Ecole Normale Supérieure

Protection of cryptographic keys recodings against physical attacks

Author:
Simon Rastikian

Supervisor:
Arnaud Tisserand

July 2018

Abstract

The main purpose of this internship is to get introduced to the matter of physical security of a few **Elliptic Curve Cryptography** (ECC) algorithms against **Side Channel Attacks** (SCA). The **window Non-Adjacent Form** (NAF) **method for point multiplication** algorithm should be coded and integrated into an ECC library called **μNaCl** and injected into an ST machine **STM32L053R8 Nucleo**.

1 Introduction

Smart cards, mobile phones, iris recognition, radio-frequency identification, smart grids, wireless sensor network ... all benefit from an outstanding mathematical invention: Elliptic Curves [7].

Public-key cryptography was conceived in 1976 by the computer engineer Whitfield Diffie and the electrical engineer Martin Hellman. After several attempts to find the one-way function using the *permutation polynomials* and the *knapsack-based* problems, the first[1] practical realization appeared in 1977 when Ron Rivest, Adi Shamir and Len Adleman proposed their RSA cryptosystem. The RSA security was based on the intractability of the *integer factorization* problem. In 1985, Neal Koblitz and Victor Miller discovered the Elliptic Curve Cryptography (ECC) relying the schemes on the public-key mechanisms. Although the same functionality as RSA is provided, the ECC security is based on the hardness of a different problem: the **Elliptic Curve Discrete Logarithm Problem** (ECDLP) [9]. In 1987, Hendrick Lenstra published a paper analysing an algorithm to factor positive integers depending on the use of elliptic curves [8]. Elliptic curves algorithms began to prosper in 2004-2005 because of the advantages they provide comparing to RSA.

These algorithms might be mathematically secure, but this security is sometimes poor against physical attacks because some information might be revealed. Algorithms should then be secured against side channel attacks and in particular in this internship against **Power Analysis Attack**.

[1]In 1973 while working at the UK Government Communication Headquarters, Clifford C. Cocks creates a public-key cryptography algorithm that is equivalent to the RSA cryptosystem. His insight remained hidden for 24 years since the algorithm was classified information [14]

2 Elliptic Curve Cryptography

2.1 What are elliptic curves?

Definition 2.1.1 *An elliptic curve E over a field K is defined by the Weierstrass equation [9]:*

$$E\colon y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

where $a_1, a_2, a_3, a_4, a_6 \in K$ and the discriminant $\Delta \neq 0$.[2] *Δ is defined as follows:*

$$\begin{cases} \Delta = -d_2^2d_8 - 8d_4^3 - 27d_6^2 + 9d_2d_4d_6 \\ d_2 = a_1^2 + 4a_2 \\ d_4 = 2a_4 + a_1a_3 \\ d_6 = a_3^2 + 4a_6 \\ d_8 = a_1^2a_6 + 4a_2a_6 - a_1a_3a_4 + a_2a_3^2 - a_4^2 \end{cases}$$

More specifically, if L is an extension field of K then the elliptic curve is the set of points $(x, y) \in L^2$ which verify the Weierstrass equation plus a conventional point called point at infinity and noted as ∞.

Definition 2.1.2 *Two elliptic curves E_1, E_2 defined over a field K are said to be isomorphic over K if $\exists u, r, s, t \in K, u \neq 0$ such that [9]:*

$$\begin{aligned} \Phi\colon K^2 &\to K^2 \\ (x, y) &\mapsto (u^2x + r, u^3y + u^2sx + t) \end{aligned}$$

transforms equation E_1 into equation E_2. The transformation Φ is called an admissible change of variables.

Thus, depending on the characteristic p of the field K, Weierstrass equation can be significantly simplified using specific admissible change of variables. Three kinds of fields are especially amenable for the efficient implementation of elliptic curve systems:

1. **Prime fields** $\mathbb{F}_p$: For p prime greater than 3, E could be simplified into the equation $y^2 = x^3 + ax + b$ where $a, b \in K$ and $\Delta = -16(4a^3 + 27b^2)$.

2. **Binary fields** $\mathbb{F}_{2^m}$: For $p = 2$, two cases occur:

[2]It is necessary for the discriminant to be non-zero in order to guarantee the smoothness of the curve.

- If $a_1 \neq 0$, then E could be transformed to the *non-supersingular* curve $y^2 + xy = x^3 + ax^2 + b$ where $a, b \in K$ and $\Delta = b$.
- If $a_1 = 0$, then E could be transformed to the *supersingular* curve $y^2 + cy = x^3 + ax + b$ where $a, b, c \in K$ and $\Delta = c^4$.

3. **Optimal extension fields** $\mathbb{F}_{3^m}$: For $p = 3$, two cases occur:

   - If $a_1^2 \neq -a_2$, then E could be transformed to the *non-supersingular* curve $y^2 = x^3 + ax^2 + b$ where $a, b \in K$ and $\Delta = -a^3 b$.
   - If $a_1^2 = -a_2$ then E could be transformed to the *supersingular* curve $y^2 = x^3 + ax + b$ where $a, b \in K$ and $\Delta = -a^3$.

2.2 ECDLP

Note: The following mathematical formulas are only true for the prime fields.

2.2.1 Group Law

Let E be an elliptic curve defined over a field $\mathbb{F}_p$. One can define an additive law $+ \colon E(K) \to E(K)$ called *point addition*. The *chord-and-tangent rule* (figure 1) explains geometrically how the point addition works:

- Let $P = (x_1, y_1)$, $Q = (x_2, y_2)$ be two distinct points on E.
- Finding geometrically the *sum* R of P and Q consists of extending the chord between P and Q and intersecting this line with the curve at a third point R'.
- R is the reflection of R' about the x-axis.

The same procedure is done if $P = Q$ but rather than drawing a chord, the tangent to E is drawn at P. We then talk about *point doubling* rather than point addition and write $R = 2P$. Thus we can define few properties to $(E, +)$ using ∞ point [9]:

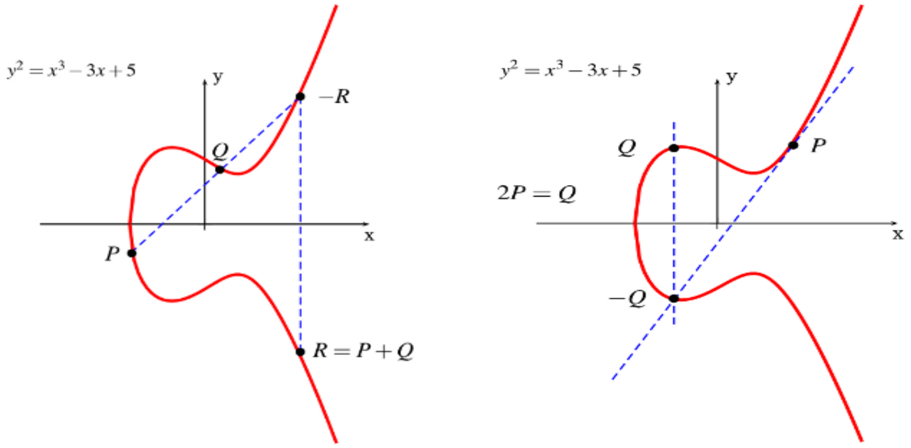
1. **Identity**: $P + \infty = \infty + P = P$, $\forall P \in E(K)$
2. **Negative**: Let $P = (x, y)$, the *negative* of P is noted $-P = (x, -y)$ and is indeed a point in $E(K)$. We then have $(x, y) + (x, -y) = \infty$.
3. **Point addition**: Let $P = (x_1, y_1), Q(x_2, y_2) \in E(K)$ such that $P \neq \pm Q$. If one of the two points is ∞ then the identity case is applied. Otherwise, $P + Q = (x_3, y_3)$, where

$$x_3 = (\frac{y_2 - y_1}{x_2 - x_1})^2 - x_1 - x_2 \text{ and } y_3 = (\frac{y_2 - y_1}{x_2 - x_1})(x_1 - x_3) - y_1$$

4. *Point doubling*: Let $P = (x_1, y_1) \in E(K)$ where $P \neq -P$. Then $[2]P = (x_3, y_3)$ where

$$x_3 = (\frac{3x_1^2 + a}{2y_1})^2 - 2x_1 \text{ and } y_3 = (\frac{3x_1^2 + a}{2y_1})(x_1 - x_3) - y_1$$

These properties imply that point addition is an associative and a commutative law; Thus (E,+) is an abelian group[3].



(a) Point addition: $R = P + Q$. (b) Point doubling: $R = [2]P$.

Figure 1: Geometric addition and doubling of elliptic curve points.

2.2.2 Ideas behind ECC

Let E be an elliptic curve defined over a field $\mathbb{F}_p$. If $P \in E(\mathbb{F}_p)$ has a prime order n, then one can define a cyclic group over $E(\mathbb{F}_p)$ generated by P.

$$\langle P \rangle = \{\infty, P, 2P, \ldots, (n-1)P\}$$

The *Elliptic Curve Discrete Logarithm Problem* (ECDLP) is based on the following idea:

[3]Using *Hasse theorem*, the group order $\#E(\mathbb{F}_q)$ can be bounded with the interval $[q + 1 - 2\sqrt{q}, q + 1 + 2\sqrt{q}]$.

- **Key pair generation**: Given E, p, P and n as a public domain parameters, one chooses a secret key $k \in_R [1, n-1]$. Then he/she computes $Q = kP$[4] and returns (Q, k).

- **ECDLP problem**: Given E, p, P and n as public domain parameter and having $Q = kP$, find k the discrete logarithm of Q to the base P.

2.3 ECC vs RSA

Unlike RSA's case, till today, no sub-exponential complexity algorithm for solving ECDLP has been discovered. Two of the fastest agorithms that had been discovered are Pollard's rho attack (time complexity is $\mathcal{O}(\sqrt{n})$) [10] and Shanks babystep-giantstep (time and space complexity is $\mathcal{O}(\sqrt{n})$) [5]. An alternative to RSA is ECC. Both key types share the same important property of being asymmetric algorithms. However, ECC can offer the same level of cryptographic strength at much smaller key size. Table 1 shows the difference between the key sizes needed for the same security type with RSA and ECC.

Symmetric key size (bits)	RSA and DH key size (bits)	ECC key size (bits)
80 (SKIPJACK)	1024	160
112 (Triple-DES)	2048	224
128 (AES-Small)	3072	256
192 (AES-Medium)	7680	384
256 (AES-Large)	15360	512

Table 1: NIST comparison of ECC, RSA and DH key sizes for different security requirements [3].

2.4 Projective-coordinates

Formulas for point addition and point doubling require many field arithmetic operations, such as addition, squaring, multiplication... But they also require field inversion which is highly time and electricity consuming compared to field multiplication. Therefore, a nice method for reducing the field inversion is using the projective-coordinate representation rather then the affine coordinates.

[4] kP is a notation for $\underbrace{P + P + \cdots + P}_{\text{k times}}$ called scalar multiplication.

Definition 2.4.1 *Let K be a field and c and d two positive integers. An equivalence relation $\sim$ on the set $K^3 \setminus \{(0,0,0)\}$ noted as $(X_1, Y_1, Z_1) \sim (X_2, Y_2, Z_2)$ exists if $X_1 = \lambda^c X_2$, $Y_1 = \lambda^d Y_2$, $Z_1 = \lambda Z_2$ for some $\lambda \in K^*$. The equivalent class containing $(X, Y, Z) \in K^3 \setminus \{(0,0,0)\}$ is*

$$(X : Y : Z) = \{(\lambda^c X, \lambda^d Y, \lambda Z) | \lambda \in K^*\}$$

$(X : Y : Z)$ is called a projective point, and (X,Y,Z) is called a representative of $(X : Y : Z)$.

We thus have a 1-1 correspondence between the projective points

$$\mathbb{P}(K)^* = \{(X : Y : Z) | X, Y, Z \in K, Z \neq 0\}$$

and the affine points

$$\mathbb{A}(K) = \{(x, y) : x, y \in K\}$$

that lie on E. The set of projective points with Z=0 which lies on E is called *line at infinity* and corresponds to the points at infinity in the affine coordinate. There are several types of projective coordinates that transforms the elliptic curve $y^2 = x^3 + ax + b$:

1. **Standard projective coordinates** ($c = 1, d = 1$): the projective point (X, Y, Z), $Z \neq 0$ corresponds to the affine point $(X/Z, Y/Z)$ and $(0 : 1 : 0)$ to ∞. The equation becomes $Y^2 Z = X^3 + aXZ^2 + bZ^3$

2. **Jacobian projective coordinates** ($c = 2, d = 3$): the projective point (X, Y, Z), $Z \neq 0$ corresponds to the affine point $(X/Z^2, Y/Z^3)$ and $(1 : 1 : 0)$ to ∞. The equation becomes $Y^2 Z = X^3 + aXZ^4 + bZ^6$. In Jacobian and standard projective coordinates, the negative of $(X : Y : Z)$ is $(X : -Y : Z)$

3. **Chudnovsky coordinates**: the Jacobian point $(X : Y : Z)$ is represented with redundancy as $(X : Y : Z : Z^2 : Z^3)$.

3 Side Channel Attacks

Modern security systems use cryptographic algorithms to provide confidentiality, integrity and authenticity of data. Breaking (partially) a cryptographic device means extracting the key of the device or some secret information about it. Evaluating the security of a cryptographic device means making assumptions about the knowledge that an attacker has about it. The strongest assumptions that can be made extends Kerckhoff's principle[5]

[5]Auguste Kerckhoff's principle is that a cryptographic algorithm and all its details except for the secret key should be publicly available.

and assumes that the attacker knows all the details about the cryptographic device and has access to it. Unfortunately, electronic circuits are inherently leaky - the emissions they produce make it possible for an attacker to figure out what data is being processed. Heat and electromagnetic emanations are both viable sources of information for an attacker.

3.1 Power Analysis Attacks

Power analysis is a form of side channel attacks in which the attacker observes and studies the power consumption of a cryptographic device. It exploits the fact that the instantaneous power consumption depends on the data processed and on the operation performed. Because they are powerful attacks, easy to perform, power analysis attacks have got much attention over the past decades. Because they can be a real threat on the security, designers and developers should be familiar with them and their countermeasures. Two types of power analysis attacks are well-known [12]:

1. **Simple Power Attack (SPA)**: It is an attack that is based on the visual examination of graphs of the current used by a device overtime. The goal is to reveal the key when given only small number of power traces. It exploits the fact that arithmetic operations are not equivalent in power consumption because some are "easy" to execute by the device and others are "hard". In this internship, this type of attacks will be focused on.

2. **Differential Power Attack (DPA)**: It is a popular type of power analysis attack in symmetric cryptography because it does not require detailed knowledge about the attacked device. It involves statistically analyzing power consumption measurements from a cryptosystem. In contrast to SPA, DPA requires a large number of power traces to exploit the data dependency of the power consumption of the cryptographic device and to analyze it at a fixed moment of time as a function of the processed data.

3.2 NAF

3.2.1 Scalar Multiplication: Double-and-Add algorithm

In ECC, classical key pair generation involves the following algorithm for calculating $Q = kP$.

Algorithm 1 Double-and-Add algorithm

```
    Input: $k = (k_{n-1}k_{n-2}\dots k_0)_2$, $P \in E(\mathbb{F}_p)$
    Output: $Q = kP$
  $Q \leftarrow \infty$
  for $i$ from $n-1$ to $0$ do
      $Q \leftarrow [2]Q$ (DBL)
      if $k_i = 1$ then $Q \leftarrow Q + P$ (ADD)
```

In each of the projective coordinates representation, point addition costs more than point doubling. For instance, adding two points represented in Jacobian coordinates costs 12 field multiplications and 4 field sums, but doubling a point represented in Jacobian coordinates costs only 4 field multiplications and 4 field sums. As figure 2 shows, this algorithm is not secure against SPA because by only looking at the power trace, one can know the successive bits of the key.

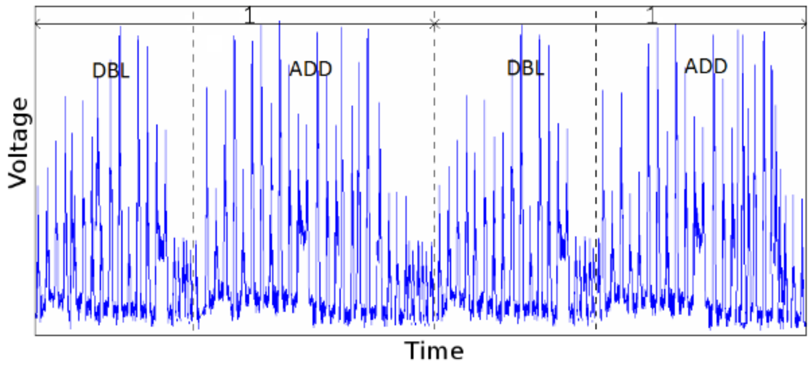


Figure 2: Power consumption measure of Double-and-Add algorithm (from left to right) coded on an FPGA [4].

3.2.2 Window Non-Adjacent Form (NAF) method for point multiplication

In this internship, we focused on understanding and coding the NAF representation algorithm in C language. The w-NAF algorithm for point multiplication is more or less the same classical Double-and-Add algorithm but with a different secret key representation. It focuses on the fact that subtracting a point has the same cost as adding a point since with Jacobian coordinates $-(X, Y, Z) = (X, -Y, Z)$.

Definition 3.2.1 *Let $w > 1$ and k be positive integers. A width-w NAF of k is the expression $k = \sum_{i=0}^{l-1} k_i 2^i$ where $|k_i| < 2^{w-1}$ and k_i are either odd or zero and $k_{l-1} \neq 0$. In this representation, at most one of any w consecutive digits is nonzero.*

Thus, the width-w NAF has few special properties like given k a positive integer, the width-w NAF of k is unique and is noted $NAF_w(k)$. Also, the length of $NAF_w(k)$ is at most one unit longer than the length of k in its binary representation. The first part of the algorithm is to calculate $NAF_w(k)$.

Algorithm 2 width-w NAF of a positive integer

```
    Input: $k$ positive integer, $w$
    Output: $NAF_w(k)$
  $i \leftarrow 0$
  while $k \geqslant 0$ do
      if $k$ is odd then $k_i \leftarrow k$ mods $2^w$, $k \leftarrow k - k_i$
      else $k_i \leftarrow 0$
      $k \leftarrow k/2$, $i \leftarrow i + 1$
  Return $NAF_w(k) = (k_{i-1} \dots k_0)$
```

Where mods is a function that keeps k_i in $[\![-2^{w-1}, 2^{w-1}[\![$.
The second part is to calculate kP.

Algorithm 3 window Non-Adjacent Form method for point multiplication

```
    Input: $k$ positive integer, $w$, $P \in E(\mathbb{F}_q)$
    Output: $kP$
  Calculate $NAF_w(k)$
  Compute and store all $P_i = iP$ $\forall i$ odd and $i < 2^{w-1}$
  $Q \leftarrow \infty$
  for $i$ from $l - 1$ downto 0 do
      $Q \leftarrow 2Q$
      if $k_i \neq 0$ then
          if $k_i > 0$ then $Q \leftarrow Q + P_{k_i}$
          else $Q \leftarrow Q - P_{-k_i}$
```

The purpose of this algorithm is to compute kP faster than the classical algorithm since point addition is used less frequently than in the classical algorithm (the density of nonzero digits among all width-w NAF representations of length l is approximately $\frac{1}{w+1}$). This algorithm trades space[6] with time.

Theoretically, looking closer at this algorithm, we can notice that it is not completely secure against SPA because the secret key can be partially discovered since the same weakness as the classical algorithm remains: computing different cost operations. Indeed, in this algorithm if $k_i = 0$, only point doubling is performed which is cheaper than point addition computed when $k_i \neq 0$. Only this time, if $k_i \neq 0$, k_i cannot be known by simply analyzing the power trace.

3.3 STM32L053R8 Nucleo

The STM32L053R8 Nucleo is one of the ST development boards that our cryptographic algorithm will be integrated to [15]. It is an Ultra-low power platform with an ARM 32-bit Cortex-M0+ processor. It is equipped with 64 Kbytes Flash, 8 Kbytes RAM, 32 MHz CPU, one user led and two buttons (one user button and one reset button).
The first step was to make the user led twinkle. This has been done by using the ARM Mbed OS that is enabled on the board. The ARM Mbed OS provides a transformative device-to-data platform that facilitates the coding by providing the necessary libraries, secures the board and gives a wide range of communication options with the drivers.

Later, we discovered that it is better (for compatibility issues with the $\mu NaCl$ library) to use the cross compiler *Clang* and the interface *ST-link* (that ports the code from the computer to the board).

3.4 Measurement setup: Digital Sampling Oscilloscope

The power consumption signal emitted by the cryptographic device needs to be recorded. Generally in measurement setups, this is done by a digital sampling oscilloscope. Such an oscilloscope converts the input analog voltage signal into an output digital signal, and stores it in it's memory. Three parameters characterize the analog-to-digital conversion:

- **Input Bandwidth:** By applying the Fourier transform, every analog signal can be viewed as a sum of sinusoidal function multiplied by some coefficient. The bandwidth of a signal is defined as the difference between the highest and the lowest frequency component that is present

[6]Storing a point takes too much space, this is why the window Non-Adjacent Form method for point multiplication is generally computed for $w \in \{2, 3, 4\}$.

in the signal. For an oscilloscope, the minimum frequency recorded is *0 Hz*. The oscilloscope that is aimed to be used in this internship is a high-end oscilloscope and can accept a maximum frequency of 1GHz. Above this frequency, distortion of the signal takes place.

- **Sampling rate:** The sampling rate determines how many points of the analog signal are recorded per second. Nyquist-Shannon sampling theorem states that the sampling rate needs to be more than twice as high as the highest frequency component of the input signal in order to avoid a loss of information [13].

- **Resolution:** It is the conversion of the time-discrete signal into time- and value-discrete signal this means that the range of sampled values is reduced from infinite number of possible values to a finite number of possible ones. The oscilloscope that we planned to use has a resolution of 8 bits, this means that each sampled value is mapped to one of 256 possible output values.

4 Application

4.1 $\mu NaCl$ library

As written before, NAF algorithms had to be coded and integrated into a library $\mu NaCl$. $\mu NaCl$ is an ongoing library made for ECC. The core $\mu NaCl$ (which is mostly the arithmetic field operations coded in ARM assembly) was written by Michael Hutter and Peter Schwabe, and the standalone Curve25519 implementation for ARM Cortex-M0 (which is mostly the operations on the curve like point multiplication, point addition . . .) was written by Björn Haase and Ana Helena Sánchez. First, we downloaded the library and worked at compiling it. $\mu NaCl$'s documentation required the installation of the cross compiler Clang. We had some difficulties compiling the code (some options weren't understood by Clang like -mfloat-abi=soft), thus we thought that this might be caused by the Clang package installation, so we installed Clang from the source. These problems were solved but some other problems related to the linker (ld) came up (like not finding libc.a). We surpassed these difficulties by installing a specific package that was missing : libnewlib-arm-none-eabi. Getting this done, the compilation finally worked.

4.2 Applying NAF algorithms

Width-w NAF algorithm has been coded for a key k of a maximum size of 255 bits and for w between 2 and 8. First, we coded $NAF_w(k)$ using basic operations such as bits manipulation, logical shifts, addition by one.... We then added some space optimization such that it will take a size of 512 bits for $w = 2$, 1024 bits for $w = 4$ and 2048 bits for the other cases. Then, we started coding the window NAF method for point multiplication. While doing so, we noticed that the Curve25519 coded in $\mu NaCl$ has no Y projective coordinate. This means that it's not possible to subtract a point since $-(X, Y, Z) = (X, -Y, Z)$.

Actually, Curve25519 is a Montgomery curve which is a form of elliptic curve (can be transformed into the short Weierstrass's model) defined over $\mathbb{F}_q$ by the equation [11] [6]:

$$M_{A,B} : By^2 = x^3 + Ax^2 + x \text{ where } A, B \in \mathbb{F}_q \text{ and } B(A^2 - 4) \neq 0.$$

Usually, Montgomery projective coordinates (standard projective coordinates but without the Y coordinate) are used with this type of curves[7]. Curve25519 is defined over the quadratic extension of the prime field $\mathbb{F}_{2^{255}-19}$ [1] and has the following formula:

$$Curve25519 : y^2 = x^3 + 486662x^2 + x.$$

This field is used because one can code fast arithmetic modulo $2^{255} - 19$ (see [1]).

4.3 Montgomery algorithm

4.3.1 Differential operations

Not stocking the Y projective coordinate implies the non-possibility of subtracting points and a loss of information and thus representing (x,y) and (x,-y) as the same point (X:Z). Having so, we thought about representing the key k in radix-8 form, precomputing the points $\{P \dots 7P\}$ and executing a similar algorithm as the Add-and-Double algorithm. Unfortunately, this was not possible because of the space a point addition can use in a Montgomery representation. Actually, in a Montgomery model, adding two points is done using *differential addition.* A differential addition is an addition where each sum is already accompanied by a difference, i.e. computing

[7]A detailed explanation about building Montgomery projective coordinates, recovering the y coordinate, adding and doubling is written in [6].

P+Q requires the presence of P, Q and $P-Q$ [2].
Note: A differential doubling can be directly done since it only needs the presence of P and ∞.
Montgomery observed that on a Montgomery curve, the x-coordinates of P, Q, $P-Q$ and $P+Q$ are related with the following equations [6]:

$$x_{P+Q}x_{P-Q}(x_P - x_Q)^2 = (x_P x_Q - 1)^2 \text{ if } P \neq Q.$$
$$4x_{[2]P}x_P(x_P^2 + Ax_P + 1) = (x_P^2 - 1)^2 \text{ if } P = Q.$$

Thus using Montgomery projective coordinates, the differential addition and differential doubling equations become each a pair of simultaneous relations. Having this defined, we can say that the loss of information of Y coordinate is recovered by the presence of $P-Q$. Therefore, the radix-8 algorithm is not a good idea because supposing we have Q and we are reading the next 3 bits of the key, we need to have $Q - iP \ \forall\ i \in [\![0, 7]\!]$. Recursively, one should store around $O(8^{length(k)/3})$ points while computing the algorithm.

4.3.2 Combining algorithms

This type of operations are made to be actually used in the Montgomery ladder algorithm:

Algorithm 4 Montgomery ladder algorithm

```
    Input: $k = \sum_{i=0}^{l-1} k_i 2^i$ with $k_{l-1} = 1$, $P \in E(\mathbb{F}_q)$
    Output: $kP$
  $Q = (Q_0, Q_1) \leftarrow (P, [2]P)$
  for $i$ from $l-2$ down-to 0 do
      if $k_i = 0$ then $(Q_0, Q_1) \leftarrow ([2]Q_0, Q_0 + Q_1)$
      else $(Q_0, Q_1) \leftarrow (Q_0 + Q_1, [2]Q_1)$
  return $Q_0$
```

Note: this algorithm is secure against SPA since in each step one point addition and one point doubling are done.
Inspired by this algorithm, we thought about combining the width-w NAF representation with this algorithm. The resulting algorithm is not better than the classical one because it computes w-1 more addition in each loop and stores w-1 more points than the classical Montgomery ladder algorithm. Thus, this algorithm has not been coded and stayed as a proposition.

Algorithm 5 width-2 NAF and Montgomery ladder algorithm

```
  Input: $k = \sum_{i=0}^{l-1} k_i 2^i$ with $k_{l-1} = 1$, $P \in E(\mathbb{F}_q)$
  Output: $kP$
compute $NAF_2(k)$
$Q = (Q_0, Q_1, Q_2) \leftarrow (P, [2]P, \infty)$
for $i$ from $length(NAF_2(k)) - 2$ down-to 0 do
    if $k_i = 0$ then
        $(Q_0, Q_1, Q_2) \leftarrow ([2]Q_0, Q_0 + Q_1, Q_2 + Q_0)$
    else if $k_i = 1$ then
        $(Q_0, Q_1, Q_2) \leftarrow (Q_0 + Q_1, [2]Q_1, Q_2 + Q_1)$
    else
        $(Q_0, Q_1, Q_2) \leftarrow (Q_0 + Q_2, Q_1 + Q_2, [2]Q_2)$
return $Q_0$
```

5 Conclusion

To conclude, when using Weierstrass equation defined over a finite field, one can get some interesting elliptic curves. Operations (point addition and point doubling) over these elliptic curves can be defined using the chord-and-tangent rule. These operations are expensive if one does not use the projective coordinates. Elliptic curves are used in cryptography because solving the ECDLP problem is hard. ECDLP might be mathematically secure but it should also be physically secure against some side channel attacks like the SPA. NAF algorithms can partially secure the secret key in ECDLP but it is not sufficient. We tried coding these algorithms in a cryptographic library ($\mu NaCl$) but we discovered some interesting mathematical and informatical points that makes it incompatible with the Montgomery curve Curve25519. Curve25519 does not stock some specific coordinate and it uses the differential addition. Even though combining the NAF representation with Montgomery ladder algorithm was not optimized in time and space, this algorithm works well and is theoretically secure against SPA. Finally, ECC is a well reputed domain nowadays because of the great level of security it can afford... but how can we extend it to be also resistant against tomorrow's quantum-computers?

References

[1] D. J. Bernstein. *Curve25519: new Diffie-Hellman speed records*. Springer, 2006.

[2] D. J. Bernstein. *Differential addition chains*. 2006. URL: https://cr.yp.to/ecdh/diffchain-20060219.pdf.

[3] D. R. L. Brown. *Standards for Efficient Cryptography: SEC 1: Elliptic Curve Cryptography*. 2009. URL: http://www.secg.org/sec1-v2.pdf.

[4] T. Chabrier. *Arithmetic recodings for ECC cryptoprocessors with protections against side-channel attacks*. 2013. URL: https://www.theses.fr/174580924.

[5] H. Cohen. *A course in computational algebraic number theory*. Springer, 2000.

[6] C. Costello and B. Smith. *Montgomery curves and their arithmetic: The case of large characteristic fields*. 2017. URL: https://arxiv.org/pdf/1703.01863.pdf.

[7] R. Harkanson and Y. Kim. "Applications of elliptic curve cryptography: a light introduction to elliptic curves and a survey of their applications". In: *ACM International Conference Proceeding Series* (2017).

[8] H. W. Lenstra. "Factoring integers with elliptic curves". In: *The Annals of Mathematics* (1987).

[9] D. Hankerson, A. Menezes and S. Vanstone. *Guide to Elliptic Curve Cryptography*. Springer, 2004.

[10] S. D. Miller and R. Venkatesan. *Spectral analysis of Pollard rho collisions*. Springer, 2006.

[11] P. L. Montgomery. "Speeding the Pollard and elliptic curve methods of factorization". In: *Mathematics of computation* (1987).

[12] S. Mangard, E. Oswald and T. Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer, 2007.

[13] A. Oppenheim, R. Schafer and J. Buck. *Discrete-time Signal Processing*. Prentice Hall, 1999.

[14] S. Singh. *The Code Book: The Secret History of Codes and Codebreaking*. Fourth Estate, 2000.

[15] *STM32L053R8*. URL: https://www.st.com/en/microcontrollers/stm32l053r8.html.