Using Modular Extension to Provably Protect Edwards Curves Against Fault Attacks

Margaux Dugardin[1,2], Sylvain Guilley[2,3], Martin Moreau[3], Zakaria Najm[4], and Pablo Rauzy[5,6]

[1] CESTI, Thales Communications & Security, 31000 Toulouse, France.
[2] LTCI, CNRS, Télécom ParisTech, Université Paris-Saclay, 75013 Paris, France.
`firstname.lastname@telecom-paristech.fr`
[3] Secure-IC SAS, 35510 Cesson-Sévigné, France. `firstname.lastname@secure-ic.com`
[4] ST-Microelectronics, 13790 Rousset, France. `zakaria.najm@st.com`
[5] Inria, CITI Lab, 69100 Villeurbanne, France.
[6] LIASD, Université Paris 8, Saint-Denis, France.
`pr@ai.univ-paris8.fr`

Abstract. Fault injection attacks are a real-world threat to cryptosystems, in particular asymmetric cryptography. In this paper, we focus on countermeasures which guarantee the integrity of the computation result, hence covering most existing and future fault attacks. Namely, we study the *modular extension* protection scheme in previously existing and newly contributed variants of the countermeasure on elliptic curve scalar multiplication (ECSM) algorithms. We find that an existing countermeasure is incorrect and we propose new "test-free" variant of the modular extension scheme that fixes it. We then formally prove the correctness and security of modular extension: specifically, the fault non-detection probability is inversely proportional to the security parameter. Finally, we implement an ECSM protected with test-free modular extension during the elliptic curve operation to evaluate the efficient of this method on Edwards and twisted Edwards curves.

Keywords: fault injection attack, modular extension, asymmetric cryptography, elliptic curve cryptography, Edwards curves.

1 Introduction

Properly used cryptography is a key building block for secure information exchange. Thus, implementation-level hacks must be considered seriously in addition to the threat of cyber-attacks. In particular, fault injection attacks target physical implementations of secured devices in order to induce exploitable errors.

Asymmetric cryptography Asymmetric cryptography addresses different needs such as key exchange and digital signature. RSA, Diffie-Hellman, and ElGamal have been used for decades, and elliptic curve cryptography (ECC) algorithms such as ECDSA [24] are more and more deployed. ECC pairing-based cryptography has recently been accelerated in practice and is thus becoming practical [36]. For example, the construction of "pairing-friendly" elliptic curves is an active subject [23]. Homomorphic encryption schemes are getting more practical and are progressively considered viable solutions for some real-world applications requiring strong privacy. All these algorithms use large numbers and take place in mathematical structures such as finite rings and fields, which enables powerful mathematical properties but also facilitates attacks.

Fault Attacks As put forward in the reference book on fault analysis in cryptography [27, Chp. 9], there are three main categories of fault attacks.
1) *Safe-error attacks* consist in testing whether an intermediate variable is dummy (usually introduced against simple power analysis [30]) or not, by faulting it and looking whether there is an effect on the final result.
2) *Cryptosystem parameter alterations* aim at weakening the algorithm in order to ease key extraction. For example [10], invalid-curve fault attacks consist in moving an ECC computation to a weaker curve, enabling the attacker to use cryptanalysis attacks exploiting the faulty outputs.
3) Finally, the most serious attacks belong to the *differential fault analysis* (DFA) category. Often the attack path consists in comparing correct and faulted outputs, like in the well-known BellCoRe attack on CRT-RSA (RSA sped up using the Chinese Remainder Theorem), or the sign-change fault attack on ECC.

The *BellCoRe attack* [15] on CRT-RSA introduced the concept of fault injection attacks. It is very powerful: faulting the computation even in a very random way yields almost certainly an exploitable result allowing to recover the secret primes of the RSA modulus $N = pq$.

The *sign-change attack* [14] on ECC consists in changing the sign of an intermediate elliptic curve point in the midst of an elliptic curve scalar multiplication (ECSM). The resulting faulted point is still on the curve so the fault is not detected by traditional point validation countermeasures. Such a fault can be achieved by for instance changing the sign in the double operation of the ECSM algorithm (line 3 of Alg. 1). If the fault injection occurs during the last iteration of the loop, then the final result $\widehat{Q} = [-2\sum_{i=1}^{n-1} k_i 2^{i-1}]P + k_0 P = -Q + 2k_0 P$, i.e., either $\widehat{Q} = -Q$ or $\widehat{Q} = -Q + 2P$ depending on k_0, which reveals the value of k_0 to the attacker. This process can be iterated to find the other bits of the scalar, and optimizations exist that trade-off between the number of necessary faulted results and the required exhaustive search.

```
Input  : $P \in \mathcal{E}$, $k = \sum_{i=0}^{n-1} k_i 2^i$ ($n$ is the scalar size in bits, where $k_i \in \{0,1\}$)
Output : $[k]P$
1 $Q \leftarrow \mathcal{O}$                                  ▷ $\mathcal{O}$ is the point at infinity
2 for $i \leftarrow n-1$ down to $0$ do
3     $Q \leftarrow 2Q$                                       ▷ ECDBL
4     if $k_i = 1$ then $Q \leftarrow Q + P$                  ▷ ECADD
5 return $Q$
```

Algorithm 1: Double-and-add left-to-right scalar multiplication on elliptic curve $\mathcal{E}$.

Both RSA and ECC algorithms continue to be the target of **many** new fault injection attacks: see [3,31,11,12,21] just for some 2014 papers. Besides, this topic is emerging and other new fault attacks will appear sooner or later. Hence, the need for efficient and practical generic countermeasures against fault attacks is obvious. David Wagner from UC Berkeley concurs in [40]: "*It is a fascinating research problem to establish a principled foundation for security against fault attacks and to find schemes that can be proven secure within that framework.*"

Countermeasures Verifications compatible with mathematical structures can be applied either at computational or at algorithmic level.

Algorithmic protections have been proposed by Giraud [22] (and many others [16,32,29]) for CRT-RSA, which naturally transpose to ECC, as shown in [28]. These protections are implementation specific (e.g., depend on the chosen exponentiation algorithm) and are thus difficult to automate, requiring specialized engineering skills.

Computational protections have been pioneered by Shamir in [37] using *modular extension*, initially to protect CRT-RSA. The idea is to carry out the same computation in two different algebraic structures allowing to check the computation before disclosing its result. For example protecting a computation in $\mathbb{F}_p$ consists in carrying out the same computation in $\mathbb{Z}_{pr}$ and $\mathbb{F}_r$ ($\mathbb{Z}_{pr}$ is the direct product of $\mathbb{F}_p$ and $\mathbb{F}_r$), where r is a small number ($r \ll p$); the computation in $\mathbb{Z}_{pr}$ must match that of $\mathbb{F}_r$ when reduced modulo r, if not an error is returned, otherwise the result in $\mathbb{Z}_{pr}$ is reduced modulo p and returned. The principle of modular extension is sketched in Fig. 1. This method operates at low level (integer arithmetic), thereby enabling countermeasures (and optimizations) to be added on top of it. They are thus easily maintained, which explains why this method is quite popular. Indeed, there is a wealth of variants for CRT-RSA stemming from this idea [1,39,26,13,17,19], as well as a few proofs-of-concept transposing it to ECC [14,2,25]. Despite the nonexistence of literature, the same idea could apply to post-quantum code-based cryptography, pairing, and homomorphic computation for instance. Therefore, our paper focuses on computational countermeasures.

On the one hand, the variety of CRT-RSA countermeasures shows that fault attacks are a threat that is taken seriously by both the academic and the industrial communities. On the other hand, it bears witness to the artisanal way these countermeasures were put together. Indeed, the absence of formal security claims and of proofs added to the necessity of writing implementations by hand results in many weaknesses in existing countermeasures and thus in many attempts to create better ones.

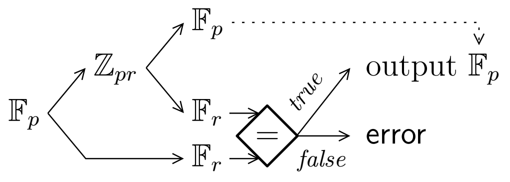

Fig. 1. Sketch of the principle of *modular extension.*

Contributions The countermeasure described in [2] can be applied only on Weierstrass curve, and the overhead computation is 48% for curve with parameters on 256 bits. The main disadvantage of this countermeasure is the need for point testing during the addition and doubling operations. These tests can differ in $\mathbb{Z}_{pr}$ and $\mathbb{F}_r$, hence a *loose security proof*, because the computation in $\mathbb{F}_r$ can be trapped in the point at infinity.

In this paper, we take advantage of the speed-up record on ECDSA computation using the twisted Edwards curve `Ed25519` [6] coded with NaCl crypto-library [9]. We propose a new countermeasure against faults injection based on modulus extension with only one "test-free" addition operation using *complete* and *unified* formulas of addition point on Edwards and twisted Edwards curves. This allows for a *synchronized* computation in $\mathbb{F}_p$ and $\mathbb{F}_r$ while computing in $\mathbb{Z}_{pr}$, as opposed to the state-of-the-art countermeasures, such as [14,2,25]. Our countermeasure is new insofar as we give explicit conditions on the prime r: they happen to be easily met in the case of Edwards curves (see Sec. 6.1), whereas they restrict the number of possible r to a little number of values for the popular twisted Edwards curves (see Sec. 6.2). The overhead computation of this countermeasure is 28% for Edwards curve and 39% for twisted Edwards curve on 32-bit processors, such as an ARM Cortex-M4.

Outline The rest of the paper is organized as follows. Section 2 described the details of the existing countermeasure for ECC. Section 3 is the theoretical analysis of our new countermeasure using the modular extension with test-free. Section 4 described the mathematical background on the Edwards curves. The description of our countermeasure to make the modular extension without test in the elliptic curve operation is in Section 5. Section 6 explains the overhead of computation and some examples of our countermeasure. Section 7 concludes.

2 Existing Countermeasures for ECC

Countermeasures against fault injection attacks have been proposed for elliptic curve computations, but they are actually incorrect. For example, in [14], Blömer, Otto, and Seifert (BOS) propose a countermeasure based on the modular extension idea of Shamir for CRT-RSA [37]. The problem is that the modular extension scheme cannot actually be applied as is to Weierstrass elliptic curve, because the tests in the ECDBL and ECADD operations are not true at the same times for the computation in $\mathbb{Z}_{pr}$ and the one in $\mathbb{F}_r$, which breaks the scheme and will yield false negatives. This behavior can be a *serious security issue* as it reveals information about the inputs.

In 2010 Joye patented [25] essentially the same countermeasure except it uses $\mathbb{F}_{r^2}$ and $\mathbb{Z}_{pr^2}$ instead of $\mathbb{F}_r$ and $\mathbb{Z}_{pr}$, which does not address the raised issues.

In [2], Baek and Vasyltsov (BV) propose a countermeasure based on modular extension and point verification. The particularity of this countermeasure is that instead of computing a sibling ECSM on a smaller curve $\mathcal{E}(\mathbb{F}_r)$ to compare with its redundant counterpart over $\mathcal{E}(\mathbb{Z}_{pr})$, it only checks whether the point obtained by reducing the result $\mathcal{E}(\mathbb{Z}_{pr})$ modulo r is on the $\mathcal{E}(\mathbb{F}_r)$ curve (i.e., whether it satisfies the curve equations modulo r). Because of that, BV does not suffer from BOS problem. However, the correctness of BV comes with a drawback: indeed, faults may go undetected if they happen before $\mathcal{O}$ (the point at infinity) is reached in the computation modulo r as the intermediate point quickly tends to $(0 : 0 : 0)$ in projective coordinates and stays there until the end.

It is actually possible to get the best of both world: what is needed is BOS approach (i.e., pure modular extension scheme) but without the problematic tests. Luckily, Edwards curves allow to perform ECC without tests thanks to a complete addition law, as will be detailed in Sec. 4. But before, we will formally analyze the security of the modular extension scheme when the implementation is test-free.

3 Security Analysis of Modular Extension

Definition 1 (Fault model). *We consider an attacker who can fault data by randomizing or zeroing any intermediate variable, and fault code by skipping any number of consecutive instructions.*

Remark 1. The three fault models have been described several times in the literature. For example, randomizing faults are discussed in [15], zeroing faults in [18], and instruction skip faults in [35].

Definition 2 (Attack order). *We call order of the attack the number of faults (in the sense of Def. 1) injected during the target execution.*

In the rest of this section, we focus mainly on the resistance to first-order attacks on data.

Definition 3 (Secure algorithm). *An algorithm is said secure if it is correct and if it either returns the right result or an* error *constant when faults have been injected, with an overwhelming probability.*

Theorem 1 (Security of test-free modular extension). *Test-free algorithms protected using the modular extension technique, are secure as per Def. 3. In particular, the probability of non-detection is inversely proportional to the security parameter r.*

Proof. Faulted results are polynomials of faults. The result of an asymmetric cryptography computation can be written as a function of a subset of the intermediate variables, plus some inputs if the intermediate variables do not suffice to finish the computation. We are interested in the expression of the result as a function of the intermediate variables which are the target of a transient or permanent fault injection. We give the formal name $\widehat{x}$ to any faulted variable x. For convenience, we denote them by $\widehat{x_i}$, $1 \leq i \leq n$, where $n \geq 1$ is the number of injected faults. The result consists in additions, subtractions, and multiplications of those formal variables (and inputs). Such expression is a multivariate polynomial. If the inputs are fixed, then the polynomial has only n formal variables. We call it $P(\widehat{x_1}, \ldots, \widehat{x_n})$. For now, let us assume that $n = 1$, i.e., that we face a single fault. Then P is a monovariate polynomial. Its degree d is the multiplicative depth of $\widehat{x_1}$ in the result.

A fault is not detected if and only if $P(\widehat{x_1}) = P(x_1) \mod r$, whereas $P(\widehat{x_1}) \neq P(x_1) \mod p$. Notice that the latter condition is superfluous insofar since if it is negated then the effect of the fault does not alter the result in $\mathbb{F}_p$.

Non-detection probability is inversely proportional to r. As the faulted variable $\widehat{x_1}$ can take any value in $\mathbb{Z}_{pr}$, the non-detection probability $\mathbb{P}_{\text{n.d.}}$ is given by:

$$\begin{aligned}
\mathbb{P}_{\text{n.d.}} &= \frac{1}{pr-1} \cdot \sum_{\widehat{x_1} \in \mathbb{Z}_{pr} \setminus \{x_1\}} \mathbb{1}_{P(\widehat{x_1}) \,=\, P(x_1) \mod r} \\
&= \frac{1}{pr-1} \cdot \Big(-1 + p \sum_{\widehat{x_1}=0}^{r-1} \mathbb{1}_{P(\widehat{x_1}) \,=\, P(x_1) \mod r} \Big). \qquad (1)
\end{aligned}$$

Here, $\mathbb{1}_{\text{condition}}$ is an indicator function: it is equal to 1 (resp. 0) if the condition is true (resp. false).

Let $\widehat{x_1} \in \mathbb{Z}_r$, if $P(\widehat{x_1}) = P(x_1) \mod r$, then $\widehat{x_1}$ is a root of the polynomial $\Delta P(\widehat{x_1}) = P(\widehat{x_1}) - P(x_1)$ in $\mathbb{Z}_r$. We denote by $\#\text{roots}(\Delta P)$ the number of roots of ΔP over $\mathbb{Z}_r$. Thus (1) computes $(p \times \#\text{roots}(\Delta P) - 1)/(pr - 1) \approx \#\text{roots}(\Delta P)/r$.

Study of the proportionality constant. A priori, bounds on this value are broad since $\#\text{roots}(\Delta P)$ can be as high as the degree d of ΔP in $\mathbb{Z}_r$, i.e., $\min(d, r-1)$. However, in practice, ΔP looks like a random polynomial over the finite field $\mathbb{Z}_r$, for several reasons:

- inputs are random numbers in most cryptographic algorithms, such as probabilistic signature schemes,
- the coefficients of ΔP in $\mathbb{Z}_r$ are randomized due to the reduction modulo r.

In such case, the number of roots is very small, despite the possibility of d being large. See for instance [33] for a proof that the number of roots tends to 1 as $r \to \infty$. Interestingly, random polynomials are still friable (i.e., they are clearly not irreducible) in average, but most factors of degree greater than one happen not to have roots in $\mathbb{Z}_r$. Thus, we have $\mathbb{P}_{\text{n.d.}} \gtrsim \frac{1}{r}$, meaning that $\mathbb{P}_{\text{n.d.}} \geq \frac{1}{r}$ but is close to $\frac{1}{r}$. A more detailed study of the theoretical upper bound on the number of roots is available in Sec. A.

The same law applies to multiple faults. In the case of multiple faults ($n > 1$), then the probability of non-detection generalizes to:

$$\begin{aligned}
\mathbb{P}_{\text{n.d.}} &= \tfrac{1}{(pr-1)^n} \cdot \sum_{\widehat{x_1},\ldots,\widehat{x_n} \in \mathbb{Z}_{pr}\setminus\{x_1\}\times\ldots\times\mathbb{Z}_{pr}\setminus\{x_n\}} \mathbb{1}_{P(\widehat{x_1},\ldots,\widehat{x_n})=P(x_1,\ldots,x_n) \bmod r} \\
&= \tfrac{1}{(pr-1)^n} \cdot \sum_{\widehat{x_2},\ldots,\widehat{x_n} \in \prod_{i=2}^{n} \mathbb{Z}_{pr}\setminus\{x_i\}} \left[\sum_{\widehat{x_1} \in \mathbb{Z}_{pr}\setminus\{x_1\}} \mathbb{1}_{P(\widehat{x_1},\ldots,\widehat{x_n})=P(x_1,\ldots,x_n) \bmod r} \right] \\
&= \tfrac{1}{(pr-1)^n} \cdot \sum_{\widehat{x_2},\ldots,\widehat{x_n} \in \prod_{i=2}^{n} \mathbb{Z}_{pr}\setminus\{x_i\}} [p \times \#\text{roots}(\mathit{\Delta} P) - 1] \\
&= \tfrac{1}{(pr-1)^n} \cdot (pr-1)^{n-1} \, [p \times \#\text{roots}(\mathit{\Delta} P) - 1] \\
&= \frac{p \times \#\text{roots}(\mathit{\Delta} P) - 1}{pr - 1}.
\end{aligned}$$

Therefore, the probability not to detect a fault when $n > 1$ is identical to that for $n = 1$. Thus, we also have $\mathbb{P}_{\text{n.d.}} \approx \frac{1}{r}$ in the case of multiple faults of the intermediate variables[7].

4 Edwards Curves

In mathematics, the Edwards curves are a family of elliptic curves studied by Harold M. Edwards in 2007 [20]. Technically, an Edwards curve is not elliptic, because it has singularities; but resolving those singularities produces an elliptic curve. The concept of elliptic curves over finite fields is widely used in elliptic curve cryptography. Applications of Edwards curves to cryptography were developed by Bernstein and Lange: they pointed out several advantages of the Edwards form in comparison to the more well known Weierstrass form.

4.1 Edwards curves over large-characteristic fields

Definition 4 (Edwards curves). *On the finite field $\mathbb{F}_p$ with p a prime number, an elliptic curve in Edwards form has parameters c, d in the finite field $\mathbb{F}_p$ and coordinates (x, y) satisfying the following equation:*

$$x^2 + y^2 = c^2(1 + dx^2y^2), \tag{2}$$

with $cd(1 - c^4d) \neq 0$.

The main advantage to use the Edwards curves is that addition formulas are *unified*: doubling and addition operations are the same. Affine unified addition formula is $(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$, where:

$$\begin{cases} x_3 = \frac{x_1y_2 + y_1x_2}{c(1+dx_1x_2y_1y_2)}, \\ y_3 = \frac{y_1y_2 - x_1x_2}{c(1-dx_1x_2y_1y_2)}. \end{cases} \tag{3}$$

The affine negation formula is as expected: $-(x_1, y_1) = (-x_1, y_1)$.

The neutral element of the curve is the point $(0, c)$. Contrary to Weierstrass curves, this point is not special (there is no abstract "*point at infinity*"), but verifies the curve equation. The point $(0, -c)$ has order 2. The points $(c, 0)$ and $(-c, 0)$ have order 4.

[7] Note that this study does not take correlated faults into account.

Bernstein and Lange [7] proved that if d is not a square in $\mathbb{F}_p$ then the unified addition law is *complete*. This means that the addition formula is valid for all points, with no exception. That is one of the advantages of Edwards curves over Weierstrass curves in which the addition law is *not complete*: a *complete* addition law provides some resistance to side-channel attacks.

4.2 Twisted Edwards curves over large-characteristic fields

Twisted Edwards curves are a generalization of Edwards curves [5].

Definition 5 (Twisted Edwards curves). *Let p a prime number. On the finite field $\mathbb{F}_p$, an elliptic curve in twisted Edwards form has parameters a, d in the finite field $\mathbb{F}_p$ and coordinates (x, y) satisfying the following equation:*

$$ax^2 + y^2 = 1 + dx^2y^2, \tag{4}$$

with $ad(a - d) \neq 0$.

Like Edwards curves, the addition formulas are unified. Affine unified addition formula is $(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$, where:

$$\begin{cases} x_3 = \frac{x_1y_2+y_1x_2}{1+dx_1x_2y_1y_2}, \\ y_3 = \frac{y_1y_2-ax_1x_2}{1-dx_1x_2y_1y_2}. \end{cases} \tag{5}$$

The neutral element is $(0, 1)$. Affine negation formula is natural: $-(x_1, y_1) = (-x_1, y_1)$.

Addition law is *complete* if a is a square and d is a non-square [7].

5 Practical Study

On Edwards curves and twisted Edwards curves, the addition law is *complete*: addition formulas work for all pairs of input points. In particular, there is no troublesome point at infinity. Another advantage of Edwards curve is the atomicity of the formula doubling and adding and the constant time to protect the classical Side-Channel Attack. The addition law is *unified*, meaning that there are no test to verify if the two input points are equal, opposite or different. To be more efficient, we use the unified projective coordinates to the addition law ECADD-complete-unified named "add-2007-bl-2" on [8] or on [7, Sec. 4, page 9].

The ECSM with modular extension protection using complete unified addition formulas is given in Alg. 2. The first phase can compute offline, because find r verifies the lemmas 1 and 2 is not trivial. The second phase is composed by two ECSM computation online. The first ECSM computation consists in multiplying the point P with the scalar k on the ring $\mathbb{Z}_{pr}$ using the parameters defined later in this section by the proprieties 1 or 2. The second ECSM computation is the multiplication of the point P' with the scalar k on the *small* curve over $\mathbb{F}_r$ using the parameters defined in the lemmas 1 or 2. It is worthwhile to note that these two ECSM share the same code (see. Alg. 3).

Given an elliptic curve over $\mathbb{F}_p$ and a point (x_G, y_G), we define by λ the multiple of p to add when the *point on curve* equation is plunged from $\mathbb{F}_p$ to $\mathbb{Z}$. Formally,

Definition 6 (Parameter λ for Edwards curves). *Given an Edwards elliptic curve of equation* (2)*, the parameter λ is the integer satisfying the relationship in $\mathbb{Z}$:*

$$x_G^2 + y_G^2 = c^2(1 + dx_G^2y_G^2) + \lambda p.$$

```
   Input  : $P \in \mathcal{E}(\mathbb{F}_p)$, $k \in \mathbb{Z}$
   Output : $Q = [k]P \in \mathcal{E}(\mathbb{F}_p)$
   ------------------------------- Offline phase -------------------------------
   Edwards Curves:
 1 Compute $\lambda p = x_G^2 + y_G^2 - c^2(1 + cx^2y^2)$
 2 repeat
 3     Choose a random prime $r < p$
 4     Compute $x'_G = X_G \mod r$
 5     Compute $y'_G = y_G \mod r$
 6     Compute $c' = c^2 + \lambda p \mod r$
 7     Compute $d' = \frac{dc^2}{c^2+\lambda p} \mod r$
   until $x'_G \neq 0$ and $y'_G \neq 0$ and $c'd'(1 - c'^4 d') \neq 0$ and $c'$ a square and $d'$ a no-square
                                                         ▷ $r$ verifies the lemma 1

   Twisted Edwards Curves:
 1 Compute $\lambda = (1 + dx_G^2 y_G^2 - ax_G^2 + y_G^2) \div p$
 2 Find all the factor $r$ smaller than $p$ of $\lambda$
 3 for each factor $r$ do
 4     Compute $x'_G = x_G \mod r$
 5     Compute $y'_G = y_G \mod r$
 6     Compute $a' = a \mod r$
 7     Compute $d' = d \mod r$
 8     if $x'_G \neq 0$ and $y'_G \neq 0$ and $a'd'(a'-d') \neq 0$ and $a'$ a square and $d'$ a no-square then
 9         break                                         ▷ $r$ verifies the lemma 2
       else
10         $r$ does not work

11 Determine the small curve $\mathcal{E}(\mathbb{F}_r)$ with parameter $c'$ (or $a'$) and $d'$ , and a point $P'(x'_G, y'_G)$ is on that curve.
12 Determine the combined curve $\mathcal{E}(\mathbb{Z}_{pr})$ with parameter $C = CRT(c, c')$ (or $A = CRT(a, a')$) and $D = CRT(d, d')$     ▷ using properties 1 and 2.
   ------------------------------- Online phase --------------------------------
13 $(X_{pr} : Y_{pr} : Z_{pr}) = \text{ECSM}(P, k, \mathcal{E}(\mathbb{Z}_{pr}))$     ▷ without test on the point and on the scalar value
14 $(X_r : Y_r : Z_r) = \text{ECSM}(P', k, \mathcal{E}(\mathbb{F}_r))$     ▷ without test on the point and on the scalar value
15 if $(X_{pr} \mod r : Y_{pr} \mod r : Z_{pr} \mod r) = (X_r : Y_r : Z_r)$ then
16     return $(X_{pr} \mod p : Y_{pr} \mod p : Z_{pr} \mod p)$
   else
17     return error
```

Algorithm 2: ECSM with modular extension protection using complete unified addition formulas.

Definition 7 (Param. λ for twisted Edwards curves). *Given a twisted Edwards elliptic curve of equation (4), the parameter λ is the integer satisfying the following relationship in $\mathbb{Z}$:*

$$ax_G^2 + y_G^2 = 1 + dx_G^2 y_G^2 + \lambda p.$$

5.1 Edwards curves

Lemma 1. *Let p be a prime and an Edwards curve over $\mathbb{F}_p$ as per definition 4, parameterized by c, d. Let λ as per definition 6.*

Let r be a prime number $r < p$, such that $c^2 + \lambda p$ is a non-zero square in $\mathbb{F}_r$, $x_G \bmod r \neq 0$ and $y_G \bmod r \neq 0$. The set of points which satisfy $\mathcal{E}_r : x^2 + y^2 = c'^2(1 + d'x^2y^2) \mod r$ with

$$\begin{cases} c'^2 = c^2 + \lambda p \mod r & \textit{and} \\ d' = \frac{dc^2}{c^2+\lambda p} \mod r \end{cases}$$

is an Edwards curve, generated by the point $(x'_G, y'_G) = (x_G \bmod r, y_G \bmod r)$.

If the parameters c' and d' satisfy $c'd'(1 - c'^4d') \neq 0$ and d' is not a square in the finite field $\mathbb{F}_r$, then the ECSM computation on this small Edwards curve $\mathcal{E}(\mathbb{F}_r)$ is complete, i.e., can be computed without point or scalar conditional tests.

Proof. Let r a prime number smaller than p. If c, d, p, λ verify the conditions given in definitions 4 and 6, and if $c^2 + \lambda p$ is a non-zero square in $\mathbb{F}_r$, then we have on $\mathbb{Z}$:

$$\begin{aligned} x_G^2 + y_G^2 &= c^2(1 + dx_G^2y_G^2) + \lambda p \\ &= (c^2 + \lambda p) + c^2 dx_G^2 y_G^2. \end{aligned}$$

As by hypothesis $(c^2 + \lambda p) \bmod r \neq 0$, then we have in $\mathbb{F}_r$:

$$x_G^2 + y_G^2 = (c^2 + \lambda p)\left(1 + \frac{c^2 d}{c^2 + \lambda p} x_G^2 y_G^2\right).$$

As in addition $(c^2 + \lambda p)$ is a square in $\mathbb{F}_r$, we have in $\mathbb{F}_r$:

$$x_G^2 + y_G^2 = \left(\sqrt{c^2 + \lambda p}\right)^2 \left(1 + \frac{c^2 d}{c^2 + \lambda p} x_G^2 y_G^2\right).$$

By definition 4, the parameter $c' = \sqrt{c^2 + \lambda p}$ in $\mathbb{F}_r$ and $d' = \frac{c^2 d}{c^2+\lambda p}$ in $\mathbb{F}_r$ with the assumption $c'd'(1 - c'^4d') \neq 0$ are the parameters of the Edwards curve $\mathcal{E}_r$ and the point $(x'_G, y'_G) = (x_G \bmod r, y_G \bmod r)$ is on the curve $\mathcal{E}(\mathbb{F}_r)$ by curve construction.

If $x_G \bmod r \neq 0$ and $y_G \bmod r \neq 0$ then the order of the point (x'_G, y'_G) is greater than 4, because the point at infinity and the point of order 2 on $\mathcal{E}(\mathbb{F}_r)$ have the x-coordinate equal to zero, and the point of order 4 on $\mathcal{E}(\mathbb{F}_r)$ has the y-coordinate equal to zero.

If d' is not a square in $\mathbb{F}_r$, then the addition law on this *small* curve is unified and complete [7]. Thus there is no need for point verification testing when performing additions in an ECSM, and the scalar can be an integer greater than the order of the *small* curve $\mathcal{E}(\mathbb{F}_r)$.

For the purpose of the modular extension countermeasure, we extend the notion of Edwards curve to rings[8] (such as $\mathbb{Z}_{pr}$).

[8] Similar idea can be found in [14,2,25]; we explicit it here for the article to be self-contained.

Proposition 1. *Let an Edwards curve defined on $\mathbb{F}_p$ with the parameters c, d and the point (x_G, y_G). If a random number r verifying the lemma 1 can be found to define the Edwards curve $\mathcal{E}(\mathbb{F}_r)$ with the parameters c', d', then $C = CRT(c, c')$ and $D = CRT(d, d')$ are the parameters of an Edwards elliptic curve over the rings $\mathbb{Z}_{pr}$. This curve parameters permit to detect a fault thanks to the comparison at line 15 in the Algorithm 2.*

Proof. We introduce the following notations:

- We denote by Pt_p with p in index a point named Pt computed on the $\mathcal{E}(\mathbb{F}_p)$;
- We denote by Pt_r with r in index a point named Pt computed on the $\mathcal{E}(\mathbb{F}_r)$;
- We denote by Pt_{pr} with pr in index a point named Pt computed on the $\mathcal{E}(\mathbb{Z}_{pr})$.

The input value of the two ECSMs verify the equality using the projective coordinates, because we have as input (x_G, y_G) for the combined curve and (x'_G, y'_G) for the *small* curve:

$$\begin{cases} X\text{- coordinate: } x'_G &= x_G \mod r, \\ Y\text{- coordinate: } y'_G &= y_G \mod r, \\ Z\text{- coordinate: } 1 &= 1 \mod r. \end{cases} \tag{6}$$

The ECSM computation over the combined curve on the ring extension $\mathbb{Z}_{pr}$ and the *small* curve over finite field $\mathbb{F}_r$ do consist in the same sequence of addition operations (ECADD-complete-unified).

Let P_{pr} and P_r be two points such that $X_{P_r} = X_{P_{pr}} \mod r, Y_{P_r} = Y_{P_{pr}} \mod r, Z_{P_r} = Z_{P_{pr}} \mod r$. Let Q_{pr} and Q_r be two points such that $X_{Q_r} = X_{Q_{pr}} \mod r, Y_{Q_r} = Y_{Q_{pr}} \mod r, Z_{Q_r} = Z_{Q_{pr}} \mod r$.

We compute R_r the result of ECADD-complete-unified between P_r and Q_r over $\mathcal{E}(\mathbb{F}_r)$, and R_{pr} the result of ECADD-complete-unified between P_{pr} and Q_{pr} over $\mathcal{E}(\mathbb{Z}_{pr})$.

The computation of the projective coordinates of R_{pr} is composed by addition, subtraction, multiplication over the ring $\mathbb{Z}_{pr}$ using the projective coordinates of P_{pr} and Q_{pr} and the two curve parameters C and D.

The computation of the projective coordinates of R_{pr} is composed by addition, subtraction, multiplication over the ring $\mathbb{Z}_{pr}$ using the projective coordinates of P_r and Q_r and the two curve parameters c' and d'.

By construction $C = CRT(c, c')$ and $D = CRT(d, d')$, we have $C \bmod r = c'$ and $D \bmod r = d'$, so the projective coordinates of R_{pr} are pairwise equal modulo r with the projective coordinates of R_r.

As the ECADD-complete-unified operation conserves the equality of the point coordinates value modulo r, we conclude that the ECSM computation conserves the equality of the point coordinates value modulo r between the computation over the ring extension and over the finite field $\mathbb{F}_r$.

5.2 Twisted Edwards curves

Lemma 2. *If a, d, p, λ verify the conditions defined in definition 7, then if we can choose a prime factor r of λp such that $x_G \bmod r \neq 0$ and such that the point $(x'_G, y'_G) = (x_G \bmod r, y_G \bmod r)$ generates the curve $\mathcal{E}(\mathbb{F}_r) : a'x^2 + y^2 = 1 + d'x^2y^2$ where $a' = a \bmod r$ and $d' = d \bmod r$.*

If the parameters a' and d' satisfy $a'd'(a' - d') \neq 0$, a' is a square and d' is a non-square in the finite field $\mathbb{F}_r$, then the ECSM computation on this small *twisted Edwards curve $\mathcal{E}(\mathbb{F}_r)$ requires no point and scalar tests.*

Proof. If a, d, p, λ verify the conditions given in definition 7, then we have we have on $\mathbb{Z}$:

$$ax_G^2 + y_G^2 = 1 + dx_G^2 y_G^2 + \lambda p.$$

Let r a prime factor of λp, then $\lambda p = 0 \mod r$. Hence we have:

$$\begin{aligned}&(a \bmod r)(x_G \bmod r)^2 + (y_G \bmod r)^2\\ &= 1 + (d \bmod r)(x_G \bmod r)^2(y_G \bmod r)^2.\end{aligned}$$

By definition 5, the parameters $(a', d') = (a \bmod r, d \bmod r)$ with the assumption $a'd'(a' - d') \neq 0$ are the parameters of the twisted Edwards curve $\mathcal{E}_r$ and the point $(x'_G, y'_G) = (x_G \bmod r, y_G \bmod r)$ is on the curve $\mathcal{E}_r$ by curve construction.

By definition 5, the parameters $(a', d') = (a \bmod r, d \bmod r)$ with the assumption $a'd'(a' - d') \neq 0$ are the parameters of the twisted Edwards curve $\mathcal{E}(\mathbb{F}_r)$ and the point $(x'_G, y'_G) = (x_G \bmod r, y_G \bmod r)$ is on the curve $\mathcal{E}(\mathbb{F}_r)$ by curve construction.

If a' is a square and d' is a non-square in $\mathbb{F}_r$, then the addition law on this *small* curve is unified and complete [7], which implies that no point verification testing is required for each addition in ECSM, and that the scalar can be an arbitrary integer, for instance greater than the order of the *small* curve.

For the purpose of the modular extension countermeasure depicted in Alg. 2, we extend the notion of twisted Edwards curve to rings[8] (such as $\mathbb{Z}_{pr}$).

Proposition 2. *Let a twisted Edwards curve defined on $\mathbb{F}_p$ with the parameters a, d and the point (x_G, y_G). If a random number r verifying the lemma 2 can be found to define the twisted Edwards curve $\mathcal{E}(\mathbb{F}_r)$ with the parameters a', d', then $A = a$ and $D = d$ are the parameters of a twisted Edwards curve over the ring $\mathbb{Z}_{pr}$.*

If $x_G \bmod r \neq 0$ then the point (x'_G, y'_G) is not the point at infinity. So, this point (x'_G, y'_G) is a generator of a non-trivial subgroup of the elliptic curve $\mathcal{E}(\mathbb{F}_r)$.

This curve parameters permit to detect a fault with the comparison at line 15 in the Algorithm 2.

Proof. The input value of the two ECSM verify the equality using the projective coordinates, because we have as input (x_G, y_G) for the combined curve and (x'_G, y'_G) for the *small* curve, as described previously in Eqn. (6).

The ECSM computation over the combined curve on the ring extension $\mathbb{Z}_{pr}$ and the *small* curve over finite field $\mathbb{F}_r$ consist in the same sequence of addition operations (ECADD-complete-unified). Namely, the sequence is given in Alg. 3, where $\mathcal{E}$ is either $\mathcal{E}(\mathbb{Z}_{pr})$ or $\mathcal{E}(\mathbb{F}_r)$.

By construction $A = a, D = d$ and we have $a \bmod r = a'$ and $d \bmod r = d'$, so the projective coordinates of R_{pr} are equal modulo r two by two with the projective coordinates of R_r.

As the ECADD-complete-unified operation conserves the equality of the point coordinates value modulo r, we conclude that the ECSM computation conserves the equality of the point coordinates value modulo r between the computation over the ring extension and over the finite field $\mathbb{F}_r$.

5.3 Discussion

About small curve requirements Both for Edwards and twisted Edwards curves, the small curve is of course not a cryptographic-grade curve. Indeed, the modulus r is too small and the curve might have points of low order. However, the small curve is not intended to be the support of a secure cryptographic operation: the computation on this curve actually remains internal to fault-detection-enabled ECSM. That is, the small curve is intended here to carry out exactly the same computation as that done in the curve on the extended ring, in order to enable the integrity verification.

Resistance to some attacks As a general guideline, additional protection against the *common point* attack [4] shall be enforced. This attack is based on curve parameters alteration, with the hope that the obtained curve is weak. Thus, to thwart this attack, the curve parameters shall be tested before and after the computation.

6 Performance

Our implementation uses the projective coordinates described in [7, Sec. 4, page 9]. Projective unified addition version takes $10\mathcal{M} + 1\mathcal{S} + 1\mathcal{C} + 1\mathcal{D} + 7\mathcal{A}$ where $\mathcal{M}$ is the cost of multiplication, $\mathcal{S}$ is the cost of square, $\mathcal{C}$ is the cost of multiplying by c, $\mathcal{D}$ is the cost of multiplying by d, and $\mathcal{A}$ abbreviates addition. The ECSM is the algorithm *add-always left-to-right* like described in Alg. 3. The bitwidth of the modulus is denoted by n (e.g., $n = 256$ for `Ed25519`). We denote by n' the number of words of the modulus, that is $n' = 256/32 = 8$ on 32-bit platforms (or $n' = 256/16 = 16$ on 16-bit platforms). We consider that cost of a multiplication of two numbers composed by n' words is ${n'}^2$, cost of a square $\mathcal{S}$ is $0.8\mathcal{M}$ and the addition $\mathcal{A}$ is n'. The Table 1 permits to compare the time of each ECADD-complete-unified, depending of the number of words n'.

Curves type	ECADD-complete-unified on $\mathbb{F}_p$	ECADD-complete-unified on $\mathbb{Z}_{pr}$	ECADD-complete-unified on $\mathbb{F}_r$	Total cost of the countermeasure	Computational overhead with:	
					$n' = 8$	$n' = 16$
Edwards	$11.8{n'}^2 + 7n'$	$11.8{n'}^2 + 30.6n' + 18.8$	19.8	$11.8{n'}^2 + 30.6n' + 38.6$	$\approx +28\%$	$\approx +13\%$
Twisted Edwards	$11.8{n'}^2 + 7n'$	$12.8{n'}^2 + 32.6n' + 29.8$	19.8	$12.8{n'}^2 + 32.6n' + 49.6$	$\approx +39\%$	$\approx +21\%$

Table 1. Theory of the elliptic curve addition cost

```
  Input  : $P \in \mathcal{E}$, $k = \sum_{i=0}^{n-1} k_i 2^i$ ($n$ is the scalar size in bits, where $k_i \in \{0,1\}$)
  Output : $[k]P$
1 $R_0 \leftarrow P$
2 $R_1 \leftarrow P$
3 $j \leftarrow n - 2$
4 $b \leftarrow 0$
5 while $j \geq 0$ do
6     $R_0 \leftarrow R_0 + R_b$                    ▷ ECADD-complete-unified
7     $b \leftarrow b \oplus k_j$
8     $j \leftarrow j + k_j - 1$
9 return $R_0$
```

Algorithm 3: Add-always left-to-right scalar multiplication on elliptic curve $\mathcal{E}$.

6.1 Edwards curve example

For our experiment, we generate a Edwards curve on the finite field $\mathbb{F}_{2^{255}-19}$ defined by $x^2 + y^2 = 1 - 6x^2y^2 \mod 2^{255} - 19$.

Using the Prop. [34, sec 3.1], this Edwards curve corresponds to an elliptic curve defined by $\mathcal{E} : v^2 = u^3 + a_2u^2 + a_4u$ on $\mathbb{F}_{2^{255}-19}$, with $a_2 = -5$ and $a_4 = 49$. The number of elements defined on the curve computed by MAGMA tool [38] is:

$$\#\mathcal{E}(2^{255} - 19) = 2^{255} + 138694172605265013181071149003381840660.$$

We find a generator point (x_G, y_G) on the Edwards curve with:

$$x_G = 53746514586250388770967951861766021561817370662802863797712166095360241234126,$$
$$y_G = 19570081233560550597987439135529516381390903225319934175948181057081969418594.$$

The co-factor of the curve is 4. For the small curve, we can choose $r = 2147499037$; hence we have $c' = 1800340494, d' = 1430405543, x'_G = 28751952$ and $y'_G = 1290929995$. These parameters verify the lemma 1. The probability of fault non-detection is about equal to 2^{-31}.

Remark: The probability namely "$c^2 + \lambda p$ is a non-zero square in $\mathbb{F}_r$" is about 1/2 and the probability namely "$\frac{dc^2}{c^2+\lambda p}$ is a non-zero no-square in $\mathbb{F}_r$" is about 1/2. To generate 500.000 random primes $r < 2^{32}$ verifying the lemma 1, using online version on MAGMA tool [38], the time is 110.769 seconds. The number of random prime number generated is 1.999.238. The probability that a random prime r meets the requirement of lemma 1 is less than 1/4 verified by this experimental part.

6.2 Twisted Edwards curve example: `Curve25519` and `Ed25519`

On the finite field $\mathbb{F}_{2^{255}-19}$, the elliptic curve `Curve25519` defined by the equation $v^2 = u^3 + 48662u^2 + u$ is birationally equivalent to the twisted Edwards Curves `Ed25519` defined by equation $-x^2 + y^2 = 1 - \frac{121665}{121666}x^2y^2$.

This equivalence is given by:

$$\begin{cases} x = \frac{u}{v}\sqrt{-48664} \\ y = \frac{u-1}{u+1} \end{cases} \quad \text{or} \quad \begin{cases} u = \frac{1+y}{1-y} \\ v = \frac{1+y}{(1-y)x}\sqrt{-48664} \end{cases} . \tag{7}$$

The `Curve25519` is a Montgomery curve, where very efficient computations can be carried out using only the X and Z coordinates. We find a generator point (x_G, y_G) on the twisted Edwards curve `Ed25519` with:

$$x_G = 24727413235106541002554574571675588834622768167397638456726423682521233608206 3,$$
$$y_G = 15549675580280190176352668710449542251549572066445060580507079593062643049417.$$

The prime factors of λ (recall definition 7) smaller[9] than p are stored in the Table 2.

For the small curve like described in Table 2, we can choose:

1. $r = 78857, a' = 32865, d' = 47471, x'_G = 71670$ and $y'_G = 16752$, or
2. $r = 843229, a' = 839079, d' = 43998, x'_G = 96826$ and $y'_G = 488894$.

These parameters verify the lemma 2.

The probability of fault non-detection is about equal to 2^{-16} for the first case and to 2^{-19} for the second case.

Important remark: we notice that the small verification field $\mathbb{F}_r$ cannot be chosen at random. Instead, the value of r is highly constrained, as shown in Tab. 2. This limitation of the ring extension countermeasure was not previously known.

[9] Actually, there is in λ only one factor larger than p, of length ≈ 900 bits, hence of no practical use—it is indeed more efficient to perform the computation several times or to verify the signature.

Prime factors r	2	3	17	47	78857	843229	159962189299
Length in bit of r	2	2	5	7	16	19	40
r verifies the lemma 2	False	False	False	False	**True**	**True**	False

Table 2. Prime Factors $< p$ of λ for the generator point (x_G, y_G) given in example (curve `Ed25519` defined in Sec. 6.2)

6.3 Comments about results

One can see in Table 1 that the global time computation increases by 28% or 39% for each addition operation using a 256-bit curve with a 32-bit processor ($n' = 8$). The computation overhead decreases when the curve parameters and the security increase. Remarkably, the implementation code is the same for the two ECSM computations. The memory storage requirement is increased by two word registers for each variable.

7 Conclusions

It is well known that detecting faults while computing elliptic curve cryptography can be achieved thanks to ring extension. In this paradigm, two entangled computations are carried out in the extended ring, allowing to tightly produce the functional result along with a redundant one, which can be checked independently. However, classical methods fail because the redundant computation evaluated standalone or entangled can be different, owing to some tests being independently evaluated when the elliptic curve formulae are not complete. Edwards curves and twisted Edwards curves have complete formulae, hence are not concerned with the issue of consistent tests requirement. Still, the application of ring extension involves some technicalities, we discuss in the paper. Namely, Edwards curves require an adaptation with Chinese reminder theorem of the curve constant parameter. As for twisted Edwards curves, the modulus extension can only be performed with a factor of λ, which is related both to the curve parameters and to the base point.

The outcome is a provable fault detection method for (twisted) Edwards curves which, despite its simplicity, is novel, elegant and effective.

References

1. C. Aumüller, P. Bier, W. Fischer, P. Hofreiter, and J.-P. Seifert. Fault Attacks on RSA with CRT: Concrete Results and Practical Countermeasures. In B. S. Kaliski, Jr., Ç. K. Koç, and C. Paar, editors, *CHES*, volume 2523 of *Lecture Notes in Computer Science*, pages 260–275. Springer, 2002.
2. Y.-J. Baek and I. Vasyltsov. How to Prevent DPA and Fault Attack in a Unified Way for ECC Scalar Multiplication - Ring Extension Method. In E. Dawson and D. S. Wong, editors, *Information Security Practice and Experience*, volume 4464 of *Lecture Notes in Computer Science*, pages 225–237. Springer Berlin Heidelberg, 2007.
3. G. Barthe, F. Dupressoir, P. Fouque, B. Grégoire, and J. Zapalowicz. Synthesis of Fault Attacks on Cryptographic Implementations. In G. Ahn, M. Yung, and N. Li, editors, *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, pages 1016–1027. ACM, 2014.
4. A. Battistello. *Constructive Side-Channel Analysis and Secure Design: 5th International Workshop, COSADE 2014, Paris, France, April 13-15, 2014. Revised Selected Papers*, chapter Common Points on Elliptic Curves: The Achilles' Heel of Fault Attack Countermeasures, pages 69–81. Springer International Publishing, Cham, 2014.

5. D. J. Bernstein, P. Birkner, M. Joye, T. Lange, and C. Peters. Twisted edwards curves. In S. Vaudenay, editor, *Progress in Cryptology - AFRICACRYPT 2008, First International Conference on Cryptology in Africa, Casablanca, Morocco, June 11-14, 2008. Proceedings*, volume 5023 of *Lecture Notes in Computer Science*, pages 389–405. Springer, 2008.
6. D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B. Yang. High-speed high-security signatures. *J. Cryptographic Engineering*, 2(2):77–89, 2012.
7. D. J. Bernstein and T. Lange. Faster addition and doubling on elliptic curves. In K. Kurosawa, editor, *Advances in Cryptology - ASIACRYPT 2007, 13th International Conference on the Theory and Application of Cryptology and Information Security, Kuching, Malaysia, December 2-6, 2007, Proceedings*, volume 4833 of *Lecture Notes in Computer Science*, pages 29–50. Springer, 2007.
8. D. J. Bernstein and T. Lange. Explicit-Formulas Database, March 2015. http://hyperelliptic.org/EFD/.
9. D. J. Bernstein, T. Lange, and P. Schwabe. The security impact of a new cryptographic library. In A. Hevia and G. Neven, editors, *Progress in Cryptology - LATINCRYPT 2012 - 2nd International Conference on Cryptology and Information Security in Latin America, Santiago, Chile, October 7-10, 2012. Proceedings*, volume 7533 of *Lecture Notes in Computer Science*, pages 159–176. Springer, 2012.
10. I. Biehl, B. Meyer, and V. Müller. Differential fault attacks on elliptic curve cryptosystems. In *CRYPTO '00: Proceedings of the 20th Annual International Cryptology Conference on Advances in Cryptology*, pages 131–146, London, UK, 2000. Springer-Verlag.
11. J. Blömer, R. Gomes Da Silva, P. Gunther, J. Krämer, and J.-P. Seifert. A Practical Second-Order Fault Attack against a Real-World Pairing Implementation. In *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2014 Workshop on*, pages 123–136, Sept 2014. Busan, Korea.
12. J. Blömer, P. Günther, and G. Liske. Tampering Attacks in Pairing-Based Cryptography. In *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2014 Workshop on*, pages 1–7, Sept 2014. Busan, Korea.
13. J. Blömer, M. Otto, and J.-P. Seifert. A new CRT-RSA algorithm secure against bellcore attacks. In S. Jajodia, V. Atluri, and T. Jaeger, editors, *ACM Conference on Computer and Communications Security*, pages 311–320. ACM, 2003.
14. J. Blömer, M. Otto, and J.-P. Seifert. Sign Change Fault Attacks on Elliptic Curve Cryptosystems. In L. Breveglieri, I. Koren, D. Naccache, and J.-P. Seifert, editors, *Fault Diagnosis and Tolerance in Cryptography*, volume 4236 of *Lecture Notes in Computer Science*, pages 36–52. Springer Berlin Heidelberg, 2006.
15. D. Boneh, R. A. DeMillo, and R. J. Lipton. On the importance of checking cryptographic protocols for faults (extended abstract). In W. Fumy, editor, *EUROCRYPT*, volume 1233 of *Lecture Notes in Computer Science*, pages 37–51. Springer, 1997.
16. A. Boscher, R. Naciri, and E. Prouff. CRT RSA Algorithm Protected Against Fault Attacks. In D. Sauveron, C. Markantonakis, A. Bilas, and J.-J. Quisquater, editors, *WISTP*, volume 4462 of *Lecture Notes in Computer Science*, pages 229–243. Springer, 2007.
17. M. Ciet and M. Joye. Practical fault countermeasures for chinese remaindering based RSA. In *Fault Diagnosis and Tolerance in Cryptography*, pages 124–131, Friday September 2nd 2005. Edinburgh, Scotland.
18. C. Clavier. Secret External Encodings Do Not Prevent Transient Fault Analysis. In *CHES*, volume 4727 of *Lecture Notes in Computer Science*, pages 181–194. Springer, 2007. Vienna, Austria.
19. E. Dottax, C. Giraud, M. Rivain, and Y. Sierra. On Second-Order Fault Analysis Resistance for CRT-RSA Implementations. In O. Markowitch, A. Bilas, J.-H. Hoepman, C. J. Mitchell, and J.-J. Quisquater, editors, *WISTP*, volume 5746 of *Lecture Notes in Computer Science*, pages 68–83. Springer, 2009.
20. H. M. Edwards. A normal form for elliptic curves. *Bulletin of the American Mathematical Society*, 44:393–422, 2007.
21. N. El Mrabet, J. J. Fournier, L. Goubin, and R. Lashermes. A survey of fault attacks in pairing based cryptography. *Cryptography and Communications*, pages 1–21, 2014.

22. C. Giraud. An RSA Implementation Resistant to Fault Attacks and to Simple Power Analysis. *IEEE Trans. Computers*, 55(9):1116–1120, 2006.
23. A. Guillevic and D. Vergnaud. Genus 2 Hyperelliptic Curve Families with Explicit Jacobian Order Evaluation and Pairing-Friendly Constructions. In M. Abdalla and T. Lange, editors, *Pairing-Based Cryptography — Pairing 2012*, volume 7708 of *Lecture Notes in Computer Science*, pages 234–253. Springer Berlin Heidelberg, 2013.
24. D. Johnson, A. Menezes, and S. Vanstone. The Elliptic Curve Digital Signature Algorithm (ECDSA). *International Journal of Information Security*, 1(1):36–63, 2001.
25. M. Joye. Fault-resistant calculations on elliptic curves, Sept. 15 2010. EP Patent App. EP20,100,155,001 ; http://www.google.com/patents/EP2228716A1?cl=en.
26. M. Joye, P. Paillier, and S.-M. Yen. Secure evaluation of modular functions. In R. Hwang and C. Wu, editors, *International Workshop on Cryptology and Network Security*, pages 227–229, September, 26-28 2001. http://joye.site88.net/papers/JPY01dfa.pdf, Taipei, Taiwan.
27. M. Joye and M. Tunstall, editors. *Fault Analysis in Cryptography*. Information Security and Cryptography. Springer, 2012. ISBN: 978-3-642-29655-0; DOI: 10.1007/978-3-642-29656-7.
28. D. Karaklajic, J. Fan, J. Schmidt, and I. Verbauwhede. Low-cost fault detection method for ECC using Montgomery powering ladder. In *Design, Automation and Test in Europe, DATE 2011, Grenoble, France, March 14-18, 2011*, pages 1016–1021. IEEE, 2011.
29. S.-K. Kim, T. H. Kim, D.-G. Han, and S. Hong. An efficient CRT-RSA algorithm secure against power and fault attacks. *J. Syst. Softw.*, 84:1660–1669, October 2011.
30. P. C. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In M. J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.
31. R. Lashermes, M. Paindavoine, N. El Mrabet, J. J. Fournier, and L. Goubin. Practical Validation of Several Fault Attacks against the Miller Algorithm. In *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2014 Workshop on*, pages 115–122, Sept 2014. Busan, Korea.
32. D.-P. Le, M. Rivain, and C. H. Tan. On double exponentiation for securing RSA against fault analysis. In J. Benaloh, editor, *CT-RSA*, volume 8366 of *Lecture Notes in Computer Science*, pages 152–168. Springer, 2014.
33. V. Leont'ev. Roots of random polynomials over a finite field. *Mathematical Notes*, 80(1-2):300–304, 2006.
34. D. Moody and D. Shumow. Isogenies on edwards and huff curves. *Computer Security Division, National Institute of Standards and Technology (NIST)*, 2011.
35. N. Moro, K. Heydemann, E. Encrenaz, and B. Robisson. Formal verification of a software countermeasure against instruction skip attacks. *J. Cryptographic Engineering*, 4(3):145–156, 2014.
36. M. Naehrig, R. Niederhagen, and P. Schwabe. New software speed records for cryptographic pairings. In M. Abdalla and P. S. Barreto, editors, *Progress in Cryptology – LATINCRYPT 2010*, volume 6212 of *Lecture Notes in Computer Science*, pages 109–123. Springer-Verlag Berlin Heidelberg, 2010. Updated version: http://cryptojedi.org/papers/#dclxvi.
37. A. Shamir. Method and apparatus for protecting public key schemes from timing and fault attacks, November 1999. US Patent Number 5,991,415; also presented at the rump session of EUROCRYPT '97 (May 11–15, 1997, Konstanz, Germany).
38. University of Sydney. Magma Computational Algebra System. http://magma.maths.usyd.edu.au/magma/, Accessed on 2014-08-22.
39. D. Vigilant. RSA with CRT: A New Cost-Effective Solution to Thwart Fault Attacks. In E. Oswald and P. Rohatgi, editors, *CHES*, volume 5154 of *Lecture Notes in Computer Science*, pages 130–145. Springer, 2008.
40. D. Wagner. Cryptanalysis of a provably secure CRT-RSA algorithm. In V. Atluri, B. Pfitzmann, and P. D. McDaniel, editors, *ACM Conference on Computer and Communications Security*, pages 92–97. ACM, 2004.

A Theoretical Upper-Bound for #roots

It is interesting to study the theoretical upper bound on the number of roots in practical cases. Leont'ev proved in [33] that if P is a random polynomial in $\mathbb{F}_p$ then $\#\text{roots}(P) \sim \text{Poisson}(\lambda = 1)$, i.e., $\mathbb{P}(\#\text{roots}(P) = k) = \frac{1}{ek!}$. In the case of $\Delta P \bmod r$, we know that there is always at least one root, when $\widehat{x_1} = x_1$, so we can rewrite $\Delta P(\widehat{x_1}) = P(\widehat{x_1}) - P(x_1) = R(\widehat{x_1}) \cdot a(\widehat{x_1} - x_1)$, where a is some constant, and R is indeed a random polynomial of degree $r - 2$, owing to the modular reduction of ΔP by r. So we know that $\#\text{roots}(\Delta P) = 1 + \#\text{roots}(R)$, hence $\mathbb{P}(\#\text{roots}(\Delta P) = k) = \mathbb{P}(\#\text{roots}(R) = k - 1)$, which is 0 if $k = 0$ and $\frac{1}{e(k-1)!}$ otherwise. We want the maximum value of k which has a "plausible" probability, let us say that is 2^{-p}, e.g., 2^{-256}. Since the values of a Poisson distribution of parameter $\lambda = 1$ are decreasing, we are looking for k such that: $\mathbb{P}(\#\text{roots}(R) = k - 1) = \frac{1}{e(k-1)!} \leq 2^{-256}$. This would suggest that $k \gtrsim 58$.

This result means that $\mathbb{P}_{\text{n.d.}}$ is predicted to be at most $\frac{57}{r}$, with r being at least a 32-bit number, i.e., that $\mathbb{P}_{\text{n.d.}}$ is at maximum $\approx 2^{-26}$, and that this worst-case scenario has a probability of $\approx 2^{-256}$ of happening, in theory.

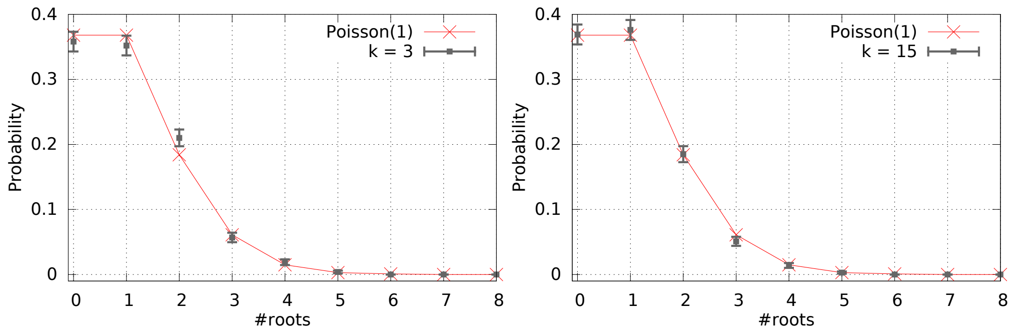

Fig. 2. #roots probability for ECSM $[k]G$.

The figure 2 shows typical number of roots (obtained with SAGE) for practical cases in ECC, and compare them to the theoretical predictions. In this figure, we chose values of k of the form $2^j - 1$, which maximize the number of operations, and thus the size and degree of the resulting ΔP polynomials. For each value of k, we expressed the polynomial ΔP corresponding to the ECSM $[k]G$, and did so for a thousand random G. We then plotted for $i = 0$ to 8 the number of $[k]G$ for which $\#\text{roots}(\Delta P) = i+1$ divided by 1000, that is the estimated probability $\hat{\mathbb{P}}(\#\text{roots}(\Delta P) = i + 1)$. Let us denote by Z the Boolean random variable which is equal to one if ΔP has a $(i + 1)$ roots, and zero otherwise. Our estimation of $\hat{\mathbb{P}}(\#\text{roots}(\Delta P) = i + 1)$ is thus the expectation of $\frac{1}{1000} \sum_{j=1}^{1000} Z_j$. This random variable follows a binomial distribution, of mean $p = \mathbb{P}(\#\text{roots}(\Delta P) = i + 1)$ and variance $p(1 - p)/1000$. The later values are used for the error bars ($[p - \sqrt{p(1 - p)/1000}, p + \sqrt{p(1 - p)/1000}]$).

The two graphs in Fig. 2 correspond to two corner-cases:

1. $k = 3 = (11)_2$: the number of roots is small because the polynomial degree is small (it is 13). (recall that $\#\text{roots}(P)$ cannot exceed the degree of P).
2. $k = 15 = (1111)_2$: the number of roots is also small, but this times because the result of Leont'ev applies. Indeed, the degree is 7213, thus the polynomial is much more random-looking.

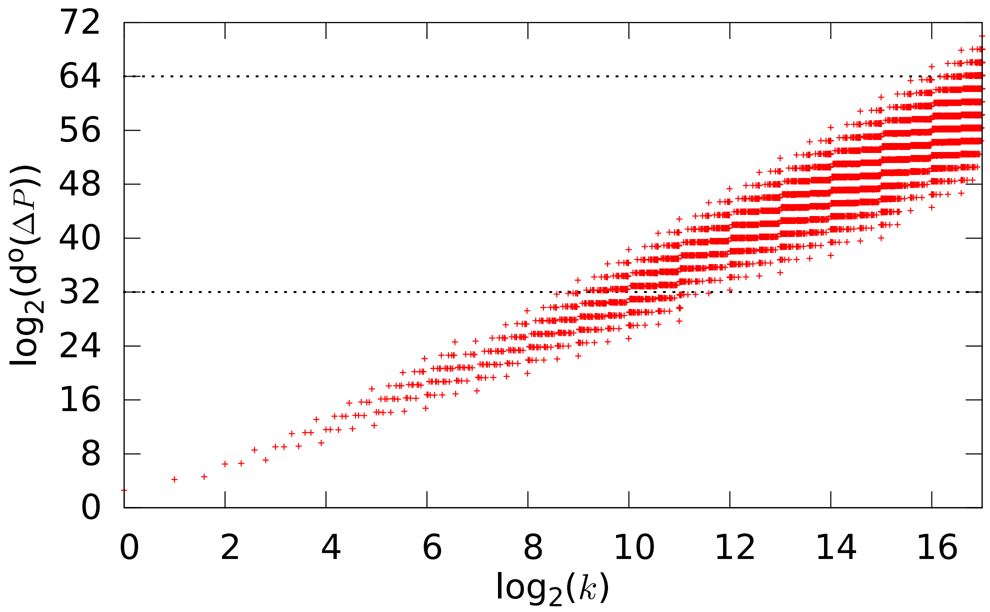

Fig. 3. Degree of the polynomial ΔP against the value of k (in log-log scale).

Actually, it is computationally hard to count the roots of polynomials of degree greater than 7213. But it can be checked that the degree of the polynomials is growing exponentially with k. This is represented in Fig. 3, where we see that the degree is about equal to $k^{3.25}$ (of course, when k has a large Hamming weight, as in $(11\ldots1)_2$, the degree is larger than when k is hollow, as in $(10\ldots0)_2$). In particular, the polynomial ΔP reaches degree 2^{32} (resp. 2^{64}) when k has about 10 (resp. 18) bits. Thus, modulo r (recall Eqn. (1)), the polynomial ΔP has maximal degree as long as the fault is injected before the last 10 (resp. 18) elliptic curve operations when r fits on 32 bits (resp. 64 bits).