between the output value of multiplication with the input of the following square in the Square-and-Multiply-Always exponentiation algorithm (Algorithm 1).

For the ML algorithm (Algorithm 2), the M_i and S_{i-1} operations depends directly on the two consecutive key bit values k_i and k_{i-1}. If the bit value k_{i-1} and its previous bit value k_i are different then the output of multiplication M_i and the input of square S_{i-1} are equal and yield strongly correlated extra-reductions; in the opposite case they yield uncorrelated extra-reductions.

Definition 4 (Guess Notation). *Let $\mathcal{G}_i$ be the "guess' Boolean random variable defined to be* `True` *(T) if the output of an operation at iteration i is equal to the input of the next operation at iteration $i-1$, and* `False` *(F) otherwise.*

Also let X_{M_i} be a random variable corresponding to the eXtra-reduction of the MMM multiplication at iteration i and $X_{S_{i-1}}$ be a random variable corresponding to the eXtra-reduction during the MMM square at iteration $(i-1)$.

Then $\mathbb{P}(X_{M_i}, X_{S_{i-1}} | \mathcal{G}_i = T)$ is their joint probability when the output value of the multiplication is equal to the input value of the square, and $\mathbb{P}(X_{M_i}, X_{S_{i-1}} | \mathcal{G}_i = F)$ is their joint probability when the output value of the multiplication is not equal to the input value of the square.

Table 1. Example of probabilities of eXtra-reduction X_{M_i} of multiply operation and $X_{S_{i-1}}$ of square operation knowing the Boolean value $\mathcal{G}_i$ for `RSA-1024-p`. The first line (correct guess) is applicable for both SMA and ML.

$(x_{M_i}, x_{S_{i-1}})$	(0,0)	(1,0)	(0,1)	(1,1)
$\mathbb{P}(x_{M_i}, x_{S_{i-1}} \mid \mathcal{G}_i = T)$	0.541575	0.191615	0.258276	0.008532
$\mathbb{P}(x_{M_i}, x_{S_{i-1}} \mid \mathcal{G}_i = F)$ for SMA	0.612756	0.120158	0.186803	0.080281
$\mathbb{P}(x_{M_i}, x_{S_{i-1}} \mid \mathcal{G}_i = F)$ for ML	0.586105	0.147246	0.213521	0.053128

The guess value $\mathcal{G}_i$ is linked to the key value depending on the regular exponentiation algorithm. For SMA and for a bit k_i, an attacker is able to estimate the probabilities $\hat{\mathbb{P}}(X_{M_i}, X_{S_{i-1}})$. This probability can be used to find the bit k_i as illustrated in Fig. 1 and explained in Sect. 4 below. For ML, $\mathcal{G}_i$ depends on two consecutive key bits as explained also in Sect. 4.

We have estimated the joint probabilities $\mathbb{P}(X_{M_i}, X_{S_{i-1}} \mid \mathcal{G}_i)$ using 1.000.000 random values for both SMA and ML algorithms and the example `RSA-1024-p` defined in [8, Sect. 2.2] for this modulus for which the ratio $p/R \simeq 0.800907$. The values of the obtained probabilities are shown in Table 1.

It is important to notice that for each $(x_{M_i}, x_{S_{i-1}}) \in \{0,1\}^2$, the conditional joint probabilities are distinct: $\mathbb{P}(X_{M_i} = x_{M_i}, X_{S_{i-1}} = x_{S_{i-1}} \mid \mathcal{G}_i = F) \neq \mathbb{P}(X_{M_i} = x_{M_i}, X_{S_{i-1}} = x_{S_{i-1}} \mid \mathcal{G}_i = T)$. Also for $\mathcal{G}_i = F$ in ML, it can be observed that $\mathbb{P}(X_{M_i}, X_{S_{i-1}} \mid \mathcal{G}_i) = \frac{p}{4R} \times \frac{p}{3R} = \mathbb{P}(X_{M_i}) \times \mathbb{P}(X_{S_{i-1}})$, which is consistent with the fact the two operations X_{M_i} and $X_{S_{i-1}}$ should be independent since they are completely unrelated.

It should be emphasized that the leakage identified in Table 1 is fairly large, since the Pearson correlations ρ of the two randoms variables are[2]:

$$\rho(X_{M_i}, X_{S_{i-1}} \mid \mathcal{G}_i = T) \approx -0.2535, \tag{2}$$

$$\rho(X_{M_i}, X_{S_{i-1}} \mid \mathcal{G}_i = F) \approx +0.1510 \text{ in SMA}, \tag{3}$$

$$\rho(X_{M_i}, X_{S_{i-1}} \mid \mathcal{G}_i = F) \approx -0.0017 \text{ in ML}. \tag{4}$$

To the best of our knowledge, such correlations have not been observed previously. A few observations are in order:

- when a square follows a multiply, and if there has been an extra-reduction in the multiplication, the result should be short, hence there is less chance for an extra-reduction to occur in the following square. This accounts for the negative correlation $\rho(X_{M_i}, X_{S_{i-1}} \mid \mathcal{G}_i = T)$;
- from Fig. 1 iteration $i = l - 2$ where $k_i = 0$, we can see that one input of the multiplication M_i equals the input of the following squaring S_{i-1}. Since a square and a multiplication share a common operand, provided it is sufficiently large, both operations are likely to have an extra-reduction at the same time, which accounts for the positive correlation $\rho(X_{M_i}, X_{S_{i-1}} \mid \mathcal{G}_i = F)$ for SMA;

[2] $\rho(X_{M_i}, X_{S_{i-1}}) = \frac{\mathbb{C}ov(X_{M_i}, X_{S_{i-1}})}{\sigma_{X_{M_i}} \sigma_{X_{S_{i-1}}}} = \frac{\mathbb{P}(X_{M_i}=1, X_{S_{i-1}}=1) - (\mathbb{P}(X_{M_i}=1) \times \mathbb{P}(X_{S_{i-1}}=1))}{\sqrt{\mathbb{P}(X_{M_i}=1)(1-\mathbb{P}(X_{M_i}=1))}\sqrt{\mathbb{P}(X_{S_{i-1}}=1)(1-\mathbb{P}(X_{S_{i-1}}=1))}}$.

– when a square and a multiply handle independent data, the extra-reductions are clearly also independent of each other, which explains the small value of $\rho(X_{M_i}, X_{S_{i-1}}|\mathcal{G}_i = F)$ for ML.

As explained next, when extra-reductions can be detected reliably, the data-flow can be analyzed accurately thereby defeating regular exponentiation protections.

3.2 Methodology to Analyze the Bias

In order to estimate the probability $\mathbb{P}(X_{M_i}, X_{S_{i-1}}|\mathcal{G}_i)$, we first determine the distribution of the output value after one MMM (following the method described by Sato et al. [15]) and then compute the joint probability for each case.

Let A, B be two independent random variables uniformly distributed in $[0, p[$ (represented in Montgomery form); let C be equal to the MMM product of A and B and U corresponds to the MMM product of A and B before eXtra-reduction (if any). Variables C and U coincide with that of Algorithm 3. As a matter of fact, an attacker cannot observe values, only extra-reductions which occur during Montgomery reduction (at line 4 of Algorithm 3). We use notations $\mathbb{P}$ for probabilities and f for probability density functions (p.d.f.'s).

Figure 2 shows histograms for C and U obtained from one million simulations; the binning consists of 100 bins of the interval $[0, 2p[$. It can be observed that

– the p.d.f. of C is uniform on $[0, p[$;
– the p.d.f. of U is a piecewise continuous function composed of a strictly increasing part, a constant part and a strictly decreasing part;
– the two conditional p.d.f.'s of C knowing $X_{M_i} \in \{0, 1\}$ (resp. $X_{S_i} \in \{0, 1\}$) are not uniform;
– for $c \in [0, p[$, one has $f(C = c) = f(U = c) + f(U = c + p)$ by definition of U;
– the maximum value of U is $p + p^2/R$, which is strictly smaller than $2p$.

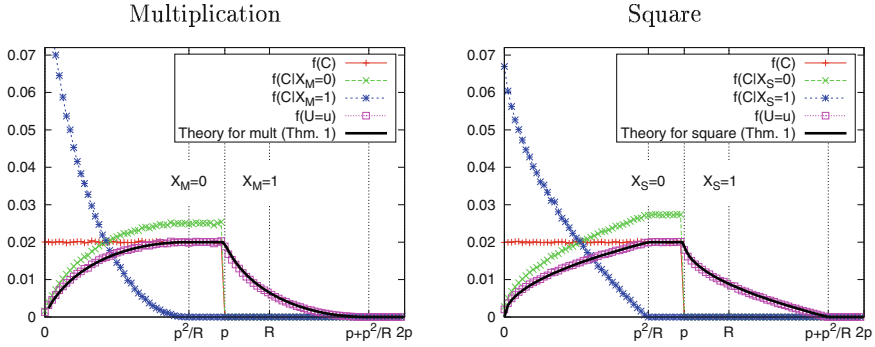


Fig. 2. Distribution of the output value of Montgomery multiplication (*left*) and square (*right*) for `RSA-1024-p`.

Recall that we use the Montgomery reduction described in Algorithm 3, where the reduction modulo p is carried out after every multiplication. This is also the case in [16,17], but *not* in [20,21] where the multiplicands lie in $[0, R[$. To complement those works, we now derive a closed-form expression of the output distribution of the Montgomery multiplication product and square (not found in [16,17]).

3.3 Mathematical Derivations

This subsection provides a mathematical justification of the biases observed in Table 1. In particular, it shows that such biases hold for all values of p and $R = 2^{\lceil \log_2(p) \rceil}$. Our closed-form expressions are derived as limits in distribution when $p \to +\infty$ that we shall write as approximations.

Theorem 1 (P.d.f. of MMM Before Extra-Reduction[3]). *Asymptotically when modulus p is large, the result of a Montgomery multiplication before the final extra-reduction (at line 2 of Algorithm 3) have piecewise p.d.f. given by*

$$f_U(u) = \begin{cases} \frac{Ru}{p^3}\left(1 - \ln(\frac{Ru}{p^2})\right) & \text{if } 0 \le u \le \frac{p^2}{R}; \\ \frac{1}{p} & \text{if } \frac{p^2}{R} \le u \le p; \\ \frac{1}{p} - \frac{R(u-p)}{p^3}\left(1 - \ln(\frac{R(u-p)}{p^2})\right) & \text{if } p \le u \le p + \frac{p^2}{R}; \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

The corresponding p.d.f. for the square is also in four pieces with the same intervals for u, and differs only from the multiplication in that it is equal to $\sqrt{Ru}/p^2$ when $0 \le u \le \frac{p^2}{R}$, and $1/p - \sqrt{R(u-p)}/p^2$ when $p \le u \le p + \frac{p^2}{R}$.

The theoretical values of Theorem 1 nicely superimpose with experimentally estimated p.d.f.'s as shown in Fig. 2.

Theorem 2 (Joint Probability of Extra-Reduction in Multiplication Followed by a Square [see Footnote 3]). *The following joint probabilities do not depend on the iteration index i, where $l - 1 \ge i > 0$.*

When $\mathcal{G}_i = T$:

$\mathbb{P}(X_{M_i}, X_{S_{i-1}})$	$X_{S_{i-1}} = 0$	$X_{S_{i-1}} = 1$
$X_{M_i} = 0$	$1 - \frac{7}{12}\frac{p}{R} + \frac{1}{48}\left(\frac{p}{R}\right)^4$	$\frac{p}{3R} - \frac{1}{48}\left(\frac{p}{R}\right)^4$
$X_{M_i} = 1$	$\frac{p}{4R} - \frac{1}{48}\left(\frac{p}{R}\right)^4$	$\frac{1}{48}\left(\frac{p}{R}\right)^4$

[3] Proof of this theorem is given in [8].

When $\mathcal{G}_i = F$ in SMA:

$\mathbb{P}(X_{M_i}, X_{S_{i-1}})$	$X_{S_{i-1}} = 0$	$X_{S_{i-1}} = 1$
$X_{M_i} = 0$	$1 - \frac{7}{12}\frac{p}{R} + \frac{1}{8}\left(\frac{p}{R}\right)^2$	$\frac{p}{3R} - \frac{1}{8}\left(\frac{p}{R}\right)^2$
$X_{M_i} = 1$	$\frac{p}{4R} - \frac{1}{8}\left(\frac{p}{R}\right)^2$	$\frac{1}{8}\left(\frac{p}{R}\right)^2$

When $\mathcal{G}_i = F$ in ML:

$\mathbb{P}(X_{M_i}, X_{S_{i-1}})$	$X_{S_{i-1}} = 0$	$X_{S_{i-1}} = 1$
$X_{M_i} = 0$	$1 - \frac{7}{12}\frac{p}{R} + \frac{1}{12}\left(\frac{p}{R}\right)^2$	$\frac{p}{3R} - \frac{1}{12}\left(\frac{p}{R}\right)^2$
$X_{M_i} = 1$	$\frac{p}{4R} - \frac{1}{12}\left(\frac{p}{R}\right)^2$	$\frac{1}{12}\left(\frac{p}{R}\right)^2$

It can be easily checked that Theorem 2 accurately matches experimental probability estimations given in Table 1.

Corollary 1. *The corresponding correlation coefficients are*

$$\rho(X_{M_i}, X_{S_{i-1}} | \mathcal{G}_i = T) = \frac{\frac{p^4}{48R^4} - \frac{p^2}{12R^2}}{\sqrt{\frac{p}{4R}\left(1 - \frac{p}{4R}\right)\frac{p}{3R}\left(1 - \frac{p}{3R}\right)}},$$

$$\rho(X_{M_i}, X_{S_{i-1}} | \mathcal{G}_i = F) = \frac{\frac{p^2}{24R^2}}{\sqrt{\frac{p}{4R}\left(1 - \frac{p}{4R}\right)\frac{p}{3R}\left(1 - \frac{p}{3R}\right)}} \text{ in SMA,}$$

$$\rho(X_{M_i}, X_{S_{i-1}} | \mathcal{G}_i = F) = 0 \text{ in ML.}$$

Proof. Apply Pearson's correlation definition on the results of Theorem 2. □

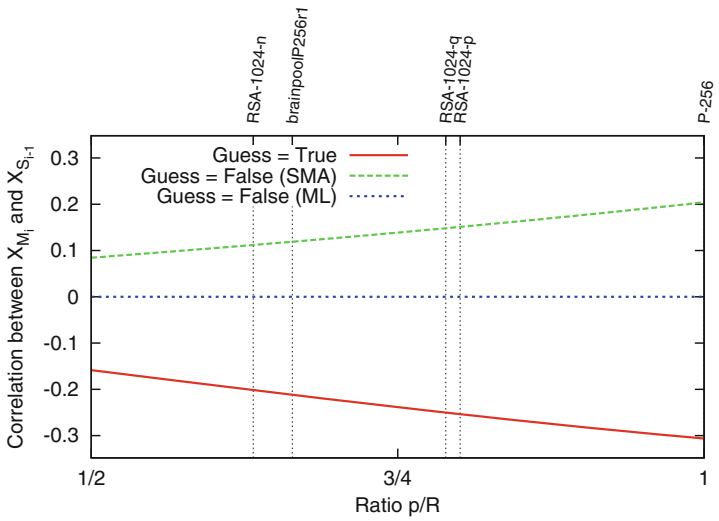


Fig. 3. Pearson's correlation between X_{M_i} and $X_{S_{i-1}}$.

When the guess is correct, $\rho(X_{M_i}, X_{S_{i-1}}|\mathcal{G}_i = T)$ is negative and increasingly negative as p/R increases, where

$$-\frac{3}{16}\sqrt{\frac{5}{7}} \approx -0.158 \quad \leq \quad \rho(X_{M_i}, X_{S_{i-1}}|\mathcal{G}_i = T) \quad \leq \quad -\frac{3}{4\sqrt{6}} \approx -0.306.$$

When the guess is incorrect, either the correlation is null (in the case of ML), or it is positive and increasing with p/R, where for $1/2 \leq p/R \leq 1$,

$$\frac{1}{2\sqrt{5\times 7}} \approx 0.085 \quad \leq \quad \rho(X_{M_i}, X_{S_{i-1}}|\mathcal{G}_i = F) \quad \leq \quad \frac{1}{2\sqrt{6}} \approx 0.204.$$

The variations of the correlation coefficients between X_{M_i} and $X_{S_{i-1}}$ in the three scenarios of Corollary 1 are plotted in Fig. 3.

Figure 3 shows that the correlation difference between guesses `True/False` is greater for the SMA algorithm than for the ML algorithm. Thus our attack on SMA should outperform that on ML. Also notice that the larger the ratio p/R, the larger the correlation difference; hence, we expect `P-256` to be easier to break than `brainpoolP256r1` with our attack.

4 Exploiting the Bias Using Our Attack

The difference between the two Pearson correlations according to the guess value $\mathcal{G}_i$ (Corollary 1) allows us to test whether some data produced by an operation is fed into the next operation. The bit value k_i can be estimated using the Pearson correlation ρ as a distinguisher, a threshold $\mathcal{T}$ depending of the knowledge of the attacker and a decision function denoted by $\mathcal{F}_{\mathcal{ALG}}$ which depends of the regular exponentiation algorithm and the used distinguisher.

Attacker's Method. An attacker calls Q times the cryptographic operation with a static key k and measures the corresponding side-channel trace. For each trace $q \in \{1, \ldots, Q\}$, $(l-1)$ pairs of extra-reductions $(x^q_{M_i}, x^q_{S_{i-1}})_{l-1 \geq i > 0}$ are captured. The complete acquisition campaign is denoted $(\underline{x_{M_i}}, \underline{x_{S_{i-1}}})$, and is a matrix of size $Q \times (l-1)$ pairs of bits. Notice that neither the input nor the output of the cryptographic algorithm is required. For all $i \in \{l-1, \ldots, 1\}$ and $q \in \{1, \ldots, Q\}, x^q_{M_i}$ is equal to 1 (resp. 0) if the eXtra-reduction is present (resp. missing) during the multiplication M_i for query q. Similarly, $x^q_{S_{i-1}}$ is equal to 1 (resp. 0) if the eXtra-reduction is present (resp. missing) during the square S_{i-1} for query q. For each pair of random variable $(X_{M_i}, X_{S_{i-1}})$, the attacker first computes the estimated probability $\hat{\mathbb{P}}(X_{M_i}, X_{S_{i-1}})$, using:

$$\hat{\mathbb{P}}(X_{M_i}, X_{S_{i-1}}) = \frac{1}{Q}\sum_{q=1}^{Q} \mathbb{1}_{(X_{M_i}=x^q_{M_i})\wedge(X_{S_{i-1}}=x^q_{S_{i-1}})}. \tag{6}$$

The attacker then computes the Pearson correlation[4] $\hat{\rho}(X_{M_i}, X_{S_{i-1}})$ for each pair $(x_{M_i}, x_{S_{i-1}}) \in \{0,1\}^2$ using the estimated probability $\hat{\mathbb{P}}(X_{M_i}, X_{S_{i-1}})$. Finally,

[4] $\hat{\rho}(X_{M_i}, X_{S_{i-1}}) = \frac{\mathbb{C}\hat{o}v(X_{M_i}, X_{S_{i-1}})}{\hat{\sigma}_{X_{M_i}}\hat{\sigma}_{X_{S_{i-1}}}} = \frac{\hat{\mathbb{P}}(X_{M_i}=1, X_{S_{i-1}}=1) - (\hat{\mathbb{P}}(X_{M_i}=1)\times\hat{\mathbb{P}}(X_{S_{i-1}}=1))}{\sqrt{\hat{\mathbb{P}}(X_{M_i}=1)(1-\hat{\mathbb{P}}(X_{M_i}=1))}\sqrt{\hat{\mathbb{P}}(X_{S_{i-1}}=1)(1-\hat{\mathbb{P}}(X_{S_{i-1}}=1))}}$.

she estimates the exponent bit k_i with her knowledge corresponding to threshold $\mathcal{T}$ and decision function $\mathcal{F}_{\mathcal{ALG}}$.

Attacker's Knowledge. In public key cryptography, the attacker wants to recover the private exponent in RSA or the private scalar in ECC. In our attacks, we assume these secret values are static, as for instance in RSA-CRT decryption or static Diffie-Hellman key agreement protocol.

– In RSA-SFM and ECC, the attacker knows the parameters p and R defined in Sect. 2.1. In RSA-SFM, p is equal to the public modulus n_{RSA}. In ECC, p equals the characteristic of the finite field over which the elliptic curve is defined. The attacker can compute the Pearson correlations $\rho(X_{M_i}, X_{S_{i-1}}|\mathcal{G}_i = T)$ and $\rho(X_{M_i}, X_{S_{i-1}}|\mathcal{G}_i = F)$ using corollary 1. The threshold for the successful attack is defined by:

$$\mathcal{T} = \frac{\rho(X_{M_i}, X_{S_{i-1}}|\mathcal{G}_i = T) + \rho(X_{M_i}, X_{S_{i-1}}|\mathcal{G}_i = F)}{2}. \tag{7}$$

– In RSA-CRT, the attacker does not know the parameters p and R defined in Sect. 2.1, because the prime factors p_{RSA} and q_{RSA} are secret parameters. Hence the determination of the probabilities by theory or simulation are impossible. However, using the Q measurements $(\underline{x_{M_i}}, \underline{x_{S_{i-1}}})$, the attacker is able to determine the mean estimated probability $\hat{\mathbb{E}}_i\hat{\mathbb{P}}(X_{M_i}, X_{S_{i-1}})$ by[5]:

$$\hat{\mathbb{E}}_i\hat{\mathbb{P}}(X_{M_i}, X_{S_{i-1}}) = \frac{\sum_{i=1}^{l-1} \hat{\mathbb{P}}(X_{M_i}, X_{S_{i-1}})}{l-1}. \tag{8}$$

The attacker then computes the mean estimated Pearson correlations using the mean estimated probability (8), and the threshold for the successful attack is defined by:

$$\mathcal{T} = \frac{\hat{\mathbb{E}}_i\hat{\mathbb{P}}(X_{M_i} = 1, X_{S_{i-1}} = 1) - (\hat{\mathbb{E}}_i\hat{\mathbb{P}}(X_{M_i} = 1) \times \hat{\mathbb{E}}_i\hat{\mathbb{P}}(X_{S_{i-1}} = 1))}{\sqrt{\hat{\mathbb{E}}_i\hat{\mathbb{P}}(X_{M_i} = 1)\hat{\mathbb{E}}_i\hat{\mathbb{P}}(X_{M_i} = 0)}\sqrt{\hat{\mathbb{E}}_i\hat{\mathbb{P}}(X_{S_{i-1}} = 1)\hat{\mathbb{E}}_i\hat{\mathbb{P}}(X_{S_{i-1}} = 0)}}. \tag{9}$$

In fact, the threshold value $\mathcal{T}$ computed in (7) or (9) does not depend on i. The indication of index i was kept as a reminder that the multiplication M_i is done in the iteration which precedes that of the square S_{i-1}.

Decision Function. The decision function depending of the regular algorithm and the used distinguisher ρ is denoted as $\mathcal{F}_{\mathcal{ALG}}$. We detail this function for the SMA (Algorithm 1) and ML (Algorithm 2) algorithms.

[5] Notice that in some cases, e.g. if the key bits happen not to be balanced, $\hat{\mathbb{E}}_i\hat{\mathbb{P}}(X_{M_i}, X_{S_{i-1}})$ can be estimated in a less biased way using $\max_i\{\hat{\mathbb{P}}(X_{M_i}, X_{S_{i-1}})\} - \min_i\{\hat{\mathbb{P}}(X_{M_i}, X_{S_{i-1}})\}$.

– In the SMA algorithm, the scalar bit k_i decides whether the output of M_i is the input of S_{i-1} or not (see Fig. 1). If the bit value k_i equals 1, then the square S_{i-1} depends on M_i ($\mathcal{G}_i = T$), otherwise the output value of M_i is different from the input value of S_{i-1} ($\mathcal{G}_i = F$). Using the Sect. 3, we see that $\rho(X_{M_i}, X_{S_{i-1}}|\mathcal{G}_i = T) < \rho(X_{M_i}, X_{S_{i-1}}|\mathcal{G}_i = F)$, so the decision function $\mathcal{F}_{SMA}$ is defined by:

$$\hat{k}_i = \mathcal{F}_{SMA}(\rho, \mathcal{T}) = \begin{cases} 0 & \text{if } \hat{\rho}(X_{M_i}, X_{S_{i-1}}) \geq \mathcal{T}, \\ 1 & \text{otherwise.} \end{cases} \tag{10}$$

– For the Montgomery Ladder (ML) algorithm, the M_i and S_{i-1} operations do not depend directly on the key bit value k_i. The dependence comes from the bit value k_{i-1} and the previous bit value k_i. If the two bits value k_{i-1} and k_i are different then the output of multiplication M_i and the input of square S_{i-1} are equal ($\mathcal{G}_i = T$), otherwise these output/input are different ($\mathcal{G}_i = F$). Using Sect. 3, we see that $\rho(X_{M_i}, X_{S_{i-1}}|\mathcal{G}_i = T) < \rho(X_{M_i}, X_{S_{i-1}}|\mathcal{G}_i = F)$, so the decision function $\mathcal{F}_{ML}$ using the previously estimated bit $\hat{k}_{i-1}$ is defined for each i ($l - 1 > i \geq 1$) by:

$$\hat{k}_i = \mathcal{F}_{ML}(\hat{k}_{i-1}, \rho, \mathcal{T}) = \begin{cases} \hat{k}_{i-1} & \text{if } \hat{\rho}(X_{M_i}, X_{S_{i-1}}) \geq \mathcal{T}, \\ \neg\hat{k}_{i-1} & \text{otherwise.} \end{cases} \tag{11}$$

Regarding the second most significant bit k_{l-1} of the exponent, either both values $k_{l-1} = 0$ and $k_{l-1} = 1$ are tested to recover the full secret key, or our attack can be applied between the first square FS (defined at line 2 of Algorithm 2) and the square S_{l-1} (line 5 of Algorithm 2).

Algorithm 4. ρ-attack

Input: $(\underline{x_{M_i}}, \underline{x_{S_{i-1}}})$, a set of Q pairs of $(l-1)$ bits

Output: An estimation $\hat{k} \in \{0,1\}^{l-1}$ of the secret exponent

1: **for** $i = l - 1$ **downto** 1 **do**
2: $\quad\hat{\mathbb{P}}(X_{M_i}, X_{S_{i-1}}) \leftarrow 0$
3: $\quad$**for** $q = 1$ **to** Q **do**
4: $\quad\quad\hat{\mathbb{P}}(X_{M_i} = x^q_{M_i}, X_{S_{i-1}} = x^q_{S_{i-1}}) \leftarrow \hat{\mathbb{P}}(X_{M_i} = x^q_{M_i}, X_{S_{i-1}} = x^q_{S_{i-1}}) + 1$
5: $\quad$**end for**
6: $\quad\hat{\mathbb{P}}(X_{M_i}, X_{S_{i-1}}) \leftarrow \hat{\mathbb{P}}(X_{M_i}, X_{S_{i-1}}) \,/\, Q$ ▷ Normalization
7: $\quad$Compute $\hat{\rho}(X_{M_i}, X_{S_{i-1}})$ using $\hat{\mathbb{P}}(X_{M_i}, X_{S_{i-1}})$
8: **end for**
9: Compute $\mathcal{T}$ depending on the attacker's knowledge
10: **for** $i = l - 1$ **downto** 1 **do**
11: $\quad\hat{k}_i \leftarrow \mathcal{F}_{\mathcal{ALG}}\left(\hat{\rho}(X_{M_i}, X_{S_{i-1}}), \mathcal{T}\right)$ ▷ Threshold
12: **end for**
13: **return** $\hat{k}$

Summary of the Attack. To estimate the exponent k by $\hat{k}$, we define two attacks:

- The attack named "ρ-attack-Hard", knowing the values of $\mathbb{P}(X_{M_i}, X_{S_{i-1}}|\mathcal{G}_i = T)$ and $\mathbb{P}(X_{M_i}, X_{S_{i-1}}|\mathcal{G}_i = F)$, using the threshold $\mathcal{T}$ computed by (7).
- The attack named "ρ-attack-Soft", when the theoretical value $\mathbb{P}(X_{M_i}, X_{S_{i-1}}|\mathcal{G}_i)$ is unknown. It uses the estimated probability $\hat{\mathbb{P}}(X_{M_i}, X_{S_{i-1}})$ to compute the threshold $\mathcal{T}$ by (9).

Algorithm 4 describes the attack to recover a full key. Lines 1-8 correspond to the computation of the estimated probabilities for each bit k_i defined by (6). Line 9 is the computation of the threshold: if the attack is ρ-attack-Hard the attacker uses (7), otherwise the attack is ρ-attack-Soft and she uses (9). The lines 10-12 compute the full estimated key using the decision function $\mathcal{F}_{\mathcal{ALG}}$, defined by the Eqs. (10) or (11).

5 Experimental Results

In the first part of this section, we detail a simulated attack which exploits the bias (explained in Corollary 1) to determine the number of queries necessary for the success of the attack. Then, we detail the side-channel part (*local timing analysis* using power consumption and electromagnetic analysis to distinguish *functional vs dummy subtractions*) in order to detect whether an eXtra-reduction is performed ($X = 1$) or not ($X = 0$) during the Montgomery reduction (Algorithm 3).

5.1 Simulations

Let `RSA-1024-p` defined at [8, Sect. 2.2] the modulus p used in the SMA algorithm (Algorithm 1). We generated one thousand random queries and saved for all MMM the information whether an extra-reduction is done or not. The length of static key k is 512 bits. As detailed in the ρ-attack (Algorithm 4) we computed the estimated probabilities $\hat{\mathbb{P}}(X_{M_i}, X_{S_{i-1}})$ and the estimated Pearson correlation $\hat{\rho}(X_{M_i}, X_{S_{i-1}})$ to retrieve each k_i. The estimated threshold $\mathcal{T}$ computed by (9) in our simulation is equal to -0.06076, which is an excellent approximation of the theoretical threshold (7). To retrieve each bit if the exponent, we used the decision function $\mathcal{F}_{SMA}$ described for ρ-attack in SMA by (10).

Figure 4 shows the estimated Pearson correlation values $\hat{\rho}(X_{M_i}, X_{S_{i-1}})$ for the first iterations. It can be easily seen that the estimated key value by this sequence corresponds to $\texttt{0}\times\texttt{1}0001111101011101110100\texttt{11}\ldots = \texttt{0}\times\texttt{11f5dd3}\ldots$ Our ρ-attack retrieves the 511 bits of the exponent using 1000 randoms queries with success rate 100%.

Success Rate Curves. We implemented ρ-attack-Hard and ρ-attack-Soft in the ideal case, i.e., without noise. The success rate to recover one bit of the exponent is represented in Fig. 5, for both SMA and ML cases. Interestingly,

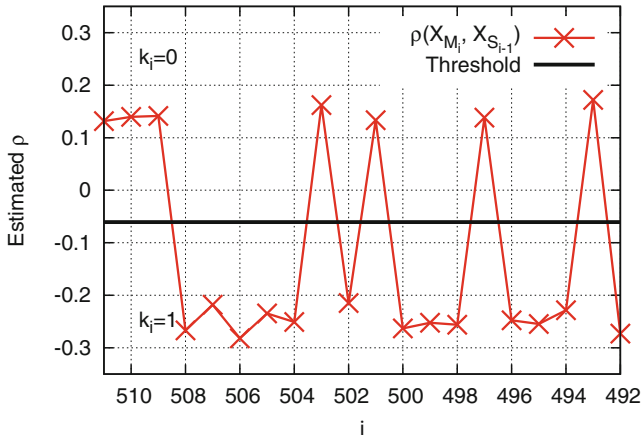


Fig. 4. Estimated Pearson correlations using 1000 randoms queries for `RSA-1024-p` for the first 20 iterations.

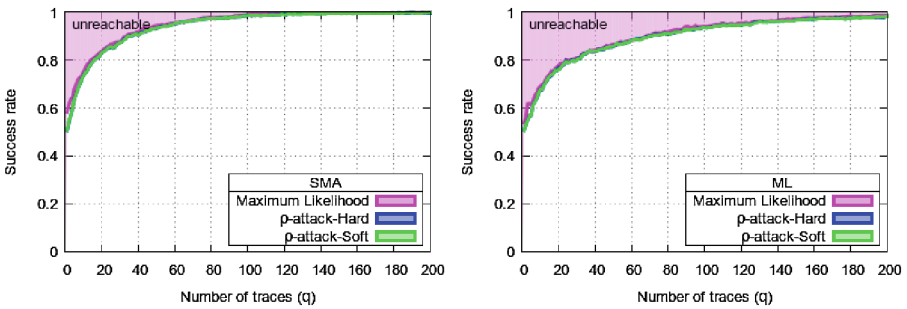
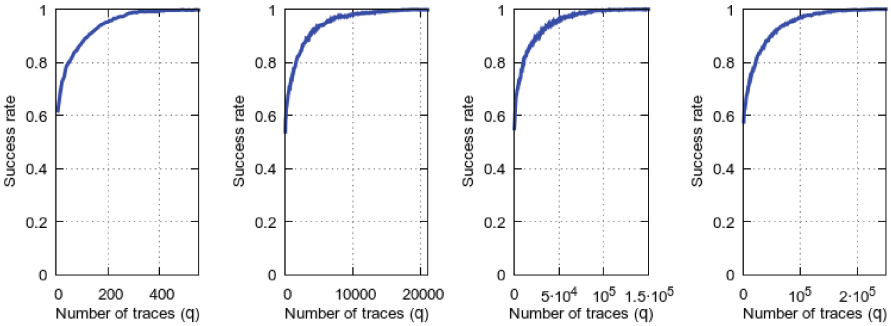


Fig. 5. Evolution of the success rate for the ρ-attack-Soft and the ρ-attack-Hard as a function of the number Q of queries (upper bound is the maximum likelihood), for `RSA-1024-p`.

ρ-attack-Hard and ρ-attack-Soft yield the same success rate, which happens to be (very close to) the optimal value. This optimal value is that obtained with the maximum likelihood distinguisher derived in [8].

The reason for the hard and soft attacks to have similar success probability is that the online estimation of the threshold is very good. Indeed, in the example of Fig. 5, the threshold $\mathcal{T}$ (Eq. (9)) is estimated based on $512Q$ traces, which is huge (one needs only to estimate 4 probabilities to get the estimation of $\mathcal{T}$). So, in the rest of this section, we make no difference between the hard and soft versions of the attacks from a success rate point of view.

The ρ-attacks are very close to the Maximum Likelihood attack for a similar reason. Estimating the difference between two random variables of very little dimensionality (recall that $(X_{M_i}, X_{S_{i-1}})$ lives in $\{0,1\}^2$) can be done almost



(a) $\mathbb{P}_{\text{noise}} = 10\%$ (b) $\mathbb{P}_{\text{noise}} = 20\%$ (c) $\mathbb{P}_{\text{noise}} = 30\%$ (d) $\mathbb{P}_{\text{noise}} = 40\%$

Fig. 6. Evolution of the success rate for the ρ-attack in function of queries Q using $p =$ `RSA-1024-p` for four increasing noise values.

equivalently in the proportional scale [23] (Pearson correlation) as in the context of information theoretic attacks (maximum likelihood attack) [8].

We may also notice that as the *distinguisher margin* [22] is larger for SMA than for ML (recall Fig. 3), the former attack requires less traces to reach a given success rate.

In practical cases, detecting an extra-reduction using only one acquisition can lead to errors. The probability to have an error is denoted by $\mathbb{P}_{\text{noise}}$. We show in Fig. 6 that the attack continues to be successful (albeit with more traces) over a large range of $\mathbb{P}_{\text{noise}}$ values. Evidently when $\mathbb{P}_{\text{noise}} = 50\,\%$ the attack becomes infeasible.

5.2 Experimental Detection of Extra-Reductions

Two Montgomery reduction implementations will be analyzed in this section. We raise the following questions.

1. How to exploit the *local timing* to distinguish the eXtra-reduction using power consumption measurements, on OpenSSL v1.0.1k-3 ([6])?
2. How to exploit the difference between a *real* and a *dummy* final subtraction using electromagnetic (EM) emanations, on mbedTLS v 2.2.0 ([7])?

(1a) Experiment Setup in Power. The target is a dual core LPC43S37 microcontroller fabricated in CMOS 90 nm Ultra Low Leakage process soldered on an LPCXpresso4337 board, and running at its maximum frequency (208 MHz). The side-channel traces where obtained measuring the instantaneous power consumption with a PICOSCOPE 6402C featuring 256 MB of memory, 500 MHz bandwidth and 5 GS/s sampling rate. We executed the private function of RSA

[6] Latest stable version at the time of submission.
[7] Latest version at the time of submission.

in OpenSSL with the private primes parameters defined by `RSA-1024-p` and `RSA-1024-q` in [8, Sect. 2.2]. The private modular exponentiation is RSA-CRT with a regular algorithm.

(1b) OpenSSL Experiment. In OpenSSL (see Listing 1.1 in Appendix A), the final subtraction is made when U is greater than p like described in Algorithm 3. A simple power analysis using the delay (referred to as "`SPA-Timing`") between two MMM operations found whether the extra-reduction is present ($X = 1$) or not ($X = 0$). On the Cortex M4 core, the delay between the M_i and S_{i-1} when $X_{M_i} = 1$ is 41.4952 μs, whereas the delay when $X_{M_i} = 0$ is 41.1875 μs. For the square operation S_{i-1}, the delay is 41.5637 μs when $X_{S_{i-1}} = 1$ and it is 41.2471 μs when $X_{S_{i-1}} = 0$. All in one, the observable timing differences are respectively 308 ns and 317 ns. When OpenSSL is offloaded on the Cortex M0 core of the LPC43S37, the timing difference is respectively 399 ns and 411 ns. The success rate of this detection attack is 100 %, hence $\mathbb{P}_{\text{noise}} = 0$.

(2a) Experiment Setup in EM. The target device is an STM32F4 microcontroller, which contains an ARM Cortex-M4 processor running at its maximum frequency (168 MHz). For the acquisition, we used a Tektronix oscilloscope and a Langer near field probe. The sampling frequency is 1 GSa/s with 50 MHz hardware input low-pass filter enabled. The position of the probe was determined to maximize the signal related to the activity of the hardware 32×32 processor. We executed the private function of RSA in mbedTLS, with the private primes parameters defined by `RSA-1024-p` and `RSA-1024-q` in [8, Sect. 2.2]. The private modular exponentiation is RSA-CRT with a regular algorithm.

(2b) mbedTLS Experiment. In order to achieve constant-time MMM, mbedTLS library implements a countermeasure using a dummy subtraction (see Listing 1.2 in Appendix A). In order to test the efficiency of the countermeasure, the duration of the real and dummy subtraction were compared as shown in Fig. 7. The durations are the same. Therefore, the `SPA-Timing` attack is not practical anymore.

In a view to differentiate the two patterns, we use a horizontal side-channel analysis [3], namely Pearson correlation (`max-corr`) [4] or the sum of the absolute differences (`min-abs-diff`). We build two reference patterns of the real subtraction $RP(X = 1)$ and dummy subtraction $RP(X = 0)$, and compare these patterns with one acquisition.

For this experiment, we use 500 acquisitions to build template $RP(X = 1)$ and again 500 acquisitions to make $RP(X = 0)$. The detection attack using one acquisition $\mathcal{A}_x$ where the extra-reduction $X = x$ is considered successful:

- when $\rho(\mathcal{A}_x, RP(X = x)) > \rho(\mathcal{A}_x, RP(X = \neg x))$ for `max-corr`, and
- when $\mathbb{E}(|\mathcal{A}_x - RP(X = x)|) < \mathbb{E}(|\mathcal{A}_x - RP(X = x)|)$ for `min-abs-diff`.

The success rate of the extra-reduction detection using 30000 acquisitions is 82.50 % for `max-corr` and 83.47 % for `min-abs-diff`, hence $\mathbb{P}_{\text{noise}} < 20\,\%$.

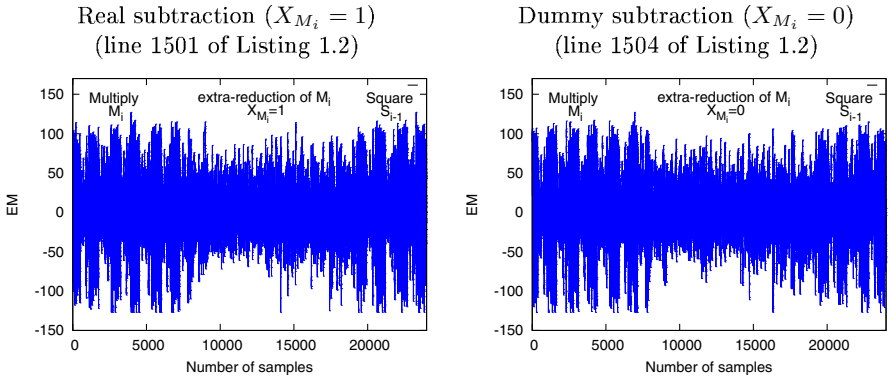


Fig. 7. Electromagnetic acquisition focus on one real subtraction (*left*) and pattern of one dummy subtraction (*right*) between two consecutive MMM operations.

5.3 Conclusions on Experiments

By combining the detection of extra-reductions using side-channel analysis (Sect. 5.2) and the theoretical attack to decide whether or not there is a dependency between various MMMs (Sect. 4), we deduce the number of queries Q needed to recover the secret exponent k. Table 2 summaries the results.

Table 2. Summary of the number of queries (see Fig. 6(b)) to retrieve all key bits of a secret exponent, as a function of side-channel detection method and regular exponentiation algorithm.

Type of attack side-channel for detection	`SPA-Timing`	`max-corr`	`min-abs-diff`
Detection probability for one query $= 1 - \mathbb{P}_{\text{noise}}$	100 %	82.50 %	83.47 %
Number of queries (SMA)	≈ 200	≈ 10000	≈ 10000
Number of queries (ML)	≈ 400	≈ 20000	≈ 20000

6 Conclusion

This paper has presented a new theoretical and practical attack against asymmetrical computation with regular exponentiation using extra-reductions as a side-channel. The working factor is the existence of a strong bias between the extra-reductions during the Montgomery Modular Multiplication of two consecutive operations. This new bias can be exploited in each regular binary algorithm, because each iteration consists in a square and a multiply whose inputs depend on the outputs of an operation from the previous iteration.

The new attacks have been detailed on RSA but are also applicable to ECC with appropriate customizations for various ECC implementations. As an example [5] for addition `madd-2004-hmv`, the Z-coordinate in output of addition is computed by a multiplication $Z3 = Z1 \times T1$ and for doubling `dbl-2007-bl`, the Z-coordinate in input of doubling is a square $ZZ = Z1 \times Z1$.

Acknowledgements. The authors would like to thank the anonymous reviewers for their useful comments that improved the quality of the paper. The first author would also like to thank François Dassance, Jean-Christophe Courrège and her colleagues for the suggestion of the main idea of this paper and their valuable insights.

A Analysis of Extra-Reduction in OpenSSL and MbedTLS Source Codes

The extra-reduction is explicit in the source code of OpenSSL, as shown in Listing 1.1.

Listing 1.1. Extra-reduction in OpenSSL code. File `crypto/bn/bn_mont.c`

```
309 if (BN_ucmp(ret, &(mont->N)) >= 0)
310 {
311     if (!BN_usub(ret,ret,&(mont->N))) goto err;
312 }
```

The big-number library of mbedTLS implements a protection against timing attacks. A subtraction is also carried out: it is either functional or dummy, as shown in Listing 1.2.

Listing 1.2. Extra-reduction in mbedTLS code. File `library/bignum.c`, function `mpi_montmul`

```
1500 if( mpi_cmp_abs( A, N ) >= 0 )
1501     mpi_sub_hlp( n, N->p, A->p );
1502 else
1503     /* prevent timing attacks */
1504     mpi_sub_hlp( n, A->p, T->p );
```

References

1. Acıiçmez, O., Schindler, W.: A vulnerability in RSA implementations due to instruction cache analysis and its demonstration on openSSL. In: Malkin, T. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 256–273. Springer, Heidelberg (2008)
2. Aciiçmez, O., Schindler, W., Koç, Ç.K.: Improving Brumley and Boneh timing attack on unprotected SSL implementations. In: Atluri, V., Meadows, C., Juels, A. (eds.) CCS 2005, pp. 139–146. ACM, New York (2005)
3. Bauer, A., Jaulmes, É., Prouff, E., Reinhard, J.-R., Wild, J.: Horizontal collision correlation attack on elliptic curves - extended version. Cryptogr. Commun. **7**(1), 91–119 (2015)
4. Belgarric, P., Bhasin, S., Bruneau, N., Danger, J.-L., Debande, N., Guilley, S., Heuser, A., Najm, Z., Rioul, O.: Time-frequency analysis for second-order attacks. In: Francillon, A., Rohatgi, P. (eds.) CARDIS 2013. LNCS, vol. 8419, pp. 108–122. Springer, Heidelberg (2014)
5. Bernstein, D.J., Lange, T.: Explicit formulas database. http://www.hyperelliptic.org/EFD/
6. Clavier, C., Feix, B., Gagnerot, G., Roussellet, M., Verneuil, V.: Horizontal correlation analysis on exponentiation. In: Soriano, M., Qing, S., López, J. (eds.) ICICS 2010. LNCS, vol. 6476, pp. 46–61. Springer, Heidelberg (2010)
7. 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)
8. Dugardin, M., Guilley, S., Danger, J.-L., Najm, Z., Rioul, O.: Correlated extra-reductions defeat blinded regular exponentiation - extended version. Cryptology ePrint Archive, Report 2016/597 (2016). http://eprint.iacr.org/2016/597
9. Fouque, P.-A., Réal, D., Valette, F., Drissi, M.: The carry leakage on the randomized exponent countermeasure. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 198–213. Springer, Heidelberg (2008)
10. Hanley, N., Kim, H.S., Tunstall, M.: Exploiting collisions in addition chain-based exponentiation algorithms using a single trace. In: Nyberg, K. (ed.) CT-RSA 2015. LNCS, vol. 9048, pp. 429–446. Springer, Heidelberg (2015)
11. 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)
12. Kocher, P.C.: On certificate revocation and validation. In: Hirschfeld, R. (ed.) FC 1998. LNCS, vol. 1465, pp. 172–177. Springer, Heidelberg (1998)
13. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1996). http://www.cacr.math.uwaterloo.ca/hac/
14. Peter, L.: Montgomery: modular multiplication without trial division. Math. Comput. **44**(170), 519–521 (1985)
15. Sato, H., Schepers, D., Takagi, T.: Exact analysis of montgomery multiplication. In: Canteaut, A., Viswanathan, K. (eds.) INDOCRYPT 2004. LNCS, vol. 3348, pp. 290–304. Springer, Heidelberg (2004)
16. Schindler, W.: A timing attack against RSA with the Chinese Remainder Theorem. In: Paar, C., Koç, Ç.K. (eds.) CHES 2000. LNCS, vol. 1965, pp. 109–124. Springer, Heidelberg (2000)
17. Schindler, W.: A combined timing and power attack. In: Naccache, D., Paillier, P. (eds.) PKC 2002. LNCS, vol. 2274, pp. 263–279. Springer, Heidelberg (2002)

18. Schindler, W.: Exclusive exponent blinding may not suffice to prevent timing attacks on RSA. In: Güneysu, T., Handschuh, H. (eds.) CHES 2015. LNCS, vol. 9293, pp. 229–247. Springer, Heidelberg (2015)
19. Schindler, W., Koeune, F., Quisquater, J.-J.: Improving divide and conquer attacks against cryptosystems by better error detection/correction strategies. In: Honary, B. (ed.) Cryptography and Coding 2001. LNCS, vol. 2260, pp. 245–267. Springer, Heidelberg (2001)
20. Schindler, W., Walter, C.D.: More detail for a combined timing and power attack against implementations of RSA. In: Paterson, K.G. (ed.) Cryptography and Coding 2003. LNCS, vol. 2898, pp. 245–263. Springer, Heidelberg (2003)
21. Walter, C.D., Thompson, S.: Distinguishing exponent digits by observing modular subtractions. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 192–207. Springer, Heidelberg (2001)
22. Whitnall, C., Oswald, E.: A comprehensive evaluation of mutual information analysis using a fair evaluation framework. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 316–334. Springer, Heidelberg (2011)
23. Whitnall, C., Oswald, E., Standaert, F.-X.: The myth of generic DPA..and the magic of learning. In: Benaloh, J. (ed.) CT-RSA 2014. LNCS, vol. 8366, pp. 183–205. Springer, Heidelberg (2014)
24. Witteman, M.F., van Woudenberg, J.G.J., Menarini, F.: Defeating RSA multiply-always and message blinding countermeasures. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 77–88. Springer, Heidelberg (2011)