Safe-Errors on SPA Protected Implementations with the Atomicity Technique

Pierre-Alain Fouque[1], Sylvain Guilley[2,3], Cédric Murdica[2], and David Naccache[4](✉)

[1] Institut Universitaire de France, Université de Rennes 1, Rennes, France
Pierre-Alain.Fouque@ens.fr
[2] Secure-IC S.A.S., 80 avenue des Buttes de Coësmes, 35700 Rennes, France
{sylvain.guilley,cedric.murdica}@secure-ic.com
[3] Département COMELEC, Institut TELECOM, TELECOM ParisTech, CNRS LTCI, Paris, France
sylvain.guilley@telecom-paristech.fr
[4] Département d'informatique, École normale supérieure, 45, rue d'Ulm, 75230 Paris Cedex 05, France
david.naccache@ens.fr

Abstract. ECDSA is one of the most important public-key signature scheme, however it is vulnerable to lattice attack once a few bits of the nonces are leaked. To protect Elliptic Curve Cryptography (ECC) against Simple Power Analysis, many countermeasures have been proposed. Doubling and Additions of points on the given elliptic curve require several additions and multiplications in the base field and this number is not the same for the two operations. The idea of the atomicity protection is to use a fixed pattern, *i.e.* a small number of instructions and rewrite the two basic operations of ECC using this pattern. Dummy operations are introduced so that the different elliptic curve operations might be written with the same atomic pattern. In an adversary point of view, the attacker only sees a succession of patterns and is no longer able to distinguish which one corresponds to addition and doubling. Chevallier-Mames, Ciet and Joye were the first to introduce such countermeasure. In this paper, we are interested in studying this countermeasure and we show a new vulnerability since the ECDSA implementation succumbs now to C Safe-Error attacks. Then, we propose an effective solution to prevent against C Safe-Error attacks when using the Side-Channel Atomicity. The dummy operations are used in such a way that if a fault is introduced on one of them, it can be detected. Finally, our countermeasure method is generic, meaning that it can be adapted to all formulæ. We apply our methods to different formulæ presented for Side-Channel Atomicity.

Keywords: Elliptic Curve Cryptography · Side-channel atomicity · Fault attacks · Infective countermeasure · Lattice attack

P.Y.A. Ryan et al. (Eds.): Kahn Festschrift, LNCS 9100, pp. 479–493, 2016.
DOI: 10.1007/978-3-662-49301-4_30

1 Introduction

As well as most of cryptosystems, Elliptic Curve Cryptography (ECC) is vulnerable to side-channel attacks. One of the first reported attack on ECC was the Simple Side-Channel Analysis (SSCA) [6]. It consists in analyzing a single trace of the execution of the Elliptic Curve Scalar Multiplication and attempts to distinguish the power consumption between a doubling and an addition of elliptic curve points.

Numerous countermeasures exist against the SSCA. The side-channel Atomicity is one of them and was proposed by Chevallier-Mames, Ciet, Joye in 2004 [4]. It consists in writing the different elliptic curve operations, such as doubling and addition, with identical block of field operations, which makes SSCA infeasible. Inspired from this paper [4], different formulæ that are more efficient, or more suitable for particular scalar multiplications, have been proposed [7,10,15]. Up to now, all these formulæ contain at least one dummy operation.

One of the most popular elliptic curve cryptographic scheme is the signature scheme ECDSA and it is well-known that this scheme is sensible to lattice attacks once some information on the most significant bits of the nonces k are known. Many attacks have been proposed since [3,8,12,13].

It is possible to use C Safe-Errors on the dummy operations added purportedly for the atomicity formulae as Yen *et al.* proposed against the CRT-RSA implementation in [18]. The attacker introduces a fault during a possibly dummy field operation. If the result is still correct, the operation was indeed dummy and the elliptic curve operation can be deduced. As a consequence, the current target bit of the secret scalar can be learned. However, such way of attacking discloses only a small number of bits of the nonce per ECSM if we allow multiple faults. Liu and Nguyen at CT-RSA 2013 in [9] show that it is possible to recover the secret key on DSA as soon as we have at least 2 bits of the nonces for 160-bit modulus. This lower bound has been proven in [14]. The number of bits increases with the size of the modulus and for 192-bit and 256-bit moduli we do not know how many bits are required. Thus, C safe errors must be improved, otherwise not enough information is collected to extract the secret key. Another alternative to lattice-based attacks consists in using Bleichenbacher attack that has been recently proposed by De Mulder *et al.* at CHES 2013 [11]. This attack allows in theory to recover the secret key as soon as a few bits of the nonces is known and according to the modulus size, it could be preferable to use this attack in comparison with lattice attacks. The main drawback of this attack is that if we want to use a very small number of bits, then the number of needed signature becomes quite large. For instance, in order to attack ECDSA on 160-bit finite field knowing only one bit of the nonce, the number of signatures is about 2^{33}. We use an interesting idea introduced in [1] to reduce the number of faulty signatures to 2^{26} if one bit is known for 160-bit moduli and to 2^{19} if two bits are known and in this case we can attack 160-bit and 192-bit moduli by increasing the time and memory complexity. When more bits are available, it is not easy to tell which one of lattice attacks and Bleichenbacher attacks is the most efficient as shown in [11] since lattice attack can also be used to makes Bleichenbacher attack more efficient.

In this paper, we also present a countermeasure against this attack for the atomicity implementations. The formulæ are rewritten such that the dummy operations no longer occur. We define some processes such that every fault induced will inevitably be detected.

The rest of the paper is organized as follows. In Sect. 2, we recall background on ECC, side-channel attacks, and the side-channel atomicity countermeasures. The attacks on protected implementations are given in Sect. 3. The classical C safe-errors when the exponent is static and our new attack when the exponent is ephemeral using previous algorithms [1,11]. Section 4 presents our proposed solution that can be applied to any formulæ. Finally, we conclude in Sect. 5.

2 Background

In this section, we present the required background to understand the attack on the Side-Channel Atomicity and the protection that we suggest.

2.1 Elliptic Curve Cryptography

An elliptic curve over a finite prime field $\mathbb{F}_p$ of characteristic $p > 3$ can be described by its reduced Weierstraß form:

$$E\colon y^2 = x^3 + ax + b. \tag{1}$$

We denote by $E(\mathbb{F}_p)$ the set of points $(x, y) \in \mathbb{F}_p^2$ satisfying Eq. (1), plus the point at infinity $\mathcal{O}$.

The points on $E(\mathbb{F}_p)$ define an additive Abelian group given by the following addition law. Let $P = (x_1, y_1) \neq \mathcal{O}$ and $Q = (x_2, y_2) \notin \{\mathcal{O}, -P\}$ be two points on $E(\mathbb{F}_p)$. Point addition $R = (x_3, y_3) = P + Q$ is defined by the formula:

$$\begin{aligned} x_3 &= \lambda^2 - x_1 - x_2 \\ y_3 &= \lambda(x_1 - x_3) - y_1 \end{aligned} \qquad \text{where } \lambda = \begin{cases} \frac{y_1 - y_2}{x_1 - x_2} & \text{if } P \neq Q, \\ \frac{3x_1^2 + a}{2y_1} & \text{if } P = Q. \end{cases}$$

The inverse of point P is defined as $-P = (x_1, -y_1)$.

To avoid modular inversions, implementers frequently work in the Jacobian projective coordinates system. The equation of an elliptic curve in the Jacobian projective coordinates system in the reduced Weierstraß form is:

$$E^{\mathcal{J}}: Y^2 = X^3 + aXZ^4 + bZ^6.$$

The projective point (X, Y, Z) corresponds to the affine point $(X/Z^2, Y/Z^3)$. The point (X, Y, Z) is equivalent to any point (r^2X, r^3Y, rZ) with $r \in \mathbb{F}_p^*$.

Let $P_1 = (X_1, Y_1, Z_1), P_2 = (X_2, Y_2, Z_2)$ be two points on $E^{\mathcal{J}}(\mathbb{F}_p)$ with $P_1 \neq \mathcal{O}, ord(P_1) > 2$ and $P_2 \notin \{\mathcal{O}, -P_1\}$. Point doubling and points addition are defined by the following formulæ:

- **ECDBL**. $P_3 = (X_3, Y_3, Z_3) = 2P_1$ can be computed as:
  $X_3 = T,\ Y_3 = -8Y_1^4 + M(S - T),\ Z_3 = 2Y_1Z_1$, where
  $S = 4X_1Y_1^2,\ M = 3X_1^2 + aZ_1^4,\ T = -2S + M^2$
- **ECADD**. $P_3 = (X_3, Y_3, Z_3) = P_1 + P_2$ can be computed as:
  $X_3 = -H^3 - 2U_1H^2 + R^2,\ Y_3 = -S_1H^3 + R(U_1H^2 - X_3),\ Z_3 = Z_1Z_2H$, where
  $U_1 = X_1Z_2^2,\ U_2 = X_2Z_1^2,\ S_1 = Y_1Z_2^3,\ S_2 = Y_2Z_1^3,\ H = U_2 - U_1,\ R = S_2 - S_1$

2.2 Elliptic Curve Digital Signature Algorithm

The Elliptic Curve Digital Signature Algorithm (ECDSA) is a signature scheme. It has been standardized in [17]. Given the following curve parameters:

- E, an elliptic curve over a prime field $\mathbb{F}_p$,
- G, a generator of a subgroup of E of order t,

the signature process is as follows:

Algorithm 1. ECDSA Signature

```
Input: private key $d$, an encoded integer $m \in \{0, p-1\}$ representing a message
Output: Signature $(r, s)$
 1: $k \overset{\mathcal{R}}{\longleftarrow} \{1, \dots, t-1\}$
 2: $Q \leftarrow [k]G$
 3: $r \leftarrow x_Q \mod t$
 4: if $r = 0$ then
 5:     go to line 1
 6: end if
 7: $s \leftarrow k^{-1}(dr + m) \mod t$
 8: if $s = 0$ then
 9:     go to line 1
10: end if
11: return $(r, s)$
```

2.3 Side-Channel Atomicity

In ECC, one has to compute scalar multiplications, *i.e.* compute $[k]P$, given P and an integer k. The Left-to-Right Double-and-Add and Right-to-Left algorithms (Algorithms 2 and 3) are ways of doing so.

Algorithm 2. Left-to-Right Double-and-Add

```
Input: a point $P$ and an integer $k = (1, k_{n-2}, \dots, k_0)_2$
Output: $[k]P$
  $R_0 \leftarrow P$
  for $i = n - 2$ downto $0$ do
      $R_0 \leftarrow 2R_0$                                  $\triangleright R_0 = [(k_{n-1}, \dots, k_{i+1}, 0)_2]P$
      if $k_i = 1$ then $R_0 \leftarrow R_0 + P$             $\triangleright R_0 = [(k_{n-1}, \dots, k_{i+1}, k_i)_2]P$
  end for
  return $R_0$
```

Algorithm 3. Right-to-Left Double-and-Add

```
Input: $k = (k_{n-1}, \dots, k_1, 1)_2, P$
Output: $[k]P$
  $R_0 \leftarrow P$
  $R_1 \leftarrow 2P$
  for $i = 1$ to $n - 1$ do
      if $k_i = 1$ then $R_0 \leftarrow R_0 + R_1$           $\triangleright R_0 = [(k_i, \dots, k_0)_2]P$
      $R_1 \leftarrow 2R_1$                                  $\triangleright R_1 = [2^{i+1}]P$
  end for
  return $R_0$
```

Both algorithms exist when the scalar is given by its Non-Adjacent Form (NAF) representation. They are given in Appendix A.

If an adversary is able to distinguish the power consumption of an addition and a doubling during the execution of such algorithm, then she is able to recover the secret scalar k [6]. In order to prevent this attack called the Simple-Power Analysis, Chevallier-Mames, Ciet and Joye suggest to write the elliptic curve formulæ with sequences of identical *atomic patterns*. An atomic pattern is defined in [4] as the sequence of the following (possibly dummy) operations:

1. modular multiplication or square
2. modular addition
3. modular opposite
4. modular addition

A point doubling requires 10 of these atomic patterns, while an addition requires 16 in the Jacobian coordinates systems. It has been later improved several times by Longa in [10], Giraud and Verneuil in [7] and Rondepierre in [15]. Hereafter, we recall Giraud and Verneuil's pattern, the state-of-the-art best atomic pattern when applied with the Right-to-Left Double-and-Add, and Rondepierre's pattern, the state-of-the-art best atomic pattern when applied with the Left-to-Right Double-and-Add.

2.4 Giraud and Verneuil's pattern [7]

Giraud and Verneuil suggest a pattern composed of two squares, six multiplications, six additions and four subtractions. An addition of points requires two patterns while a doubling requires only one. The points are given in modified Jacobian coordinates: $P = (X_1, Y_1, Z_1, W_1 = aZ_1^4)$, for faster doubling [5]. These coordinates are suitable for the Right-to-Left Double-and-Add algorithms (Algorithms 3 and 5). We recall the formulæ in Fig. 1. From $P = (X_1, Y_1, Z_1)$ and $Q = (X_2, Y_2, Z_2)$, one can compute $P + Q = (X_3, Y_3, Z_3)$ and $2P = (X_3, Y_3, Z_3, W_3 = aZ_3^4)$.

1.	$T_1 \leftarrow Z_2^2$	$T_1 \leftarrow T_6^2$	$T_1 \leftarrow X_1^2$
2.	$\star \leftarrow \star + \star$	$\star \leftarrow \star + \star$	$T_2 \leftarrow Y_1 + Y_1$
3.	$T_2 \leftarrow Y_1 \times Z_2$	$T_4 \leftarrow T_5 \times T_1$	$Z_3 \leftarrow T_2 \times Z_1$
4.	$\star \leftarrow \star + \star$	$\star \leftarrow \star + \star$	$T_4 \leftarrow T_1 + T_1$
5.	$T_5 \leftarrow Y_2 \times Z_1$	$T_5 \leftarrow T_1 \times T_6$	$T_3 \leftarrow T_2 \times Y_1$
6.	$\star \leftarrow \star + \star$	$\star \leftarrow \star + \star$	$T_6 \leftarrow T_3 + T_3$
7.	$T_3 \leftarrow T_1 \times T_2$	$T_1 \leftarrow Z_1 \times T_6$	$T_2 \leftarrow T_6 \times T_3$
8.	$\star \leftarrow \star + \star$	$\star \leftarrow \star + \star$	$T_1 \leftarrow T_4 + T_1$
9.	$\star \leftarrow \star + \star$	$\star \leftarrow \star + \star$	$T_1 \leftarrow T_1 + W_1$
10.	$T_4 \leftarrow Z_1^2$	$T_6 \leftarrow T_2^2$	$T_3 \leftarrow T_1^2$
11.	$T_5 \leftarrow T_5 \times T_4$	$Z_3 \leftarrow T_1 \times Z_2$	$T_4 \leftarrow T_6 \times X_1$
12.	$\star \leftarrow \star + \star$	$T_1 \leftarrow T_4 + T_4$	$T_5 \leftarrow W_1 + W_1$
13.	$T_2 \leftarrow T_2 - T_3$	$T_6 \leftarrow T_6 - T_1$	$T_3 \leftarrow T_3 - T_4$
14.	$T_5 \leftarrow T_1 \times X_1$	$T_1 \leftarrow T_5 \times T_3$	$W_3 \leftarrow T_2 \times T_5$
15.	$\star \leftarrow \star - \star$	$X_3 \leftarrow T_6 - T_5$	$X_3 \leftarrow T_3 - T_4$
16.	$\star \leftarrow \star - \star$	$T_4 \leftarrow T_4 - X_3$	$T_6 \leftarrow T_4 - X_3$
17.	$T_6 \leftarrow X_2 \times T_4$	$T_3 \leftarrow T_4 \times T_2$	$T_4 \leftarrow T_6 \times T_1$
18.	$T_6 \leftarrow T_6 - T_5$	$Y_3 \leftarrow T_3 - T_1$	$Y_3 \leftarrow T_4 - T_2$

Fig. 1. Addition and doubling operations written with Giraud and Verneuil's pattern ($\star$ represents a dummy operand). Each column is an atomic pattern.

2.5 Rondepierre's pattern [15]

Rondepierre suggests a pattern composed of two squares, eight multiplications, five additions and five subtractions. An addition of points requires one pattern, as well as a doubling. From $P = (X_1, Y_1, Z_1, Z_1^2, Z_1^3), Q = (X_2, Y_2, 1)$ and $I = \sqrt{-a3^{-1}}$, Rondepierre proposes formulæ to compute $P + Q = (X_3, Y_3, Z_3, Z_3^2, Z_3^3), P - Q = (X_3, Y_3, Z_3, Z_3^2, Z_3^3)$ or $2P = (X_3, Y_3, Z_3, Z_3^2, Z_3^3)$. The subtraction of points is suitable for the Right-to-Left method (Algorithms 3 and 5). The formulæ are suitable for the Right-to-Left Double-and-Add algorithms (Algorithms 3 and 5). They are given in Fig. 2.

1.	$T_1 \leftarrow X_2 \times Z_1^2$	$T_1 \leftarrow X_2 \times Z_1^2$	$T_0 \leftarrow I \times Z_1^2$
2.	$T_1 \leftarrow T_1 - X_1$	$T_1 \leftarrow T_1 - X_1$	$T_1 \leftarrow X_1 - T_0$
3.	$\star \leftarrow \star + \star$	$Z_1^2 \leftarrow Y_1 + Y_1$	$T_2 \leftarrow Y_1 + Y_1$
4.	$T_2 \leftarrow T_1 \times T_1$	$T_2 \leftarrow T_1 \times T_1$	$Z_3^2 \leftarrow Y_1 \times T_2$
5.	$\star \leftarrow \star + \star$	$\star \leftarrow \star + \star$	$Y_3 \leftarrow Z_3^2 + Z_3^2$
6.	$T_3 \leftarrow X_1 \times T_2$	$T_3 \leftarrow X_1 \times T_2$	$T_3 \leftarrow T_2 \times Z_1$
7.	$T_0 \leftarrow Y_2 \times Z_1^3$	$T_0 \leftarrow Y_2 \times Z_1^3$	$T_2 \leftarrow Y_3 \times X_1$
8.	$\star \leftarrow \star + \star$	$T_0 \leftarrow Z_1^2 + T_0$	$X_3 \leftarrow X_1 + T_0$
9.	$Z_1^3 \leftarrow T_1 \times T_2$	$Z_1^3 \leftarrow T_1 \times T_2$	$T_0 \leftarrow T_1 \times X_3$
10.	$T_2 \leftarrow Z_1 \times T_1$	$T_2 \leftarrow Z_1 \times T_1$	$T_1 \leftarrow Z_3^2 \times Y_3$
11.	$X_3 \leftarrow T_3 + T_3$	$X_3 \leftarrow T_3 + T_3$	$T_2 \leftarrow T_0 + T_0$
12.	$X_3 \leftarrow Z_1^3 + X_3$	$X_3 \leftarrow Z_1^3 + X_3$	$T_0 \leftarrow T_0 + T_2$
13.	$Z_3^2 \leftarrow (T_0)^2$	$Z_3^2 \leftarrow (T_0)^2$	$X_3 \leftarrow (T_0)^2$
14.	$T_0 \leftarrow T_0 - Y_1$	$T_0 \leftarrow T_0 - Y_1$	$X_3 \leftarrow X_3 - T_2$
15.	$T_1 \leftarrow (T_0)^2$	$T_1 \leftarrow (T_0)^2$	$Z_3^2 \leftarrow (T_3)^2$
16.	$X_3 \leftarrow T_1 - X_3$	$X_3 \leftarrow T_1 - X_3$	$X_3 \leftarrow X_3 - T_2$
17.	$T_1 \leftarrow T_3 - X_3$	$T_1 \leftarrow T_3 - X_3$	$T_2 \leftarrow T_2 - X_3$
18.	$T_3 \leftarrow T_1 \times T_0$	$T_3 \leftarrow T_1 \times T_0$	$Z_3^2 \leftarrow Z_3^2 \times T_3$
19.	$T_0 \leftarrow Y_1 \times Z_1^3$	$T_0 \leftarrow Y_1 \times Z_1^3$	$Y_3 \leftarrow T_0 \times T_2$
20.	$Y_3 \leftarrow T_3 - T_0$	$Y_3 \leftarrow T_3 - T_0$	$Y_3 \leftarrow Y_3 - T_1$
21.	$Z_3 \leftarrow T_2$	$Z_3 \leftarrow T_2$	$Z_3 \leftarrow T_3$

Fig. 2. Addition, subtraction and doubling operations written with Rondepierre's pattern ($\star$ represents a dummy operand). Each column is an atomic pattern.

3 Attacks on Side-Channel Atomicity

3.1 C Safe-Error

The C Safe-Error attack was first published by Yen, Kim, Lim and Moon [18]. They target an RSA implementation which contains dummy operations to prevent the SPA. A fault is introduced during an operation which is possibly a dummy one. If the result of the cryptographic operation is correct, the operation was indeed a dummy operation and some information on the private key can be deduced.

C Safe-Error on the side-channel atomicity countermeasure for ECC relies on the same principle.

C Safe-Error on Giraud and Verneuil's Pattern. Suppose that the Right-to-Left Double-and-Add (Algorithm 3) is used, with the patterns of Fig. 1. Regarding the Right-to-Left Double-and-Add, the last pattern is necessarily a doubling. However, regarding the trace during the execution of the penultimate pattern, the attacker cannot deduce that it is a doubling or the second part of an addition.

Suppose that the attacker injects a fault on the arithmetic module unit during the execution of the first addition of the penultimate pattern (line 6 of Fig. 1).

If the pattern is indeed the second part of an addition, the error has no effect on the result. The fault is *safe*. In this case, the most significant bit of the scalar is 1.

On the other hand, if the result is incorrect, the pattern was a doubling and the most significant bit is 0.

The attacker can repetitively perform this attack during several ECDSA signature generations. She can collect several signatures and keep only the correct ones (the ones where the error was safe). She then got several signatures knowing that the most significant bit of the ephemeral scalar is 1.

C Safe-Error on Rondepierre's Pattern. The attack on this pattern is analogous to the previous one.

Suppose that the Left-to-Right Double-and-Add (Algorithm 2) is used, with the patterns of Fig. 2. Regarding the trace during the execution of the last pattern, the attacker cannot deduce that it is a doubling an addition.

Suppose that the attacker injects a fault on the arithmetic module unit during the execution of the first subtraction of the last pattern (line 3 of Fig. 2). If the pattern is indeed an addition, the error has no effect on the result. The fault is *safe*. In this case, the least significant bit of the scalar is 1.

On the other hand, if the result is incorrect, the pattern was a doubling and the least significant bit is 0.

The attacker can repetitively performs this attack during several ECDSA signature generations. She can collect several signatures and keep only the correct ones (the ones where the error was safe). Hence she has got several signatures such that the least significant bit is 1.

Extension to Several Bits. Of course, the attacker can inject several faults at different times during the algorithm.

For Giraud and Verneuil's patterns, two patterns are required for the addition. The attacker can inject one fault on the penultimate pattern and one fault on the fifth last pattern. If the result is correct, it means that the last patterns are $\mathcal{A}1; \mathcal{A}2; \mathcal{D}; \mathcal{A}1; \mathcal{A}2; \mathcal{D}$, thus the two most significant bits are 1.

For Rondepierre's patterns, the attacker can inject a fault on the last pattern and on the third last pattern. If the result is correct, it means that the last patterns are $\mathcal{A}; \mathcal{D}; \mathcal{A}$, thus the two least significant bits are 1.

Injecting the Fault at the Right Time. We describe here the issue of injecting the fault at the right time. As a matter of fact, we said before that the attacker needs to inject a fault on the last or penultimate pattern. How does she know that this is the last or penultimate pattern before the end of the ECSM? Indeed, a fault cannot be injected retrospectively, i.e., after noticing that the ECSM is finished.

In fact, she can suppose that the Hamming weight of the n-bit scalar is $n/2$ which happens with high probability. In this case, there will be n doubling and

$n/2$ additions. This gives a total of $2n$ Giraud and Verneuil's pattern (because two patterns are required for the addition) and $n + n/2$ Rondepierre's pattern. The last pattern is thus the $2n^{th}$ pattern (Giraud and Verneuil) and the $(n + n/2)^{th}$ pattern (Rondepierre).

The attacker can verify afterwards that the Hamming weight of the scalar was is indeed $n/2$ counting the patterns by SPA. If it is not the case, she throws out the signature[1].

3.2 Lattice Attacks Knowing only Two Bits per Value of the Ephemeral Nonces

The attack works as follows: in a first step, a small number of bits ℓ (e.g., $\ell = 1, 2, 3, 4, 5,$ or 6) is gathered about the nonce k used in ECDSA. Namely, one bit is tested through the effectiveness (or not) of an injection at a given field operation in one ECDBL or ECADD atomic pattern. Then, a lattice attack is launched using only these ℓ bits of information about the ephemeral nonce per ECSM.

There are basically two different strategies to recover the secret key d. The first one consists in solving the Hidden Number Problem (HNP), which can be described as follows: given (t_i, u_i) pairs of integers such that

$$|dt_i - u_i|_q \leq q/2^{\ell+1},$$

where ℓ denotes the number of bits we recovered by C Safe-Errors, d denotes the hidden number we are looking for and $|\cdot|_q$ denotes the distance to $q\mathbb{Z}$, *i.e.* $|z|_q = \min_{a\in\mathbb{Z}} |z - aq|$. Such problem can be cast as a Closest Vector Problem (CVP) in a lattice and the LLL algorithm can be used to solve it in practice very efficiently. We recall the basic attack in Appendix 5 and its extensive presentation can be found in [14]. The main advantage of this technique is that the number of signatures required is usually very small, but it cannot be used all the time when the number of bits becomes very small. Indeed, in this case for 160-bit modulus for instance, Liu and Nguyen used BKZ 2.0 to solve such lattice and the dimension becomes very high for lattice algorithms [9].

When the number of bits is very small, which is the case here if we try to reduce the number of faults, another technique due to Bleichenbacher can be used. This technique has been described in [11] for attacking a smartcard using ECDSA on 384-bit modulus. The idea is that there is a bias on distribution of the nonces k_j. If we correctly guess the value of the secret d is large and all other biases are small (close to 0) according to the correct definition of bias $B_q(D) = E(\exp^{2i\pi D/q})$ where E is the expectation of the random variable $\exp^{2i\pi D/q}$ and D is the random variable representing the choice of d. We can approximate this bias experimentally using many signatures by computing $B_q(d) = (1/m) \cdot \sum_{j=0}^{m-1} \exp^{2i\pi(h_j + c_j d)/q}$ where $h_j = H(m_j)/s_j \mod q$ and $c_j = r_j/s_j \mod q$ for

[1] Notice that the atomicity countermeasure does not execute in constant time. However, the only information that is leaked is the Hamming weight of the scalar, which is not enough to design an attack (at least with state-of-the-art knowledge).

signature (r_j, s_j) of message m_j and m the number of such signatures. The idea is just to compute all the bias $B_q(d)$ for all possible values of d and pick the largest one. Due to the special form of the bias, it is possible to perform all these computations using Fast Fourier Transform, however the time complexity of this task is out of reach since there are 2^{160} different values for d. Bleichenbacher proposes a first phase which consists in reducing the range of the value d (we are looking for the, say 32 most significant bits of d, by reducing the bias of d. This operation will also widen the width of the pick of the bias d in the frequence domain. In the first stage of this attack, we are looking for a linear combination of the values c_j which is small, less than 32 bits. In this case, it has been shown in [11] that we can recover the 32 most significant bits of d. However, the number of required signatures becomes very high and De Mulder *et al.* use a lattice reduction technique to reduce the number of signature contrary to Bleichenbacher original attack which uses more Generalized Birthday Paradox (GBP) ideas [16]. For instance, given (h_j, c_j) such that $h_j + dc_j = k_j$, if c_j and $c_{j'}$ have 32 bits in common, then $h_j - h_{j'} + d(c_j - c_{j'}) = k_j - k_{j'}$ is a new relation where the new value $(c_j - c_{j'})$ has been reduced by 32 bits and since we add the k_js, the initial bias b is increased to b^2 according to the Piling-up lemma. In [1], the authors show that it is possible to recover a 160-bit secret value with only *one bit* of the nonces. However, the number of required signatures grows up to 2^{33}. They also show that it is possible to reduce the number of signatures required in Bleichenbacher algorithm by using time-memory/signature tradeoff.

The idea is that the first iteration will allow us to make many signature samples (h_j, c_j) by increasing the bias. For instance, given m signatures, we can generate m^2 samples by performing addition and subtraction $\bmod q$ of the initial signatures.

In Fig. 3, we give the minimal number m of signatures required for number of known bits ℓ of the nonce.

q	160 bits			192 bits	256 bits
ℓ	1	2	2	2	3
m	2^{26}	2^{14}	200	2^{16}	2^{16}
Tech.	Bleich.	Bleich.	Latt.	Bleich.	Bleich.
Compl.	2^{40}	2^{28}	Few hr	2^{33}	2^{33}

Fig. 3. Minimal number of signatures d required depending on the number of bits ℓ using Brainpool curves.

4 Our Protection

We propose in this section our protection. It consists in using the dummy operations to perform a check at the end of the ECSM.

4.1 Generalized Protection

In the patterns of all known atomic side-channel protections, the dummy operations are either field additions or field subtractions and are only on patterns of the addition and subtraction of points. The underlying reason is that those operations are more furtive than multiplications. Thus, it is unlikely that an attacker manages to distinguish between dummy and functional operations in the patterns.

Our idea is to perform a check at the end of the ECSM such that if an error occurred, the circuit detects it and no result is returned.

Let addition and doubling formulæ using some patterns such that an addition of points contains l dummy field additions and m dummy subtractions. This means that, at the end of the ECSM, there are (l times the number of additions of points) dummy field additions and (m times the number of additions of points) dummy field subtractions.

We propose to add two temporary registers T_{add} and T_{sub} first initialized with $T_{add} \leftarrow r_{add}$, $T_{sub} \leftarrow -r_{sub}$; r_{add}, r_{sub} being two random integers. Every dummy addition $\star \leftarrow \star + \star$ is replaced by $T_{add} \leftarrow T_{add} + r_{add}$ and every subtraction $\star \leftarrow \star - \star$ is replaced by $T_{sub} \leftarrow T_{sub} - r_{sub}$. In this way, at the end of the ECSM, T_{add} should be equal to $l \times r_{add}$ times the number of additions performed during the ECSM and T_{sub} should be equal to $m \times r_{sub}$ times the number of additions.

A counter is added for each pattern to count the number of additions and doubling performed. Another method is that the number of patterns is related to the Hamming weight (HW) of the scalar used.

The protection consists in verifying that the equality is satisfied at the end of the ECSM.

4.2 The Protection with Giraud and Verneuil's Pattern

With those formulæ, there are 11 dummy additions and 2 dummy subtractions for the addition of points. The number of addition of points is HW(k) for the Right-to-Left Double-and-Add algorithm (Algorithms 3), k being the scalar.

Thus the protection consists in verifying that T_{add} is equal to $11 \times \mathrm{HW}(k) \times r_{add}$ and T_{sub} is equal to $2 \times \mathrm{HW}(k) \times r_{sub}$ at the end of the ECSM.

4.3 The Protection with Rondepierre's Pattern

With those formulæ, there are 3 dummy additions for the addition of points. The number of addition of points is HW(k) for the Left-to-Right Double-and-Add algorithm (Algorithm 2), k being the scalar.

Thus the protection consists in verifying that T_{add} is equal to $3 \times \mathrm{HW}(k) \times r_{add}$.

5 Conclusion

In this paper, we show how to use C Safe-Errors on the atomicity side-channel countermeasure to recover a few bits of ephemeral scalars used during ECDSA signatures. With only two bits of the scalar, we are able to recover the secret key.

Then, we propose a protection to thwart C Safe-Errors that target the atomicity countermeasure. The method consists in replacing the dummy operations of the atomic patterns by chained secret operations that are verified in a final check. In this case, the C Safe-error is no longer applicable.

References

1. Aranha, D.F., Fouque, P.-A., Gérard, B., Kammerer, J.-G., Tibouchi, M., Zapalowicz, J.-C.: GLV/GLS decomposition, power analysis, and attacks on ECDSA signatures with single-bit nonce bias. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8873, pp. 262–281. Springer, Heidelberg (2014)
2. Boneh, D., Venkatesan, R.: Hardness of computing the most significant bits of secret keys in Diffie-Hellman and related schemes. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 129–142. Springer, Heidelberg (1996)
3. Brumley, B.B., Tuveri, N.: Remote timing attacks are still practical. In: Atluri, V., Diaz, C. (eds.) ESORICS 2011. LNCS, vol. 6879, pp. 355–371. Springer, Heidelberg (2011)
4. Chevallier-Mames, B., Ciet, M., Joye, M.: Low-cost solutions for preventing simple side-channel analysis: side-channel atomicity. IEEE Trans. Comput. **53**(6), 760–768 (2004)
5. Cohen, H., Miyaji, A., Ono, T.: Efficient elliptic curve exponentiation using mixed coordinates. In: Ohta, K., Pei, D. (eds.) ASIACRYPT 1998. LNCS, vol. 1514, pp. 51–65. Springer, Heidelberg (1998)
6. Coron, J.-S.: Resistance against differential power analysis for elliptic curve cryptosystems. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 292–302. Springer, Heidelberg (1999)
7. Giraud, C., Verneuil, V.: Atomicity improvement for elliptic curve scalar multiplication. In: Gollmann, D., Lanet, J.-L., Iguchi-Cartigny, J. (eds.) CARDIS 2010. LNCS, vol. 6035, pp. 80–101. Springer, Heidelberg (2010)
8. Howgrave-Graham, N., Smart, N.P.: Lattice attacks on digital signature schemes. Des. Codes Cryptogr. **23**(3), 283–290 (2001)
9. Liu, M., Nguyen, P.Q.: Solving BDD by enumeration: an update. In: Dawson, E. (ed.) CT-RSA 2013. LNCS, vol. 7779, pp. 293–309. Springer, Heidelberg (2013)
10. Longa, P.: Accelerating the scalar multiplication on elliptic curve cryptosystems over prime fields. Master's thesis, School of Information and Engineering, University of Ottawa, Canada (2007)
11. De Mulder, E., Hutter, M., Marson, M.E., Pearson, P.: Using Bleichenbacher's solution to the hidden number problem to attack nonce leaks in 384-bit ECDSA: extended version. J. Cryptogr. Eng. **4**(1), 33–45 (2014)
12. Naccache, D., Nguyên, P.Q., Tunstall, M., Whelan, C.: Experimenting with faults, lattices and the DSA. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 16–28. Springer, Heidelberg (2005)

13. Nguyen, P.Q., Shparlinski, I.: The insecurity of the elliptic curve digital signature algorithm with partially known nonces. Des. Codes Cryptogr. **30**(2), 201–217 (2003)
14. Nguyen, P.Q., Tibouchi, M.: Lattice-based fault attacks on signatures. In: Joye, M., Tunstall, M. (eds.) Fault Analysis in Cryptography. Information Security and Cryptography, pp. 201–220. Springer, Heidelberg (2012)
15. Rondepierre, F.: Revisiting atomic patterns for scalar multiplications on elliptic curves. In: Francillon, A., Rohatgi, P. (eds.) CARDIS 2013. LNCS, vol. 8419, pp. 171–186. Springer, Heidelberg (2014)
16. Wagner, D.: A generalized birthday problem. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 288–303. Springer, Heidelberg (2002)
17. ANSI X9.62. Public key cryptography for the financial services industry: the elliptic curve digital signature algorithm (ECDSA). ANSI, USA (1998)
18. Yen, S.-M., Kim, S., Lim, S., Moon, S.-J.: A countermeasure against one physical cryptanalysis may benefit another attack. In: Kim, K. (ed.) ICISC 2001. LNCS, vol. 2288, pp. 414–427. Springer, Heidelberg (2002)

Reminder About the Lattice-Based Attack on ECDSA

Using the ℓ least significant bits of k (the attack also works with the most significant bits), we can write $k = 2^\ell(k \gg \ell) + \mathrm{lsb}_\ell\, k = 2^\ell b + \mathrm{lsb}_\ell\, k$ for some integer $b \geq 0$. We then get from $dr = sk - h \mod q$:

$$dr \cdot 2^{-\ell} s^{-1} = b - h \cdot 2^{-\ell} s^{-1} + \mathrm{lsb}_\ell\, k \cdot 2^{-\ell} \mod q.$$

Now let t and u two values which can be computed from known or retrieved information, such as:

$$t = r \cdot 2^{-\ell} s^{-1} \mod q, \ u = -h \cdot 2^{-\ell} s^{-1} + \mathrm{lsb}_\ell\, k \cdot 2^{-\ell} \mod q.$$

The inequality $b < q/2^\ell$ can be expressed in terms of t and u as:

$$0 \leq dt - u \mod q < q/2^\ell.$$

Therefore, if we denote by $|\cdot|_q$ the distance to $\mathbb{Z}/q\mathbb{Z}$, i.e. $|z|_q = \min_{a\in\mathbb{Z}} |z - aq|$, we have:

$$\begin{aligned} |dt - u - q/2^{\ell+1}|_q &\leq q/2^{\ell+1}, \\ |dt - v/2^{\ell+1}|_q &\leq q/2^{\ell+1}, \end{aligned}$$

where v is the integer $2^{\ell+1}u + q$. Given a number of faulty signatures (r_i, s_i) of various messages, say m of them, the same method yields pairs of integers (t_i, v_i) such that

$$|dt_i - v_i/2^{\ell+1}|_q \leq q/2^{\ell+1}. \tag{2}$$

The goal is to recover d from this data. The problem is very similar to the hidden number problem considered by Boneh and Venkatesan in [2], and is approached by transforming it into a lattice closest vector problem.

More precisely, consider the $(m+1)$-dimensional lattice L spanned by the rows of the following matrix:

$$\begin{pmatrix} 2^{\ell+1}q & 0 & \cdots & 0 & 0 \\ 0 & 2^{\ell+1}q & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & \vdots \\ 0 & \cdots & 0 & 2^{\ell+1}q & 0 \\ 2^{\ell+1}t_1 & \cdots & \cdots & 2^{\ell+1}t_m & 1 \end{pmatrix}$$

Inequality (2) implies the existence of an integer c_i such that:

$$|2^{\ell+1}dt_i - v_i - 2^{\ell+1}c_iq| \leq q. \tag{3}$$

Now note that the row vector, called *hidden vector*,

$$\mathbf{c} = (2^{\ell+1}dt_1 + 2^{\ell+1}c_1q, \cdots, 2^{\ell+1}dt_m + 2^{\ell+1}c_mq, d)$$

belongs to L and $\mathbf{c}$ is very close to the row vector $\mathbf{v} = (v_1, \cdots, v_m, 0)$. Indeed, by (3), the distance from $\mathbf{c}$ to $\mathbf{v}$ is bounded as:

$$\|\mathbf{v} - \mathbf{c}\| \leq q\sqrt{m+1}.$$

We thus have a CVP to solve. In practice, we use an embedding technique to reduce CVP to SVP. This technique consists in computing the $(m+2)$-dimensional lattice L' spanned by the rows of the matrix

$$\begin{pmatrix} L & 0 \\ \mathbf{v} & 1 \end{pmatrix}$$

The row vector $(\mathbf{v} - \mathbf{c}, 1)$ is short, belongs to L' and we hope this is the shortest vector of L'. This assumption implies a condition on the required number of signatures depending on the parameter ℓ and the modulus. An estimate which makes it possible to recover the private key is:

$$m \gtrsim \frac{n}{\ell - \log_2 \sqrt{\pi e/2}}.$$

The above estimate is heuristic, but it is possible to give parameters for which attacks of this kind can be proved rigorously [13].

A Elliptic Curve Scalar Multiplications in NAF

We recall the definition and the NAF of integers.

Definition 1. *A non-adjacent form (NAF) of a positive integer k is an expression $k = \sum_{i=0}^{l-1} k_i 2^i$ where $k_i \in \{-1, 0, 1\}$, $k_{l-1} \neq 0$, and no two consecutive digits k_i are nonzero. The length of the NAF is l. The NAF of an integer k is denoted NAF(k) or $(k_{l-1}, \ldots, k_0)_{NAF}$.*

Algorithm 4. Left-to-Right NAF scalar multiplication

```
Input: $k = (1, k_{l-2}, \ldots, k_0)_{\text{NAF}}, P$
Output: $[k]P$
  $Q \leftarrow P$
  $i \leftarrow l - 2$
  while $i \geq 0$ do
      $Q \leftarrow 2Q$
      if $k_i = 1$ then $Q \leftarrow Q + P$
      if $k_i = -1$ then $Q \leftarrow Q - P$
      $i \leftarrow i - 1$
  end while
  return $Q$
```

The following algorithm computes the width NAF representation of the scalar on the fly.

Algorithm 5. Right-to-Left NAF scalar multiplication

```
Input: $k = (k_{n-1}, \ldots, k_0)_2, P$
Output: $[k]P$
  $R \leftarrow P$
  $Q \leftarrow \mathcal{O}$
  while $k \geq 1$ do
      if $k_0 = 1$ then
          $u \leftarrow (k \mod 4)$
          $k \leftarrow k - u$
          if $u = 1$ then
              $Q \leftarrow Q + R$
          else
              $Q \leftarrow Q - R$
          end if
      end if
      $R \leftarrow 2R$
      $k \leftarrow k/2$
  end while
  $Q \leftarrow Q + R$
  return $Q$
```