A Hole in the Ladder: Interleaved Variables in Iterative Conditional Branching (Extended Version)

Yoann Marquer, Tania Richmond, Pascal Véron

A Hole in the Ladder:
Interleaved Variables in Iterative Conditional Branching (Extended Version)

Yoann Marquer*, Tania Richmond†, Pascal Véron‡
*TAMIS team then DiverSE team, TeamPlay project, Inria, Univ. Rennes, CNRS, IRISA, France, yoann.marquer@inria.fr,
†TAMIS team, TeamPlay project Inria, Univ. Rennes, CNRS, IRISA, France, then DGA - Maîtrise de l'Information, Bruz, France, tania.richmond@inria.fr,
‡Laboratoire IMath, Université de Toulon, France, pascal.veron@univ-tln.fr

***Abstract*—The iterative conditional branchings appear in various sensitive algorithms, like the modular exponentiation in the RSA cryptosystem or the scalar multiplication in elliptic-curve cryptography. In this paper, we abstract away the desirable security properties achieved by the Montgomery ladder, and formalize systems of equations necessary to obtain what we call the semi-interleaved and fully-interleaved ladder properties. This fruitful approach allows us to design novel fault-injection attacks, able to obtain some/all bits of the secret against different ladders, including the common Montgomery ladder. We also demonstrate the generality of our approach by applying the ladder equations to the modular exponentiation and the scalar multiplication, both in the semi- and fully-interleaved cases, thus proposing novel and more secure algorithms.**

***Index Terms*—Security and Privacy Protection, Public key cryptosystems, Computer arithmetic, Fault injection**

I. Introduction

We present common algorithms used to compute the modular exponentiation, based on an *iterative conditional branching*, where a conditional branching depending on the secret updates the value of a variable x on every iteration of one/several loop(s). Amongst these algorithms the Montgomery ladder, that uses a fresh variable y, satisfies desirable properties against side-channel or fault-injection attacks.

A. Contribution

In this paper, we formalize these properties with equations corresponding to two families of cases: the *semi-interleaved ladders* where the value of x or y may (depending on the secret) depend only on its previous value, and the *fully-interleaved ladders* where the value of x and y both depend on both previous values.

We propose 1) an attacker model using fault-injection able to obtain some bits of the secret in the semi-interleaved cases (including the Montgomery ladder) but none in the fully-interleaved cases, and 2) a stronger attacker model able to obtain all bits of the secret in the semi-interleaved cases, and even in the fully-interleaved cases if the fault can be injected in the key register.

This work was supported by the EU Horizon 2020 project *TeamPlay* (https://www.teamplay-h2020.eu), grant number 779882.

We also use our formalization to design novel algorithms for the modular exponentiation, including a semi-interleaved ladder with a mask updated at every iteration, and a fully-interleaved ladder. We demonstrate the generality of the approach by also applying the ladder equations to the scalar multiplication used in elliptic-curve cryptography (ECC), obtaining a semi-interleaved ladder suitable for practical applications, and a fully-interleaved ladder.

The ladder equations, the first two attacks and the algorithms for the modular exponentiation have been published at ARITH 2020 [1]. This paper is an extended version, including novelties like:

- the third attack (for the second attacker model) against all (including fully-interleaved) ladders (Subsection IV-B),
- in the case of the modular exponentiation a proof that the probability to obtain randomly a suitable ladder constant for our fully-interleaved ladder candidate is almost 1 when used in concrete applications like RSA or DSA cryptosystems (Subsections V-D and V-E),
- and original semi- and fully-interleaved ladder candidates for the scalar multiplication over elliptic curves (Section VI).

B. Organization of the Paper

As introduction, we present in Section II some related works on the modular exponentiation and the Montgomery ladder. In Section III we formalize the iterative conditional branching to deduce the equations satisfied by the semi-interleaved and fully-interleaved ladders, and thus the requirements for ladderizable algorithms. In Section IV we introduce two attacker models using fault injection techniques, and compare the vulnerability of the non-, semi- and fully-interleaved ladders. Finally, we detail how to produce examples of the semi- and fully-interleaved ladders in Section V for the modular exponentiation and in Section VI for the scalar multiplication.

II. Related Works

In this section, we introduce known algorithms for the modular exponentiation, their relevance regarding security, and the desirable properties of the usual Montgomery ladder.

input public a, n; secret k
1: $x \leftarrow 1$
2: **for** $i = d$ **to** 0 **do**
3: $\quad x \leftarrow x^2 \bmod n$
4: $\quad$ **if** $k[i] = 1$ **then**
5: $\quad\quad x \leftarrow ax \bmod n$
6: $\quad$ **end if**
7: **end for**
8: **return** x
output $x = a^k \bmod n$

TABLE I
SQUARE AND MULTIPLY

input public a, n; secret k
1: $x \leftarrow 1$
2: **for** $i = d$ **to** 0 **do**
3: $\quad x \leftarrow x^2 \bmod n$
4: $\quad$ **if** $k[i] = 1$ **then**
5: $\quad\quad x \leftarrow ax \bmod n$
6: $\quad$ **else**
7: $\quad\quad y \leftarrow ax \bmod n$
8: $\quad$ **end if**
9: **end for**
10: **return** x
output $x = a^k \bmod n$

TABLE II
SQUARE AND MULTIPLY ALWAYS

input public a, n; secret k
1: $x \leftarrow 1$
2: $y \leftarrow a \bmod n$
3: **for** $i = d$ **to** 0 **do**
4: $\quad$ **if** $k[i] = 1$ **then**
5: $\quad\quad x \leftarrow xy \bmod n$
6: $\quad\quad y \leftarrow y^2 \bmod n$
7: $\quad$ **else**
8: $\quad\quad y \leftarrow xy \bmod n$
9: $\quad\quad x \leftarrow x^2 \bmod n$
10: $\quad$ **end if**
11: **end for**
12: **return** x
output $x = a^k \bmod n$

TABLE III
MONTGOMERY LADDER

1: $x \leftarrow$ init
2: **for** $i = 1$ **to** n **do**
3: $\quad \ddots$
4: $\quad$ **if** secret **then**
5: $\quad\quad x \leftarrow \theta(x)$
6: $\quad$ **else**
7: $\quad\quad x \leftarrow \varepsilon(x)$
8: $\quad$ **end if**
9: $\quad \iddots$
10: **end for**

TABLE IV
ITERATIVE CONDITIONAL BRANCHING

A. Modular Exponentiation

Let k be a secret key, and $k = \sum_{0 \le i \le d} k[i]\, 2^i$ be its binary expansion of size $d + 1$, i.e. $k[i]$ is the bit i of k. The *square-and-multiply* algorithm described in Table I computes the (left-to-right) modular exponentiation $a^k \bmod n$, by using $a^{\sum_{0 \le i \le d} k[i] 2^i} = \prod_{0 \le i \le d} (a^{2^i})^{k[i]}$. This exponentiation is commonly used in cryptosystems like RSA [2].

For every iteration, the multiplication ax is computed only if $k[i] = 1$, which can be detected by observing execution time[1] [3] or power profiles[2] by means of e.g. SPA (Simple Power Analysis[3]) [9], and thus leads to information leakage from both time and power side-channel attacks.

To prevent SPA, regularity of the modular exponentiation algorithms is required, which means that both branches of the sensitive conditional branching perform the same operations, independently from the value of the exponent. Thus, an `else` branch is added with a dummy instruction [7] in the *square-and-multiply-always* algorithm described in Table II.

But countermeasures developed against a given attack may benefit another one [10]. Because the multiplication in the `else` branch of this algorithm is a dummy operation, a fault injected [11] in the register containing ax will eventually propagate through successive iterations and alter the final result only if $k[i] = 1$, thus leaking some information. Therefore, an attacker (see the attacker models in Section IV) able to inject a fault in a given register at a given iteration can obtain the digits of the secret key by comparing the final output with or without fault (technique known as safe-error attack).

B. Montgomery Ladder

This is not the case in the algorithm proposed by Montgomery [12] and described in Table III, where a fault injected in a register will eventually propagate to the other one, and thus will alter the final result in any case, preventing the attacker to obtain information. But, as described in Section IV, some information on the last digits may still leak, weakening the protection obtained from the ladder.

[1]Even observing the duration of the whole execution can leak the Hamming weight of the secret key, and thus can narrow down the exploration space.

[2]A multiplication can be distinguished from a squaring, hence variants with squaring only have been proposed in [4] and improved in [5].

[3]The power profile depends also on the values in the considered registers, so computing a multiplication in every case is better against SPA but not against CPA (Correlation Power Analysis) [6], even if standard blinding techniques can prevent differential attacks [7], [8].

The Montgomery ladder is algorithmically equivalent [13] to the square-and-multiply(-always) algorithm(s), in the sense that x has the same value for every iteration. Actually, some variants [14] of the square-and-multiply-always algorithm[4] may be as resistant as the Montgomery ladder [18], both by checking invariants [19] violated if a fault is injected. In the case of the Montgomery ladder, the invariant $y = ax$ is satisfied for every iteration. These invariants are important for the self-secure exponentiation countermeasures [20].

Note that the `else` branch in Table III is identical to the `then` branch, except that x and y are swapped, which provides also (partial[5]) protection against timing and power leakage. Moreover, the variable dependency makes these variables interleaved, so this exponentiation is algorithmically (but partially, as we will demonstrate) protected against safe-error attacks. Finally, as opposed to square-and-multiply-always in Table II, the code in the `else` branch in Table III is not dead, so will not be removed by compiler optimizations.

III. LADDER EQUATIONS

In this section, we formalize the iterative conditional branching occuring in algorithms like the Montgomery ladder, and deduce the requirements to optimize them with semi- or fully-interleaved ladders.

A. Iterative Conditional Branching

In this paper, we focus on algorithms as in Table IV called *iterative conditional branching*. It appears in algorithms like square-and-multiply for the modular exponentiation, its counterpart double-and-add for elliptic curve point multiplication [21], [22], or the secure bit permutation in the McEliece cryptosystem [23] attacked in [24]. It also appears naturally (Table V) when trying to turn a loop on the secret into a conditional branching depending on the secret, that can itself be balanced to remove or reduce the dependency on the secret.

But our approach does not depend on the number/depth of the considered loops, hence the dots in Table IV. We assume only that the conditional branching uses only one variable x, the multivariate case being future work (see Section VII).

[4]See [15] for highly regular right-to-left variants, [16] for a generalization to any basis and left-to-right/right-to-left variants, and [17] for their duality.

[5]The variables x and y may have different access time, which hinders protection against cache-timing leakage.

```
1: assert(secret ≤ bound)
2: for i = 0 to secret do
3:     ⋮
4: end for
```

```
1: for i = 0 to bound do
2:     if i ≤ secret then
3:         ⋮
4:     end if
5: end for
```

TABLE V
LOOP BOUNDED BY A SENSITIVE VARIABLE

```
1: x ← init
2: y ← ℓ(init)
3: for i = 1 to n do
4:     ...
5:     if secret then
6:         x ← f(x, y)
7:         y ← ε(y)
8:     else
9:         y ← f(y, x)
10:        x ← ε(x)
11:    end if
12:    ...
13: end for
```

TABLE VI
SEMI-INTERLEAVED LADDERS

```
1: x ← init
2: y ← ℓ(init)
3: for i = 1 to n do
4:     ...
5:     if secret then
6:         x ← f(x, y)
7:         y ← g(x, y)
8:     else
9:         y ← f(y, x)
10:        x ← g(y, x)
11:    end if
12:    ...
13: end for
```

TABLE VII
FULLY-INTERLEAVED LADDERS

Definition 1 (Iterative Conditional Branching)**.** An algorithm as in Table IV is said with an (univariate) *iterative conditional branching* with two (unary) functions θ and ε.

B. Semi-Interleaved Ladders

To prevent information leakage from side-channel analysis or fault injections, we use another variable y in the algorithm described in Table VI. As in the Montgomery ladder (see Table III), we need to find two functions ℓ and f such that for every iteration:

- $y = \ell(x)$, and
- x has the same value for every iteration as in Table IV.

$y = \ell(x)$ is satisfied at the initialization. By induction, let's assume $y = \ell(x)$ at the beginning of an iteration. In the `then` branch[6] we have $x \leftarrow f(x, y)$ then $y \leftarrow \varepsilon(y)$, thus in order to have $y = \ell(x)$ satisfied at the end of an iteration the following equation must hold:

$$\forall x, \varepsilon(\ell(x)) = \ell(f(x, \ell(x)))$$

and to have x updated to $\theta(x)$ during the iteration the following equation must hold:

$$\forall x, f(x, \ell(x)) = \theta(x)$$

In the `else` branch we have $y \leftarrow f(y, x)$ then $x \leftarrow \varepsilon(x)$, thus in order to have $y = \ell(x)$ satisfied at the end of an iteration the following equation must hold:

$$\forall x, f(\ell(x), x) = \ell(\varepsilon(x))$$

and x is already updated to $\varepsilon(x)$ during the iteration.

Definition 2 (Semi-Ladderizable)**.** Let A be an algorithm with a univariate iterative conditional branching with two unary functions denoted θ and ε. A is *semi-ladderizable* if there exists a unary function ℓ and a binary function f such that, for every considered value x:

$$\begin{cases} \varepsilon(\ell(x)) = \ell(\theta(x)) & (1) \\ f(x, \ell(x)) = \theta(x) & (2) \\ f(\ell(x), x) = \ell(\varepsilon(x)) & (3) \end{cases}$$

For the square-and-multiply algorithm we have $\theta(x) = ax^2$ and $\varepsilon(x) = x^2$, and we know that it can be semi-ladderized by using the Montgomery ladder with $\ell(x) = ax$ and $f(x, y) = xy$, but we demonstrate in Section V that there are other solutions. Note that to respect the form of the semi-interleaved ladder, we should have written $y \leftarrow yx$ and not $y \leftarrow xy$ in the `else` branch[7] of the Montgomery ladder.

C. Fully-Interleaved Ladders

Unfortunately, the semi-interleaved ladder is vulnerable to fault injection techniques, because in every branch at least one variable depends only on its previous value and not the previous value of both variables (see Section IV). Moreover, an attacker able to determine whether the output of one operation is used as the input to another one can [26] apply collision attacks[8] to deduce whether two following bits are the same. To solve these issues, we propose in Table VII a *fully-interleaved ladder* using three functions ℓ, f and g.

As for the semi-interleaved ladders, $y = \ell(x)$ is satisfied at the initialization. We assume again by induction that $y = \ell(x)$ at the beginning of an iteration. In the `then` branch we have $x \leftarrow f(x, y)$ then $y \leftarrow g(x, y)$, thus in order to have $y = \ell(x)$ satisfied at the end of an iteration the following equation must hold:

$$\forall x, g(f(x, \ell(x)), \ell(x)) = \ell(f(x, \ell(x)))$$

and to have x updated to $\theta(x)$ during the iteration the following equation must hold:

$$\forall x, f(x, \ell(x)) = \theta(x)$$

In the `else` branch we have $y \leftarrow f(y, x)$ then $x \leftarrow g(y, x)$, thus, in order to have $y = \ell(x)$ satisfied at the end of an iteration, the following equation must hold:

$$\forall x, f(\ell(x), x) = \ell(g(f(\ell(x), x), x))$$

and to have x updated to $\varepsilon(x)$ during the iteration the following equation must hold:

$$\forall x, g(f(\ell(x), x), x) = \varepsilon(x)$$

Definition 3 (Fully-Ladderizable)**.** Let A be an algorithm with a univariate iterative conditional branching with two unary functions denoted θ and ε. A is *fully-ladderizable* if there

[6]Because the condition depends on the secret, we will not assume that in the `then` branch the condition is satisifed but not in the `else` branch, to prevent data dependencies that could be attacked.

[7]The former is actually better regarding vulnerability to the M safe-error [18] or collision [25] attacks, showing that our methodology is good practice.

[8]A countermeasure proposed in [27] is to randomly blend variants of the ladder, or compute the exponentiation by taking a random (bounded) walk.

exists a unary function ℓ and two binary functions f and g such that, for every considered input value x:

$$\begin{cases} g(\theta(x), \ell(x)) = \ell(\theta(x)) & (4) \\ f(x, \ell(x)) = \theta(x) & (5) \\ f(\ell(x), x) = \ell(\varepsilon(x)) & (6) \\ g(f(\ell(x), x), x) = \varepsilon(x) & (7) \end{cases}$$

Note that Equation (2) is Equation (5) and Equation (3) is Equation (6). Without surprise, if g is chosen such that $g(x, y) = \varepsilon(y)$ then Equation (1) is a special case of Equation (4), and Equation (7) is satisfied. Thus, semi-interleaved ladders are subcases of fully-interleaved ladders.

D. Ladderizable Algorithms

Theorem 1. *Let A be an algorithm with an iterative conditional branching with two unary functions denoted θ and ε. If A is semi-ladderizable with ℓ and f, or fully-ladderizable with ℓ, f and g, then for every iteration of the ladder variant:*

- $y = \ell(x)$
- *x is updated as in A:*

$$x \leftarrow \begin{cases} \theta(x) & \textit{if secret} \\ \varepsilon(x) & \textit{otherwise} \end{cases}$$

Thus, for every algorithm with an iterative conditional branching, if there exists ℓ and f satisfying the equations in Definition 2 then the conditional branching can be rewritten as a semi-interleaved ladder. Even better, if there exists ℓ, f and g satisfying the equations in Definition 3 then it can be rewritten as a fully-interleaved ladder.

Because in a semi- or fully-interleaved ladder the operations performed are the same for both the `then` and the `else` branches, this transformation is an algorithmic countermeasure against side-channel attacks [3], [9]. We detail in Section IV the impact of the semi- and fully-interleaved ladderization against fault injection techniques. Then, to demonstrate the concept, we construct examples of semi- and fully-interleaved ladders in Sections V and VI.

IV. Fault Injection

In this section, we introduce two attacker models using fault injection techniques, and compare the vulnerability of the non-, semi- and fully-interleaved ladders.

According to [11] a *fault* is a physical defect, imperfection or flaw that occurs within some hardware or software component, while an *error* is a deviation from accuracy or correctness, and is the manifestation of a fault. Hardware/physical faults can be permanent, transient or intermittent, while software faults are the consequence of incorrect design, at specification or at coding time. *Fault injection* is defined [28] as the validation technique of the dependability of fault tolerant systems, which consists in performing controlled experiments where the observation of the system's behavior in presence of faults is induced explicitly by the writing introduction (called injection) of faults in the system.

A. First Attacker Model

For the iterative conditional branching (Table IV) and the semi- and fully-interleaved ladders (Tables VI and VII), we assume as in the Montgomery ladder (Table III) that "secret" is the condition $k[i] = 1$, where $k[i]$ is the i-th bit of the secret key k.

Definition 4 (First Attacker Model)**.** We assume that:

- The attacker wants to obtain the secret key stored in the chip and copied in the register k.
- The attacker can run the program any number of times:
  - inputting x_{init} and y_{init}, the initial values in the register x and y,
  - obtaining x_{final} and/or y_{final}, the final value(s) returned by the program.
- A run consists of iterations i over[9] $1, \ldots, n$, where:
  - $k[i]$ is the i-th bit of k,
  - x_i (resp. y_i) denotes the value in the register x (resp. y) between the iterations $i-1$ and i.
- The attacker can $\lightning x_i$ (resp. $\lightning y_i$) inject a random fault[10] (the affected variable is set to a random value) in the register of x (resp. y) between the iterations $i-1$ and i.

The attacker might be unlucky and obtain the same value as before, but this is unlikely and can be fixed by several tries, so we will assume for simplicity that the new value is different.

The attacker can run a program for the square-and-multiply(-always) (Tables I and II) algorithm(s) for a given input x_{init}, obtain a value x_{final}, then run the program again with the same input while injecting a fault $\lightning y_i$ in the register $y = ax$ between the iterations $i-1$ and i, and obtain a value x_{fault}. If $x_{\text{fault}} = x_{\text{final}}$ then $k[i] = 0$, otherwise $k[i] = 1$. This can be done for every iteration (in any order), thus the attacker can obtain that way all the bits of the secret key.

In this algorithm, because the current value in x determines the next value in x, a faulted value $\lightning x_i$ for an iteration i always propagates to the next iteration $\lightning x_{i+1}$, which we denote by $\lightning x_i \Rightarrow \lightning x_{i+1}$. To obtain the bit $k[i]$, the attacker has exploited the fact that a fault in $y = ax$ propagates to x, denoted by $\lightning y_i \Rightarrow \lightning x_{i+1}$, only if $k[i] = 1$. As opposed to this non-ladderized variant, in Tables VI and VII the values of x and y are interleaved, and the fault propagation patterns are the following:

1) For the semi-interleaved ladder:

$$\begin{aligned} &\lightning x_i \Rightarrow \lightning x_{i+1} \text{ and } (\lightning y_{i+1} \text{ only if } k[i] = 0) \\ &\lightning y_i \Rightarrow \lightning y_{i+1} \text{ and } (\lightning x_{i+1} \text{ only if } k[i] = 1) \end{aligned}$$

which means that a fault always propagates to the same register, but propagates to the other depending on the current bit of the secret key.

[9] To simplify the notations, we assume in this section that the counter is incremented (from 1 to n with a step 1), but the argument is similar for other initial or final values, a decremented counter, and/or other step values.

[10] In the Montgomery ladder (Table III) the invariant $y = ax$ is satisfied for every iteration. So, if a random fault is injected in x or y, the violation of the invariant allows the algorithm to detect the fault [20] and thus to enhance the appropriate fault policy, as stopping the program or switching to a random key for the rest of the computation. The same argument holds for our semi- and fully-interleaved ladders with the invariant $y = \ell(x)$.

2) For the fully-interleaved ladder:

$$\lightning x_i \text{ or } \lightning y_i \Rightarrow \lightning x_{i+1} \text{ and } \lightning y_{i+1}$$

which means that any fault in one register propagates in every case to both.

The fully-interleaved ladder has a lower *fault tolerance*, i.e. it is easier to disrupt the computation. This is not convenient for properties like functionality, availability or redundancy, but prevents an attacker from obtaining the secret key, thus increases security. Thus, we reduced the information leakage from fault injection by reducing also the fault tolerance. The semi-interleaved ladder is more robust, but this comes at the price of a fault propagation pattern depending on the secret key, which can be exploited.

Indeed, the attacker can $\lightning y_n$ as in Figure 1 and compare the x output. We assume, as in the exponentiation examples, that changing a read intermediate value would lead to a different output, because if this was not the case then the attacker should have used better x_{init} and y_{init} inputs. So, if $x_{\text{fault}} = x_{\text{final}}$ then $k[n] = 0$, otherwise $k[n] = 1$. Thus, the attacker can obtain $k[n]$, the last bit of the secret key.

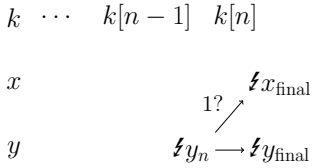

Fig. 1. Attack the Last Bit

If the obtained bit was 0 as in the left part of Figure 2, the attacker can $\lightning y_{n-1}$. In that case if $x_{\text{fault}} = x_{\text{final}}$ then $k[n-1] = 0$, otherwise $k[n-1] = 1$. This process can be repeated until a 1 is found. If the obtained bit was 1 as in the right part of Figure 2, the attacker can $\lightning x_{n-1}$. In that case if $y_{\text{fault}} = y_{\text{final}}$ then $k[n-1] = 1$, otherwise $k[n-1] = 0$. This process can be repeated until a 0 is found.

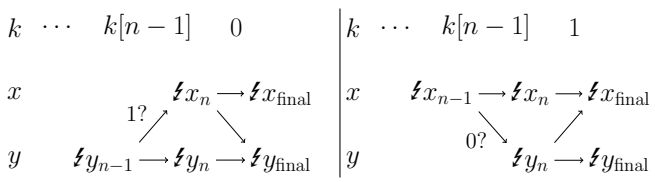

Fig. 2. Attack the Penultimate Bit

One may think that these processes can be alternated in order to recover all the bits of the secret key, but if there is a bit alternation, i.e. $(k[i], k[i+1]) = (0,1)$ or $(1,0)$, then the bits before i cannot be obtained, as illustrated in Figures 3 and 4. So, the attacker can obtain the final bits $10\ldots0$ if x_{final} can be read, and the final bits $01\ldots1$ if y_{final} can be read.

The vulnerability to fault injection is summarized in Table X. Against the first attacker model described in Definition 4, fully-interleaved ladders are more secure than semi-interleaved ladders, which are more secure than without interleaving at all. But we show in the next subsection that a stronger attacker is able to obtain all the bits of the key from both the semi- and fully-interleaved ladders.

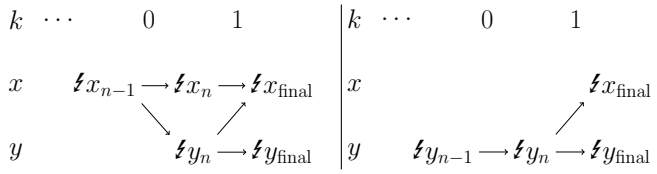

Fig. 3. $\lightning x$ or $\lightning y$ before 01

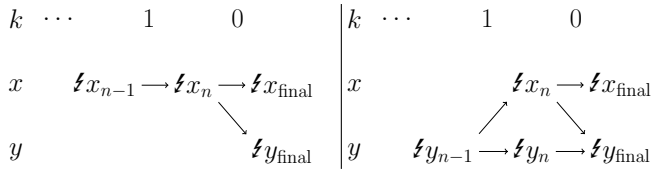

Fig. 4. $\lightning x$ or $\lightning y$ before 10

B. Second Attacker Model

Definition 5 (Second Attacker Model)**.** We assume that:

- The second attacker has the same goal and means that the attacker in Definition 4.
- The second attacker can also $\lightning k_{>i} = 0$ (resp. $\lightning k_{>i} = 1$) *stuck-at* [11] 0 (resp. 1) all the bits of the register k between iterations i and $i+1$.

This stronger attacker is able to break the semi-interleaved ladders by using the attack protocol described in Table VIII in order to recover all the bits of the secret key[11]. EXE($x_{\text{init}}, y_{\text{init}}, \lightning k_{>i} = 1$) at Line 4 means that the studied program is executed with inputs $x_{\text{init}}, y_{\text{init}}$ and a stuck-at $\lightning k_{>i} = 1$, and EXE($x_{\text{init}}, y_{\text{init}}, \lightning k_{>i} = 1, \lightning x_i$) at Line 5 is the same but with a fault $\lightning x_i$. In particular, if both x_{final} and y_{final} can be read, then the Montgomery Ladder (Table III) can be broken (by iterating over 0 to d). This attack does not work against fully-interleaved ladders, but a fully-interleaved ladder is more difficult to obtain (when possible), as discussed in Subsections V-C and VI-C.

Actually, being able to write deterministically in the key register is enough to obtain with a third attack all the bits of the key, even against fully-interleaved ladders. Indeed, let's assume that the second attacker knows the first i bits $k[1]\ldots k[i]$, where $i = 0$ for the initialization. By $\lightning k_{>i} = 0$ (resp. $\lightning k_{>i} = 1$) and $\lightning k_{>i+1} = 0$ (resp. $\lightning k_{>i+1} = 1$) the second attacker can compare $x_{\text{fault}0}$ and $x_{\text{final}0}$ (resp. $x_{\text{fault}1}$ and $x_{\text{final}1}$) obtained respectively for $k[1]\ldots k[i]\,0\,0\ldots0$ and $k[1]\ldots k[i]\,k[i+1]\,0\ldots0$ (resp. with 1 instead of 0 at the end), and the same for y. Thus, the second attacker can (almost) determine whether $k[i+1] = 1$ (resp. 0):

- if $x_{\text{fault}0} \neq x_{\text{final}0}$ or $y_{\text{fault}0} \neq y_{\text{final}0}$ then $k[i+1] = 1$,
- if $x_{\text{fault}1} \neq x_{\text{final}1}$ or $y_{\text{fault}1} \neq y_{\text{final}1}$ then $k[i+1] = 0$.

If $x_{\text{fault}0} = x_{\text{final}0}$, $x_{\text{fault}1} = x_{\text{final}1}$, $y_{\text{fault}0} = y_{\text{final}0}$ and $y_{\text{fault}1} = y_{\text{final}1}$ then another inputs $x_{\text{init}}, y_{\text{init}}$ should be chosen in the protocol described in Table IX.

That way all the bits of the key k can be obtained, except the ones such that for every possible x_{init}, $x_{\text{fault}0} = x_{\text{final}0}$ and $x_{\text{fault}1} = x_{\text{final}1}$. But this case is unlikely, it does not prevent

[11]Note that the iterations are reversed in this attack protocol.

```
FIx ← 0
for i = n to 1 do
    if FIx = 1 then
        (x_final, y_final) ← EXE(x_init, y_init, ↯k_{>i} = 1)
        (x_fault, y_fault) ← EXE(x_init, y_init, ↯k_{>i} = 1, ↯x_i)
        if y_fault = y_final then
            k[i] ← 1
        else
            k[i] ← 0
            FIx ← 0
        end if
    else
        (x_final, y_final) ← EXE(x_init, y_init, ↯k_{>i} = 0)
        (x_fault, y_fault) ← EXE(x_init, y_init, ↯k_{>i} = 0, ↯y_i)
        if x_fault = x_final then
            k[i] ← 0
        else
            k[i] ← 1
            FIx ← 1
        end if
    end if
end for
return k
```

TABLE VIII
PROTOCOL TO ATTACK THE SEMI-INTERLEAVED LADDERS

```
for i = 0 to n − 1 do
    (x_final, y_final) ← EXE(x_init, y_init, ↯k_{>i} = 0)
    (x_fault, y_fault) ← EXE(x_init, y_init, ↯k_{>i+1} = 0)
    if x_fault ≠ x_final or y_fault ≠ y_final then
        k[i + 1] ← 1
    else
        (x_final, y_final) ← EXE(x_init, y_init, ↯k_{>i} = 1)
        (x_fault, y_fault) ← EXE(x_init, y_init, ↯k_{>i+1} = 1)
        if x_fault ≠ x_final or y_fault ≠ y_final then
            k[i + 1] ← 0
        end if
    end if
end for
return k
```

TABLE IX
PROTOCOL TO ATTACK BOTH SEMI- AND FULLY-INTERLEAVED LADDERS

the attacker from discovering the other bits, and these bits make no difference in the result, thus probably the attacker does not care. Therefore, the second attacker can obtain all the (relevant) bits of the key, even against semi- and fully-interleaved ladders. The vulnerabilities against these attacks are summarized in Table X.

	Obtained Bits		
	Model 1	Model 2	
Interleaving	Attack 1	Attack 2	Attack 3
non	all	all	all
semi	some	all	all
fully	none	none	all

TABLE X
VULNERABILITY AGAINST FAULT-INJECTION ATTACKS

V. EXTENDING THE MONTGOMERY LADDER

The purpose of this section is to provide concrete examples of the ladder equations for cryptography, and to generalize the idea behind the Montgomery ladder to improve its protection against side-channel and fault injection attacks.

We demonstrate that by using the modular exponentiation. In the following for any n we consider $\mathbb{Z}/n\mathbb{Z}$, i.e. integers modulo n, and denote $(\mathbb{Z}/n\mathbb{Z})^\star$ the integers invertible modulo n. We assume that $\theta(x)$, $\varepsilon(x)$, $f(x,y)$, $g(x,y)$ and $\ell(x)$ are quadratic polynomials with the following coefficients:

$$\begin{aligned}
\theta(x) &= ax^2 \\
\varepsilon(x) &= x^2 \\
f(x,y) &= f_{20}x^2 + f_{11}xy + f_{02}y^2 + f_{10}x + f_{01}y + f_{00} \\
g(x,y) &= g_{20}x^2 + g_{11}xy + g_{02}y^2 + g_{10}x + g_{01}y + g_{00} \\
\ell(x) &= \ell_1 x + \ell_0
\end{aligned}$$

More general quadratic polynomials, e.g. $\ell_2 \neq 0$ or more complex $\theta(x)$ and $\varepsilon(x)$, can be investigated but the general systems of equations tend to be complicated, and in this section we want to focus on the exponentiation algorithms. To ensure that $\theta(x) = ax^2$ and $\ell(x) = \ell_1 x + \ell_0$ depends on x, we will assume that $a \neq 0$ and $\ell_1 \neq 0$. Moreover, in the following we focus on solutions without constraint on a to preserve the generality of the original algorithm.

Finally, note that $\mathbb{Z}/n\mathbb{Z}$ may not be an integral domain i.e. there exists $x, y \in \mathbb{Z}/n\mathbb{Z}\backslash\{0\}$ such that $xy = 0 \bmod n$, which is the case for RSA (see Subsection V-D) with p and q such that $n = pq$. Because n cannot be easily factorized into p and q, and because we are looking for general solutions for any n, if $x \neq 0$ and $xy = 0$ we will look for solutions $y = 0$, thus loosing some generality but simplifying the analysis.

A. Both Semi- and Fully-Interleaved Ladders

Equation (2) (resp. Equation (5)) $f(x, \ell(x)) = \theta(x)$ for the semi- (resp. fully-) interleaved cases is equivalent to:

$$\begin{cases}
f_{20} + f_{11}\ell_1 + f_{02}\ell_1^2 = a \\
f_{11}\ell_0 + 2f_{02}\ell_1\ell_0 + f_{10} + f_{01}\ell_1 = 0 \\
f_{02}\ell_0^2 + f_{01}\ell_0 + f_{00} = 0
\end{cases}$$

Equation (3) (resp. Equation (6)) $f(\ell(x), x) = \ell(\varepsilon(x))$ for the semi- (resp. fully-) interleaved cases is equivalent to:

$$\begin{cases}
f_{20}\ell_1^2 + f_{11}\ell_1 + f_{02} = \ell_1 \\
2f_{20}\ell_1\ell_0 + f_{11}\ell_0 + f_{10}\ell_1 + f_{01} = 0 \\
f_{20}\ell_0^2 + f_{10}\ell_0 + f_{00} = \ell_0
\end{cases}$$

B. Semi-Interleaved Ladder

The remaining Equation (1) $\varepsilon(\ell(x)) = \ell(\theta(x))$ for the semi-interleaved cases is equivalent to:

$$\begin{cases}
\ell_1^2 = \ell_1 a \\
2\ell_1\ell_0 = 0 \\
\ell_0^2 = \ell_0
\end{cases}$$

So, because $\ell_1 \neq 0$, we assume $\ell_1 = a$ and $\ell_0 = 0$. Therefore Equation (2) is equivalent to:

$$\begin{cases}
f_{20} + f_{11}a + f_{02}a^2 = a \\
f_{10} + f_{01}a = 0 \\
f_{00} = 0
\end{cases}$$

and Equation (3) is equivalent to:

$$\begin{cases}
f_{20}a^2 + f_{11}a + f_{02} = a \\
f_{10}a + f_{01} = 0 \\
f_{00} = 0
\end{cases}$$

input public a, n; secret k
$x \leftarrow 1$
$y \leftarrow a \bmod n$
$c \leftarrow a^2 + 1 \bmod n$
for $i = d$ **to** 0 **do**
  $m \leftarrow$ random$([0, n-1])$
  if $k[i] = 1$ **then**
    $z \leftarrow y^2 \bmod n$
    $x \leftarrow ma(x^2 + z) + (1 - mc)xy \bmod n$
    $y \leftarrow z$
  else
    $z \leftarrow x^2 \bmod n$
    $y \leftarrow ma(y^2 + z) + (1 - mc)yx \bmod n$
    $x \leftarrow z$
  end if
end for
return x
output $x = a^k \bmod n$

TABLE XI
SEMI-INTERLEAVED LADDER FOR THE EXPONENTIATION

From $f_{10} + f_{01}a = f_{10}a + f_{01}$ we deduce $f_{10}(a-1) = f_{01}(a-1)$, which is always true without constraint on a if $f_{10} = f_{01}$. Moreover, from $f_{10} + f_{01}a = 0$ we obtain $f_{10} = -f_{01}a$, thus from $f_{10}a + f_{01} = 0$ we obtain $f_{01}(a^2 - 1) = 0$, which is always true without constraint on a if $f_{01} = 0$. Therefore we assume $f_{10} = 0 = f_{01}$.

From $f_{20} + f_{11}a + f_{02}a^2 = f_{20}a^2 + f_{11}a + f_{02}$ we deduce $f_{20}(a^2 - 1) = f_{02}(a^2 - 1)$, which is always true without constraint on a if $f_{20} = f_{02}$. The remaining constraint is $f_{20}(a^2 + 1) + f_{11}a = a$, which is equivalent to $f_{20} = \frac{a}{a^2+1}(1 - f_{11})$. By assuming that the coefficients are integers there exists m such that $1 - f_{11} = m(a^2 + 1)$. Thus, we have $f_{20} = ma$ and $f_{11} = 1 - m(a^2 + 1)$.

Theorem 2. *The square-and-multiply algorithm in Table II for the modular exponentiation with $\theta(x) = ax^2$ and $\varepsilon(x) = x^2$ is semi-ladderizable with:*

$$\ell(x) = ax$$
$$f(x, y) = ma(x^2 + y^2) + (1 - m(a^2 + 1))xy$$

producing the algorithm in Table XI.

Note that if $m = 0$ then $f(x, y) = xy$ so this solution is the common Montgomery ladder. Note also that m can be chosen randomly for every iteration, providing a mask for the intermediate values and thus reducing the opportunity of information leakage.

C. Fully-Interleaved Ladder (General Case)

Equation (4) $g(\theta(x), \ell(x)) = \ell(\theta(x))$ for the fully-interleaved cases is equivalent to:

$$\begin{cases} g_{20}a^2 = 0 \\ g_{11}\ell_1 a = 0 \\ g_{11}\ell_0 a + g_{02}\ell_1^2 + g_{10}a = \ell_1 a \\ 2g_{02}\ell_1\ell_0 + g_{01}\ell_1 = 0 \\ g_{02}\ell_0^2 + g_{01}\ell_0 + g_{00} = \ell_0 \end{cases}$$

So, because $a \neq 0$ and $\ell_1 \neq 0$, we assume $g_{20} = 0$ from the first equation, $g_{11} = 0$ from the second, and the remaining constraints are:

$$\begin{cases} g_{02}\ell_1^2 + g_{10}a = \ell_1 a \\ 2g_{02}\ell_1\ell_0 + g_{01}\ell_1 = 0 \\ g_{02}\ell_0^2 + g_{01}\ell_0 + g_{00} = \ell_0 \end{cases}$$

According to Equation (6) $f(\ell(x), x) = \ell(\varepsilon(x)) = \ell_1 x^2 + \ell_0$, so by using $g_{20} = g_{11} = 0$ we have:

$$g(f(\ell(x), x), x) = (g_{02} + g_{10}\ell_1)x^2 + (g_{01})x + (g_{10}\ell_0 + g_{00})$$

Thus, Equation (7) $g(f(\ell(x), x), x) = \varepsilon(x)$ for the fully-interleaved cases is equivalent to:

$$\begin{cases} g_{02} + g_{10}\ell_1 = 1 \\ g_{01} = 0 \\ g_{10}\ell_0 + g_{00} = 0 \end{cases}$$

Because $g_{20} = g_{11} = g_{01} = 0$ and we require $g(x, y) = g_{02}y^2 + g_{10}x + g_{00}$ to depend on both x and y, we have to assume $g_{02} \neq 0$ and $g_{10} \neq 0$. By using $g_{01} = 0$ the remaining constraints from Equation (4) are:

$$\begin{cases} g_{02}\ell_1^2 + g_{10}a = \ell_1 a \\ g_{02}\ell_1\ell_0 = 0 \\ g_{02}\ell_0^2 + g_{00} = \ell_0 \end{cases}$$

Because $g_{02} \neq 0$ and $\ell_1 \neq 0$, the second equation implies that $\ell_0 = 0$. Thus, the remaining constraints from Equation (4) and Equation (7) are:

$$\begin{cases} g_{02}\ell_1^2 + g_{10}a = \ell_1 a \\ g_{02} + g_{10}\ell_1 = 1 \\ g_{20} = g_{11} = g_{01} = g_{00} = 0 \end{cases}$$

By using the second equation $g_{02} = 1 - g_{10}\ell_1$, the first one is equivalent to:

$$g_{10}(a - \ell_1^3) = \ell_1(a - \ell_1)$$

Because $g_{10} \neq 0$ and $\ell_1 \neq 0$, note that $a = \ell_1^3 \Leftrightarrow a = \ell_1$, in which case $\ell_1^3 = \ell_1$, thus $a = \ell_1 = \pm 1$. To remove the constraint on a, we assume that $a \neq \ell_1^3$, and thus g_{10} and g_{02} depends only on a and ℓ_1:

$$g_{10} = \ell_1 \frac{a - \ell_1}{a - \ell_1^3}$$
$$g_{02} = 1 - g_{10}\ell_1 = a\frac{1 - \ell_1^2}{a - \ell_1^3}$$
$$\begin{aligned} g(x, y) &= g_{02}y^2 + g_{10}x \\ &= \frac{1}{a - \ell_1^3}\left(a(1 - \ell_1^2)y^2 + \ell_1(a - \ell_1)x\right) \end{aligned}$$

Note that because $\ell_1 \neq 0$ and $g_{10} \neq 0$ we have to assume that, as opposed to the semi-interleaved cases, $\ell_1 \neq a$.

Moreover, because $\ell_0 = 0$, the constraints from the common Equations (5) and (6) can be simplified:

$$\begin{cases} f_{20} + f_{11}\ell_1 + f_{02}\ell_1^2 = a & (L_1) \\ f_{20}\ell_1^2 + f_{11}\ell_1 + f_{02} = \ell_1 & (L_2) \\ f_{10} + f_{01}\ell_1 = 0 & (L_3) \\ f_{10}\ell_1 + f_{01} = 0 & (L_4) \\ f_{00} = 0 \end{cases}$$

In order to have a non-trivial ladder $\ell(x) = \ell_1 x$, we will assume that $\ell_1 \neq \pm 1$. With $(L_4) - (L_3)$ we obtain $(f_{10} - f_{01})(\ell_1 - 1) = 0$ thus $f_{10} = f_{01}$, and with (L_3) we have $f_{10}(\ell_1 + 1) = 0$, so $f_{10} = f_{01} = 0$. With $(L_1) - (L_2)$ we obtain $(f_{02} - f_{20})(\ell_1^2 - 1) = a - \ell_1$, so $f_{02} = f_{20} + \frac{a-\ell_1}{\ell_1^2-1}$. Therefore, remain the following constraints:

$$\begin{cases} f_{20}(\ell_1^2 + 1) + f_{11}\ell_1 = a - \ell_1^2 \dfrac{a-\ell_1}{\ell_1^2-1} \\ \qquad\qquad\qquad\qquad = \ell_1 - \dfrac{a-\ell_1}{\ell_1^2-1} \\ f_{02} = f_{20} + \dfrac{a-\ell_1}{\ell_1^2-1} \\ f_{10} = f_{01} = f_{00} = 0 \end{cases}$$

The first line being obtained from (L_1) and the second from (L_2). Note that $a - \ell_1^2 \frac{a-\ell_1}{\ell_1^2-1} = \ell_1 - \frac{a-\ell_1}{\ell_1^2-1}$ is satisfied and thus is not a constraint. Moreover, because $\ell_1 \neq 0$, by using the second line, f_{11} can be written as:

$$\begin{aligned} f_{11} &= \frac{1}{\ell_1}\left(\ell_1 - \frac{a-\ell_1}{\ell_1^2-1} - f_{20}(\ell_1^2+1)\right) \\ &= \frac{1}{\ell_1}\left(\frac{\ell_1^3 - a}{\ell_1^2-1} - f_{20}(\ell_1^2+1)\right) \end{aligned}$$

Thus, we have:

$$\begin{aligned} f(x,y) &= f_{20}x^2 + f_{11}xy + f_{02}y^2 \\ &= f_{20}x^2 + \frac{1}{\ell_1}\left(\frac{\ell_1^3-a}{\ell_1^2-1} - f_{20}(\ell_1^2+1)\right)xy \\ &\quad + \left(f_{20} + \frac{a-\ell_1}{\ell_1^2-1}\right)y^2 \\ &= f_{20}\left(x^2 - \frac{\ell_1^2+1}{\ell_1}xy + y^2\right) \\ &\quad + \frac{1}{\ell_1^2-1}\left(\frac{\ell_1^3-a}{\ell_1}xy + (a-\ell_1)y^2\right) \end{aligned}$$

Finally, f_{20} has no constraint. It could have been used as a mask for f as in Theorem 2, but unfortunately not for g. So, for sake of simplicity, we assume $f_{20} = 0$ and obtain the following functions:

Theorem 3. *The square-and-multiply algorithm in Table II for the modular exponentiation with $\theta(x) = ax^2$ and $\varepsilon(x) = x^2$ is fully-ladderizable with:*

$$\begin{aligned} \ell(x) &= \ell_1 x \\ f(x,y) &= \frac{1}{\ell_1^2-1}\left(\frac{\ell_1^3-a}{\ell_1}xy - (\ell_1 - a)y^2\right) \\ g(x,y) &= \frac{1}{\ell_1^3 - a}\left(a(\ell_1^2-1)y^2 + \ell_1(\ell_1 - a)x\right) \end{aligned}$$

input public a, n; secret k
1: **do**
2: $\quad \ell \leftarrow \text{random}([2, n-2] \setminus \{a\})$
3: $\quad v_0 \leftarrow \ell - a \bmod n$
4: $\quad (d_1, u_1) \leftarrow \text{EEA}(\ell, n)$
5: $\quad v_2 \leftarrow \ell^2 - 1 \bmod n$
6: $\quad (d_2, u_2) \leftarrow \text{EEA}(v_2, n)$
7: $\quad v_3 \leftarrow \ell^3 - a \bmod n$
8: $\quad (d_3, u_3) \leftarrow \text{EEA}(v_3, n)$
9: **while** $v_0 \bmod n = 0 \vee d_1 \neq 1 \vee d_2 \neq 1 \vee d_3 \neq 1$
10: $c_0 \leftarrow u_1 u_2 v_3 \bmod n$
11: $c_1 \leftarrow -v_0 u_2 \bmod n$
12: $c_2 \leftarrow a v_2 u_3 \bmod n$
13: $c_3 \leftarrow \ell v_0 u_3 \bmod n$
14: $x \leftarrow 1$
15: $y \leftarrow \ell$
16: **for** $i = d$ **to** 0 **do**
17: $\quad$ **if** $k[i] = 1$ **then**
18: $\quad\quad x \leftarrow c_0 xy + c_1 y^2 \bmod n$
19: $\quad\quad y \leftarrow c_2 y^2 + c_3 x \bmod n$
20: $\quad$ **else**
21: $\quad\quad y \leftarrow c_0 yx + c_1 x^2 \bmod n$
22: $\quad\quad x \leftarrow c_2 x^2 + c_3 y \bmod n$
23: $\quad$ **end if**
24: **end for**
25: **return** x
output $x = a^k \bmod n$

TABLE XII
FULLY-INTERLEAVED LADDER FOR THE MODULAR EXPONENTIATION (GENERAL CASE)

where ① $\ell_1 \neq a \bmod n$, ② $\ell_1 \in (\mathbb{Z}/n\mathbb{Z})^\star$, ③ $\ell_1^2 - 1 \in (\mathbb{Z}/n\mathbb{Z})^\star$ and ④ $\ell_1^3 - a \in (\mathbb{Z}/n\mathbb{Z})^\star$, producing the algorithm in Table XII.

The first constraint on the ladder constant (denoted ℓ in the following) can be satisfied by checking whether $(\ell - a) \bmod n = 0$. The other constraints can be satisfied by using the Extended Euclidean Algorithm $(d, u) \leftarrow \text{EEA}(v, n)$, where $uv = d \bmod n$, such that if $d = 1$ then v is invertible modulo n and $v^{-1} = u \bmod n$. Because $\ell \neq a \bmod n$, $0 \notin (\mathbb{Z}/n\mathbb{Z})^\star$, and if $\ell = \pm 1$ then $\ell_1^2 - 1 \notin (\mathbb{Z}/n\mathbb{Z})^\star$, we can assume that $\ell \in \mathbb{Z}/n\mathbb{Z}$ can be chosen in $\mathrm{L}_{a,n} = [2, n-2] \setminus \{a\}$, then check whether ℓ satisfies the constraints of Theorem 3 (Lines 1–9). Actually, we prove in the RSA case (Subsection V-D) and the DSA case (Subsection V-E) that a ladder constant verifying these constraints is almost always obtained after one iteration.

Note that after a suitable ℓ is chosen, the coefficients of f and g are constant during the iterations, thus can be pre-computed (Lines 10–13). These computations to find suitable constants may cause a significant overhead (Lines 1–15) but that does not depend on the secret, thus trading execution time and energy consumption for more security.

We provide in Table XIII a comparison of complexities for the Montgomery, semi- and fully-interleaved ladders. The complexities are given in terms of cost per key bit, where M stands for multiplication, S for squaring and A for addition/subtraction, all modulo n. These costs are given after memoization, for instance the squarings in Line 18 and Line 19 in Table XII are the same, so could have been memorized by using $z \leftarrow y^2$ and then z used instead of the squares. Because the costs are given per key bit, we did not include the cost of the pre-computations, which is far from negligible for the fully-interleaved ladder in Table XII.

Montgomery	Semi-Interleaved	Fully-Interleaved
$M+S$	$5M+2S+3A$	$5M+S+2A$

TABLE XIII
COST PER BIT OF THE MONTGOMERY, SEMI- AND FULLY-INTERLEAVED LADDERS (AFTER MEMOIZATION)

D. Fully-Interleaved Ladder (RSA Case)

In the context of RSA [2], we assume that $n = pq$, where p, q are distinct primes. The modular exponentiation aims at computing $a^k \bmod n$, so we assume that the input a is already given modulo n, i.e. $0 \leq a \leq n-1$. Because 0^k and $(\pm 1)^k$ are fairly trivial to compute, to simplify the following reasoning we assume that $2 \leq a \leq n-2$, so $\text{card}(\mathrm{L}_{a,n}) = n-4$, where $\mathrm{L}_{a,n} = [2, n-2] \setminus \{a\}$.

To satisfy constraints ① and ② of Theorem 3, i.e. $\ell \neq a$ and $\ell \in (\mathbb{Z}/n\mathbb{Z})^\star$, ℓ has to be chosen in $\mathrm{L}_{a,n}$ such that $\gcd(\ell, n) = 1$. Thus, in the following we will assume that ℓ is chosen randomly in $\mathrm{L}_{a,n}$ and determine the probability that ℓ satisfies the ladder constraints. Within this range, the only integers such that $\gcd(\ell, n) > 1$ are jp for $j \in [1, q-1]$ or jq for $j \in [1, p-1]$, so the probability to pick randomly a ladder constant not verifying ② is:

$$Pr(\text{not } ②) \leq \frac{(p-1)+(q-1)}{n-4} \approx \frac{1}{p} + \frac{1}{q}$$

In RSA context, it means that a random $\ell \in \mathrm{L}_{a,n}$ is invertible modulo n with probability almost equal to 1. Indeed, according to the National Institute of Standards and Technology (NIST) recommendations [29], if $\text{nBits} = \lfloor \log_2 n \rfloor + 1$ denotes the number of bits of $n = pq$, then p and q should verify:

$$2^{\frac{\text{nBits}-1}{2}} < p < 2^{\frac{\text{nBits}}{2}}$$
$$2^{\frac{\text{nBits}-1}{2}} < q < 2^{\frac{\text{nBits}}{2}}$$

Note that if one can compute ℓ such that $\gcd(\ell, n) > 1$ then $\gcd(\ell, n) = p$ or q, and is able to factorize n. The same problem occurs in RSA cryptosystem: if Alice can generate a message m such that $\gcd(m, n) > 1$, where n is Bob's public key, then Alice can factorize n and find Bob's secret key. Such an event could occur with a probability nearly equal to 0.

To satisfy constraint ③ of Theorem 3, i.e. $\ell^2 - 1 \in (\mathbb{Z}/n\mathbb{Z})^\star$, note that according to the chinese remainder theorem the integers ℓ such that $\ell^2 - 1$ is not invertible modulo n verify $\ell^2 = 1 \bmod p$ and $\ell^2 = 1 \bmod q$. There are at most four such integers: 1, $n-1$, γ, $n-\gamma$, where $\gamma = 1 \bmod p$ and $\gamma = -1 \bmod q$. So, the probability to pick randomly a ladder constant in $\mathrm{L}_{a,n}$ not verifying ③ is:

$$Pr(\text{not } ③) \leq \frac{2}{n-4}$$

Finally, to satisfy the last constraint ④ of Theorem 3, i.e. $\ell^3 - a \in (\mathbb{Z}/n\mathbb{Z})^\star$, we use the following lemma from Gauss:

Definition 6 (r^{th} Residues)**.** Let n and $r \geq 2$ be two integers. An integer $a \in (\mathbb{Z}/n\mathbb{Z})^\star$ is a r^{th} *residue* modulo n if there exists an integer $\ell \in \mathbb{Z}/n\mathbb{Z}$ such that $\ell^r = a \bmod n$, otherwise it is a *non-residue* modulo n.

Lemma 1 (Gauss [30])**.** *Let p be a prime and $r \geq 2$ be an integer. We denote $b = \gcd(p-1, r)$. We have:*

1) *For every integer $a \in (\mathbb{Z}/p\mathbb{Z})^\star$, a is a r^{th} residue modulo p if and only if $a^{\frac{p-1}{b}} = 1 \bmod p$,*
2) *there exists exactly $\frac{p-1}{b}$ r^{th} residues in $(\mathbb{Z}/p\mathbb{Z})^\star$,*
3) *if $a \in (\mathbb{Z}/p\mathbb{Z})^\star$ is a r^{th} residue modulo p then there exists exactly b integers $\ell \in \mathbb{Z}/p\mathbb{Z}$ such that $\ell^r = a \bmod p$.*

According to the chinese remainder theorem, $\ell^3 - a$ is invertible modulo $n = pq$ if and only if $\ell^3 - a$ is invertible modulo p and $\ell^3 - a$ is invertible modulo q. Moreover, because p is prime, $\ell^3 - a$ is invertible $\bmod p$ if and only if $\ell^3 - a \neq 0 \bmod p$, i.e. a is a 3^{rd} non-residue modulo p. Therefore, the constraint ④ is not satisfied only if a is a 3^{rd} residue modulo p or q.

For a fixed, because $b_p = \gcd(p-1, 3) = 1$ or 3 and $b_q = \gcd(q-1, 3) = 1$ or 3, according to Lemma 1 there are at most 9 non invertible $\ell^3 - a$ integers modulo n, so the probability to pick randomly a ladder constant in $\mathrm{L}_{a,n}$ not verifying ④ is:

$$Pr(\text{not } ④) \leq \frac{9}{n-4}$$

In the worst case, the unfavorable cases are distinct, thus the probability to pick randomly a suitable ladder constant is:

$$\begin{aligned} Pr(①②③④) &\geq 1 - \left(\frac{p+q-2}{n-4} + \frac{2}{n-4} + \frac{9}{n-4}\right) \\ &= 1 - \frac{p+q+9}{n-4} \approx 1 - \frac{1}{p} - \frac{1}{q} - \frac{1}{n} \end{aligned}$$

which is almost equal to 1 in a cryptographic context. Thus, in the RSA case $n = pq$ finding a suitable ladder constant (Lines 1–9) in Table XII costs only one iteration with probability almost equal to 1.

E. Fully-Interleaved Ladder (DSA Case)

In Digital Signature Algorithm (DSA) [31] context (or for Diffie-Hellman key exchange mechanism [32]), the exponentiation is computed modulo $n = p$, where p is prime.

As in the RSA case we prove that finding a suitable ladder constant costs only one iteration with probability almost equal to 1. The argument is similar to the previous one but even simpler. If $\ell \in \mathrm{L}_{a,n} = [2, n-2] \setminus \{a\}$ then ① $\ell \neq a$. Moreover $\ell \neq 0 \bmod n$ so ② ℓ is invertible modulo n. Finally, $\ell \neq \pm 1$ so ③ $\ell^2 - 1 \neq 0 \bmod n$ thus $\ell^2 - 1$ is invertible modulo n.

Remains the last constraint ④ $\ell^3 - a$ is invertible modulo n. In the following we denote $\mathrm{R}_n^3(a)$ that a is a cubic residue modulo n, and let $b = \text{pgcd}(n-1, 3)$.

According to Lemma 1, there exists $\frac{n-1}{b}$ cubic residue in $(\mathbb{Z}/n\mathbb{Z})^\star$. -1 and 1 are cubic residue modulo n, as opposed to 0. Thus, the probability that an integer $a \in [2, n-2]$ is a cubic residue is:

$$Pr\left(\mathrm{R}_n^3(a)\right) = \frac{\frac{n-1}{b} - 2}{n-3} = \frac{1}{b} - \frac{2\left(1 - \frac{1}{b}\right)}{n-3}$$

According to Lemma 1, if a is a cubic residue then there exists b cubic roots ℓ of a in $(\mathbb{Z}/n\mathbb{Z})^\star$. If $\ell = 0, \pm 1$ then $a = \ell^3 = 0, \pm 1$, which is excluded. If $\ell = a$ then $a = a^3$

i.e. $a(a+1)(a-1) = 0 \bmod p$, thus $a = 0, \pm 1$ which is excluded. So the cubic roots ℓ of a are automatically in $\mathrm{L}_{a,n} = [2, n-2] \setminus \{a\}$. Thus the probability that an integer $\ell \in \mathrm{L}_{a,n}$ is a cubic root of a is $\frac{b}{n-4}$. Therefore, if a is a cubic residue then the probability that $\ell \in \mathrm{L}_{a,n}$ satisfies the last constraint is:

$$Pr\big(④ \mid \mathrm{R}_n^3(a)\big) = 1 - \frac{b}{n-4}$$

Finally, if a is not a cubic residue then every $\ell \in \mathrm{L}_{a,n}$ satisfies ④. Therefore, the probability to pick randomly a suitable ladder constant is:

$$\begin{aligned} Pr(①②③④) &= Pr(④) \\ &= Pr\big(④ \mid \mathrm{R}_n^3(a)\big) \times Pr\big(\mathrm{R}_n^3(a)\big) \\ &\quad + Pr\big(④ \mid \text{not } \mathrm{R}_n^3(a)\big) \times Pr\big(\text{not } \mathrm{R}_n^3(a)\big) \\ &= \left(1 - \frac{b}{n-4}\right)\left(\frac{1}{b} - \frac{2\left(1-\frac{1}{b}\right)}{n-3}\right) \\ &\quad + 1\left(1 - \frac{1}{b} + \frac{2\left(1-\frac{1}{b}\right)}{n-3}\right) \\ &= 1 - \frac{1}{n-4} + \frac{2(b-1)}{(n-3)(n-4)} \end{aligned}$$

which is almost equal to 1 in a cryptographic context.

VI. Scalar Multiplication in ECC

As in Section V we demonstrate concrete examples of the ladder equations, this time for the scalar multiplication used in elliptic curve cryptography (ECC).

ECC was independently introduced in 1985 by Neal Koblitz [22] and Victor Miller [21]. It is nowadays considered as an excellent choice for key exchange or digital signatures, especially when these mechanisms run on resource-constrained devices. The security of most cryptocurrencies is based on ECC, which has been standardized by the NIST [31], [33].

Definition 7 (Elliptic Curve)**.** Let p be a prime. An elliptic curve in short Weierstrass form over a finite field $\mathbb{F}_p$ is defined by the set $E(\mathbb{F}_p) = \big\{(x,y) \in \mathbb{F}_p \times \mathbb{F}_p \mid y^2 = x^3 + ax + b\big\} \cup \mathcal{O}$, with $a, b \in \mathbb{F}_p$ satisfying $4a^3 + 27b^2 \neq 0$ and $\mathcal{O}$ being called the point at infinity.

The set $E(\mathbb{F}_p)$ is an additive abelian group with an efficiently computable group law. Point addition $P + Q$ or point doubling $2P$ involves additions and multiplications over $\mathbb{F}_p$. The point $\mathcal{O}$ is the identity element of the group law. Depending on the parameters a and b of the curve, there exists many formulas to optimize these two operations. A large and updated survey can be found on the elliptic curves explicit formulas database web site: https://www.hyperelliptic.org/EFD/

The main operation in ECC is scalar multiplication $kA = A + \cdots + A$, where A is a point on a curve and k is an integer. It can be performed by using the *double-and-add* algorithm, similar to the square-and-multiply algorithm in Table II p.2. The initialization $x \leftarrow 1$ is replaced by $P \leftarrow \mathcal{O}$, the squaring $x \leftarrow x^2$ is replaced by a doubling $P \leftarrow 2P$ and the multiplication $x \leftarrow ax$ is replaced by an addition $P \leftarrow A + P$, producing the output $P = kA$.

Thus, for a fixed point A we consider in this section the following known functions:

$$\begin{aligned} \theta(P) &= 2P + A \\ \varepsilon(P) &= 2P \end{aligned}$$

And we consider ladder functions of the following form:

$$\begin{aligned} f(P,Q) &= f_P P + f_Q Q + f_A A \\ g(P,Q) &= g_P P + g_Q Q + g_A A \\ \ell(P) &= \ell_P P + \ell_A A \end{aligned}$$

with $f_P, f_Q, \ell_P \neq 0$ for both cases, and $g_P, g_Q \neq 0$ for the fully-interleaved ladders case, allowing in principle non-secure scalar multiplications as intermediate computations for the secure one. The efficiency of the obtained solutions are discussed in Subsection VI-D.

A. Both Semi- and Fully-Interleaved Ladders

Equation (2) (resp. Equation (5)) $f(x, \ell(x)) = \theta(x)$ and Equation (3) (resp. Equation (6)) $f(\ell(x), x) = \ell(\varepsilon(x))$ for the semi- (resp. fully-) interleaved cases imply that:

$$\begin{cases} f_P + f_Q \ell_P = 2 \\ f_Q = (2 - f_P)\ell_P \end{cases}$$

So $(2 - f_P)(\ell_P^2 - 1) = 0$. If $f_P = 2$ then $f_Q = 0$, thus we assume $f_P \neq 2$. To make no assumption about the finite field, we assume for the following $\ell_P = \pm 1$. They also imply that:

$$\begin{cases} f_Q \ell_A + f_A = 1 \\ f_A = (2 - f_P)\ell_A - \ell_A \end{cases}$$

So $(2 - f_P)(\ell_P + 1)\ell_A = 1 + \ell_A$.

- If $\ell_P = -1$ then $\ell_A = -1$, $f_Q = f_P - 2$ and $f_A = (f_P - 2) + 1$, thus the candidate ladder functions are:
$$\begin{aligned} f(P,Q) &= f_P(P + Q + A) - (2Q + A) \\ \ell(P) &= -(P + A) \end{aligned}$$
- If $\ell_P = 1$ then by assuming that $3 - 2f_P$ is invertible we obtain $\ell_A = \frac{1}{3-2f_P}$, $f_Q = 2 - f_P$ and $f_A = 1 - \frac{2-f_P}{3-2f_P}$, thus the candidate ladder functions are:
$$\begin{aligned} f(P,Q) &= f_P(P - Q) + 2Q + \frac{1 - f_P}{3 - 2f_P} A \\ \ell(P) &= P + \frac{1}{3 - 2f_P} A \end{aligned}$$

B. Semi-Interleaved Ladder

The remaining Equation (1) $\varepsilon(\ell(x)) = \ell(\theta(x))$ for the semi-interleaved ladders is already satisfied for the $\ell_P = -1$ case. For the $\ell_P = 1$ case, it implies that $(\frac{1}{3-2f_P} - 1)A = 0$, thus to avoid constraint on A we assume $f_P = 1$.

Theorem 4. *The double-and-add algorithm with $\theta(P) = 2P + A$ and $\varepsilon(P) = 2P$ is semi-ladderizable with two solutions:*

1)
$$\begin{aligned} f(P,Q) &= f_P(P + Q + A) - (2Q + A) \\ \ell(P) &= -(P + A) \end{aligned}$$
where $f_P \neq 0, 2$, and:

2)
$$f(P,Q) = P + Q$$
$$\ell(P) = P + A$$

The second solution is actually the common Montgomery ladder for the scalar multiplication. The first one for $f_P = 1$ corresponds to the ladder function $f(P,Q) = P - Q$ which is similar to the common Montgomery ladder except that the point Q has an opposite sign.

C. Fully-Interleaved Ladder

In the case $\ell_P = -1$, Equation (4) $g(\theta(x), \ell(x)) = \ell(\theta(x))$ for the fully-interleaved ladders is equivalent to:

$$\begin{cases} 2g_P - g_Q = -2 \\ g_P - g_Q + g_A = -2 \end{cases}$$

So $g_Q = 2(1 + g_P)$ and $g_A = g_P$. Thus Equation (7) $g(f(\ell(x), x), x) = \varepsilon(x)$ for the fully-interleaved ladders implies that $f_P g_P = 0$. Because we assumed $f_P, g_P \neq 0$ and we are looking for a solution independent from the chosen field for the coefficients, we have no solution for this case.

In the case $\ell_P = 1$, Equation (4) $g(\theta(x), \ell(x)) = \ell(\theta(x))$ for the fully-interleaved ladders is equivalent to:

$$\begin{cases} 2g_P + g_Q = 2 \\ g_P + \dfrac{g_Q}{3 - 2f_P} + g_A = 1 + \dfrac{1}{3 - 2f_P} \end{cases}$$

So $g_Q = 2(1 - g_P)$ and $g_A = 1 + \frac{(2f_P - 1)g_P - 1}{3 - 2f_P}$. Thus Equation (7) $g(f(\ell(x), x), x) = \varepsilon(x)$ for the fully-interleaved ladders implies that $\frac{2(f_P(g_P - 1) + 1)}{3 - 2f_P} A = 0$. To avoid constraint on A we assume $f_P(g_P - 1) + 1 = 0$, i.e. $g_P = 1 - \frac{1}{f_P}$.

Theorem 5. *The double-and-add algorithm with $\theta(P) = 2P + A$ and $\varepsilon(P) = 2P$ is fully-ladderizable with:*

$$f(P,Q) = f_P \left(P - Q - \frac{1}{3 - 2f_P} A \right) + 2Q + \frac{1}{3 - 2f_P} A$$
$$g(P,Q) = P + \frac{1}{f_P} \left(-P + 2Q + \frac{1}{3 - 2f_P} A \right) - \frac{1}{3 - 2f_P} A$$
$$\ell(P) = P + \frac{1}{3 - 2f_P} A$$

where $f_P \neq 0, 1, 2$, and where f_P and $3 - 2f_P$ are invertible.

The ladder functions of this solution for $f_P = 1$ are $f(P,Q) = P + Q$, $g(P,Q) = 2Q$ and $\ell(P) = P + A$, i.e. the common Montgomery ladder for the scalar multiplication, but is no longer a fully-interleaved ladder.

D. Application to ECC

The novel candidates are the first semi-interleaved ladder in Theorem 4 requiring a scalar multiplication by f_P, and the fully-interleaved ladder in Theorem 5 requiring a scalar multiplication by $\frac{1}{3 - 2f_P}$ then f_P and $\frac{1}{f_P}$.

Scalar multiplications cA for various coefficients c can be precomputed if A was a fixed point, as in the DSA signing step or the first step of Diffie-Hellman protocol. But both candidates involve scalar multiplications for non-fixed points R, linear combination of A, P, and Q.

A scalar multiplication cR for a specific coefficient c can be efficiently computed (i.e. with very few field operations) by using an elliptic curve endowed with an efficient endomorphism. On such curves, there exists a constant λ, depending on the modulus p and the parameters of the curve (see Definition 7), such that λR can be computed with very few multiplications (only one in most cases). This is detailed in [34], where the authors describe the GLV method used in various standards, like the Bitcoin protocol specification or TLS [35].

Another way to optimize the cost of cR is to use a short addition chain [36], so that the number of point operations to compute cR is equal to the length of the chain. The problem of computing the shortest addition chain for a given integer is hard [37] but for "small" integers there exists tables giving the corresponding chain, like https://bo.blackowl.org/random/ln.

Therefore, for the first semi-interleaved ladder in Theorem 4, one can use an elliptic curve endowed with an efficient endomorphism λ and assign $f_P = \lambda$. Or, for a non-specific curve, one can choose f_P amongst the integers corresponding to "very short" addition chains. In that case, because as in Theorem 2 f_P does not occur in $\ell(P)$, f_P can be chosen randomly for every iteration, thus increasing security compared to the common Montgomery ladder. However, this comes at the price of drastically increasing the computational cost of the scalar multiplication.

For the fully-interleaved ladder candidate, neither an efficient endomorphism nor an addition chain can be used. Indeed, that $f_P R$ can be efficiently computed does not imply that $\frac{1}{f_P} R$ can also be efficiently computed, because both coefficients are dependent. Therefore, the fully-interleaved ladder candidate might not be applicable in practice to ECC.

VII. Conclusion and Future Work

We abstracted away the algorithmic strength of the Montgomery ladder against side-channel and fault-injection attacks, by defining semi- and fully-ladderizable algorithms. We designed also fault-injection attacks able to obtain some/all bits of the secret key from the semi-interleaved ladders like the Montgomery ladder, and even from the fully-interleaved ladders if the attacker is able to stuck-at the key register.

As examples, we provided for the modular exponentiation a better semi-interleaved ladder using a random mask updated at every iteration, and a fully-interleaved ladder depending on an appropriate ladder constant. We investigated the properties required by the ladder constant, and discovered that they are almost always satisfied in a cryptographic context.

To demonstrate that generality of our approach, we provided new algorithms for the scalar multiplication in elliptic curves: a novel and effective semi-interleaved ladder, and a fully-interleaved ladder. The applicability of the last candidate depends on the efficient computations of several dependent coefficients for the scalar multiplications, which is an open and interesting problem.

Finally, we only investigated the univariate case for conditional branching, but the multivariate case may be of interest. For instance, a multi-variable polynomial $\sum_{0 \leq n \leq d} \sum_{n_1 + \cdots + n_k = n} c_{n_1, \ldots, n_k} \prod_{1 \leq i \leq k} x_i^{n_i}$ can be represented by a multidimensional array of coefficients, thus the

manipulation of ladder equations could be handled by using matrix operations.

ACKNOWLEDGMENTS

The authors want to thank the reviewers of ARITH 2020 for their relevant and very useful comments.

REFERENCES

[1] Y. Marquer and T. Richmond, "A Hole in the Ladder: Interleaved Variables in Iterative Conditional Branching," in *ARITH 2020 - 27th IEEE Symposium on Computer Arithmetic*. Portland, Oregon, USA, United States: IEEE, Jun. 2020, pp. 56–63.

[2] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[3] P. C. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems," in *Advances in Cryptology — CRYPTO '96*, N. Koblitz, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 104–113.

[4] C. Clavier, B. Feix, G. Gagnerot, M. Roussellet, and V. Verneuil, "Square Always Exponentiation," in *Progress in Cryptology – INDOCRYPT 2011*, D. J. Bernstein and S. Chatterjee, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 40–57.

[5] J. Ha, Y. Choi, D. Choi, and H. Lee, "Power Analysis Attacks on the Right-to-Left Square-Always Exponentiation Algorithm," *J. Internet Serv. Inf. Secur.*, vol. 4, pp. 38–51, 2014.

[6] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *International workshop on cryptographic hardware and embedded systems*. Springer, 2004, pp. 16–29.

[7] J.-S. Coron, "Resistance Against Differential Power Analysis For Elliptic Curve Cryptosystems," in *Cryptographic Hardware and Embedded Systems*, Ç. K. Koç and C. Paar, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 292–302.

[8] T. S. Messerges, E. A. Dabbish, and R. H. Sloan, "Power Analysis Attacks of Modular Exponentiation in Smartcards," in *Cryptographic Hardware and Embedded Systems*, Ç. K. Koç and C. Paar, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 144–157.

[9] P. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," in *Advances in Cryptology — CRYPTO' 99*, M. Wiener, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 388–397.

[10] Y. Sung-Ming, S. Kim, S. Lim, and S. Moon, "A Countermeasure against One Physical Cryptanalysis May Benefit Another Attack," in *Information Security and Cryptology — ICISC 2001*, K. Kim, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 414–427.

[11] H. Ziade, R. Ayoubi, and R. Velazco, "A survey on fault injection techniques," *International Arab Journal of Information Technology*, vol. Vol. 1, No. 2, July, pp. 171–186, 2004. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00105562

[12] P. L. Montgomery, "Montgomery, P.L.: Speeding the Pollard and Elliptic Curve Methods of Factorization. Math. Comp. 48, 243-264," *Mathematics of Computation - Math. Comput.*, vol. 48, pp. 243–243, 01 1987.

[13] Y. Marquer, "Algorithmic Completeness of Imperative Programming Languages," *Fundamenta Informaticae*, vol. 168, no. 1, pp. 51–77, July 2019.

[14] A. Boscher, R. Naciri, and E. Prouff, "CRT RSA Algorithm Protected Against Fault Attacks," in *Information Security Theory and Practices. Smart Cards, Mobile and Ubiquitous Computing Systems*, D. Sauveron, K. Markantonakis, A. Bilas, and J.-J. Quisquater, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 229–243.

[15] M. Joye, "Highly Regular Right-to-Left Algorithms for Scalar Multiplication," in *Cryptographic Hardware and Embedded Systems - CHES 2007*, P. Paillier and I. Verbauwhede, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 135–147.

[16] ——, "Highly Regular m-Ary Powering Ladders," in *Selected Areas in Cryptography*, M. J. Jacobson, V. Rijmen, and R. Safavi-Naini, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 350–363.

[17] C. D. Walter, "The Montgomery and Joye Powering Ladders are Dual," *IACR ePrint Archive*, vol. 1081, pp. 1–6, 2017. [Online]. Available: https://eprint.iacr.org/2017/1081.pdf

[18] M. Joye and S.-M. Yen, "The Montgomery Powering Ladder," in *Cryptographic Hardware and Embedded Systems - CHES 2002*, B. S. Kaliski, ç. K. Koç, and C. Paar, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 291–302.

[19] Á. Kiss, J. Krämer, P. Rauzy, and J.-P. Seifert, "Algorithmic Countermeasures Against Fault Attacks and Power Analysis for RSA-CRT," in *Constructive Side-Channel Analysis and Secure Design*, F.-X. Standaert and E. Oswald, Eds. Cham: Springer International Publishing, 2016, pp. 111–129.

[20] C. Giraud, "An RSA Implementation Resistant to Fault Attacks and to Simple Power Analysis," *IEEE Transactions on Computers*, vol. 55, no. 9, pp. 1116–1120, Sep. 2006.

[21] V. S. Miller, "Use of Elliptic Curves in Cryptography," in *Advances in Cryptology — CRYPTO '85 Proceedings*, H. C. Williams, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1986, pp. 417–426.

[22] N. Koblitz, "Elliptic Curve Cryptosystems," *Math. Comp*, vol. 48, pp. 243–264, 01 1987.

[23] F. Strenzke, E. Tews, H. G. Molter, R. Overbeck, and A. Shoufan, "Side Channels in the McEliece PKC," in *Post-Quantum Cryptography*, J. Buchmann and J. Ding, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 216–229.

[24] M. Petrvalsky, T. Richmond, M. Drutarovsky, P. Cayrel, and V. Fischer, "Differential power analysis attack on the secure bit permutation in the McEliece cryptosystem," in *2016 26th International Conference Radioelektronika (RADIOELEKTRONIKA)*, April 2016, pp. 132–137.

[25] H. Kim, T. H. Kim, J. C. Yoon, and S. Hong, "Practical Second-Order Correlation Power Analysis on the Message Blinding Method and Its Novel Countermeasure for RSA," *ETRI Journal*, vol. 32, no. 1, pp. 102–111, 2010. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.4218/etrij.10.0109.0249

[26] N. Hanley, H. Kim, and M. Tunstall, "Exploiting Collisions in Addition Chain-Based Exponentiation Algorithms Using a Single Trace," in *Topics in Cryptology — CT-RSA 2015*, K. Nyberg, Ed. Cham: Springer International Publishing, 2015, pp. 431–448.

[27] D.-P. Le, C. H. Tan, and M. Tunstall, "Randomizing the Montgomery Powering Ladder," in *Information Security Theory and Practice*, R. N. Akram and S. Jajodia, Eds. Cham: Springer International Publishing, 2015, pp. 169–184.

[28] J. Arlat, "Validation de la sûreté de fonctionnement par injection de fautes : méthode, mise en oeuvre, application," Ph.D. dissertation, Institut national polytechnique (Toulouse, France), 1990.

[29] E. B. Barker, L. Chen, A. L. Roginsky, R. Davis, and S. Simon, *Recommendation for Pair-Wise Key Establishment Using Integer Factorization Cryptography*, ser. Special Publication. NIST SP, March 2019, no. 800-56B Rev. 2. [Online]. Available: https://doi.org/10.6028/NIST.SP.800-56Br2

[30] C. F. Gauss and A. A. Clarke, *Disquisitiones Arithmeticae*. Yale University Press, 1965. [Online]. Available: http://www.jstor.org/stable/j.ctt1cc2mnd

[31] "Digital signature standard (DSS)," July 2013, FIPS PUB 186-4, U.S.Department of Commerce/National Institute of Standards and Technology. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf

[32] E. Rescorla, "RFC2631: Diffie-Hellman key agreement method," USA, 1999.

[33] E. B. Barker, L. Chen, A. L. Roginsky, A. Vassilev, and R. Davis, *Recommendation for Pair-Wise Key Establishment Using Dicrete Logarithm Cryptography*, ser. Special Publication. NIST SP, April 2018, no. 800-56A Rev. 3. [Online]. Available: https://doi.org/10.6028/NIST.SP.800-56Ar3

[34] R. P. Gallant, R. J. Lambert, and S. A. Vanstone, "Faster point multiplication on elliptic curves with efficient endomorphisms," in *Advances in Cryptology — CRYPTO*, ser. LNCS, vol. 2139. Springer, 2001, pp. 190–200.

[35] Y. Nir, S. Josefsson, and M. Pégourié-Gonnard, "RFC8422: Elliptic curve cryptography (ECC) cipher suites for transport layer security (TLS) versions 1.2 and earlier," USA, 2020.

[36] D. E. Knuth, *The Art of Computer Programming: Fundamental Algorithms*, 3rd ed. Addison Wesley, 07 1997, vol. 2.

[37] H. M. Bahig, "Improved generation of minimal addition chains," *Computing*, vol. 78, no. 2, pp. 161–172, 2006.