Blinded Fault Resistant Exponentiation

Guillaume Fumaroli and David Vigilant

Axalto
6 rue de la Verrerie, F-92190 Meudon, France.
{gfumaroli,divigilant}@axalto.com

Abstract. As the core operation of many public key cryptosystems, group exponentiation is central to cryptography. Attacks on its implementation in embedded device setting is hence of great concern. Recently, implementations resisting both simple side-channel analysis and fault attacks were proposed. In this paper, we go further and present an algorithm that also inherently thwarts differential side-channel attacks in any finite abelian group with only limited time and storage overhead.

1 Introduction

In traditional cryptanalysis, only inputs and outputs of cryptographic algorithms are available to the attacker. Unfortunately, this assumption is inaccurate when the hardware implementation is in the hands of the attacker. A new range of attacks known as implementation attacks is then applicable. Embedded devices such as smartcards are especially targeted by these implementation attacks, that may be either passive or active.

Passive attacks are based on side-channel analysis, introduced by Kocher in [1], whose principle consist in monitoring the device to find correlations between some physical information leakage and the secret key manipulated by the device. While simple side-channel analysis refers to a correlation involving a single acquisition, differential side-channel analysis recovers the secret in several attempts by using the correlation between different acquisitions and a part of the secret.

Active attacks or fault attacks consist in carefully forcing the cryptographic device to perform erroneous operations such that the result leaks information about the secret data involved in the computation.

Group exponentiation is at the basis of many public key cryptosystems such as RSA, ECC or the Diffie-Hellman key exchange in some group. Cryptosystems based on exponentiation are particularly sensitive to implementation attacks both active [2] and passive [3].

In this paper, we present an exponentiation algorithm that resists all forementionned implementation attacks in finite abelian groups.

Our countermeasure features a novel base point blinding technique, based on the so-called Montgomery ladder introduced in [4], that requires fewer group operations than other techniques achieving the same level of protection.

In any finite abelian group whose order is unknown, our technique becomes the most efficient requiring no pre-computations. In particular, this is the case for RSA as the factorization of the modulus is rarely available to the device. Note that our countermeasure also fully applies to the ECC setting since the randomization of projective coordinates, introduced by Coron in [5], was later proven insufficiant by Goubin in [6]. As pointed out recently by Dupuy and Kuntz-Jacques [7], when the attacker can tamper with the base element, scalar point multiplications also require randomization of the computation flow to provide DPA resistance.

In section 2, the history of exponentiation algorithms targetting constrained embedded devices is reviewed. Section 3 presents our algorithm and analyses its security and efficiency. Section 4 concludes.

2 A review of previous work

Though more refined algorithms for computing group exponentiations exist in the litterature, only those based on binary ladders are relevant in constrained environments such as smartcards.

Square-and-multiply algorithms (Fig. 1) have first been considered for implementation, but they are easily broken by simple side-channel attacks.

```
Input: $x \in \mathbb{G}$, $k = \sum_{i=0}^{t-1} k_i 2^i \in \mathbb{N}$
Output: $x^k \in \mathbb{G}$
$R_0 \leftarrow 1$; $R_1 \leftarrow x$
for $j = t-1$ down to $0$ do
    $R_0 \leftarrow {R_0}^2$
    if $k_j = 1$ then $R_0 \leftarrow R_0 R_1$
end for
return $R_0$
```

Fig. 1. Square-and-multiply

Further, square-and-multiply-always algorithms (Fig. 2) introduced by Coron [5] were designed to prevent simple side-channel attacks by performing dummy operations. However, such algorithms bring specific weaknesses with respect to so-called safe-error attacks [8].

Montgomery ladder [4] was initially developped for elliptic curve scalar multiplication. Later, Joye et al. [8] extended it to exponentiation in any abelian group and pointed out its intrinsic resistance to simple side-channel attacks and safe-error attacks leveraging a slight modification. Let $L_j = \sum_{i=j}^{t-1} k_i 2^{i-j}$ and $H_j = L_j + 1$. As pointed out in [8], the principle of Montgomery ladder is based

```
Input: $x \in \mathbb{G}$, $k = \sum_{i=0}^{t-1} k_i 2^i \in \mathbb{N}$
Output: $x^k \in \mathbb{G}$
$R_0 \leftarrow 1$; $R_2 \leftarrow x$
for $j = t-1$ down to $0$ do
    $R_0 \leftarrow {R_0}^2$
    $R_{\bar{k_j}} \leftarrow R_{\bar{k_j}} R_2$
end for
return $R_0$
```

Fig. 2. Square-and-multiply-always

on the following observation:

$$(x^{L_j}, x^{H_j}) = \begin{cases} \left(\left(x^{L_{j+1}}\right)^2, x^{L_{j+1}} x^{H_{j+1}}\right) & \text{if } k_j = 0 \\ \left(x^{L_{j+1}} x^{H_{j+1}}, \left(x^{H_{j+1}}\right)^2\right) & \text{if } k_j = 1 \ . \end{cases}$$

This formula leads to Fig. 3 algorithm. The registers R_0 and R_1 contain the values of x^{L_j} and x^{H_j} respectively. (R_0, R_1) is initialized with $(x^{L_t}, x^{H_t}) = (1, x)$. After t iterations, (R_0, R_1) contains $(x^{L_0}, x^{H_0}) = (x^k, x^{k+1})$.

```
Input: $x \in \mathbb{G}$, $k = \sum_{i=0}^{t-1} k_i 2^i \in \mathbb{N}$
Output: $x^k \in \mathbb{G}$
$R_0 \leftarrow 1$; $R_1 \leftarrow x$
for $j = t-1$ down to $0$ do
    $R_{\bar{k_j}} \leftarrow R_{\bar{k_j}} R_{k_j}$
    $R_{k_j} \leftarrow R_{k_j}^2$
end for
return $R_0$
```

Fig. 3. Joye *et al.* Montgomery ladder

However, Montgomery ladder remains sensitive to differential side-channel analysis. As for group exponentiations, differential side-channel analysis may be prevented by randomizing either the group, the exponent or the base element. Randomization of the group structure was not explored in this paper. Known techniques targetting the exponent and the base element are presented in more details in [5, 9]. Blinding the exponent is not well suited for exponentiations in finite abelian groups. Indeed, the group order is generally unknown and its computation may be difficult. So blinding the base seems to be the most appropriate countermeasure. Let $\mathbb{G}$ be some abelian group closed under a multiplicative law and $d \in \mathbb{N}$ be a secret exponent. Blinding the base element consists in multiplying the input $x \in \mathbb{G}$ with a random element r picked at random from $\mathbb{G}$; the value of $x^d \in \mathbb{G}$ is then obtained as $(xr)^d \times (r^{-1})^d$. This countermeasure introduced in [5] requires two balanced group exponentiations, or a subtle but unpractical pre-computation technique that may be difficult to handle by the personalization process.

3 Our algorithm

As in the previous section, $L_j = \sum_{i=j}^{t-1} k_i 2^{i-j}$ and $H_j = L_j + 1$. Let us consider Fig. 3 algorithm and suppose (R_0, R_1) contains $\rho(x^{L_{j+1}}, x^{H_{j+1}})$ at the beginning of some iteration for some $\rho \in \mathbb{G}$. Then, (R_0, R_1) will contain $\rho^2(x^{L_j}, x^{H_j})$ at the beginning of the next iteration. This remark leads to Fig. 4 algorithm.

```
**Input:** $x \in \mathbb{G}$, $k = \sum_{i=0}^{t-1} k_i 2^i \in \mathbb{N}$
**Output:** $x^k \in \mathbb{G}$
Pick a random $r \in \mathbb{G}$
$R_0 \leftarrow r$; $R_1 \leftarrow rx$; $R_2 \leftarrow r^{-1}$
**for** $j = t-1$ **down to** $0$ **do**
    $R_{\bar{k_j}} \leftarrow R_{\bar{k_j}} R_{k_j}$
    $R_{k_j} \leftarrow R_{k_j}^2$
    $R_2 \leftarrow R_2^2$
**end for**
**return** $R_2 R_0$
```

Fig. 4. Side-channel analysis resistant Montgomery ladder

As will be shown in the sequel, the more refined Fig. 5 algorithm will have to be considered as Fig. 4 algorithm fails to detect some fault attacks.

At initialization, the couple of registers (R_0, R_1) is multiplicatively blinded by a secret random element $r \in \mathbb{G}$. Throughout the computation, (R_0, R_1) is then instrinsically multiplicatively masked by the element $r^{2^{t-j}} \in \mathbb{G}$. The register R_2 is initialized with $r^{-1} \in \mathbb{G}$ and holds the compensative factor $r^{-2^{t-j}} \in \mathbb{G}$ such that $R_2(R_0, R_1)$ equals $(x^{L_j}, x^{H_j}) \in \mathbb{G}^2$. At the end of the computation, the actual multiplication $R_2 R_0$ hence evaluates to $x^k \in \mathbb{G}$.

3.1 Security Analysis

Some masking elements $r \in \mathbb{G}$ exhibit the undesirable property that $r^{2^j} = 1$ for some $j \in \mathbb{N}$. For such elements, (R_0, R_1) is permanently unmasked after j iterations.

```
Input: $x \in \mathbb{G}$, $k = \sum_{i=0}^{t-1} k_i 2^i \in \mathbb{N}$,
       $\mathrm{CKS}_{\mathrm{ref}}$ the checksum of $k$.
Output: $x^k \in \mathbb{G}$
---
Pick a random $r \in \mathbb{G}$
$R_0 \leftarrow r$; $R_1 \leftarrow rx$; $R_2 \leftarrow r^{-1}$
init(CKS)
for $j = t-1$ down to $0$ do
    $R_{\bar{k_j}} \leftarrow R_{\bar{k_j}} R_{k_j}$
    $R_{k_j} \leftarrow R_{k_j}^2$
    $R_2 \leftarrow R_2^2$
    update(CKS, $k_j$)
end for
$R_2 \leftarrow R_2 \oplus \mathrm{CKS} \oplus \mathrm{CKS}_{\mathrm{ref}}$
return $R_2 R_0$
```

Fig. 5. Side-channel analysis and fault attacks resistant Montgomery ladder

Definition 1 (Weak mask). *Let $\mathcal{W}_{\mathbb{G}} = \bigcup_{i \in \mathbb{N}} \left\{ x \in \mathbb{G} \mid x^{2^i} = 1 \right\}$. Any element $x \in \mathcal{W}_{\mathbb{G}}$ is called a weak mask.*

Theorem 1 (Weak mask probability in finite abelian groups). *Let $\mathbb{G}$ be a finite abelian group with $|\mathbb{G}| = \alpha 2^{\beta}$ for some odd α. Let $\Pr_{r \leftarrow \mathbb{G}} \{r \in \mathcal{W}_{\mathbb{G}}\}$ denote the probability that r be a weak mask when r is picked randomly uniformly from $\mathbb{G}$. We have*

$$\Pr_{r \leftarrow \mathbb{G}} \{r \in \mathcal{W}_{\mathbb{G}}\} = \frac{1}{\alpha} \ .$$

Proof. See Appendix A.

In our context, the fraction $1/\alpha$ where α denotes the greatest odd factor of $|\mathbb{G}|$ is necessarily negligible. Otherwise, $|\mathbb{G}|$ would be smooth and the discrete logarithm in $\mathbb{G}$ would be efficiently solved by the Pohlig–Hellman algorithm [10].

Suppose $\beta = 100$ and $|\mathbb{G}| \simeq 2^{1024}$. Then, the probability of picking a weak mask is about $1/2^{924} < 10^{-278}$. This shows that weak masks never happen in practice.

Simple side-channel analysis and safe-error attacks The square-and-multiply algorithm (Fig. 1) is sensitive to simple side-channel analysis. Indeed, it contains a conditional branching on the multiplication that directly depends on the secret exponent. Then, since the physical leakage of a square can be distinguished from that of a multiplication, the secret data can be easily retrieved from just one acquisition.

The square-and-multiply-always algorithm (Fig. 2) perfectly balances the former conditional branching by adding dummy multiplications. However, it introduces a specific weakness toward safe-error attacks that consist in carefully injecting a fault during the execution and checking whether it impacts on the result. In particular, the so-called M-safe-error attack consist in disturbing the multiplication. The value of the secret exponent can then be retrieved by distinguishing between required and dummy multiplications, corresponding to an exponent bit equal to 1 and 0, respectively.

Because of its high regularity, the Montgomery ladder algorithm (Fig. 3) is intrinsically resistant to simple side-channel attacks. It is also insensitive to safe-error attacks [8]. If a fault is injected at any time during the computation, the result is necessarily faulty. As it keeps the same structure, our algorithm (Fig. 4) clearly remains equivalent to Montgomery ladder in terms of simple side-channel analysis and fault attack resistance.

Differential side-channel analysis The intermediate variables are masked by r^{2^i} at each step i of the computation, and are hence statistically independant from the input and the output throughout the computation, so they cannot be exploited by the attacker.

Only those acquisitions for which r is a weak mask may be relevant to an attacker. In this case, the intermediate variables are unmasked after some steps of our algorithm. Clearly, the expected number of acquisitions to mount a differential side-channel attack against our algorithm grows inversely proportional with the probability that r be a weak mask. Since the probability of picking a weak mask is negligible, differential side-channel attacks are infeasible in practice.

Fault attacks The resistance of our algorithm against fault attacks is based on the relationship $R_2(R_0, R_1) = (x^\kappa, x^{\kappa+1})$ for some $\kappa \in \mathbb{N}$. If an error occurs on a temporary result or during one of the group operations at any time during the computation, the mutual coherence of R_0, R_1, and R_2 is definitively lost. As a consequence, the result of the last multiplication R_2R_0 is just some perfect random number to an attacker that cannot be exploited as such, at least if we assume the input was not blinded with a weak mask. Again, as weak masks are extremely unlikely in practice, any such error will be caught by our countermeasure.

However, as in [11], Fig. 4 algorithm fails to thwart exponent or loop counter disturbance as it does preserve the former relationship. Such faults hence have to be handled by other techniques. As pointed out in [12], avoiding conditional branching is safer since modifying the result of a comparison or the value of a loop counter by tampering with the associated register is easy. On the contrary, in order to by-pass an instruction, the attacker would have to increment the program counter. Such a precise modification is hardly feasible in practice. For that reason, we propose combining the on-the-fly checksum computation of [11] with the infective computation technique of [12] (Fig. 5). Let $\gamma = \mathrm{CKS} \oplus \mathrm{CKS}_{\mathrm{ref}}$ be the difference between the re-computed checksum CKS and the reference checksum value $\mathrm{CKS}_{\mathrm{ref}}$. The most significant bits of R_2 are xored with γ before the last multiplication. Hence, the final result will be spoiled whenever $\gamma \neq 0$ i.e. whenever the exponent or the loop counter has been tampered with.

3.2 Efficiency Analysis

Time Montgomery ladder (Fig. 3) requires t multiplications and t squarings. Our algorithm (Fig. 4) requires t more squarings for computing the compensative factor. The inversion and the multiplication involved in the initialization phase as well as the final multiplication can be neglected with respect to the cost of the overall computation. Let M denote the cost of a multiplication. The cost of a squaring can be approximated to $0.8M$. Each step of our algorithm costs $2.6M$ compared to $1.8M$ for Montgomery ladder, that is a 44.44% time complexity increase.

Storage Compared to Montgomery ladder, our algorithm requires one more register R_2 for the compensative factor, that is a 50% storage increase.

Note however that many cryptographic co-processors cannot store the result of some operations – as the modular multiplication or squaring – at the address of the operands. With such architectures, three registers for the standard Montgomery ladder and four registers for our algorithm are needed, corresponding to a 33% storage increase.

4 Conclusion

This paper presents an algorithm for computing exponentiations in any finte abelian group, especially relevant in the RSA and ECC setting, that is intrinsically resistant to all known simple and differential side-channel analysis and fault attacks, while requiring at most 50% more time and storage compared to traditional balanced implementations.

Our countermeasure is especially suited when only the parameters needed for the computation itself are known, which is extremely valuable as additional parameters are rarely available to the cryptographic device. In particular, neither the group order nor the public exponent are required.

Acknowledgment

The authors would like to thank Jean Creignou for many helpful remarks on the preliminary version of this paper.

A Proof of Theorem 1

Lemma 1 (Cauchy's Lemma). *Any finite group whose order is divisible by a prime number p contains an element of order p.*

Definition 2. *Let $\mathbb{G}$ be a finite abelian group and p be a prime number. Let $\mathbb{G}_p$ denote the subgroup of all elements of $\mathbb{G}$ whose order is a power of p. Any element $x \in \mathbb{G}_p$ is called a p-torsion element of $\mathbb{G}$.*

Lemma 2. *Let $\mathbb{G}$ be a finite abelian group. We have*

$$\mathbb{G} \cong \prod_{p \mid |\mathbb{G}|} \mathbb{G}_p \ .$$

Proof. Let $|\mathbb{G}| = \prod_{i=1}^{n} p_i^{\beta_i}$ where $p_{i,i\in\{1,\ldots,n\}}$ are prime numbers.

Let us show that the homomorphism ψ defined as

$$\prod_{i=1}^{n} \mathbb{G}_{p_i} \xrightarrow{\psi} \mathbb{G}$$

$$(x_1, \ldots, x_n) \longmapsto \prod_{i=1}^{n} x_i$$

is an isomorphism. First, we show that ψ is a monomorphism.

Let x and y be in the abelian group $\mathbb{G}$. Let $|\langle x \rangle| = a$ be the order of x and $|\langle y \rangle| = b$ the order of y in $\mathbb{G}$. First, observe that if a and b are coprime, then $xy = 1 \Rightarrow x = y = 1$. This is a consequence of Bézout's identity. Since a and b are coprime, there exists integers u and v such that $au + bv = 1$. We have

$$xy = 1 \Rightarrow (xy)^{au} = 1 \tag{1}$$

and as $x^a = 1$, we have $(xy)^{au} = x^{au}y^{au} = y^{au}$. Then, since $y^b = 1$, we get

$$(xy)^{au} = y^{au} = y^{au}y^{bv} = y^{au+bv} = y \ . \tag{2}$$

From (1) and (2), we have $y = 1$. In the same way, we show $xy = 1 \Rightarrow x = 1$.

Now let us suppose that $\psi(x_1, \ldots, x_n) = \prod_{i=1}^{n} x_i = 1$. Clearly the order of x_1 and the order of $\prod_{i=2}^{n} x_i$ are coprime, and as we have shown above, $x_1 \times \prod_{i=2}^{n} x_i = 1 \Rightarrow x_1 = 1$ and $\prod_{i=2}^{n} x_i = 1$. In particular $x_1 = 1$. Then, by induction on the relation $\prod_{i=2}^{n} x_i = 1$, we get the expected result

$$\psi(x_1, \ldots, x_n) = \prod_{i=1}^{n} x_i = 1 \Rightarrow x_1 = \ldots = x_n = 1 \ .$$

Now let us show that ψ is an epimorphism.

For all $y \in \mathbb{G}$, $|\langle y \rangle|$ divides $|\mathbb{G}|$. Hence, $|\langle y \rangle| = \prod_{i=1}^{n} p_i^{\gamma_i}$ where $\gamma_i \leq \beta_i$ for all $i \in \{1, \ldots, n\}$. Let $u_i = \prod_{j \neq i} p_j^{\gamma_j}$ for $i \in \{1, \ldots, n\}$. Then, $y^{u_i} \in \mathbb{G}_{p_i}$ since $|\langle y^{u_i} \rangle| = p_i^{\gamma_i}$. Moreover, the $u_{i,i\in\{1,\ldots,n\}}$ are coprime as there exists no integer dividing all u_i. According to Bézout's identity, there exists $a_1, \ldots, a_n$ such that $\sum_{i=1}^{n} a_i u_i = 1$. Hence, for all $y \in \mathbb{G}$, $y = \prod_{i=1}^{n} x_i$ with $x_i = y^{u_i a_i} \in \mathbb{G}_{p_i}$. □

Lemma 3. *Let $\mathbb{G}$ be a group with $|\mathbb{G}| = \prod_{i=1}^{n} p_i^{\beta_i}$ where $p_{i,i=1\ldots n}$ are prime numbers. Then, $|\mathbb{G}_{p_i}| = p_i^{\beta_i}$.*

Proof. Necessarily, $|\mathbb{G}_{p_i}|$ is a power of p_i, say $p_i^{\gamma_i}$. Indeed, from Cauchy's Lemma, if $|\mathbb{G}_{p_i}|$ were divisible by some prime number $p \neq p_i$, it would contain an element of order p, which is contradictory with the definition of $\mathbb{G}_{p_i}$. Then, since $\mathbb{G} \cong \prod_{p \mid |\mathbb{G}|} \mathbb{G}_p$, we have $\prod_i p_i^{\beta_i} = \prod_i p_i^{\gamma_i}$, so $\gamma_i = \beta_i$ for all i. $\square$

We have $\mathcal{W}_{\mathbb{G}} = \mathbb{G}_2$ and, from lemma 3, $|\mathbb{G}_2| = 2^{\beta}$. Finally,

$$\Pr_{r \leftarrow \mathbb{G}} \{r \in \mathcal{W}_{\mathbb{G}}\} = \frac{|\mathcal{W}_{\mathbb{G}}|}{|\mathbb{G}|} = \frac{1}{\alpha} .$$

References

1. Kocher, P.C.: Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In: CRYPTO. Volume 1109 of Lecture Notes in Computer Science. (1996) 104–113
2. Boneh, D., DeMillo, R.A., Lipton, R.J.: On the Importance of Checking Cryptographic Protocols for Faults. Lecture Notes in Computer Science **1233** (1997) 37–51
3. Kocher, P., Jaffe, J., Jun, B.: Differential Power Analysis. Lecture Notes in Computer Science **1666** (1999) 388–397
4. Montgomery, P.L.: Speeding the Pollard and elliptic curve methods of factorization. Mathematics of Computation **48(177)** (January 1987) 243264
5. Coron, J.S.: Resistance Against Differential Power Analysis for Elliptic Curve Cryptosystems. In Ç.K. Koç, Paar, C., eds.: Cryptographic Hardware and Embedded Systems — CHES 2002. Volume 1717 of Lecture Notes in Computer Science. (1999) 292–302
6. Goubin, L.: A refined power analysis attack on elliptic curve cryptosystems. In Springer-Verlag, ed.: Public Key Cryptography PKC 2003. Volume 2567 of Lecture Notes in Computer Science. (2003) 199211
7. Dupuy, W., Kunz-Jacques, S.: Resistance of Randomized Projective Coordinates Against Power Analysis. In B.S. Kaliski Jr., c.K., Paar, C., eds.: Cryptographic Hardware and Embedded Systems — CHES 2005. Volume 3659 of Lecture Notes in Computer Science. (2005) 1–12
8. Joye, M., Yen, S.M.: The Montgomery Powering Ladder. In B.S. Kaliski Jr., c.K., Paar, C., eds.: Cryptographic Hardware and Embedded Systems — CHES 2002. Volume 2523 of Lecture Notes in Computer Science. (2002) 291–302
9. Trichina, E., Bellezza, A.: Implementation of elliptic curve cryptography with built-in counter measures against side channel attacks. In B.S. Kaliski Jr., c.K., Paar, C., eds.: Cryptographic Hardware and Embedded Systems — CHES 2002. Volume 2523 of Lecture Notes in Computer Science. (2002) 98–113
10. Pohlig, S.C., Hellman, M.E.: An improved algorithm for computing logarithms over GF(p) and its cryptographic significance. IEEE Transactions on Information Theory **24** (1978) 106–110
11. Giraud, C.: Fault Resistant RSA Implementation. In Breveglieri, L., Koren, I., eds.: 2nd Workshop on Fault Diagnosis and Tolerance in Cryptography — FDTC 2005. (2005) 142–151
12. Ciet, M., Joye, M.: Practical Fault Countermeasures for Chinese Remaindering Based RSA. In Breveglieri, L., Koren, I., eds.: 2nd Workshop on Fault Diagnosis and Tolerance in Cryptography — FDTC 2005. (2005) 124–131