



Brandenburgische
Technische Universität
Cottbus - Senftenberg



Brandenburg University of Technology Cottbus-Senftenberg
Faculty 1 - MINT - Mathematics, Computer Science, Physics,
Electrical Engineering and Information Technology
Chair of Wireless Systems

Distinguishability Investigation on Longa's Atomic Patterns when used as a Basis for Implementing Elliptic Curve Scalar Multiplication Algorithms

MASTER'S THESIS

Sze Hei LI

This thesis is submitted for the degree of Master of Science in Cyber Security

August 5, 2024

Supervisor: Hon. Prof. Dr.-Ing. Zoya DYKA
Second Examiner: Dr.-Ing. Ievgen KABIN

Acknowledgements

I am deeply grateful to Hon. Prof. Dr.-Ing. Zoya Dyka and Dr.-Ing. Ievgen Kabin for their invaluable guidance and support throughout my research. Their expertise, encouragement, and constructive feedback have been instrumental in shaping this thesis. I sincerely appreciate their time and dedication, which have significantly contributed to the success of this work.

Abstract

In the evolving landscape of cryptographic security, the robustness of Elliptic Curve Cryptography (ECC) against side-channel analysis (SCA) attacks is of paramount importance due to the widespread use of ECC and the growing sophistication of SCAs. This thesis delves into the investigation of Longa's atomic patterns applied within Elliptic Curve scalar multiplication algorithms, assessing their resistance to horizontal SCAs. The research employs these atomic patterns in practical implementation on a microcontroller (Texas Instruments Launchpad F28379 board) using the open-source cryptographic library FLECC in C.

In our analysis, we only focused on the distinguishability of the first atomic block in the Elliptic Curve point doubling and point addition patterns. Due to various technical limitations, we were unable to determine significant differences in the execution time and the shapes of the atomic blocks. Further investigations of the SCA-resistance can be performed based on this work.

A significant contribution of this work is the identification and correction of several discrepancies in Longa's original atomic patterns. This thesis marks the first practical implementation of Longa's patterns, extending the theoretical research into empirical analysis.

Contents

List of Figures	x
List of Tables	xii
List of Algorithms	xiii
List of Abbreviations	xv
1 Introduction	1
1.1 Motivation	1
1.2 Contributions	1
1.3 Structure of This Thesis	2
2 Background	3
2.1 Elliptic Curve Cryptosystem	3
2.1.1 Binary Algorithms for EC Scalar Multiplication	4
2.2 Side-Channel Analysis Attacks	5
2.3 Vulnerabilities of kP Implementations to SCA	5
2.4 Atomicity as a Countermeasure Against SSCA	7
2.5 Vulnerabilities of the Atomic Patterns	8
2.5.1 Key-dependent Data Processing	8
2.5.2 Key-dependent Addressing of Registers	9
3 State-of-the-art Literature Overview	11
3.1 Analysis of MNAMNAA-based Algorithms	12
4 Board for Experiments	17
5 Selection of a Suitable Open-source Cryptographic Library	19
5.1 FLECC in C	20
5.2 Mbed TLS	20
5.3 Botan	20
6 Investigations	21
6.1 Implemented Atomic Patterns kP Algorithm	21
6.2 Measurements	28
6.2.1 Experimental Setup	28
6.2.2 Study of FLECC Constant-runtime Operations	31
6.3 Simple Electromagnetic Analysis Attack	33
6.3.1 Analysis of kP Operation Executed in RAM	33
Synchronization Alignment	37
Identification of Operations in Atomic Blocks	46
Automated Simple SCA	47
6.3.2 Analysis of kP Operation Executed in Flash Memory	52

	Synchronization Alignment	52
6.3.3	Comparison of the Results to Sigourou et al.'s Work [27]	53
7	Conclusion	55
8	Appendix	57
8.1	Results of Literature Overview	57
8.2	Revision of Longa's Atomic Pattern Algorithms	60
8.2.1	Longa's Appendix B2 algorithm revised	60
8.2.2	Longa's Appendix B3 algorithm revised	61
8.2.3	Longa's Appendix B4 algorithm revised	62
8.2.4	Longa's Appendix B5 algorithm revised	63
8.2.5	Longa's Appendix B6 algorithm revised	64
8.2.6	Longa's Appendix B7 algorithm revised	65
8.2.7	Longa's Appendix B8 algorithm revised	66
8.3	Distinguishability of Doubling 1 and Doubling 2	67
8.4	Execution Time of No Operations	68
8.5	Attack Scenario	68

List of Figures

3.1	Number of published papers that referenced [2] over the years.	11
3.2	Distribution of themes of literature referencing [2].	12
4.1	Front side and back side of the board attacked.	17
6.1	The Integrated Circuit Scanner positioning the board and the EM-probe.	28
6.2	The oscilloscope connected to the probe showing the captured trace. .	29
6.3	Near-field micro probe placed near capacitor C78 on the board.	29
6.4	Distribution of execution time of the first X in $\Delta 1$ of all point doublings (left), all point additions (middle) and all EC point operations (right). .	31
6.5	Measurement of the kP execution: EM trace of the implemented atomic patterns kP algorithm executed in RAM of the board.	33
6.6	A part of the captured EM trace: the noise before the start of the kP operation, the processing of the most significant bit $k_{l-1}=1$, and the atomic patterns of the first 3 point doublings are shown; kP executed in RAM of the board.	34
6.7	EM trace representing a part of the atomic pattern kP algorithm executed in RAM of the board.	35
6.8	The distribution of measured execution time of point doubling operations (left) and point addition operations (right) in a kP algorithm executed in RAM.	37
6.9	The distribution of measured execution time of point doubling operations (left) and point addition operations (right) in a kP algorithm executed in flash memory.	37
6.10	Above: alignment of 4,000 anchor samples in Synchronization A. Below: zoomed in at samples 200-400.	38
6.11	Method used in Synchronization B: Identification and alignment of rising segments.	39
6.12	Above: alignment of 4,000 anchor samples in Synchronization B. Below: zoomed in at samples 200-400.	39
6.13	Method used in Synchronization C: Identification of local minimum in each clock cycle (Above). Connection of local minimums (Middle). Synchronization done by shifting Addition 2 one unit to the left. (Below)	40
6.14	Above: alignment of 4,000 anchor samples in Synchronization C. Below: zoomed in at samples 200-400.	40
6.15	Overlay of $\Delta 1$ of all sub-traces synchronized using Synchronization A, with the algorithm executed in RAM.	41
6.16	Overlay of $\Delta 1$ of all sub-traces synchronized using Synchronization A, with point addition traces laying on top of point doubling traces, and the algorithm executed in RAM.	41
6.17	Overlay of $\Delta 1$ of all sub-traces synchronized using Synchronization B, with the algorithm executed in RAM.	42

6.18 Comparison of amplitudes of mean of sub-traces between Synchronization A and Synchronization B.	43
6.19 Overlay of $\Delta 1$ of all sub-traces synchronized using Synchronization C, with the algorithm executed in RAM.	43
6.20 Atomic blocks of the synchronized point addition and point doubling sub-traces overlaid (additions: red; doublings: blue) and the mean trace of their sub-traces (addition: yellow; doubling: green).	44
6.21 The samples at the beginning of $\Delta 1$ at location A in Figure 6.20, lines overlaid in different orders.	45
6.22 The samples at the end of $\Delta 1$ at location B in Figure 6.20, lines overlaid in different orders.	45
6.23 The samples at the end of $\Delta 4$ at location C in Figure 6.20, lines overlaid in different orders.	45
6.24 Identification of field operations based on the measured execution time of the operations.	47
6.25 An example of identifying a distinguishable sample using a 3-sample window.	48
6.26 Indication of the locations of samples in Figures 6.27 (β) and 6.28 (α) in $\Delta 1$, with each atomic pattern's duration estimated from the execution time measured in Chapter 6.2.2.	49
6.27 Minimum and maximum voltages range for point addition (orange) and point doubling (blue) operations showing a distinguishable sample at index 338,287 with $v \geq 0.004$ V in a 5-sample window.	50
6.28 Minimum and maximum voltages range for point addition (orange) and point doubling (blue) operations showing distinguishable samples with indices 186,810 and 186,811, in voltage range $0.002 \text{ V} < v \leq 0.003$ V in a 5-sample window.	51
6.29 EM trace of noise and the implemented atomic pattern kP algorithm executed in flash memory of the board attacked.	52
6.30 EM trace of noise and the beginning of the kP algorithm executed in flash memory of the board attacked.	52
6.31 EM trace of atomic blocks of point operations of the kP algorithm executed in flash memory of the board attacked.	53
8.1 (a) Showing Doubling 1 (blue) and Doubling 2 (red) are non-synchronous at 20,000 samples after the previous synchronization, (b) synchronous again after shifting Doubling 2 by 50 samples left.	67

List of Tables

2.1	MNAMNAA-based atomic point doubling, taken from [2].	7
2.2	MNAMNAA-based atomic point addition, taken from [2].	8
3.1	Summary of literature citing Longa's atomic patterns as state-of-the-art, analysing or improving Longa's atomic patterns.	15
5.1	Summary of cryptographic libraries studied.	20
6.1	MNAMNAA atomic patterns sequence for EC point doubling implemented in this work.	22
6.2	MNAMNAA atomic patterns sequence for EC point addition implemented in this work.	22
6.3	Implemented sequence of operations using constant-time FLECC functions for EC point doubling and EC point addition operations applying the atomic pattern MNAMNAA.	26
6.4	Settings applied in our measurements.	30
6.5	Execution time of FLECC constant-runtime operations in $\Delta 1$ of Doubling 2, measured 10 times.	31
6.6	Execution time of the first X operation in $\Delta 1$ of each point operation, measured in clock cycles.	32
6.7	Atomic patterns point operations execution time measured in clock cycles, using breakpoints.	36
6.8	Methods used for the fine synchronization of EC point operations. . .	37
6.9	Number of distinguishable samples found using different sample window values, with each sample's distinguishability segmented by the voltage difference v between the two separated sets of voltages (measured in Volts).	49
6.10	Count of occurrences of consecutive distinguishable samples (s) across different sample windows.	50
6.11	Summary of comparison between [27] by Sigourou et al. and our work.	53
8.1	Literature overview: aspects relevant for this thesis were taken as the basis for classification of papers referring to [2]	59
8.2	Revised MNAMNAA-based atomic point doubling. The newly revised register is shown in bold with yellow highlight.	60
8.3	Revised MANA-based atomic mixed addition. The newly revised registers are shown in bold with yellow highlight.	61
8.4	Revised MNAMNAA-based atomic mixed addition. The newly revised registers are shown in bold with yellow highlight.	62
8.5	Revised MANA-based atomic special addition with identical Z-coordinate. The newly revised registers are shown in bold with yellow highlight. .	63

8.6	Revised MNAMNAA-based atomic special addition with identical Z-coordinate. The newly revised registers are shown in bold with yellow highlight.	64
8.7	Revised MANA-based atomic point tripling. The newly revised register is shown in bold with yellow highlight.	65
8.8	Revised MNAMNAA-based atomic point tripling. The newly revised register is shown in bold with yellow highlight.	66
8.9	Measured execution time of 2,000 NOPs.	68
8.10	Measured execution time of 5,000 NOPs.	68

List of Algorithms

1	Left-to-right binary double-and-add kP algorithm	4
2	Modified left-to-right binary double-and-add kP algorithm	21

List of Abbreviations

A	Addition
CCS	Code Composer Studio
DPA	Differential Power Analysis
DSCA	Differential Side-Channel Analysis
EC	Elliptic Curve
ECC	Elliptic Curve Cryptosystem
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
EM	Electromagnetic
FLECC	FLECC in C
Frac-wmbNAF	Fractional Window-w multibase Non Adjacent Form
HCCA	Horizontal Collision Correlation Analysis
ICS	Integrated Circuit Scanner
IDE	Integrated Development Environment
LM	Longa-Miri precomputation
M	Multiplication
mbNAF	multibase Non Adjacent Form
N	Negation
NOP	No Operation
PA	Point Addition
PD	Point Doubling
RAM	Random Access Memory
RSA	Rivest-Shamir-Adleman cryptosystem
SCA	Side-Channel Analysis
SOS	Separated Operand Scanning
SPA	Simple Power Analysis
SSCA	Simple Side-Channel Analysis
wmbNAF	Window-w multibase Non Adjacent Form

Chapter 1

Introduction

1.1 Motivation

Based on the side-channel atomicity principle introduced by Chevallier-Mames et al. [1], Longa developed Simple Side-Channel Analysis (SSCA)-resistant Elliptic Curve (EC) scalar multiplication atomic formulae in his work [2], which have up to 22% faster speed than using traditional atomic structures and improved the security of Elliptic Curve scalar multiplication (kP) against side-channel analysis (SCA) attacks. Building resistance for kP algorithms against simple SCA attacks is the primary goal of the atomicity principle.

Recent work by Kabin [3] has shown that the address-bit vulnerability can be exploited using horizontal, i.e., single trace, SCA attacks. Key-dependent addressing of registers is an inherent part of regular as well as atomic patterns-based kP algorithms. Although many studies in SCA have focused on improving the efficiency and security of EC scalar multiplication theoretically, there are very few studies focusing on the practical implementation and analysis of the existing approaches for hardware or embedded devices. Thus, Longa's atomic patterns can be an insufficient SCA countermeasure. This work fills that research gap by performing horizontal SCA attacks against an implementation of a binary scalar multiplication kP algorithm using Longa's MNAMNAA atomic pattern [2].

1.2 Contributions

The contributions of this thesis are:

- An overview of the literature that referenced Longa's master's thesis [2]
- The correction of Longa's atomic patterns
- The implementation of a kP algorithm using Longa's MNAMNAA atomic patterns on a Texas Instruments Launchpad F28379 evaluation board TMS320F28379D with a 32-bit microcontroller using the open-source cryptographic library FLECC in C
- The implementation of SSCA attacks analysing an electromagnetic trace of a kP execution

To the best of our knowledge, we are the first to implement Longa's point doubling and mixed point addition atomic patterns in an embedded device.

1.3 Structure of This Thesis

Following the introduction, this thesis is organized as follows: Chapter 2 explains the basic concepts regarding Elliptic Curve Cryptosystem, SCA, kP algorithm vulnerabilities and atomicity principle. In Chapter 3, we review some state-of-the-art literature related to Longa's atomic pattern algorithms. Chapter 4 describes our experimental setup regarding the implementation of Longa's atomic pattern algorithms. In Chapter 5, we look into a selected number of open-source cryptographic libraries to choose a suitable one for our constant-time atomic kP algorithm implementation on our target device. Chapter 6 describes the practical implementation of Longa's atomic pattern algorithms and our SCA methodology. Chapter 7 summarizes this thesis. Finally, Chapter 8 serves as the appendix, where the detailed descriptions of the literature overview, corrections of Longa's atomic patterns, the distinguishability between Doubling 1 and Doubling 2, the execution time of "no operations", and an attack scenario are presented. These contents are placed in the final chapter to improve the overall readability of the thesis. The appendix represents a significant part of the investigations performed in this work.

Chapter 2

Background

2.1 Elliptic Curve Cryptosystem

Elliptic Curve Cryptosystem (ECC) was introduced independently by Miller [4] and Koblitz [5] in 1985. It is a public-key cryptosystem used to perform critical security functions including encryption, authentication and digital signatures. Not only can ECC achieve higher security per key bit compared to RSA, its other benefits such as reduced memory requirements, lower energy consumption and faster execution time have made it suitable for implementation of cryptographic protocols for resource-constrained embedded devices.

In this thesis, we discuss an implementation of elliptic curve point multiplication using an elliptic curve E over a prime finite field \mathbb{F}_p , where p is a large prime and represents the number of elements of the finite field. The EC over \mathbb{F}_p is denoted by $E(\mathbb{F}_p)$ and defined as follows [6]:

$$E(\mathbb{F}_p) : y^2 = x^3 + ax + b \quad (2.1)$$

where $x, y, a, b \in \mathbb{F}_p$, $p > 3$ and $\Delta = 4a^3 + 27b^2 \neq 0$.

A point P on an EC is a solution $P = (x, y)$ to the equation $E(\mathbb{F}_p)$. The set of points satisfying equation (2.1), along with the so-called point at infinity O , form the EC over the finite field \mathbb{F}_p . The total number of elements in this set, known as the cardinality, is referred to as the order of the elliptic curve E and is usually denoted by ε . For the points on the EC, two operations are defined: point doubling and point addition. In cryptographic protocols, the primary operation involving EC points is scalar multiplication, denoted as kP (also known as the kP -operation, EC point multiplication, or EC scalar multiplication). This operation can be computed through a series of point doubling ($Q' = 2P$) and point addition ($Q' = P+Q$) operations. All points on an EC over a finite field form an additive group. Therefore, the result Q' is also a point on $E(\mathbb{F}_p)$ of order ε .

An important parameter of an EC point is its order. The order q^1 of an EC point P is an integer q defined as the following: if $qG = P + \dots + P = O$ but $kP \neq O$ for all integers $1 \leq k < q$, where O is the point at infinity. If such a q exists, then P has finite order. In EC scalar multiplication, we make use of this property to generate a unique resulting point on the EC for each secret scalar k used. The EC scalar multiplication is executed to generate a private/public key pair as follows:

$$P = kG \quad (2.2)$$

¹For NIST standardized elliptic curves over prime finite field, $q=\varepsilon$.

where P and G are points on $E(\mathbb{F}_p)$ of order ε , and k is a randomly generated big binary secret scalar smaller than order q of the point G . For standardized ECs, point G is the publicly known base point with its coordinates given [7]. The private key is the secret scalar k ; the public key is the point P .

The security of ECCs, particularly the key pair generation protocol, is based on the complex mathematical problem known as the Elliptic Curve Discrete Logarithm Problem (ECDLP), that claims: for $Q = kP$, given P and Q , finding k is computationally infeasible. Thus, for the key pair generation protocol (2.2), given the public key P and the base point G , finding the private key k is computationally infeasible. The same is true for the rG operation in Elliptic Curve Digital Signature Algorithm (ECDSA)[8]:

$$R = rG \quad (2.3)$$

with publicly known EC points R and G , and a random integer r .

Both operations (2.2) and (2.3) are critical from the security point of view: The scalars k from (2.2) and r from (2.3) have to be kept secret. Note that the knowledge of r allows to calculate the private key used in the EC signature generation process.

Thus, kP operation, which is the main resource-consuming operation in elliptic-curve-based cryptosystems, is also security-critical. Secure kP algorithms have to be resistant against different attacks, particularly SCA attacks, which can reveal the scalar k through statistical or machine learning analysis of side-channel information obtained during the execution of kP algorithms. More detailed SCA attacks are described in Chapter 2.2.

2.1.1 Binary Algorithms for EC Scalar Multiplication

The binary method is a traditional scalar multiplication method based on the binary expansion of the l -bit long scalar $k = k_{l-1}k_{l-2}\dots k_2k_1k_0$, with $k_i \in \{0, 1\}$. Given the binary representation of the scalar k , the kP operation can be computed by processing the scalar k bit-by-bit, for example, from left to right using the traditional double-and-add algorithm, as shown in Algorithm 1, taken from [6].

Algorithm 1: Left-to-right binary double-and-add kP algorithm

```

Input:  $k = (k_{l-1}, \dots, k_0)_2, P \in E(\mathbb{F}_p)$ 
Output:  $kP$ 
1  $Q = \infty$                                 // Point at infinity  $O$ 
2 for  $i = (l - 1)$  downto 0 do
3    $Q = 2Q$                                 // Point doubling
4   if  $k_i = 1$  then
5      $Q = Q + P$                           // Point addition
6 return  $Q$ 
```

Numerous other binary kP algorithms exist, as well as kP algorithms that utilize different scalar coding methods, such as the Window- w Multibase Non-Adjacent Form ($wmbNAF$) proposed in [2]. In this work, we focus on the binary left-to-right kP algorithm implemented for the P-256 curve, which is an EC over prime finite field and currently recommended for use for ECDSA and EC key establishment by NIST [7, 9]. For this curve, all mathematical operations involving point arithmetic

(i.e., point doublings and point additions in Algorithm 1) use elements of the prime finite field \mathbb{F}_p , with the Mersenne prime $p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$ [10].

Point doublings and point additions can be represented as sequences of mathematical operations, whereby the sequence of the operations for a point doubling differs from the one for a point addition. In addition to mathematical operations, each point doubling and point addition operation also requires storing and reading data to and from registers. The sequence of these operations depends on the processed scalar k (further also denoted as key), whereby the sequence corresponding to the processing of a key bit value ‘1’ differs from the sequence corresponding to the processing of a key bit value ‘0’. This can be used to reveal the processed scalar k by performing various attacks.

2.2 Side-Channel Analysis Attacks

Side-Channel Analysis (SCA) is a security exploit which aims to extract secrets by analysing physical parameters (e.g., power consumption, electromagnetic emanation and timing, etc.) measured during the execution of a cryptographic algorithm running on a physical device. A trace refers to a set of side-channel information (e.g., power consumption, electromagnetic emanation or timing) measurements taken during a cryptographic operation. Depending on the number of traces needed for the analysis, the analysis can be classified into two categories: horizontal SCA and vertical SCA. Horizontal SCA utilizes only one single trace on a single algorithm execution to extract sensitive information. Whereas vertical SCA requires several² traces to perform the statistical analysis. Key randomization, EC point blinding and randomization of EC point coordinates proposed by Coron [11] are some effective countermeasures against vertical SCA. However, these countermeasures are ineffective against horizontal SCA [12, 13].

Depending on the analysis method, SCA attacks can be classified into two categories: Simple Side-Channel Analysis (SSCA) and Differential Side-Channel Analysis (DSCA). SSCA is a technique that uses a trace for direct interpretation of the behavior to infer the secret key. In contrast, DSCA exploits side-channel information leakages by using statistical or clustering methods across many hundreds or even millions of traces to deduce the processed key bits. It is important to note that various statistical and machine learning methods can also be applied to analyse a single kP trace, for example [12, 14-16]. The differences in the shape of the measured trace(s) are searched and analysed using statistical or machine learning methods. The similarities in small parts of traces are usually searched using correlation power analysis, e.g., Pearson’s correlation coefficient.

2.3 Vulnerabilities of kP Implementations to SCA

The shape of each measured side-channel trace of a kP execution depends on the processed input data, the scalar k , the kP algorithm implementation being attacked, target frequency, and the target platform or device, i.e., the technology used for the circuit implementation and the optimization options applied for the design synthesis [17].

Simple SCA attacks on a kP implementation exploit differences in kP trace parts corresponding to the processing of ‘0’ and ‘1’ values of k , allowing attackers to reveal

²At least two.

the key k . Primarily, binary kP algorithms are vulnerable due to the distinguishability of sequences of mathematical operations for point doubling and point addition. If these point operations' power shapes are distinguishable, attackers can infer the processed bits of k from the sequence of these operations, revealing the private key.

To counteract SCA, the power shapes of the processing of key bit value '0' should be indistinguishable from that of '1' through regularization or randomization. In another word, an SCA-resistant algorithm aims to generate either very similar shapes for the processing of both '0' and '1', or totally random-looking shapes. The double-and-add-always algorithm proposed in 1999 by Coron [11] is a regular kP algorithm performing a $2P$ and a $P + Q$ for each bit, but it is inefficient due to dummy operations [6]. Atomic pattern algorithms, introduced in [1], are a faster and more energy-efficient kind of kP algorithms with a reduced number of dummy operations. Different algorithmic techniques can further increase the resistance of kP implementations by removing, reducing or hiding the algorithm's dependency on k . Examples include key masking or blinding, and randomizing steps within the kP algorithm.

In summary, the distinguishability of processed key bits arises from:

1. Differences in sequences of mathematical operations ($2P$ vs. $P + Q$), resulting in different shapes of profiles for the computations.

Existing countermeasures:

- Dummy EC point addition operations: double-and-add-always [6]
- Universal formulae for $2P$ and $P + Q$ calculations using the same sequence of field operations [18]
- Atomicity, e.g., [1]

2. Key-dependent data processing differences for $2P$ vs. $P + Q$ (data-bit vulnerability), usually exploited by vertical SCA.

Algorithmic data randomization techniques [11] are well-known effective countermeasures:

- Key randomization
- Projective coordinates randomization
- Point blinding

3. Key-dependent differences in the addressing of different blocks/registers in $2P$ vs. $P + Q$ (address-bit vulnerability), usually exploited by vertical SCA [19]. Algorithmic address randomization [19, 20] is a known effective countermeasure, but it is not effective against horizontal attacks according to [21].

Binary kP algorithms, such as regular Montgomery ladder using Lopez-Dahab projective coordinates [22], atomic pattern algorithms [23] or even Montgomery ladder with randomized processing of point operations utilizing a universal point addition formula [24], are vulnerable to horizontal address-bit SCA [25-27], at least when implemented in hardware.

This work only focuses on atomic pattern algorithms [2] as SSCA countermeasures for embedded devices. The goal is to investigate if the atomic patterns are vulnerable to SSCA and which kind of vulnerability - data-bit or address-bit - is dominant.

2.4 Atomicity as a Countermeasure Against SSCA

Numerous proposals have been put forward to prevent SSCA. The side-channel atomicity principle introduced in 2004 by Chevallier-Mames et al. [1] aims to counteract SSCA and is especially suitable for embedded devices, due to the significantly reduced computation burden in comparison to traditional double-and-add-always kP algorithms. Extensive research have been dedicated to enhancing this countermeasure by Longa [2] and by Giraud and Verneuil [28], who proposed new – optimized – atomic pattern algorithms to improve the computation efficiency of scalar multiplication.

In the atomicity principle, each EC point operation is viewed as a process, whereby a process (e.g., EC point doubling) is viewed as a set of blocks consisting of the same sequence of instructions. Two instruction sequences are said to be side-channel equivalent if they are indistinguishable through side-channel analysis. Each of these side-channel equivalent instruction sequences is called a common side-channel atomic block, and a series of these atomic blocks forms an atomic pattern for an EC operation. Each complex calculation process (e.g., $2P$ or $P + Q$ operation) can then be expressed as a repetition of the side-channel atomic blocks represented as a repetition of the same atomic patterns, which carries a regular sequence of basic field operations.

Side-channel atomicity involves careful design of cryptographic operations implementation such that individual operations (e.g., point doubling and point addition) are side-channel equivalent, which means upon execution, it is impossible to distinguish one operation from the other or even to identify the start and end of these operations. Note that this principle is based on the assumption that the loading/storing of data from/to different registers are equivalent operations [18].

Since this work focuses on investigating the potential vulnerabilities of the atomic pattern algorithms proposed by Longa [2], we specifically selected the MNAMNAA-based atomic pattern algorithms to discuss in greater detail in this thesis.

Table 2.1 shows the atomic point doubling algorithm, which uses standard Jacobian coordinates. Table 2.2 presents the atomic mixed point addition algorithm, which uses a mix of Jacobian and affine coordinates.

Input: $P = (X_1, Y_1, Z_1)$
Output: $2P = (X_3, Y_3, Z_3) = (T_1, T_2, T_3)$
 $T_1 \leftarrow X_1, T_2 \leftarrow Y_1, T_3 \leftarrow Z_1$

	$\Delta 1$	$\Delta 2$	$\Delta 3$	$\Delta 4$
M	$T_4 = T_3^2$	$T_5 = T_4 \times T_5$	$T_5 = T_4^2$	$T_2 = T_2^2$
N	*	*	*	$T_5 = -T_1$
A	$T_5 = T_1 + T_4$	$T_4 = T_5 + T_5$	$T_6 = T_2 + T_2$	$T_5 = T_5 + T_6$
M	$T_6 = T_2^2$	$T_3 = T_2 \times T_3$	$T_6 = T_1 \times T_6$	$T_5 = T_4 \times T_5$
N	$T_4 = -T_4$	*	$T_1 = -T_6$	$T_2 = -T_2$
A	$T_2 = T_2 + T_2$	$T_4 = T_4 + T_5$	$T_1 = T_1 + T_1$	$T_2 = T_2 + T_2$
A	$T_4 = T_1 + T_4$	$T_2 = T_6 + T_6$	$T_1 = T_1 + T_5$	$T_2 = T_2 + T_4$

TABLE 2.1: MNAMNAA-based atomic point doubling, taken from [2].

Input: $\mathbf{P} = (X_1, Y_1, Z_1)$ and $\mathbf{Q} = (X_2, Y_2)$
Output: $\mathbf{P} + \mathbf{Q} = (X_3, Y_3, Z_3, X'_1, Y'_1) = (T_1, T_2, T_3, T_4, T_5)$
 $T_1 \leftarrow X_1, T_2 \leftarrow Y_1, T_3 \leftarrow Z_1, T_x \leftarrow X_2, T_y \leftarrow Y_2$

	$\Delta 1$	$\Delta 2$	$\Delta 3$
M	$T_4 = T_3^2$	$T_6 = T_5^2$	$T_9 = T_5 \times T_6$
N	*	*	*
A	*	*	$T_8 = T_8 + T_9$
M	$T_5 = T_x \times T_4$	$T_7 = T_1 \times T_6$	$T_4 = T_3 \times T_4$
N	$T_6 = -T_1$	*	*
A	$T_5 = T_5 + T_6$	$T_8 = T_1 + T_1$	*
A	*	*	*
	$\Delta 4$	$\Delta 5$	$\Delta 6$
M	$T_4 = T_y \times T_4$	$T_8 = T_2 \times T_9$	$T_3 = T_3 \times T_5$
N	$T_{10} = -T_2$	$T_6 = -T_1$	$T_4 = -T_7$
A	$T_4 = T_4 + T_{10}$	$T_6 = T_6 + T_7$	$T_4 = T_1 + T_4$
M	$T_{10} = T_4^2$	$T_6 = T_6 \times T_{10}$	$T_5 = T_4^2$
N	$T_8 = -T_8$	$T_9 = -T_8$	$T_6 = -T_8$
A	$T_1 = T_6 + T_8$	$T_2 = T_6 + T_9$	$T_6 = T_2 + T_6$
A	*	*	*

TABLE 2.2: MNAMNAA-based atomic point addition, taken from [2].

Each Δi with $i \in \{1, 2, 3, 4, 5, 6\}$ in Table 2.1 and Table 2.2 represents an atomic block. Within an atomic block, each line contains one field operation. In this case, it is either a multiplication (M), a negation (N) or an addition (A). Each * represents a dummy field operation conforming to the atomic pattern as a disguise, but the dummy operations are unnecessary for the overall result calculation. Following the atomic pattern MNAMNAA, point doubling can be computed in 4 atomic blocks, and point addition in 6 atomic blocks. Attackers can then only observe a sequence of side-channel atomic block executions and are unable to distinguish the underlying operations using SSCA, since side-channel information like timing, power consumption and memory access patterns are theoretically equivalent for every atomic block.

2.5 Vulnerabilities of the Atomic Patterns

2.5.1 Key-dependent Data Processing

Key-dependent differences in the shape of measured traces can arise not only from different sequences of instructions but also from processing different data within the same instruction or operation. These differences are typically small and dependent on the Hamming distance of the data processed. Various statistical methods, such as correlation analysis, can exploit these differences to reveal the key. This is often referred to as exploiting data-dependent vulnerabilities through data-bit SCA. While data-dependent vulnerabilities are usually exploited in vertical attacks, there are known successful horizontal attacks against kP implementations that also leverage data-dependent vulnerabilities.

Horizontal Collision Correlation Analysis (HCCA) [12] is an attack that exploits data-bit vulnerabilities on elliptic curves atomic implementations with input randomization analysing a single kP execution trace. The main assumption is that the adversary can detect when two field multiplications have at least one operand in

common or no shared operand at all. The HCCA attack can be used to detect distinguishing property of sharing of operands among EC point doubling and point addition operations, even if they are SCA-protected by applying atomicity principle. It can thus reveal the underlying secret key bit successively and expose the entire secret key. Bauer et al.'s work [12] demonstrated through experiments that the atomic pattern proposed by Chevallier et al. in [1] is vulnerable to HCCA. In addition, they pointed out that Longa's MANA atomic pattern [2] and Giraud and Verneuil's atomic pattern [28] are theoretically vulnerable to HCCA too, due to their similarities to Chevallier et al.'s atomic pattern.

2.5.2 Key-dependent Addressing of Registers

An address-bit vulnerability in devices performing cryptographic operations arises from the unintended leakage of information through observable memory access patterns. For instance, an attacker can analyse the power consumption associated with key-dependent addressing registers to gain insights into cryptographic keys or other confidential data. This is possible because the addresses of different registers induce varying bus activity for the connection of the addressed blocks, particularly in hardware implementations. Consequently, this can serve as a source of side-channel leakage.

The power consumption or electromagnetic emanation during a kP execution fluctuates based on the executed operations and the Hamming distances of data transmitted within the device. Additionally, this also relates to the Hamming distances of the registers' or blocks' addresses used during the kP execution. The address-bit vulnerability stems from the requirement that a register must have an address assigned to it and has to be "called" before data can be loaded into or from it. If different register addresses are used within the same instruction sequence, the power consumption or electromagnetic emanation of the device will vary according to the differences in the Hamming weights of these register addresses. This dependency is indirect and often challenging to measure.

In 1999, the side-channel analysis attack - address-bit Differential Power Analysis (DPA) was first introduced by Messerges et al. [29]. The correlation between the addresses of registers and the secret information was analysed, using hundreds of power traces of a single encryption operation³. They claimed that the horizontal address-bit SCA attack is powerful in revealing bit transition information. However, due to the lack of concrete results in their experiments, the address-bit DPA did not attract significant attention. Then, Itoh et al. [19] extended address-bit DPA to ECC design implementing Montgomery ladder, showing that the multiple-trace attack is successful when a small number of registers are used in the algorithm. Further research by Kabin et al. [26] proposed that the addressing of source and destination registers or blocks going through the bus in a kP hardware implementation can be distinguishable using horizontal SCA attacks, i.e., analysing only a single kP execution trace. In 2023, Kabin [3] described successful horizontal address-bit SCA attacks against hardware accelerators of kP algorithms implementing highly regular Montgomery ladder as well as an atomic pattern kP algorithm.

³The same inputs were used every time, without input randomization.

Chapter 3

State-of-the-art Literature Overview

We carried out the literature overview to understand the progress made by researchers over the years regarding the atomic pattern algorithms proposed in [2], to validate our initial idea that no one before us has ever done a SCA on an implementation of Longa's MNAMNAA-based atomic pattern algorithm on an embedded device. Also, we used the literature overview to assess if there was any previous work we could compare our thesis's analysis results with, to help us draw more comprehensive conclusions.

Using Google Scholar and Semantic Scholar as search engines at the end of 2023, we reviewed all of the known literature which cited Longa's master's thesis [2]. There are 75 literature cited [2] reported by Google Scholar and 64 reported by Semantic Scholar, summing to 84 distinct records in 4 languages altogether. However, only 63 out of the 84 records are nonduplicate, available to view and have truly cited [2]. From Figure 3.1, we can see that [2] demonstrated certain influence on others' research activities over the past 15 years continually.

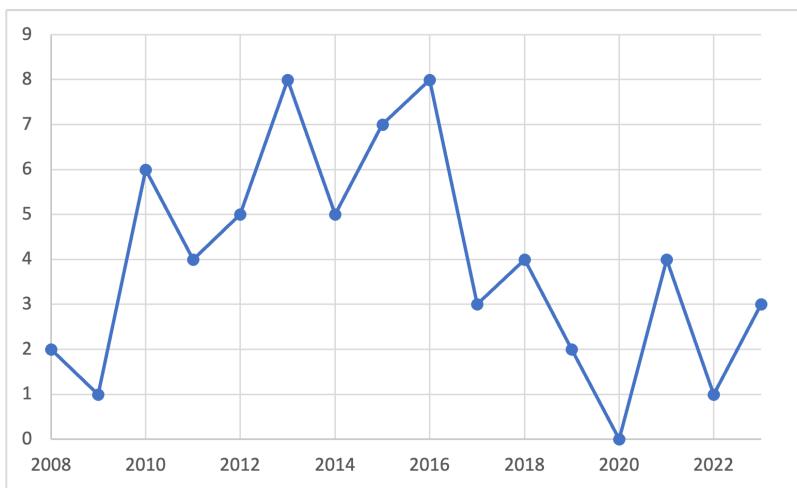


FIGURE 3.1: Number of published papers that referenced [2] over the years.

We have studied all these 63 reference papers [3, 12, 21, 23, 27, 28, 30–86] and classified them corresponding to the context, in which the atomic pattern algorithm [2] was referred.

The themes of reference papers citing [2] include:

- SCA and/or SCA countermeasures,

- EC kP algorithm efficiency optimization or analysis without any modifications of the atomic patterns,
- recoding algorithms, and
- other miscellaneous topics

The results of the literature investigation are represented in Chapter 8.1 (see Table 8.1). Figure 3.2 represents the results of the literature investigation graphically.

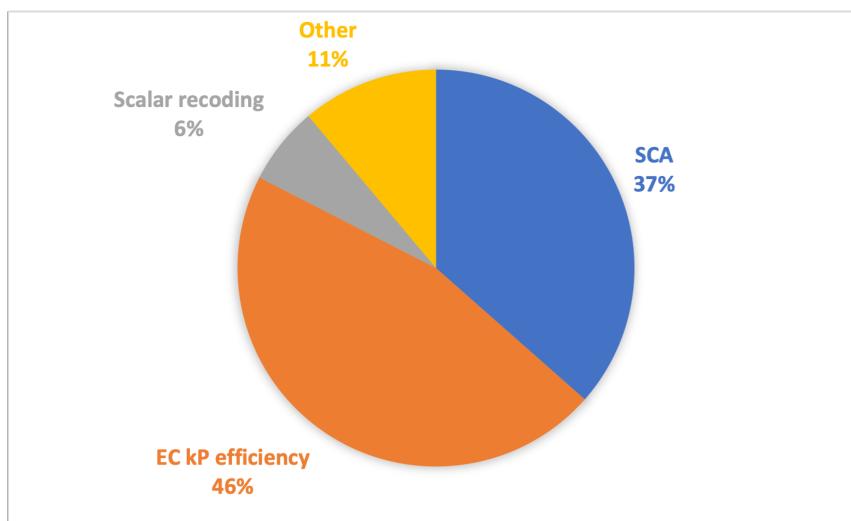


FIGURE 3.2: Distribution of themes of literature referencing [2].

As shown in Figure 3.2, of all the reference papers citing [2], 23 papers, i.e., 37%, are about SCA and/or SCA countermeasure [3, 12, 21, 23, 27, 28, 30–34, 37–40, 77–83]. It prompted us to further examine the improvements, implementations and analysis made based on Longa's atomic pattern algorithms. 15 papers [3, 12, 21, 23, 27, 28, 30–34, 37–40] out of 23 mentioned [2] as an example of the state-of-the-art countermeasure against SSCA. In the following section of this thesis, we will delve into the enhancement of SCA resistance and the implementation of atomicity as a SCA countermeasure, specifically focusing on Longa's [2] atomic patterns. We selected 15 out of 23 papers from the category "SCA" (see Figure 3.2) for a detailed overview as they are highly relevant to this work.

3.1 Analysis of MNAMNAA-based Algorithms

In this subsection, we summarize the reference papers [3, 12, 21, 23, 27, 28, 30–40] that cited Longa's atomic pattern algorithms as state-of-the-art, used it as a basis for improvement, and/or those which analysed an atomic patterns algorithm.

In 2010, Giraud and Verneuil [28] claimed that Longa's atomic patterns cannot defeat DSCA, as the DSCA countermeasures - projective coordinates randomization or random curve cannot be applied on his atomic pattern algorithms. They improved Longa's algorithms by introducing new fast doubling, general doubling and readdition algorithms using Longa's atomic patterns MNAMNAA, SNAMNAA and MNAMNAA, respectively. By maximizing the use of field squarings to replace multiplications, and minimizing the use of field additions and negations, they proposed a right-to-left mixed scalar multiplication algorithm. It was proven to be 10% more

efficient than any previous methods at the time. Later in 2012, Verneuil published his PhD thesis [31] covering the same contributions he made in [28].

Lu et al. [30] introduced a general framework of side-channel atomicity to support the proposed τ -scalar, ξ -base representation for scalar multiplication, which has shown improved resistance to simple power analysis and supports Longa's single scalar multiplication algorithms. They used AMD Athlon X2 245-based hardware to test SPA-resistance on a 2-scalar multiplication algorithm. It shows that the implementation is SPA-resistant but with overheads. However, Lu et al. only implemented Chevallier-Mames et al.'s atomic pattern "MNAA" [1] due to hardware space limitation. They claimed that their algorithm hypothetically should be able to accommodate Longa's atomic pattern without changing its effectiveness.

In 2013, Rondepierre [23] proposed a new atomic pattern for EC point doubling and point addition operations for double scalar multiplications, which can be adapted to process single scalar multiplication too. He used the Straus-Shamir trick and scalar recoding to optimize the efficiency of his atomic pattern algorithms. He showed that the implementation of his single scalar multiplication algorithm on a smart card has an efficiency improvement of 40% when compared to Giraud and Verneuil's algorithm implementation. Although he only showed memory cost reduction by optimizing intermediate registers used in his pattern and Giraud and Verneuil's pattern, he expected that this optimization can also be applied on Longa's pattern.

Bauer et al. [12] introduced HCCA, and performed a theoretical analysis on Longa's atomic pattern point doubling and point addition algorithms. They pointed out that Longa's algorithms should be susceptible to HCCA. However, they only implemented Chevallier-Mames et al.'s atomic pattern scheme [1] but not Longa's in their experiments. They demonstrated that HCCA against Chevallier-Mames et al.'s atomic pattern algorithm works for 8- and 32-bit microprocessors and elliptic curves of size 160, 250 and 384 bits.

Das et al. [32] recalled the proposition made by Bauer et al. in [12] that Longa's atomic pattern kP algorithm is vulnerable to HCCA, since the operand-sharing property among field operations can be detected. They observed that side-channel leakage between two field multiplications can be quantified by computing the distance between two leakages using Pearson correlation coefficient. In addition, they proposed a new protected atomic pattern to demonstrate how to make Giraud and Verneuil's right-to-left atomic scheme safe against HCCA and the improved Big Mac attack [87], with a small overhead.

Kabin et al. [33, 34] implemented the atomic pattern from [23] in hardware, performed a horizontal SCA attack (adapting the technique *comparison to the mean* to atomic patterns) and mentioned that all atomic pattern algorithms may be vulnerable to horizontal address-bit SCAs, at least when implemented in hardware.

Longa [35] evaluated the costs of point doubling and doubling-addition from [2] in three different processors, and optimized the kP computing costs in the proposed Longa-Miri precomputation (LM) scheme.

Faye [36] evaluated Longa's special addition and point tripling to provide a summary of state-of-the-art algorithms for EC scalar multiplication in terms of efficiency. Also, he studied parallel calculation of kP using Longa's point doubling algorithm on a 3-processor architecture theoretically.

Venelli [37], Houssain [38], Houssain et al. [39] and Lo'Ai et al. [40] all mentioned Longa's atomic patterns as state-of-the-art SCA countermeasure, but their works only serve as review papers without any implementation or improvement on the atomic pattern algorithms.

Kabin [3], Kabin et al. [21] and Sigourou et al. [27] mentioned Longa's atomic patterns as state-of-the-art SCA countermeasure, performed successful SSCA against an atomic pattern kP algorithm (see [3, 21]) or Montgomery ladder (see [27]) in their works, but did not implement Longa's atomic pattern algorithms.

Author(s)	Year	Contributions
Giraud and Verneuil [28]	2010	<ul style="list-style-type: none"> Claimed that Longa's atomic patterns cannot defeat DSCA. Proposed new composite operations using Longa's atomic patterns. Proposed right-to-left mixed scalar multiplication algorithm.
Verneuil [31]	2012	<ul style="list-style-type: none"> Covered the same contributions he made in [28].
Lu et al. [30]	2013	<ul style="list-style-type: none"> Introduced a general framework of kP side-channel atomicity. Experimented with AMD Athlon X2 245-based hardware. Only implemented the atomic pattern "MNAA". Their algorithm hypothetically can use Longa's atomic pattern.
Rondeepierre [23]	2014	<ul style="list-style-type: none"> Proposed a new atomic pattern for scalar multiplications. Used Straus-Shamir trick and scalar recoding to optimize algorithms. Implemented single scalar multiplication algorithm on a smart card. Showed memory cost reduction by optimizing intermediate registers.
Bauer et al. [12]	2015	<ul style="list-style-type: none"> Introduced HCCA. Performed theoretical analysis on Longa's atomic pattern algorithms. Claimed Longa's algorithms should be susceptible to SCCA. Only implemented Chevallier-Mames et al.'s atomic pattern scheme[1]. Demonstrated that HCCA against atomic pattern algorithm[1] works.
Das et al. [32]	2016	<ul style="list-style-type: none"> Recalled that Longa's atomic pattern algorithm is vulnerable to HCCA. Quantified side-channel leakage between two field multiplications. Proposed a new atomic pattern to safeguard the atomic scheme[28].
Kabin et al. [33, 34]	2021	<ul style="list-style-type: none"> Implemented the atomic pattern from [23] in hardware. Performed horizontal SCA attack. All atomic pattern algorithms may be vulnerable to their attack.

Author(s)	Year	Contributions
Longa [35]	2011	<ul style="list-style-type: none"> Evaluated the costs of $2P$ and $2P + Q$ from [2] in 3 different processors. Optimized the kP computing costs in the proposed LM scheme.
Faye [36]	2014	<ul style="list-style-type: none"> Evaluated Longa's special addition and point tripling. Summarized state-of-the-art kP algorithms in terms of efficiency. Studied parallel calculation of kP using Longa's $2P$ algorithm.
Venelli [37], Houssain [38], Houssain et al. [39] and Lo'Ai et al. [40]	2011- 2016	<ul style="list-style-type: none"> Reviewed numerous papers related to SCA. Cited Longa's atomic patterns as state-of-the-art SCA countermeasure.
Kabin [3], Kabin et al. [21] and Sigourou et al. [27]	2023	<ul style="list-style-type: none"> Cited Longa's atomic patterns as state-of-the-art SCA countermeasure. Performed successful SSCA against atomic pattern kP algorithm [3, 27] or against Montgomery ladder [21].

TABLE 3.1: Summary of literature citing Longa's atomic patterns as state-of-the-art, analysing or improving Longa's atomic patterns.

We found that none of the papers listed in Table 3.1 mentioned or reported about the implementation of Longa's atomic pattern algorithms. Thus, the indistinguishability of Longa's atomic pattern was never investigated practically. The paper which is the most suitable for the comparison to our experiments is [27], in which the authors implemented an atomic pattern kP algorithm for an embedded device and performed SSCA analysing a measured electromagnetic trace.

Chapter 4

Board for Experiments

As the target platform for our experiments, we used a Texas Instruments LAUNCHXL-F28379D LaunchPad. It is a development kit that provides on-board emulation for programming and debugging, along with a TMS320F28379D C2000 32-bit dual-core microcontroller as our target device. The microcontroller has 1 MB flash memory, 204 kB random-access memory (RAM) and operates real-time maximum at 200 MHz. Figure 4.1 shows the board attacked.

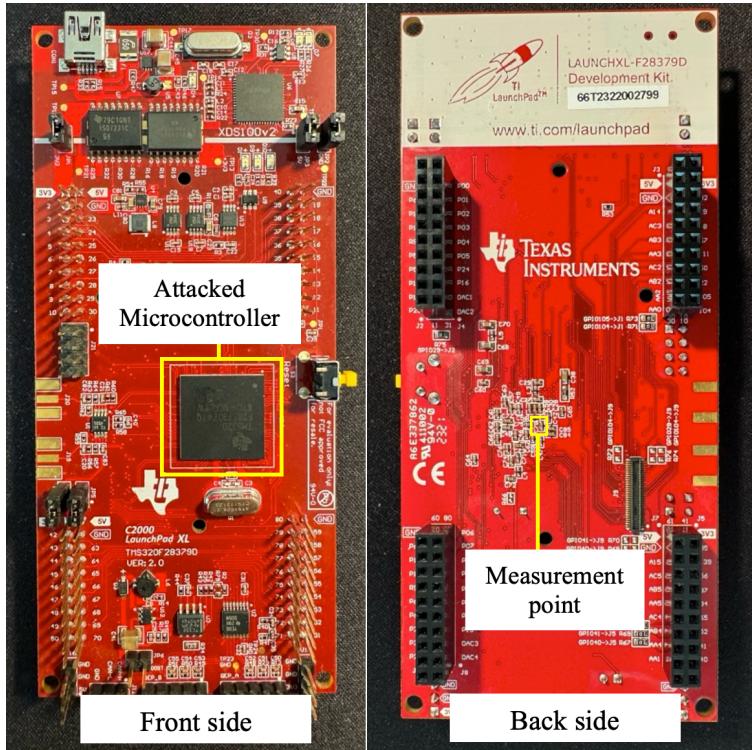


FIGURE 4.1: Front side and back side of the board attacked.

In Figure 4.1, we can see the attacked microcontroller on the front side of the board, and our measurement point at the back of the microcontroller as indicated. More details for the measurement setup will be given in Chapter 6.2.

We used Code Composer Studio (CCS) version 12.4.0.00007 as our integrated development environment (IDE) for the microcontroller. It supports compilation of C and C++.

Since the target device is resource constrained, we need to find a suitable cryptographic library to implement the atomic pattern kP algorithm [2] such that the program is portable to the target device, among other criteria.

Chapter 5

Selection of a Suitable Open-source Cryptographic Library

A recent work by Sigourou et al. [27] describes the implementation of the atomic pattern kP algorithm from Rondepierre [23]. They have explored the five most popular open-source cryptographic libraries at the time, namely OpenSSL [88], MIRACL [89], Crypto++ [90], Cryptlib [91] and FLECC in C (FLECC) [92], regarding their suitability for implementation of ECC protocols (e.g., ECDSA, ECDH) on resource-constrained devices. Their study shows that FLECC is the best suited library among others. They considered each library's suitability for embedded devices in terms of source code language used, portability to the attacked device in terms of memory size, access to modular arithmetic functions, and presence of constant-runtime functions. Their study used FLECC for the EC kP implementation of Rondepierre's atomic pattern [23] on a LAUNCHXL-F280025C 32-bit low-cost real-time microcontroller board with 128 kB flash memory and 24 kB RAM. However, even though some modular arithmetic functions from FLECC library are claimed to be constant-runtime, the authors of [27] observed that these functions are not truly time-constant at hardware level.

Our work focuses on examining open-source cryptographic libraries to find the libraries which might truly have constant-runtime functions we need at hardware level, for the implementation of our chosen atomic pattern algorithms as a counter-measure against timing attacks on our target device, which is similar to but not the same as the one used in [27]. Since the study results on cryptographic libraries in [27] state that FLECC fulfilled all the criteria for the implementation of atomic kP algorithm on their resource-constrained device, we examined this library to see if our kP implementation developed using it can be run in our environment and our target device. Additionally, we examined the suitability of using two other libraries – Botan [93] and Mbed TLS [94] – which also have constant-runtime functions according to a study by Jancar et al. [95]. Table 5.1 shows the investigation results of the cryptographic libraries we looked into. Eventually, we selected FLECC for the algorithm implementation on our board. The reasons behind will be elaborated in the subsections.

Library	Modular arithmetic functions	Constant-runtime modular arithmetic functions	Suitable for our device	Portable to our board
FLECC in C	✓	✓	✓	✓
Mbed TLS	✓	None	✓	Not checked
Botan	✓	✗ Negation ✗ Multiplication	✓	Not checked

TABLE 5.1: Summary of cryptographic libraries studied.

5.1 FLECC in C

FLECC in C (FLECC) [92] is a C library supporting constant-runtime modular arithmetic functions. However, it does not have a single function supporting the modular multiplication of two numbers (i.e., $a \cdot b \bmod p$) in constant time, but the available constant-time Montgomery modular multiplication can be applied twice for a single field product calculation. Efficiency is not our concern in this thesis, as we only focus on studying the SCA-resistance provided by atomicity. Thus, we selected FLECC for our implementation on hardware, as according to the library description¹, it provides all the constant-time functions we needed for the algorithm implementation. We used the first version of FLECC and CCS version 12.4.0.00007 for the implementation.

5.2 Mbed TLS

Mbed TLS [94] is a C library with modular arithmetic functions available and with some constant-runtime functions. In this thesis, we looked into Mbed TLS version 3.5.0. However, none of its modular arithmetic functions is declared as constant-runtime. Hence, this library was not selected for our implementation.

5.3 Botan

Botan [93] is a C++ library with modular arithmetic functions available and with a range of constant-runtime functions. We looked into Botan version 3.1.1 and found that the suitable constant-time functions provided are big integers modulo operation, big integers addition and Montgomery modular reduction. But it does not offer any constant-time field element multiplication or negation operation which are necessary for the implementation of our chosen atomic pattern [2]. Furthermore, it requires C++ 11 standard for compilation, which our CCS version does not support. Therefore, it was not selected for our implementation.

¹https://github.com/IAIK/flecc_in_c/blob/develop/src/gfp/gfp_const_runtime.c

Chapter 6

Investigations

6.1 Implemented Atomic Patterns kP Algorithm

We implemented Longa's [2] MNAMNAA atomic patterns for EC point doubling (PD) and EC point addition (PA) computations required in binary kP algorithms, as mentioned in Chapter 2. To investigate the distinguishability of the atomic patterns, we used them in the binary left-to-right kP algorithm corresponding to Algorithm 2, which differs from Algorithm 1 in its initialization phase (see line 1).

Algorithm 2: Modified left-to-right binary double-and-add kP algorithm

Input: $k = (k_{l-1}, \dots, k_0)_2, P \in E(\mathbb{F}_p)$
Output: kP

```

1  $Q = P$ 
2 for  $i = (l - 2)$  downto 0 do
3    $Q = 2Q$                                 // Point doubling atomic pattern (4 atomic blocks)
4   if  $k_i = 1$  then
5      $Q = Q + P$                           // Point addition atomic pattern (6 atomic blocks)
6 return  $Q$ 
```

Our point doubling implementation follows formula (2.13) from [2]:

$$X_3 = \alpha^2 - 2\beta, Y_3 = \alpha(\beta - X_3) - 8Y_1^4, Z_3 = 2Y_1Z_1 \quad (6.1)$$

where $\alpha = 3X_1^2 + aZ_1^4, \beta = 4X_1Y_1^2$

During the implementation and functionality tests, we found four erroneous registers used in Longa's MNAMNAA atomic patterns point doubling and point addition [2]. We examined the correspondence of the atomic patterns to formulae proposed in [2] and corrected the atomic patterns correspondingly. The corrected atomic patterns are represented in Table 6.1 and Table 6.2.

Input: $P = (X_1, Y_1, Z_1)$
Output: $2P = (X_3, Y_3, Z_3) = (T_1, T_2, T_3)$
 $T_1 \leftarrow X_1, T_2 \leftarrow Y_1, T_3 \leftarrow Z_1$

	$\Delta 1$	$\Delta 2$	$\Delta 3$	$\Delta 4$
M	$T_4 = T_3^2$	$T_5 = T_4 \cdot T_5$	$T_5 = T_4^2$	$T_2 = T_2^2$
N	*	*	*	$T_5 = -T_1$
A	$T_5 = T_1 + T_4$	$T_4 = T_5 + T_5$	$T_6 = T_2 + T_2$	$T_5 = T_5 + T_6$
M	$T_6 = T_2^2$	$T_3 = T_2 \cdot T_3$	$T_6 = T_1 \cdot T_6$	$T_5 = T_4 \cdot T_5$
N	$T_4 = -T_4$	*	$T_1 = -T_6$	$T_2 = -T_2$
A	$T_2 = T_2 + T_2$	$T_4 = T_4 + T_5$	$T_1 = T_1 + T_1$	$T_2 = T_2 + T_2$
A	$T_4 = T_1 + T_4$	$T_2 = T_6 + T_6$	$T_1 = T_1 + T_5$	$T_2 = T_2 + \mathbf{T}_5$

TABLE 6.1: MNAMNAA atomic patterns sequence for EC point doubling implemented in this work.

Input: $P = (X_1, Y_1, Z_1)$ and $Q = (X_2, Y_2)$
Output: $P + Q = (X_3, Y_3, Z_3, X'_1, Y'_1) = (T_1, T_2, T_3, T_4, T_5)$
 $T_1 \leftarrow X_1, T_2 \leftarrow Y_1, T_3 \leftarrow Z_1, T_x \leftarrow X_2, T_y \leftarrow Y_2$

	$\Delta 1$	$\Delta 2$	$\Delta 3$
M	$T_4 = T_3^2$	$T_6 = T_5^2$	$T_9 = T_5 \cdot T_6$
N	*	*	*
A	*	*	$T_8 = T_8 + T_9$
M	$T_5 = T_x \cdot T_4$	$T_7 = T_1 \cdot T_6$	$T_4 = T_3 \cdot T_4$
N	$T_6 = -T_1$	*	*
A	$T_5 = T_5 + T_6$	$T_8 = \mathbf{T}_7 + \mathbf{T}_7$	*
A	*	*	*

	$\Delta 4$	$\Delta 5$	$\Delta 6$
M	$T_4 = T_y \cdot T_4$	$T_8 = T_2 \cdot T_9$	$T_3 = T_3 \cdot T_5$
N	$T_{10} = -T_2$	$T_6 = -T_1$	$T_4 = -T_7$
A	$T_4 = T_4 + T_{10}$	$T_6 = T_6 + T_7$	$T_4 = T_1 + T_4$
M	$T_{10} = T_4^2$	$T_6 = T_6 \cdot \mathbf{T}_4$	$T_5 = T_4^2$
N	$T_8 = -T_8$	$T_9 = -T_8$	$T_6 = -T_8$
A	$T_1 = \mathbf{T}_{10} + T_8$	$T_2 = T_6 + T_9$	$T_6 = T_2 + T_6$
A	*	*	*

TABLE 6.2: MNAMNAA atomic patterns sequence for EC point addition implemented in this work.

The corrections are shown in bold with yellow highlight. The corrected patterns for EC operations were then verified to be able to produce correct results in point doubling and point addition by being implemented additionally as software code.

In this work, we examined the correctness of all the other atomic patterns proposed in [2]. The corrected patterns are presented in Chapter 8.2.

Accordingly, we implemented point doubling and point addition operations corresponding to Longa's MNAMNAA atomic patterns as shown in Table 6.1 and Table 6.2 using the open-source library FLECC to form the core part of kP computation of Algorithm 2. The implementation was done for the secp256r1 elliptic curve in FLECC, which is the EC denoted as P-256 by NIST [7]. For the calculation of a single field product, we always used the constant-runtime Montgomery modular multiplication function available in FLECC twice, similar to the implementation in [27].

Montgomery modular multiplication [96] works by first transforming the multipliers into Montgomery space, where modular multiplication can be performed inexpensively, and then transforming them back when their actual values are needed. The Montgomery space is defined by the modulo n and a positive integer $R \geq n$ coprime to n . The Montgomery multiplication based on Separated Operand Scanning (SOS) method [97] provided by FLECC library that we use involves modulo and division by R , where $R = 2^{sw}$, w is the computer's word size in bits and s is the number of words required to represent a large prime. In our case, we use the default values from the library $R = 2^{(256)(8)} = 2^{2048}$ and n^1 . In order to implement the field product calculation as a constant-runtime function, we have to execute two Montgomery modular multiplications for each finite field multiplication in Longa's atomic patterns. We denote here a single Montgomery modular multiplication function as "X". This approach highly deteriorates the performance of the original design of the atomic pattern MNAMNAA, as the atomic pattern now becomes XXNAXXNAA, with doubled the number of multiplications, which are the most computationally expensive operations among other operations used. Using the FLECC library, a multiplication ($c = a \cdot b \bmod p$) will be implemented as the following two steps: $c = a \cdot b \cdot R^{-1}$ and $c = c \cdot R^2 \cdot R^{-1}$. All implementation steps are shown in Table 6.3 in details. The implemented *kP* algorithm can be found in [98].

¹ $n = 0xFFFFFFFF\ 00000000\ FFFFFFFF\ FFFFFFFF\ BCE6FAAD\ A7179E84\ F3B9CAC2\ FC632551$, the same number as modulo p .

Atomic blocks	Operations	EC Point Doubling Input: $P = (X_1, Y_1, Z_1)$ Output: $2P = (X_3, Y_3, Z_3) = (T_1, T_2, T_3)$ $T_1 \leftarrow X_1, T_2 \leftarrow Y_1, T_3 \leftarrow Z_1$		EC Point Addition Input: $P = (X_1, Y_1, Z_1), Q = (X_2, Y_2)$ Output: $P + Q = (X_3, Y_3, Z_3, X'_1, Y'_1) = (T_1, T_2, T_3, T_4, T_5)$ $T_1 \leftarrow X_1, T_2 \leftarrow Y_1, T_3 \leftarrow Z_1, T_x \leftarrow X_2, T_y \leftarrow Y_2$	
		Original atomic pattern corresponding to Table 6.1	Our implementation using constant-time FLECC functions	Original atomic pattern corresponding to Table 6.2	Our implementation using constant-time FLECC functions
$\Delta 1$	OP1:M	$T_4 \leftarrow T_3 \cdot T_3$	$T_4 \leftarrow T_3 \cdot T_3 \cdot R^{-1}$ $T_4 \leftarrow T_4 \cdot R^2 \cdot R^{-1}$	$T_4 \leftarrow T_3 \cdot T_3$	$T_4 \leftarrow T_3 \cdot T_3 \cdot R^{-1}$ $T_4 \leftarrow T_4 \cdot R^2 \cdot R^{-1}$
	OP2:N	*	$T_0 \leftarrow -T_1$	*	$T_0 \leftarrow -T_1$
	OP3:A	$T_5 \leftarrow T_1 + T_4$	$T_5 \leftarrow T_1 + T_4$	*	$T_5 \leftarrow T_1 + T_4$
	OP4:M	$T_6 \leftarrow T_2 \cdot T_2$	$T_6 \leftarrow T_2 \cdot T_2 \cdot R^{-1}$ $T_6 \leftarrow T_6 \cdot R^2 \cdot R^{-1}$	$T_5 \leftarrow T_x \cdot T_4$	$T_5 \leftarrow T_x \cdot T_4 \cdot R^{-1}$ $T_5 \leftarrow T_5 \cdot R^2 \cdot R^{-1}$
	OP5:N	$T_4 \leftarrow -T_4$	$T_4 \leftarrow -T_4$	$T_6 \leftarrow -T_1$	$T_6 \leftarrow -T_1$
	OP6:A	$T_2 \leftarrow T_2 + T_2$	$T_2 \leftarrow T_2 + T_2$	$T_5 \leftarrow T_5 + T_6$	$T_5 \leftarrow T_5 + T_6$
	OP7:A	$T_4 \leftarrow T_1 + T_4$	$T_4 \leftarrow T_1 + T_4$	*	$T_0 \leftarrow T_1 + T_4$
$\Delta 2$	OP8:M	$T_5 \leftarrow T_4 \cdot T_5$	$T_5 \leftarrow T_4 \cdot T_5 \cdot R^{-1}$ $T_5 \leftarrow T_5 \cdot R^2 \cdot R^{-1}$	$T_6 \leftarrow T_5 \cdot T_5$	$T_6 \leftarrow T_5 \cdot T_5 \cdot R^{-1}$ $T_6 \leftarrow T_6 \cdot R^2 \cdot R^{-1}$
	OP9:N	*	$T_0 \leftarrow -T_2$	*	$T_0 \leftarrow -T_2$
	OP10:A	$T_4 \leftarrow T_5 + T_5$	$T_4 \leftarrow T_5 + T_5$	*	$T_0 \leftarrow T_5 + T_5$
	OP11:M	$T_3 \leftarrow T_2 \cdot T_3$	$T_3 \leftarrow T_2 \cdot T_3 \cdot R^{-1}$ $T_3 \leftarrow T_3 \cdot R^2 \cdot R^{-1}$	$T_7 \leftarrow T_1 \cdot T_6$	$T_7 \leftarrow T_1 \cdot T_6 \cdot R^{-1}$ $T_7 \leftarrow T_7 \cdot R^2 \cdot R^{-1}$
	OP12:N	*	$T_0 \leftarrow -T_4$	*	$T_0 \leftarrow -T_4$
	OP13:A	$T_4 \leftarrow T_4 + T_5$	$T_4 \leftarrow T_4 + T_5$	$T_8 \leftarrow \textcolor{yellow}{T_7 + T_7}$	$T_8 \leftarrow T_7 + T_7$
	OP14:A	$T_2 \leftarrow T_6 + T_6$	$T_2 \leftarrow T_6 + T_6$	*	$T_0 \leftarrow T_0 + T_0$

	Operations	EC Point Doubling		EC Point Addition	
		Original atomic pattern corresponding to Table 6.1	Our implementation using constant-time FLECC functions	Original atomic pattern corresponding to Table 6.2	Our implementation using constant-time FLECC functions
$\Delta 3$	OP15:M	$T_5 \leftarrow T_4 \cdot T_4$	$T_5 \leftarrow T_4 \cdot T_4 \cdot R^{-1}$ $T_5 \leftarrow T_5 \cdot R^2 \cdot R^{-1}$	$T_9 \leftarrow T_5 \cdot T_6$	$T_9 \leftarrow T_5 \cdot T_6 \cdot R^{-1}$ $T_9 \leftarrow T_9 \cdot R^2 \cdot R^{-1}$
	OP16:N	*	$T_0 \leftarrow -T_1$	*	$T_0 \leftarrow -T_1$
	OP17:A	$T_6 \leftarrow T_2 + T_2$	$T_6 \leftarrow T_2 + T_2$	$T_8 \leftarrow T_8 + T_9$	$T_8 \leftarrow T_8 + T_9$
	OP18:M	$T_6 \leftarrow T_1 \cdot T_6$	$T_6 \leftarrow T_1 \cdot T_6 \cdot R^{-1}$ $T_6 \leftarrow T_6 \cdot R^2 \cdot R^{-1}$	$T_4 \leftarrow T_3 \cdot T_4$	$T_4 \leftarrow T_3 \cdot T_4 \cdot R^{-1}$ $T_4 \leftarrow T_4 \cdot R^2 \cdot R^{-1}$
	OP19:N	$T_1 \leftarrow -T_6$	$T_1 \leftarrow -T_6$	*	$T_0 \leftarrow -T_6$
	OP20:A	$T_1 \leftarrow T_1 + T_1$	$T_1 \leftarrow T_1 + T_1$	*	$T_0 \leftarrow T_1 + T_1$
	OP21:A	$T_1 \leftarrow T_1 + T_5$	$T_1 \leftarrow T_1 + T_5$	*	$T_0 \leftarrow T_1 + T_5$
$\Delta 4$	OP22:M	$T_2 \leftarrow T_2 \cdot T_2$	$T_2 \leftarrow T_2 \cdot T_2 \cdot R^{-1}$ $T_2 \leftarrow T_2 \cdot R^2 \cdot R^{-1}$	$T_4 \leftarrow T_y \cdot T_4$	$T_4 \leftarrow T_y \cdot T_4 \cdot R^{-1}$ $T_4 \leftarrow T_4 \cdot R^2 \cdot R^{-1}$
	OP23:N	$T_5 \leftarrow -T_1$	$T_5 \leftarrow -T_1$	$T_{10} \leftarrow -T_2$	$T_{10} \leftarrow -T_2$
	OP24:A	$T_5 \leftarrow T_5 + T_6$	$T_5 \leftarrow T_5 + T_6$	$T_4 \leftarrow T_4 + T_{10}$	$T_4 \leftarrow T_4 + T_{10}$
	OP25:M	$T_5 \leftarrow T_4 \cdot T_5$	$T_5 \leftarrow T_4 \cdot T_5 \cdot R^{-1}$ $T_5 \leftarrow T_5 \cdot R^2 \cdot R^{-1}$	$T_{10} \leftarrow T_4 \cdot T_4$	$T_{10} \leftarrow T_4 \cdot T_4 \cdot R^{-1}$ $T_{10} \leftarrow T_{10} \cdot R^2 \cdot R^{-1}$
	OP26:N	$T_2 \leftarrow -T_2$	$T_2 \leftarrow -T_2$	$T_8 \leftarrow -T_8$	$T_8 \leftarrow -T_8$
	OP27:A	$T_2 \leftarrow T_2 + T_2$	$T_2 \leftarrow T_2 + T_2$	$T_1 \leftarrow \textcolor{blue}{T_{10}} + T_8$	$T_1 \leftarrow T_{10} + T_8$
	OP28:A	$T_2 \leftarrow T_2 + \textcolor{blue}{T_5}$	$T_2 \leftarrow T_2 + T_5$	*	$T_0 \leftarrow T_2 + T_5$

	Operations	EC Point Doubling		EC Point Addition	
		Original atomic pattern corresponding to Table 6.1	Our implementation using constant-time FLECC functions	Original atomic pattern corresponding to Table 6.2	Our implementation using constant-time FLECC functions
$\Delta 5$	OP29:M			$T_8 \leftarrow T_2 \cdot T_9$	$T_8 \leftarrow T_2 \cdot T_9 \cdot R^{-1}$ $T_8 \leftarrow T_8 \cdot R^2 \cdot R^{-1}$
	OP30:N			$T_6 \leftarrow -T_1$	$T_6 \leftarrow -T_1$
	OP31:A			$T_6 \leftarrow T_6 + T_7$	$T_6 \leftarrow T_6 + T_7$
	OP32:M			$T_6 \leftarrow T_6 \cdot \textcolor{blue}{T_4}$	$T_6 \leftarrow T_6 \cdot T_4 \cdot R^{-1}$ $T_6 \leftarrow T_6 \cdot R^2 \cdot R^{-1}$
	OP33:N			$T_9 \leftarrow -T_8$	$T_9 \leftarrow -T_8$
	OP34:A			$T_2 \leftarrow T_6 + T_9$	$T_2 \leftarrow T_6 + T_9$
	OP35:A			*	$T_0 \leftarrow T_2 + T_5$
$\Delta 6$	OP36:M			$T_3 \leftarrow T_3 \cdot T_5$	$T_3 \leftarrow T_3 \cdot T_5 \cdot R^{-1}$ $T_3 \leftarrow T_3 \cdot R^2 \cdot R^{-1}$
	OP37:N			$T_4 \leftarrow -T_7$	$T_4 \leftarrow -T_7$
	OP38:A			$T_4 \leftarrow T_1 + T_4$	$T_4 \leftarrow T_1 + T_4$
	OP39:M			$T_5 \leftarrow T_4 \cdot T_4$	$T_5 \leftarrow T_4 \cdot T_4 \cdot R^{-1}$ $T_5 \leftarrow T_5 \cdot R^2 \cdot R^{-1}$
	OP40:N			$T_6 \leftarrow -T_8$	$T_6 \leftarrow -T_8$
	OP41:A			$T_6 \leftarrow T_2 + T_6$	$T_6 \leftarrow T_2 + T_6$
	OP42:A			*	$T_0 \leftarrow T_2 + T_5$

TABLE 6.3: Implemented sequence of operations using constant-time FLECC functions for EC point doubling and EC point addition operations applying the atomic pattern MNAMNAA.

Two columns denoted as "Our implementation using constant-time FLECC functions" show which operations we used as the dummy operations (instead of the operations indicated by * in Longa's patterns). These dummy operations aim at making point doubling and point addition indistinguishable to each other. They do not change the computation results. Please note that we tried to apply identical operands and resultant registers on the arithmetic operations on both point doubling and point addition operations whenever any side has a dummy operation, to try to make both algorithms as indistinguishable as possible against address-bit SCA. For example, at point addition OP7 ($T_0 \leftarrow T_1 + T_4$), we are not able to use the same register T_4 as resultant as in point doubling OP7 ($T_4 \leftarrow T_1 + T_4$), since it would ruin other calculation steps, we use a dummy register T_0 instead. But we still try to match the operands T_1 and T_4 in the modular addition operation.

Additionally, we inserted "no operations" (NOPs) between each atomic block within a point operation, and at the end of each EC point doubling and EC point addition operation. The NOPs are necessary for simplifying the identification of atomic blocks and point operations in the measured trace.

6.2 Measurements

6.2.1 Experimental Setup

Measurements of electromagnetic (EM) emanation of cryptographic chips, unlike power measurements, do not require any modification of the board of the attacked device. Thus, we decided to measure and analyse the EM emanation of our implementation captured during a single kP algorithm execution. In our experimental setup, we used a near-field micro-probe to measure EM emanation, an Integrated Circuit Scanner (ICS) to precisely position the board and the EM-probe, and an oscilloscope to capture the EM emanation. Figures 6.1 and 6.2 show the equipment used. The list of equipment is as follows:

- Langer ICS 105 [99]
- Langer EM-probe MFA-R 0.2-75 [100]
- Teledyne LeCroy oscilloscope WavePro 604HD [101]



FIGURE 6.1: The Integrated Circuit Scanner positioning the board and the EM-probe.



FIGURE 6.2: The oscilloscope connected to the probe showing the captured trace.

To find the most optimal position to place the probe, we first looked at the schematic [102] of the F28379D LaunchPad to find the list of capacitors in the power line for internal logic (1.2 V) as the power supply capacitors are the most suitable component for EM trace measurement. To find the best signal-to-noise ratio, we placed the probe near each of the capacitors C42, C46-49 and C75-78, while executing the *kP* algorithm code in RAM, to see which position received the highest amplitude of EM signals. We found that placing the probe at the side of C78, as shown in Figure 6.3, gave us the strongest signals. Therefore, we used this position for the measurement of the EM trace.

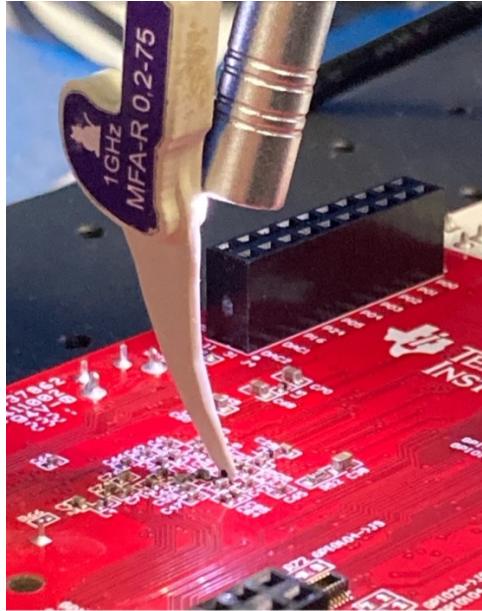


FIGURE 6.3: Near-field micro probe placed near capacitor C78 on the board.

For our *kP* algorithm implementation, without applying NOPs, for the board with a frequency of 100 MHz, the execution time of a point doubling operation run in RAM is around 290,944 clock cycles (2.91 ms in our setup), while that of a point addition operation is about 436,416 (4.36 ms). The estimated processing time of a 256-bit scalar k is between $290944 \cdot 255 = 74,190,720$ clock cycles (742 ms) for the 256-bit scalar $k = 10\ldots0$ and up to $(290944 + 436416) \cdot 255 = 185,476,800$ clock cycles (1855 ms) for the 256-bit scalar $k = 11\ldots1$. When using a sufficiently high sampling rate,

e.g., 10 GS/s which results into 100 samples per clock cycle, the average required memory to store the captured data is 391 GB, which takes tremendous time and resource to process and analyse, even if the oscilloscope could meet the memory requirement.

We are aware that attackers need to deal with technical limits like the above mentioned when extracting a 256-bit long key. We assume that these demanding technical requirements are the reason why there is a lack of experimental investigation on the resistance of different atomic patterns.

Therefore, for the reasons explained above, we only used a 22-bit long scalar k in our experiments. For the kP algorithm executed in RAM, we captured the trace with an oscilloscope sampling rate of 1 GS/s. By using a board with 100 MHz clock signal frequency, it results in only 10 samples per clock cycle. Taking into account that NOPs are inserted to the algorithm, the file size of the measured trace is about 6 GB. Similarly, for the kP algorithm executed in flash memory, we captured the trace with an oscilloscope sampling rate of 100 MS/s. By using a board with 20 MHz clock signal frequency, it results in only 5 samples per clock cycle and a trace file of 5.86 GB.

As the inputs for the implemented kP algorithm in our experiments, we used a 22-bit binary scalar k^2 and the base point G of the EC P-256 [7]³. In total, 15 point additions and 21 point doublings are performed executing our kP algorithm with these inputs.

However, since the full length of the kP operation exceeds the limit we could capture with our measurement settings when being executed in RAM, the last point addition was not measured completely. We were only able to capture 14 point additions and 21 point doublings when the code was being executed in RAM.

For the kP operation executed using the board's flash memory, we were able to measure the EM emanation during a whole kP operation execution (15 point additions and 21 point doublings). Table 6.4 summarizes our measurement settings.

	RAM	Flash
Frequency of microcontroller attacked	100 MHz	20 MHz
Sampling rate of the oscilloscope	1 GS/s	100 MS/s
Samples per clock cycle	10	5
Samples captured	200 M	200 M
kP execution time	>200 ms	1.35 s
Size of raw data captured	6.02 GB	5.86 GB

TABLE 6.4: Settings applied in our measurements.

Note that we chose to capture only 200 M samples using the oscilloscope, although our oscilloscope supports capturing a maximum of 500 M samples, due to memory limitations of the oscilloscope. This decision is made to compromise between the completeness of the whole kP operation trace and the number of sampling points per clock cycle for data quality of our analysis. Since the whole kP operation executed in RAM takes a bit more than 200 ms, it is much faster than that run in flash memory which takes about 1.35 s, we could use a higher rate on samples per second on the kP operation algorithm executed in RAM.

² $k = k_{21}k_{20}k_{19}\dots k_2k_1k_0 = 100110110101111110111$

³In hexadecimal: $G = (x, y) = (6b17d1f2e12c4247f8bce6e563a440f277037d812deb33a0f4a13945d898c296, 4fe342e2fe1a7f9b8ee7eb4a7c0f9e162bce33576b315ecccbb6406837bf51f5)$

6.2.2 Study of FLECC Constant-runtime Operations

To determine whether the constant-runtime field operation functions from FLECC we used in our implementation are truly constant-runtime, we measured the execution time of those used in Doubling 2 of our atomic patterns algorithm. These measurements were performed using a hardware breakpoint in CCS with the code executed in RAM. We repeatedly measured the first Montgomery modular multiplication (X), the second Montgomery modular multiplication (X'), the first negation (N) and the first addition (A) in $\Delta 1$ of Doubling 2, performing each measurement 10 times. We obtained the same results for each operation in all our measurements. The results are presented in Table 6.5. The execution time of each operation was measured in number of clock cycles and then converted into time in millisecond according to our experimental equipment configurations. We observed a 1-clock cycle difference between the runtimes of X and X' , as shown in Table 6.5, which means the second Montgomery modular multiplication (X') required always 1 clock cycle less than the first one (X).

	No. of clock cycles	Time(ms)
1 st Multiplication (X)	16570	0.166
2 nd Multiplication (X')	16569	0.166
Negation (N)	1197	0.0120
Addition (A)	1353	0.0135

TABLE 6.5: Execution time of FLECC constant-runtime operations in $\Delta 1$ of Doubling 2, measured 10 times.

Please note that we selected $\Delta 1$ in Doubling 2 for our measurements to avoid any potential impact of the special operand $Z_1 = 1$ (from $\Delta 1$ in Doubling 1) on the execution time of the multiplication operation. By analyzing the sub-traces of $\Delta 1$ in Doubling 1 and Doubling 2 over time, we observed that Doubling 2 has a delay of 5 clock cycles compared to Doubling 1 at 20,000 indices after the synchronization anchor samples, which will be introduced in Chapter 6.3.1. This finding indicates that the duration of Doubling 1 is likely shorter than Doubling 2. A detailed description can be found in Chapter 8.3.

Additionally, we measured the execution time of the first X in $\Delta 1$ of each point operation, recorded the results in Table 6.6 and summarized them in Figure 6.4. As illustrated in Figure 6.4, the X operation consistently took either 16,565 or 16,570 clock cycles in our 36 measurements, with an overall 75% likelihood of falling on the lower end.

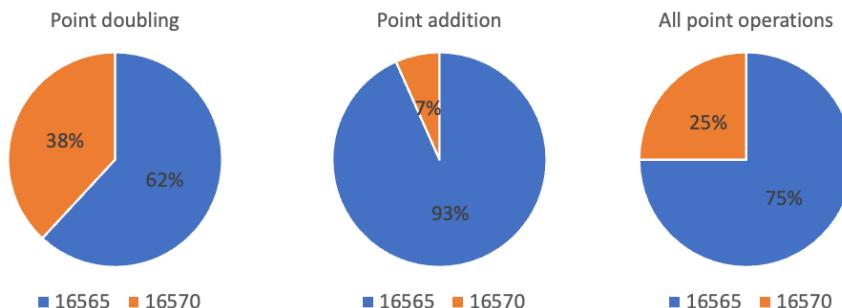


FIGURE 6.4: Distribution of execution time of the first X in $\Delta 1$ of all point doublings (left), all point additions (middle) and all EC point operations (right).

However, these differences in the execution time between X and X' , and among X itself are so negligible (≤ 5 clock cycles) and consistent that we can classify the multiplication function as constant-runtime, with negation and addition functions also demonstrating constant-runtime behavior.

X	Start	End	Duration
Doubling 1	533427	549992	16565
Doubling 2	1052586	1069156	16570
Doubling 3	1571720	1588285	16565
Doubling 4	2673531	2690101	16570
Doubling 5	3775323	3791893	16570
Doubling 6	4294499	4311069	16570
Doubling 7	5396316	5412881	16565
Doubling 8	6498128	6514693	16565
Doubling 9	7017274	7033839	16565
Doubling 10	8119076	8135641	16565
Doubling 11	8638222	8654787	16565
Doubling 12	9740014	9756579	16565
Doubling 13	10841816	10858381	16565
Doubling 14	11943608	11960178	16570
Doubling 15	13045420	13061990	16570
Doubling 16	14147227	14163797	16570
Doubling 17	15249054	15265619	16565
Doubling 18	16350836	16367406	16570
Doubling 19	16870007	16886572	16565
Doubling 20	17971794	17988359	16565
Doubling 21	19073625	19090190	16565
Addition 1	2020871	2037436	16565
Addition 2	3122673	3139238	16565
Addition 3	4743655	4760225	16570
Addition 4	5845472	5862037	16565
Addition 5	7466429	7482994	16565
Addition 6	9087376	9103941	16565
Addition 7	10189168	10205733	16565
Addition 8	11290970	11307535	16565
Addition 9	12392772	12409337	16565
Addition 10	13494584	13511149	16565
Addition 11	14596411	14612976	16565
Addition 12	15698193	15714758	16565
Addition 13	17319150	17335715	16565
Addition 14	18420947	18437512	16565
Addition 15	19522734	19539299	16565

TABLE 6.6: Execution time of the first X operation in $\Delta 1$ of each point operation, measured in clock cycles.

6.3 Simple Electromagnetic Analysis Attack

6.3.1 Analysis of *kP* Operation Executed in RAM

To ease the separation of EM traces for our analysis, in the atomic patterns *kP* algorithm, we separated each atomic block by placing a sequence of 2,000 redundant counter increment operations in between. We see these increment operations as no operations (NOPs) as they do very simple instructions that leak rather little EM emanation. Also, we placed a significantly longer sequence of 5,000 NOPs between each point doubling and addition operation to separate between different point operations, and 10,000 NOPs between two consecutive point doubling operations. Figure 6.5 shows the entire EM trace we captured during the *kP* operation. By noticing the NOPs inserted between each point operation as well as each atomic block, we can clearly identify each point doubling operation and point addition operation, and each of their atomic blocks.

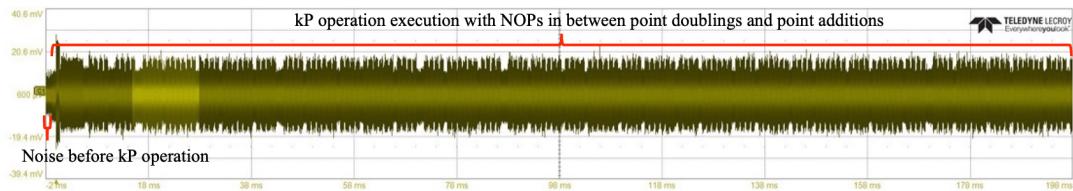


FIGURE 6.5: Measurement of the *kP* execution: EM trace of the implemented atomic patterns *kP* algorithm executed in RAM of the board.

By using a hardware breakpoint in CCS, we measured once the duration of each sequence of NOPs between the end of Doubling 2 and the start of Doubling 4, covering Doubling 3 and Addition 1. We found that the duration of the execution of 2,000 NOPs inserted between two atomic blocks has a median of 29,995 clock cycles (ranged from 27,996 to 30,006 clock cycles); while 5,000 NOPs between two EC point operation takes a median of 69,996 clock cycles (ranged from 69,996 to 70,012 clock cycles). The raw data of the measurements can be found in Table 8.9 and 8.10 in Chapter 8.4. We observed minor fluctuations in execution time for the same number of NOPs at different stages in the algorithm, which we attribute to the wait states of the memory causing the execution time to be slightly irregular.

Microcontrollers usually require a non-zero wait state when the CPU's operating frequency is higher than the memory speed, which means the CPU has to wait for a number of clock cycles to complete data read/write operations at the memory address. Modern RAM uses sophisticated techniques to minimize wait states. Furthermore, for RAM, the wait time can differ depending on factors like the type of operation being performed (e.g., read, write) and the access patterns (e.g., sequential access, random access). Therefore, it is inevitable to have some varied amount of operational time added to the operations being performed, hindering the performance of SCA.

Figure 6.6 magnifies the beginning of the trace in Figure 6.5, showing the shapes of the noise before the start of the *kP* operation, the processing of the most significant bit of the scalar *k*, and the shapes of some atomic patterns in the main loop of the *kP* algorithm.

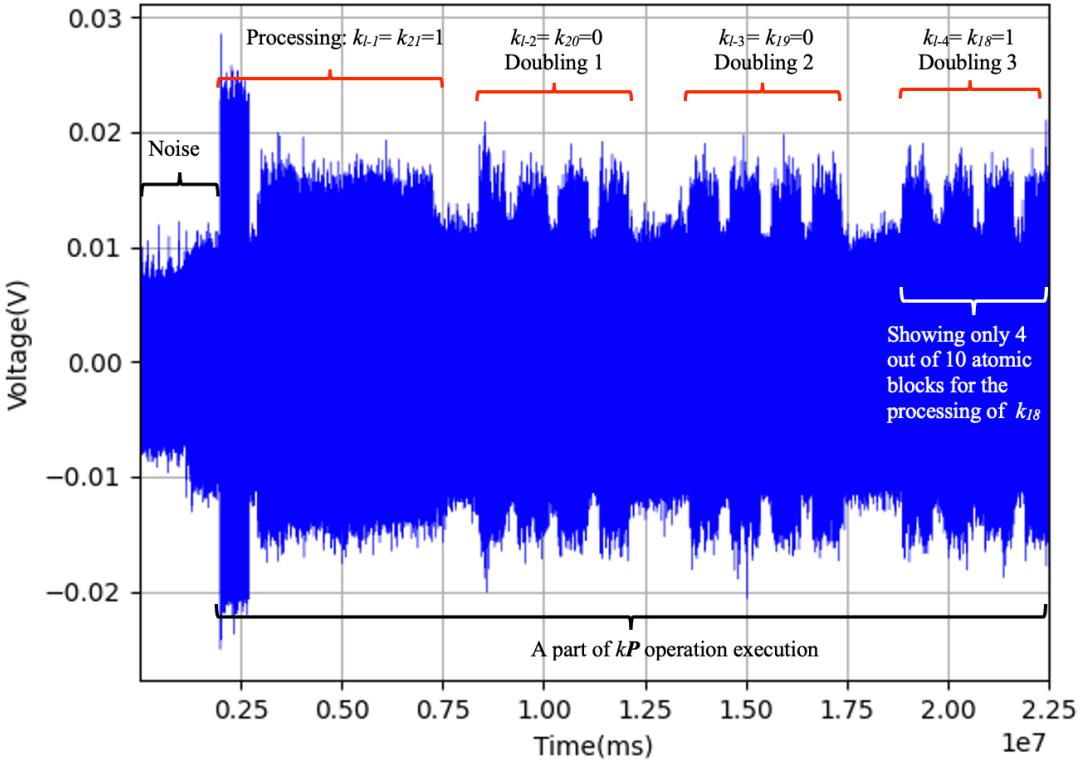


FIGURE 6.6: A part of the captured EM trace: the noise before the start of the kP operation, the processing of the most significant bit $k_{l-1}=1$, and the atomic patterns of the first 3 point doublings are shown; kP executed in RAM of the board.

Before the atomic patterns in the kP operation begin, there is a significant peak and some continuous peaks without drops in amplitudes. This part of the kP operation refers to the initialization phase of the kP operation before the main loop iterations. In the main loop, 21 of 22 bits of the scalar k are processed, executing 21 point doublings and 15 point additions as atomic patterns. We denote in our analysis all atomic patterns as “Doubling i ” with $1 \leq i \leq 21$ or “Addition j ” with $1 \leq j \leq 15$. In Figure 6.6, the first 3 EC point doubling operations are shown, namely Doubling 1, Doubling 2 and Doubling 3.

Figure 6.7 shows the part of the captured EM trace corresponding to the processing of the bits $k_{l-4} = k_{18} = 1$ and $k_{l-5} = k_{17} = 1$ of the key. Processing a key bit value ‘1’ requires the execution of a point doubling and a point addition. Thus, Figure 6.7 shows the atomic patterns Doubling 3, Addition 1, Doubling 4 and Addition 2, corresponding to our numbering of the atomic patterns. Each doubling consists of 4 atomic blocks and each addition consists of 6 atomic blocks. The short NOP sequences inserted between each atomic block help us separate the atomic blocks $\Delta 1$ to $\Delta 4$ or $\Delta 1$ to $\Delta 6$. The long NOP sequences inserted between atomic patterns help us separate the EC point operations executed in the main loop iterations, so the doubling operations and the addition operations are visibly distinguishable by observing the presence of long troughs of NOPs in between and counting the number of atomic blocks they have. Each atomic block of the corresponding atomic pattern, the short NOP blocks between the atomic blocks and the long NOP blocks between different operations are shown in Figure 6.7 too.

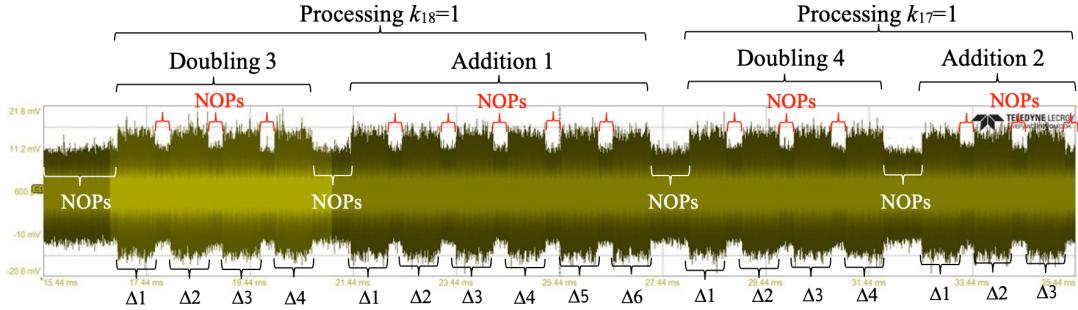


FIGURE 6.7: EM trace representing a part of the atomic pattern kP algorithm executed in RAM of the board.

We are aware that in a real-world scenario there would not be such long breaks between the atomic blocks as well as between atomic patterns. So, it would not be easy for an adversary to separate the atomic blocks or distinguish the operations. Nevertheless, our work concentrates on the investigation of the distinguishability of the atomic patterns. Thus, it is reasonable to ignore the difficulty in separating the atomic blocks and only focus on analysing the differences between the atomic blocks of point doubling and point addition operations.

Additionally, in order to analyse the shapes of point doubling operations and point addition operations, breakpoints were added in the CCS program of our kP algorithm implementation to help us measure the clock cycles needed until the first point operation starts and the clock cycles each point operation then takes. We used these measured clock cycles, as shown in Table 6.7, to calculate the actual time each point operation approximately takes, and then split the whole captured trace into sub-traces of single point operations according to the time calculated, resulting in 21 separate point doubling sub-traces and 14 separate point addition sub-traces.

	RAM			Flash memory		
	Start	End	Duration	Start	End	Duration
Doubling 1	642930	3364774	376912	3364774	5830998	2466224
Doubling 2	1169884	1546776	376892	6631158	9097318	2466160
Doubling 3	1696818	2073735	376917	9897478	12363718	2466240
Doubling 4	2804121	3181028	376907	16943334	19409542	2466208
Doubling 5	3911407	4288339	376932	23989126	26455414	2466288
Doubling 6	4438382	4815299	376917	27255574	29721814	2466240
Doubling 7	5545693	5922610	376917	34301446	36767686	2466240
Doubling 8	6652999	7029901	376902	41347302	43813494	2466192
Doubling 9	7179943	7556855	376912	44613654	47079878	2466224
Doubling 10	8287239	8664141	376902	51659478	54125670	2466192
Doubling 11	8814183	9191090	376907	54925830	57392038	2466208
Doubling 12	9921469	10298376	376907	61971622	64437830	2466208
Doubling 13	11028765	11405672	376907	69017446	71483654	2466208
Doubling 14	12136051	12512968	376917	76063238	78529478	2466208
Doubling 15	13243357	13620274	376917	83109094	85575334	2466240
Doubling 16	14350658	14727595	376937	90154934	92621238	2466304
Doubling 17	15457979	15834871	376892	97200838	99666998	2466160
Doubling 18	16565255	16942182	376927	104246598	106712870	2466272
Doubling 19	17092224	17469116	376892	107513030	109979190	2466160
Doubling 20	18199505	18576407	376902	114558806	117024998	2466192
Doubling 21	19306776	19683688	376912	121604550	124070774	2466224
Addition 1	2148757	2729118	580361	12763798	16543366	3779568
Addition 2	3256050	3836401	580351	19809622	23589158	3779536
Addition 3	4890321	5470687	580366	30121894	33901478	3779584
Addition 4	5997632	6577993	580361	37167766	40947334	3779568
Addition 5	7631877	8212233	580356	47479958	51259510	3779552
Addition 6	9266112	9846463	580351	57792118	61571654	3779536
Addition 7	10373398	10953759	580361	64837910	68617478	3779568
Addition 8	11480694	12061045	580351	71883734	75663270	3779536
Addition 9	12587990	13168351	580361	78929558	82709126	3779568
Addition 10	13695296	14275652	580356	85975414	89754966	3779552
Addition 11	14802617	15382973	580356	93021318	96800870	3779552
Addition 12	15909893	16490249	580356	100067078	103846630	3779552
Addition 13	17544138	18124499	580361	110379270	114158838	3779568
Addition 14	18651429	19231770	580341	117425078	121204582	3779504
Addition 15	19758710	20339063	580353	124470854	128250406	3779552

TABLE 6.7: Atomic patterns point operations execution time measured in clock cycles, using breakpoints.

Figures 6.8 and 6.9 represent the duration of EC point operations given in Table 6.7 graphically. We can see that the execution time of point operations in RAM fluctuates within 50 clock cycles, while those executed in flash memory has a bigger range of 150 clock cycles for the fluctuation.

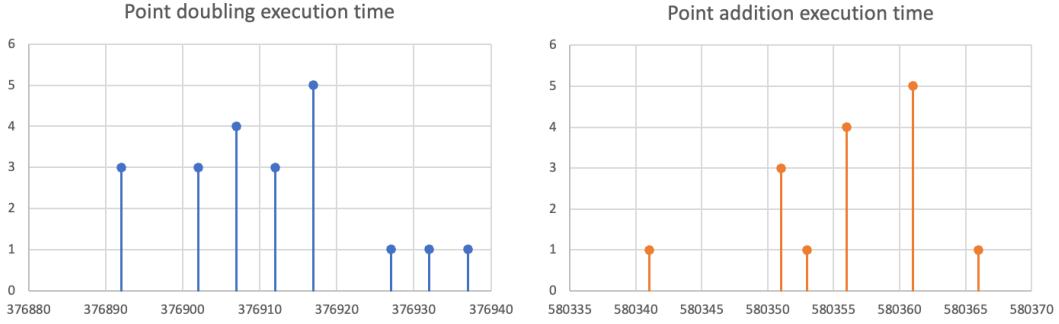


FIGURE 6.8: The distribution of measured execution time of point doubling operations (left) and point addition operations (right) in a kP algorithm executed in RAM.

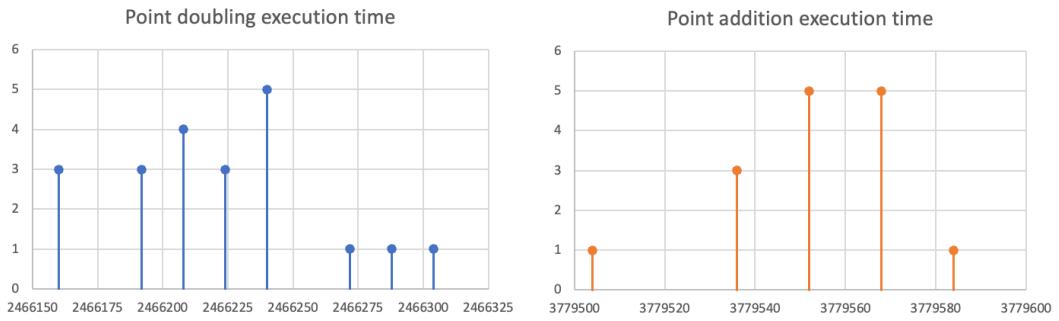


FIGURE 6.9: The distribution of measured execution time of point doubling operations (left) and point addition operations (right) in a kP algorithm executed in flash memory.

Synchronization Alignment

In all the following analysis, we excluded the sub-trace of Doubling 1 from the dataset because Doubling 1 consists of operations with operands using initialization values, i.e., OP1 in Table 6.3, which computes essentially $1 \times 1 \times R^{-1}$ and $R^{-1} \times R^2 \times R^{-1}$ that take much less computing power than any other multiplication operations with more complicated operands. Removing it from the dataset facilitates an unbiased analysis on typical doubling operations.

All the sub-traces were then finely synchronized manually using Microsoft Excel charts to have the shapes of their atomic blocks and NOPs aligned to precision for further analysis. We used three methods to synchronize the sub-traces, i.e., Synchronization A, B and C, see Table 6.8.

Sync.	Addition operations	Doubling operations
A	Align local maximums and minimums using bare eyes	Align local maximums and minimums using bare eyes
B	Based on Sync. A, align the rise of the lines between some samples	Based on Sync. A, align the rise of the lines between some samples
C	Same as Sync. A	Based on Sync. A, align according to the fixed clock period of the samples

TABLE 6.8: Methods used for the fine synchronization of EC point operations.

In Synchronization A, we started the alignment of sub-traces by fixating a close observation on a range of 4,000 anchor samples in one of the point operation sub-traces, which is at around sample 198,000 to sample 202,000 from the start of the sub-trace, so it is well assured that they are not noise but samples within the point operation in $\Delta 1$. Then, we overlaid other sub-traces on top of it to synchronize them one by one, by shifting each of them to the left or right to fit the shapes of our anchoring sub-trace. Figure 6.10 shows parts of the Excel chart after the synchronization, with point addition traces in orange and point doubling traces in blue.

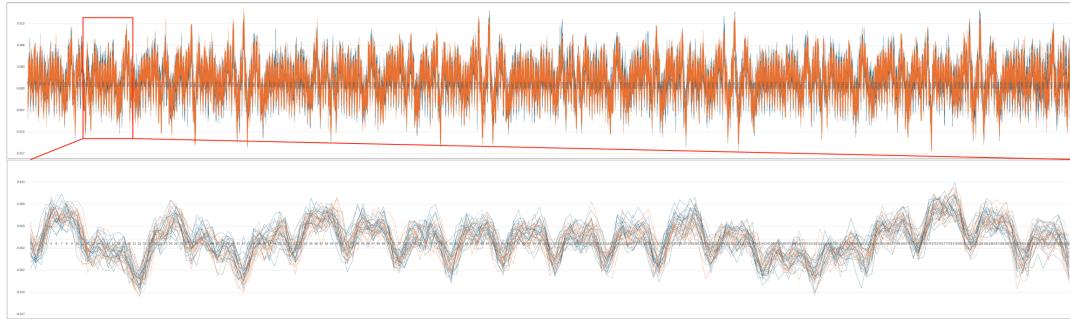


FIGURE 6.10: Above: alignment of 4,000 anchor samples in Synchronization A. Below: zoomed in at samples 200-400.

In Synchronization B, we first took the results from Synchronization A, then started aligning the rising segments of specific anchor samples in a smaller range (2,000 samples) within the same anchor samples range as in Synchronization A. Figure 6.11 depicts the essential steps involved in Synchronization B of Addition 1 (orange) and Addition 2 (blue). First, the rising segments of the data are identified, keeping all consecutive samples that exhibit increasing values in the plot. Subsequently, we observe the difference and decide to shift Addition 2 by one unit to the left in order to align the slopes and lengths of the majority of the rising segments in Addition 1. The overall synchronization alignment results are shown in Figure 6.12.

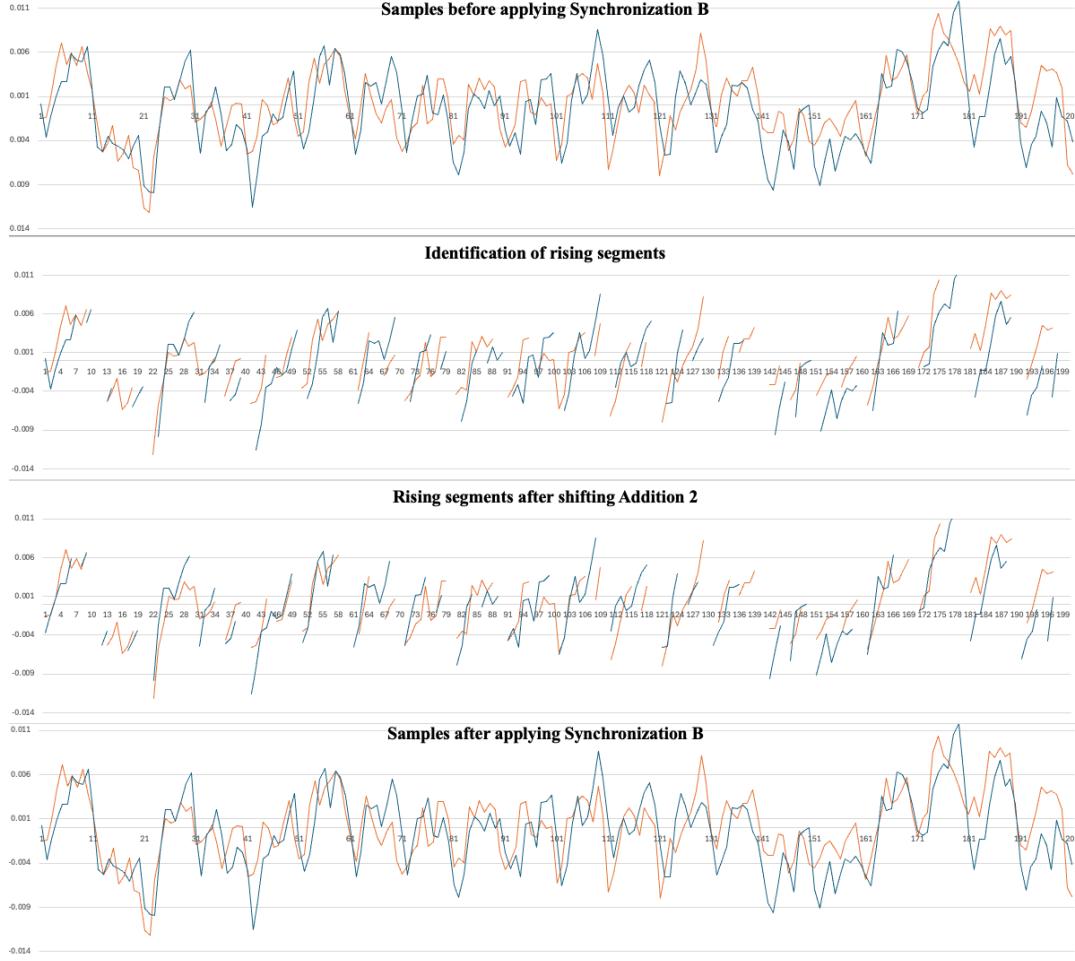


FIGURE 6.11: Method used in Synchronization B: Identification and alignment of rising segments.

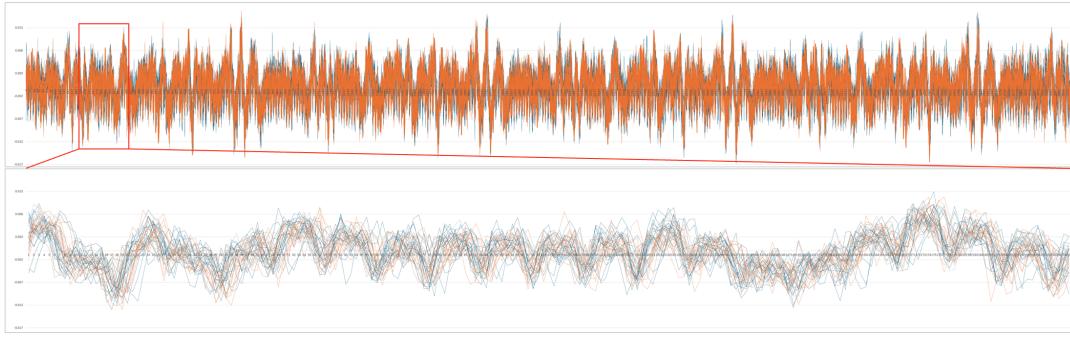


FIGURE 6.12: Above: alignment of 4,000 anchor samples in Synchronization B. Below: zoomed in at samples 200-400.

In Synchronization C, we employed the same synchronization method used in Synchronization A for point addition sub-traces. While for point doubling, based on Synchronization A, we aligned the sub-traces according to the fixed clock period (10 samples/period) of the samples. Similar to Synchronization B, we focused on the synchronization of specific samples within 2,000 samples in the chosen anchor samples. We first find out the first local minimum of each plot within the chosen range, then indicate the next local minimums at an interval of 10 samples, which

corresponds to the period of the clock signal. Finally, we connected the local minimums obtained in each plot as a new plot, compared these plots and synchronized them. An example of the alignment of Addition 1 and Addition 2 using this method is illustrated in Figure 6.13, first indicating the identified local minimums in every 10 samples with dots, then moving these dots into another plot for better visualization and finally shifting one of the sub-traces to finely align the majority of its minimums to that of the other sub-trace. The overall synchronization alignment results are shown in Figure 6.14.

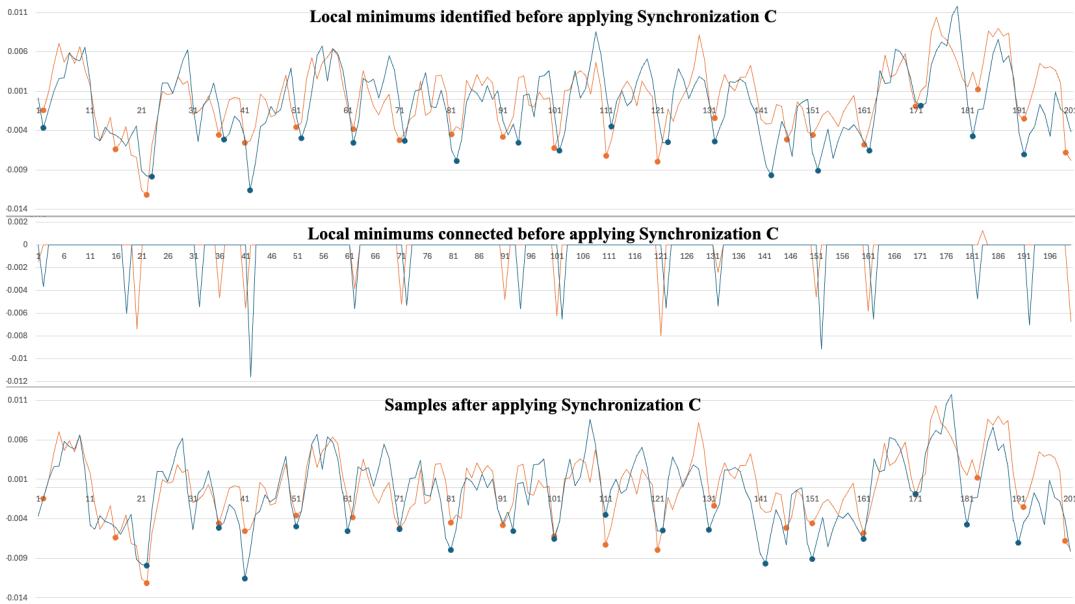


FIGURE 6.13: Method used in Synchronization C: Identification of local minimum in each clock cycle (Above). Connection of local minimums (Middle). Synchronization done by shifting Addition 2 one unit to the left. (Below)

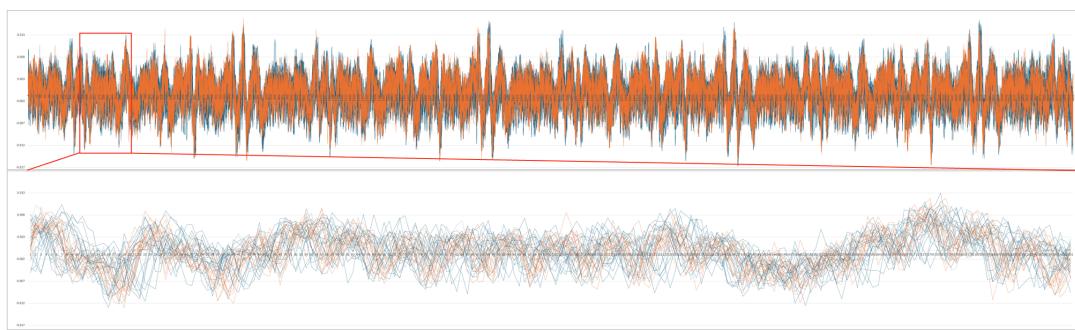


FIGURE 6.14: Above: alignment of 4,000 anchor samples in Synchronization C. Below: zoomed in at samples 200-400.

After that, we calculated for the case "Synchronization A" the average atomic patterns for point doublings as well as for point additions to examine the synchronization of the overlaid sub-traces of the first atom $\Delta 1$ of each atomic pattern. In Figure 6.15, the red lines represent the sub-traces of all point addition operations; the blue lines represent that of all point doubling operations; the yellow line marks the mean trace of all the point addition sub-traces; the green line marks the mean trace of all the point doubling sub-traces; the grey vertical area indicates the anchor

samples we used for the fine synchronization (the rough synchronization was done at the start of the atomic patterns by counting clock cycles, as explained right before this sub-section). The values on the x -axis represent the sample's index numbers, counted from the beginning of each point operation sub-trace we separated for $\Delta 1$.

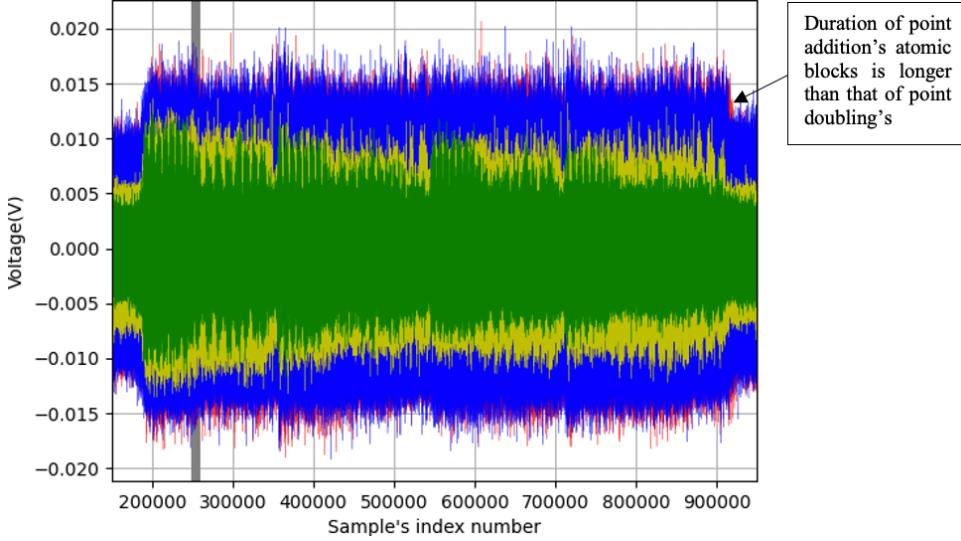


FIGURE 6.15: Overlay of $\Delta 1$ of all sub-traces synchronized using Synchronization A, with the algorithm executed in RAM.

The part of the red traces at the end of the atomic block for point additions demonstrates that the duration of $\Delta 1$ in point addition patterns is slightly longer than (and not equal to) that of point doubling patterns. In Figure 6.16, we inverted the order of lines overlay regarding point addition and point doubling in Figure 6.15.

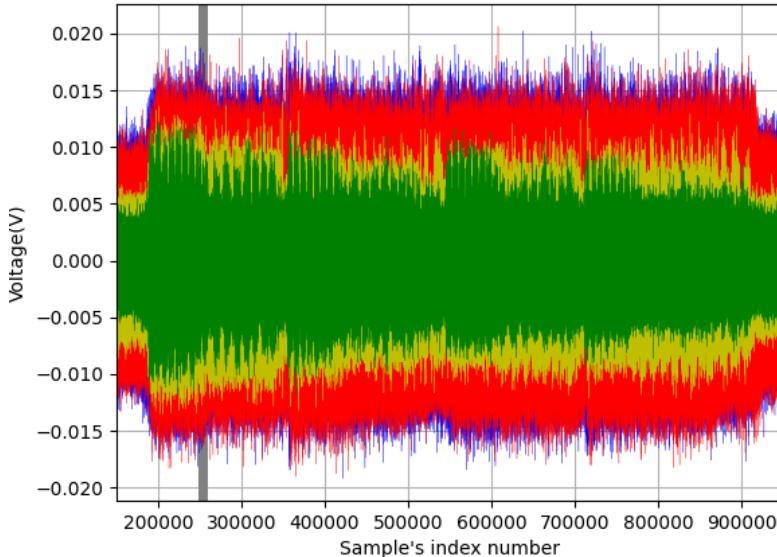


FIGURE 6.16: Overlay of $\Delta 1$ of all sub-traces synchronized using Synchronization A, with point addition traces laying on top of point doubling traces, and the algorithm executed in RAM.

From Figure 6.15 and 6.16, we noticed via visual observation in a fine scale that the shapes of the red and blue lines are adequately aligned. However, the amplitude of the green line (i.e., the mean trace of EC point doubling sub-traces) in Figure 6.16 diminishes gradually and gets closer to zero as it is further away from the synchronization anchor samples. Having a mean value close to zero potentially means that the sub-traces are not well synchronized at the end of the atomic blocks, as the samples in the sub-traces possibly have more distinct values, resulting in cancelling each other when a mean value is being calculated. Thus, we examined other methods, i.e., “Synchronization B” and “Synchronization C” (see Table 6.8) in hope of improving the synchronization.

Figure 6.17 shows the results of Synchronization B in $\Delta 1$, colored in the same way as in Figure 6.15 for Synchronization A. All figures in this sub-section adhere to the same format.

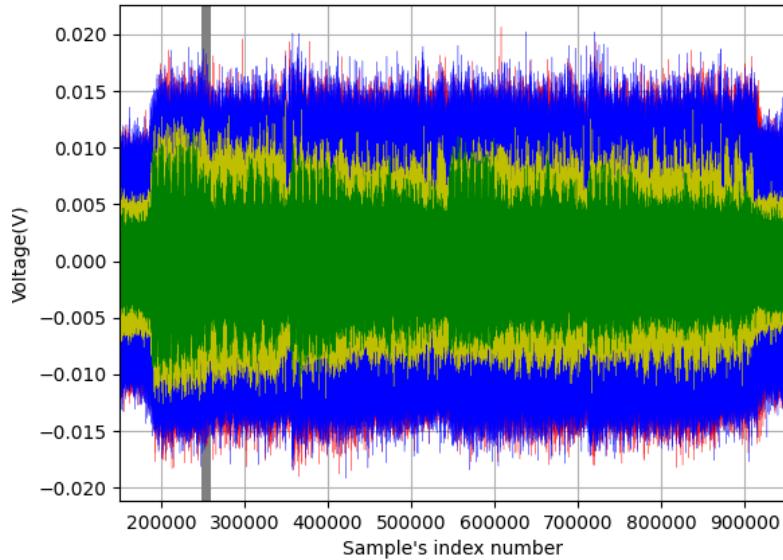


FIGURE 6.17: Overlay of $\Delta 1$ of all sub-traces synchronized using Synchronization B, with the algorithm executed in RAM.

We noticed that Synchronization B does not show any improvements when compared to Synchronization A. In Figure 6.18, the minimum value of the mean trace of point addition sub-traces (yellow) and the maximum value of the mean trace of point addition sub-traces (green) from Synchronization A are marked by the horizontal grey dashed lines. The amplitudes of the mean values of both the point addition and point doubling sub-traces from Synchronization B are visibly much smaller than those from Synchronization A, especially at the beginning of the atomic blocks. Therefore, we consider that Synchronization A is better than Synchronization B. However, the mean trace of point doubling sub-traces has always smaller amplitudes than that of point addition's. One of the reasons of this phenomenon can be a larger number of sub-traces being used in point doubling for calculating the means than in point addition.

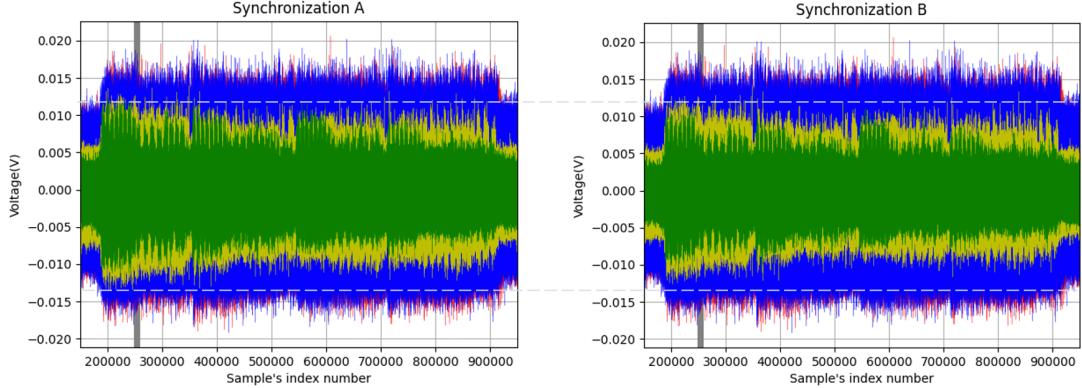


FIGURE 6.18: Comparison of amplitudes of mean of sub-traces between Synchronization A and Synchronization B.

Figure 6.19 shows the results of Synchronization C. This synchronization relying on the fixed clock period on the point doubling sub-traces is not better than the method used in Synchronization A, showing even smaller amplitudes in the mean trace of point doublings.

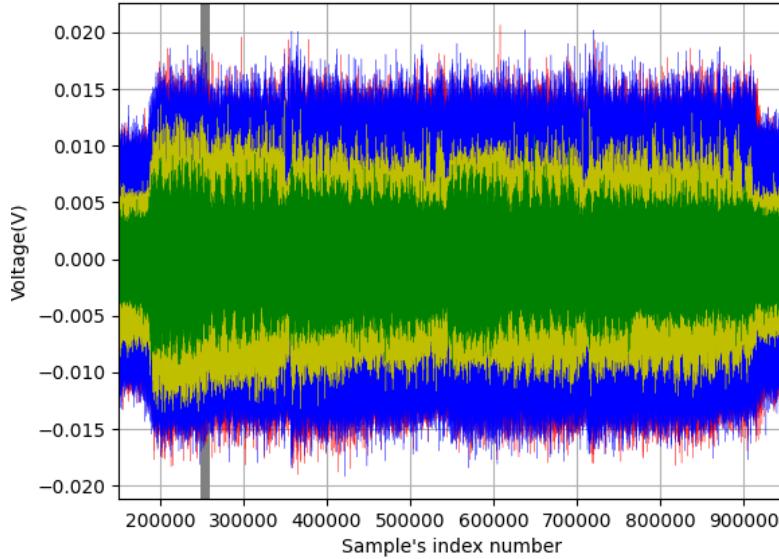


FIGURE 6.19: Overlay of $\Delta 1$ of all sub-traces synchronized using Synchronization C, with the algorithm executed in RAM.

Comparing the mean trace of $\Delta 1$ sub-traces of all three synchronizations, we assert that Synchronization A is our best-effort synchronization, which has decent alignment of point addition and point doubling sub-traces. Therefore, we would use Synchronization A for our plots and analysis going forward. It is important to note that point addition's atomic blocks are visibly slightly longer than point doubling's atomic blocks for all three synchronizations, i.e., the shapes of $\Delta 1$ in point doublings and in point additions are distinguishable. Exploring the factors contributing to this distinguishability could be a topic for future research.

Next, we extended the plot of synchronized sub-traces in $\Delta 1$ to all the atomic blocks $\Delta 1$ to $\Delta 6$, as shown in Figure 6.20. We observed that the mean trace of point

addition sub-traces and mean trace of point doubling sub-traces have the most distinct peaks in $\Delta 1$. From $\Delta 2$ onwards, the synchronization does not work as good as in $\Delta 1$ anymore, as seen in diminishing peaks in both mean traces, possibly due to different execution time of the NOPs between atomic blocks. Therefore, we considered focusing our distinguishability analysis solely on samples in $\Delta 1$. We will next zoom in Figure 6.20 to gain more insights from the sub-traces at locations A, B and C.

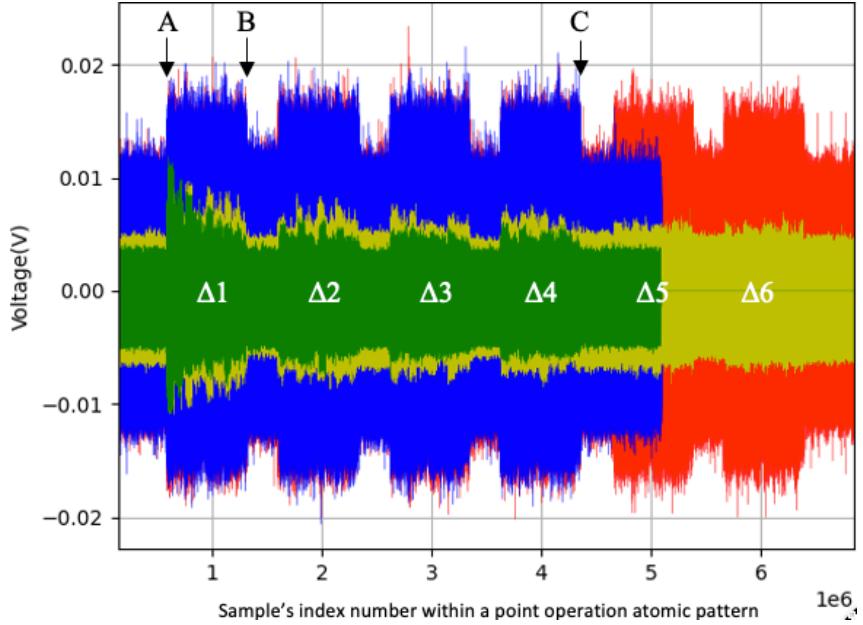


FIGURE 6.20: Atomic blocks of the synchronized point addition and point doubling sub-traces overlaid (additions: red; doublings: blue) and the mean trace of their sub-traces (addition: yellow; doubling: green).

The start and the end of $\Delta 1$ are particularly interesting to us. In Figure 6.21, there are the first 60,000 samples of $\Delta 1$ (from location A in Figure 6.20) showing how well the synchronization works in the beginning. We can see that the peaks of both the voltage and the average voltage of both operations are generally well aligned, despite some blue peaks with slightly greater amplitudes from point doubling at around sample index 186,000-191,000 (5,000 samples) in the beginning. In Figure 6.22, oppositely, we can see some red peaks with greater amplitudes from point addition at around sample index 918,000-923,000 (5,000 samples) at the end of $\Delta 1$ (from location B in Figure 6.20). Notably, these outstanding peaks have values quite similar to the noise, making it challenging to determine whether these observations could contribute to distinguishing between point addition and point doubling operations. If these outstanding peaks can be indicators for different duration of $\Delta 1$ in point doublings in comparison to point additions, an attacker can use them to reveal the key. A possible attack scenario is described in Chapter 8.5.

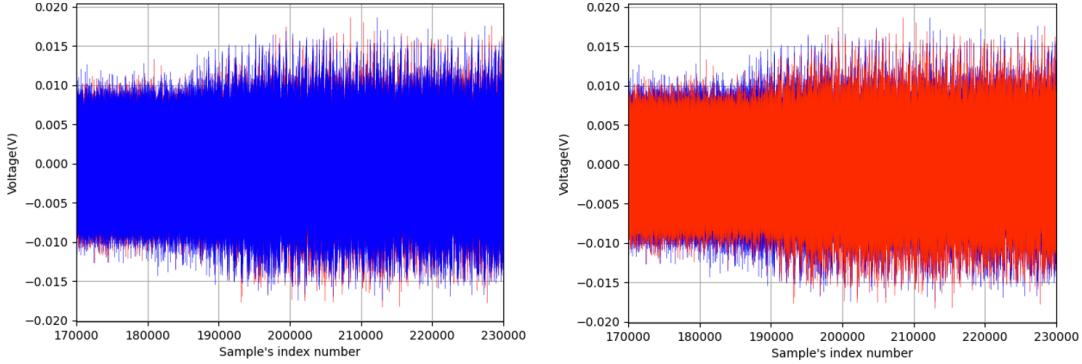


FIGURE 6.21: The samples at the beginning of Δ_1 at location A in Figure 6.20, lines overlaid in different orders.

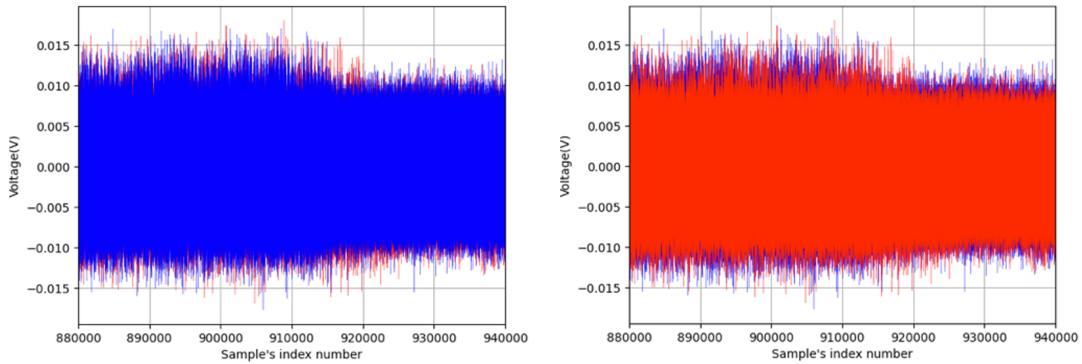


FIGURE 6.22: The samples at the end of Δ_1 at location B in Figure 6.20, lines overlaid in different orders.

Similarly, we used Figure 6.23 to examine the synchronization at the end of Δ_4 to see if both point addition and point doubling sub-traces remain aligned. We observed that the sub-traces are still reasonably synchronized. Furthermore, similar to Figure 6.22, red peaks from point addition stand out at the last 5,000 samples of the atomic block (from location C in Figure 6.20).

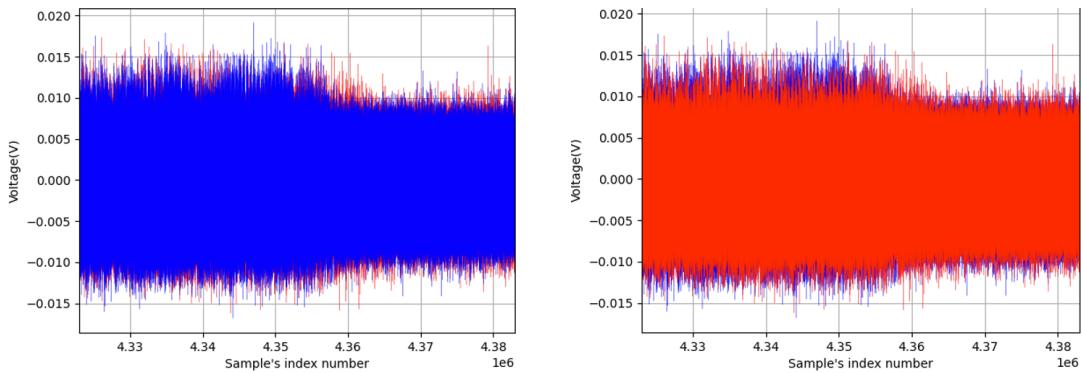


FIGURE 6.23: The samples at the end of Δ_4 at location C in Figure 6.20, lines overlaid in different orders.

Through visual comparison of the execution time in the sub-traces, we observed that the atomic patterns of point doubling begin approximately 5,000 samples earlier in Δ_1 compared to point addition. i.e., red lines are delayed in comparison to blue lines. Both EC point operations take the same execution time, but point additions

have a delay of 500 clock cycles in comparison to the point doublings. Additionally, $\Delta 1$ of point additions show slightly more obvious peaks before their shape is fully aligned with that of point doubling. This discrepancy (or delay) could be due to possible alternative synchronization alignments between the two sets, suggesting that multiple locations within the sub-traces in $\Delta 1$ could exhibit very similar characteristic shapes and our current alignment may not be optimal.

We cannot explain these discrepancies (delays of point additions) observed at the beginning and the end of the atomic blocks clearly. The red and blue lines were perfectly aligned at multiple locations within an atomic block. Despite our best efforts to synchronize the sub-traces and minimize discrepancies, we were unable to identify a better alignment option. The reasons behind these discrepancies require further investigation in future research.

In summary, the point doubling and point addition sub-traces are generally well synchronized throughout $\Delta 1$ using Synchronization A. Therefore, we decided to use these synchronized sub-traces in $\Delta 1$ for further SSCA.

Identification of Operations in Atomic Blocks

The shape of $\Delta 1$ does not allow us to identify any single field operation performed. The mean trace of point doubling sub-traces and the mean trace of point addition sub-traces indicating the synchronization are helpful to identify the multiplication operations: during the multiplication, all sub-traces are better synchronized than during other operations. Figure 6.24 represents our assumption regarding the start and end of each operation within $\Delta 1$. We used the execution time measured using breakpoints to estimate the duration of each field operation, see Table 6.5. In Figure 6.24, we marked the areas of the atomic block's sub-traces in white fonts and dashed lines to indicate the execution time of each field operation according to their execution time measured using breakpoints.

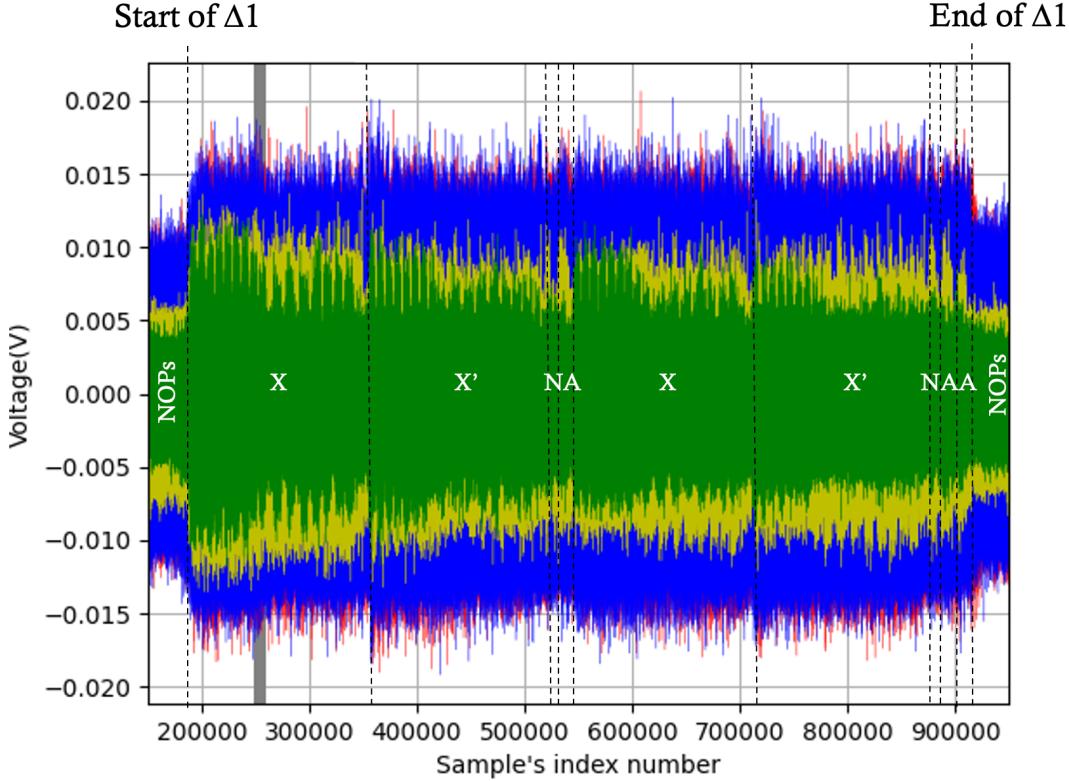


FIGURE 6.24: Identification of field operations based on the measured execution time of the operations.

Automated Simple SCA

After applying Synchronization A on all the sub-traces, we adapted the Automated Simple SCA described in [3] to analyse our synchronized and uncompressed sub-traces, focusing on the first atomic block only. Similar to [3], we search for the samples' index numbers, which the amplitudes of the set of point addition sub-traces will be completely separated from the set of point doubling sub-traces'. The notation "completely separated" means:

- at sample's index number i , the maximum value of all point doubling samples is less than the minimum value of all point addition samples, i.e., $\max_d(i) < \min_a(i)$ or
- the minimum value of all point doubling samples is greater than the maximum value of all point addition samples, i.e., $\max_a(i) < \min_d(i)$

By doing so, we can identify potentially distinguishable samples in the measured sub-traces of the atomic pattern point operations. We developed a Python program to look for the occurrence of the above two cases. However, given that perfect synchronization for every sample is unlikely, slight alignment discrepancies may arise between sub-traces over time due to unknown factors. Our program was unable to detect any occurrences of the two cases described above, when comparing the maximum and minimum values of each sample in point doubling sub-traces directly against the corresponding values in point addition sub-traces using Synchronization A. To account for these potential synchronization issues, we extended our analysis to compare a window of sample values instead of individual samples. Figure 6.25

illustrates how a 3-sample window works. The orange area represents the range between the minimum and maximum values of the point addition sub-traces, and the blue area represents the corresponding range for the point doubling sub-traces. In this example, we compare the maximum value of point doubling at sample number 3 (indicated by a yellow circle) to the minimum values of point addition at sample numbers 2 to 4 (indicated by red circles). We can see that the maximum value of point doubling at sample number 3 is lower than the minimum value of point addition at sample number 4. Please note that comparing to the traditional method difference-of-means [103] (Section 5.3.2), our proposed and implemented method allows to determine areas with a trend of capability of being distinguished. More investigations and comparison of the quality of the distinguishability of the point operation sets were not done in this work.

We performed this comparison using windows ranging from 1 to 6 samples and recorded the number of occurrences and the corresponding differences between the maximum and minimum values.

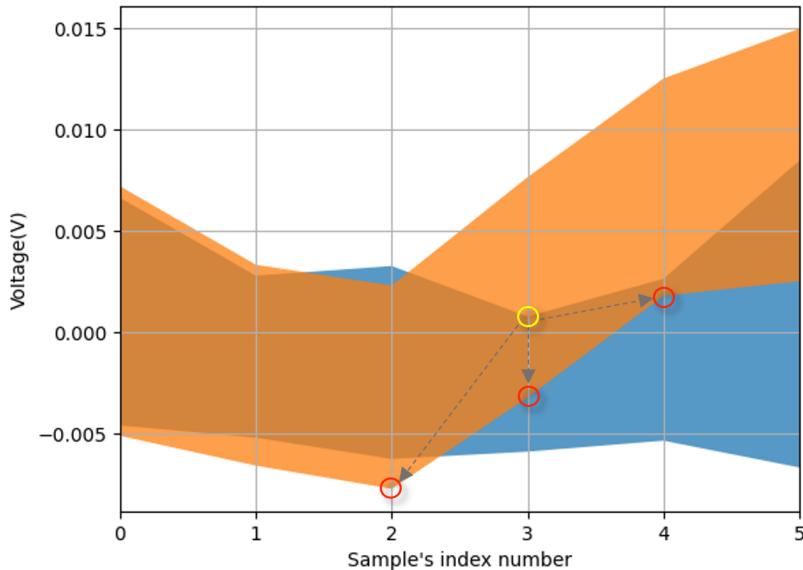


FIGURE 6.25: An example of identifying a distinguishable sample using a 3-sample window.

With the help of the program, a lot of the target samples were identified, where the set of point doubling sub-traces can be distinguished from the point addition sub-traces. We selected an area in $\Delta 1$ of all synchronized sub-traces, which assumingly refers to the first field multiplication in Montgomery space (between sample index 180,000 and 360,000, see Figure 6.26, locations α and β) to demonstrate some examples of the automatically determined separations of the atomic pattern sub-traces.

	$v \leq 0.002$	$0.002 < v \leq 0.003$	$0.003 < v \leq 0.004$	$v > 0.004$
1-sample window	0	0	0	0
2-sample window	0	0	0	0
3-sample window	6	0	0	0
4-sample window	22	0	0	0
5-sample window	84	2	0	1
6-sample window	296	16	3	1

TABLE 6.9: Number of distinguishable samples found using different sample window values, with each sample's distinguishability segmented by the voltage difference v between the two separated sets of voltages (measured in Volts).

Table 6.9 shows the number of distinguishable samples found when using the sample window values 1 to 6. Each column shows the magnitude of the distinguishability, i.e., the distance between the minimum voltage of the point addition sub-traces and the maximum voltage of the point doubling sub-traces at sample index i , or oppositely, between the minimum voltage of the point doubling sub-traces and the maximum voltage of the point addition sub-traces at sample index i . This helps us determine how distinguishable these samples are. Then, we further looked into the samples with a high degree of distinguishability.

Using a 5-sample window, we identified a notable case at sample index 338,287, where the minimum voltage of point addition sub-trace exceeded the maximum voltage of point doubling sub-trace by more than 0.004 V. This difference serves as a fairly significant indicator of distinguishability. See Figure 6.27, which is located at indicator α in Figure 6.26, approximately at the end of the first Montgomery modular multiplication operation in $\Delta 1$. We outlined the distinguishable area of the graph in red.

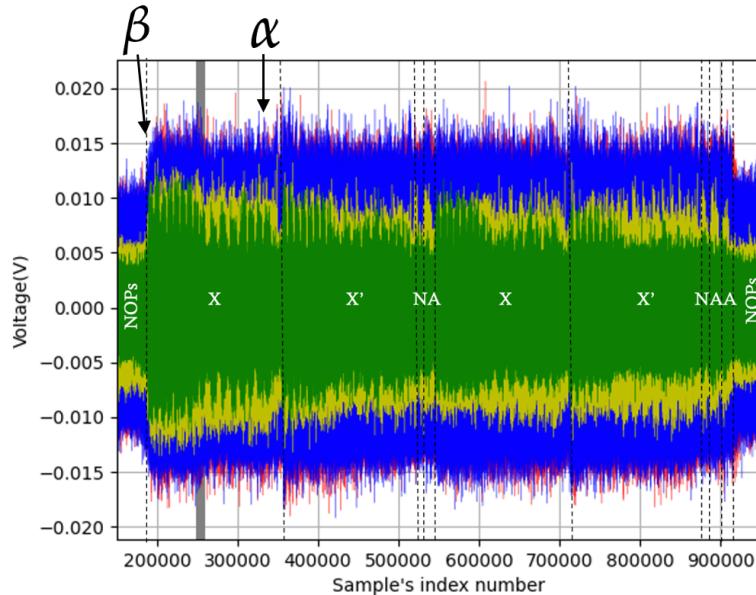


FIGURE 6.26: Indication of the locations of samples in Figures 6.27 (β) and 6.28 (α) in $\Delta 1$, with each atomic pattern's duration estimated from the execution time measured in Chapter 6.2.2.

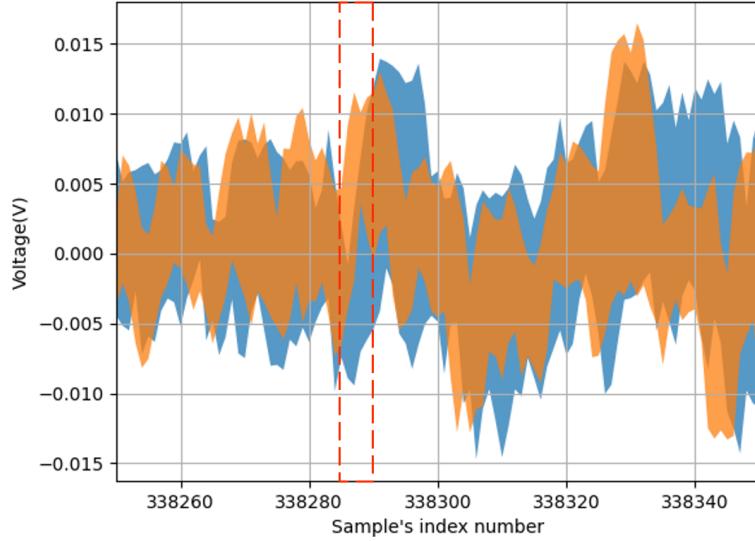


FIGURE 6.27: Minimum and maximum voltages range for point addition (orange) and point doubling (blue) operations showing a distinguishable sample at index 338,287 with $v \geq 0.004$ V in a 5-sample window.

Having identified many other potential distinguishability indicators in Table 6.9 by analyzing the distances between maximum and minimum voltages of the samples from two point operation sets, we used another technique to narrow our focus on identifying the most effective indicators. In addition to comparing these voltage differences, we also investigated consecutive distinguishable samples using various sample windows to find long sequences of distinguishable samples that could signal success in SCA. The results are shown in Table 6.10. Our analysis shows that no consecutive distinguishable samples are found when using a sample window smaller than 5. With a 5-sample window, we identified 3 instances of two consecutive distinguishable samples; this number increased to 15 occurrences with a 6-sample window. However, no sequences longer than two consecutive distinguishable samples were identified.

	$s = 2$	$s \geq 2$
1-sample window	0	0
2-sample window	0	0
3-sample window	0	0
4-sample window	0	0
5-sample window	3	0
6-sample window	15	0

TABLE 6.10: Count of occurrences of consecutive distinguishable samples (s) across different sample windows.

By combining the techniques of distance comparison and the detection of consecutive distinguishable samples, we identified the only case that exhibits both a relatively large distance between the minimum and maximum voltages of different operations and two consecutive distinguishable samples, as shown in Figure 6.28. This case is located at the beginning of Δ_1 , marked as β in Figure 6.26. At sample numbers 186,810 and 186,811, the minimum voltages of point addition sub-traces

are higher than the maximum voltages of point doubling sub-traces in a 5-sample window, by a distinguishable distance of 0.002 V to 0.003 V.

Although no longer streaks were found to indicate more significant distinguishability, it is noteworthy that in these two cases, which represent our best distinguishability indicators, the adjacent samples from point doublings and point additions appear well synchronized, with their maximum and minimum ranges closely aligned.

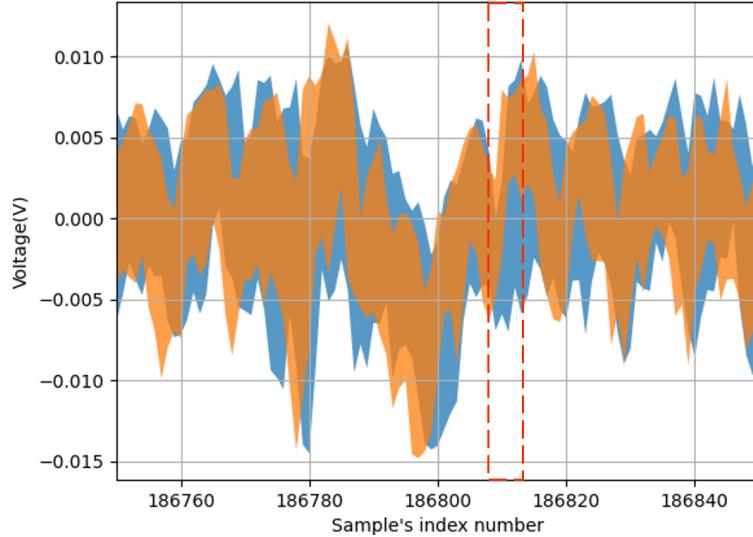


FIGURE 6.28: Minimum and maximum voltages range for point addition (orange) and point doubling (blue) operations showing distinguishable samples with indices 186,810 and 186,811, in voltage range $0.002 \text{ V} < v \leq 0.003 \text{ V}$ in a 5-sample window.

6.3.2 Analysis of *kP* Operation Executed in Flash Memory

Synchronization Alignment

Since we used 100 Mega samples (MS) per second in the oscilloscope and 20 MHz clock rate in the CCS program run in flash memory of the board, the trace was captured with only 5 samples per clock cycle, which is very low and makes it difficult to obtain meaningful insights from any analysis. Nonetheless, we split the trace into sub-traces using similar methods we used on RAM traces as in Chapter 6.3.1.

First, we identified the noise and the *kP* operation in the full trace, as shown in Figure 6.29 and a magnified version in Figure 6.30. Then, with the help of NOPs inserted as described in Chapter 6.3.1, we were able to separate each atomic block in point doubling and point addition operations into sub-traces, as shown in Figure 6.31.

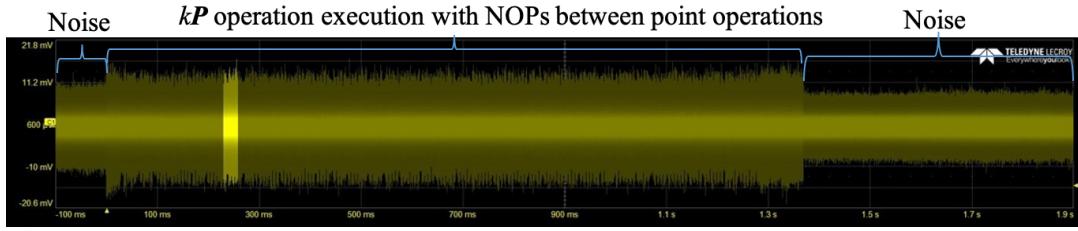


FIGURE 6.29: EM trace of noise and the implemented atomic pattern *kP* algorithm executed in flash memory of the board attacked.

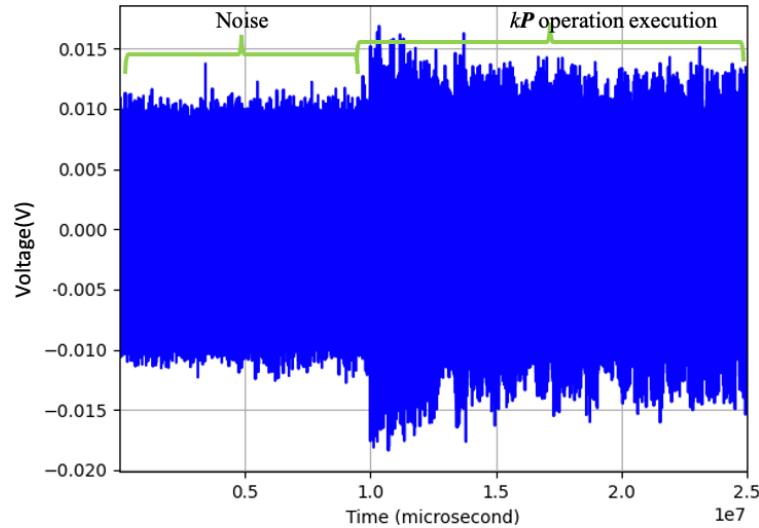


FIGURE 6.30: EM trace of noise and the beginning of the *kP* algorithm executed in flash memory of the board attacked.

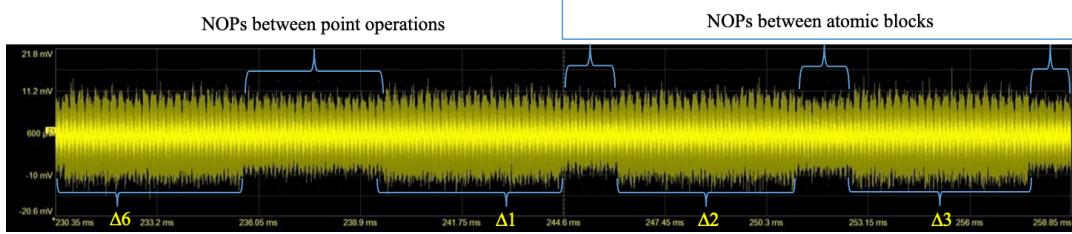


FIGURE 6.31: EM trace of atomic blocks of point operations of the kP algorithm executed in flash memory of the board attacked.

We tried to synchronize the sub-traces by using a simple averaging method, which is to first convert each sample in each sub-trace into the average voltage value of its closest 10 samples, and then try to align the samples with another sub-trace's manually using Excel charts. Despite our efforts in manual synchronization, the processed sub-traces cannot be synchronized. It is important to recognize that both the low number of samples per clock cycle and the high noise level are significant problems hindering the success of simple SCA. Therefore, unfortunately we cannot further carry out simple electromagnetic analysis attack on the kP operation executed in flash memory like what we did for the kP operation executed in RAM.

6.3.3 Comparison of the Results to Sigourou et al.'s Work [27]

As mentioned in Chapter 3, no investigation of the SCA-resistance of Longa's atomic patterns was performed experimentally prior to this work. Our objectives, code implementation and experimental setup closely resemble those used in the SCA study by Sigourou et al. [27]. In this section, we compare our work to [27] to identify any overarching findings. A short summary is given in Table 6.11 below.

	Sigourou et al. [27]	This work
kP algorithm	Double-and-add L-to-R	Double-and-add L-to-R
Atomic patterns	Rondepierre [23]	Longa [2]
Cryptographic library	FLECC	FLECC
Embedded device	LAUNCHXL-F280025C	LAUNCHXL-F28379D
Samples per clock cycle	50	RAM: 10 Flash: 5
Signal-to-noise ratio	≈ 2.69	≈ 1.36
No. of samples per point operation	PD: 1,970,000 PA: 1,970,000	PD: 3,200,000 PA: 4,800,000
File size	2.7 GB	RAM: 6.02 GB Flash: 5.86 GB
SCA type	Horizontal	Horizontal
SCA method	Simple Power Analysis	Automated Simple SCA
Major findings	- SCA leakage in 1 st and 4 th OPs - Desynchronization found - FLECC is not constant-time	- Negative SSCA results - Desynchronization found - FLECC is constant-time

TABLE 6.11: Summary of comparison between [27] by Sigourou et al. and our work.

Both works use FLECC for the implementation of the binary double-and-add left-to-right (L-to-R) kP algorithm (Algorithm 2). Sigourou et al. used Rondepierre's

atomic patterns [23] while we used Longa's [2], both with the insertion of NOPs in the algorithm to ease the separation of sub-traces.

Similar to the finding of Sigourou et al., we noticed some desynchronization in our aligned sub-traces, which led us to investigate the execution time of the supposed constant-runtime functions in FLECC. Our experimental board enables us to use one hardware breakpoint within the implemented code to measure the duration of different operations, offering an alternative to visually analysing the trace. We repeatedly measured the execution time of the "constant-runtime" Montgomery modular multiplication, negation and addition functions in FLECC and confirmed that they are indeed constant-runtime, with only a minor fluctuation of 1 clock cycle in the runtime of Montgomery modular multiplication. This evidence supports that FLECC is truly constant-runtime, contrary to the claim made in [27].

Sigourou et al. performed a Simple Power Analysis in their work, while we applied Automated Simple SCA. They identified the part of the kP trace, which allows to clearly differentiate between PD and PA operations. We were not able to show a high degree of distinguishability using the distinguishability indicators we found. Thus, our atomic patterns kP implementation executed in our target embedded device is resistant against our applied SSCA techniques. They assumed the distinguishability is caused by the different addresses of the registers in memory and the Hamming distance between these addresses. Further research is required to validate these assumptions. In our work, due to a small number of samples per clock cycle, the vulnerability to horizontal address-bit SCA cannot be confirmed nor ruled out.

Chapter 7

Conclusion

Atomic patterns were introduced as a countermeasure against simple SCA attacks targeting EC kP algorithms. This work primarily aims to experimentally investigate the resistance of Longa's atomic patterns [2] to simple SCA attacks, with the goal of applying these atomic patterns in EC-based cryptographic protocols for embedded devices using an open-source cryptographic library. In Longa's work, atomic pattern algorithms with 4, 6 and 8 atomic blocks were proposed to realize EC point doubling, point addition and point tripling operations respectively. Given the frequent use of binary kP algorithms in the implementation of cryptographic operations for embedded devices, our work focused specifically on Longa's atomic patterns for point doubling and point addition which are essential operations for double-and-add kP algorithms, selecting the atomic pattern MNAMNAA for our investigation.

As the first step of our investigation, we performed an exhaustive search of literature citing Longa's work [2]. Our goal was to determine whether the resistance of Longa's atomic patterns to SSCA or other horizontal SCA attacks had been experimentally tested. As a result, we identified 63 papers citing [2] across various contexts. Some papers referred [2] as state-of-the-art SSCA countermeasure, some others focused on improving the efficiency of kP algorithms. Only a few papers discussed the potential vulnerabilities of the atomic patterns to horizontal SCA attacks. However, none of these papers experimentally examined the SCA-resistance of Longa's atomic patterns.

To address this gap, we implemented a binary left-to-right double-and-add kP algorithm using P-256, which is a recommended EC for ECDSA and EC key establishment by NIST since 2024. Our implementation was tested on a LAUNCHXL-F28379D LaunchPad board with a MS320F28379D C2000 32-bit dual-core microcontroller. We evaluated three open-source cryptographic libraries for their suitability to meet our objectives and selected the library FLECC in C, which is proven to offer constant-runtime functions.

During the implementation, we identified several mistakes in Longa's atomic patterns published in [2]. The atomic patterns were corrected corresponding to the point doubling and mixed point addition formulae proposed by Longa for the following cases:

- MNAMNAA-based atomic point doubling ([2], Appendix B2)
- MANA-based atomic mixed addition ([2], Appendix B3)
- MNAMNAA-based atomic mixed addition ([2], Appendix B4)
- MANA-based atomic special addition with identical Z-coordinate ([2], Appendix B5)
- MNAMNAA-based atomic special addition ([2], Appendix B6)

- MANA-based atomic point tripling ([2], Appendix B7)
- MNAMNAA-based atomic point tripling ([2], Appendix B8)

To evaluate the resistance of the implemented kP algorithm against SSCA, we performed a horizontal electromagnetic analysis attack, measuring the EM emanations of the target board during a kP execution. Due to technical limitations, we used a 22-bit scalar k and focused on analysing only the first atomic blocks in EC point doublings and additions. Automated Simple SCA was conducted to identify sample indices where the sub-traces of point doubling significantly differ from those of point addition. However, no significant difference was found simply using this technique. We further enhanced this technique by developing two additional methods - distance comparison and detection of consecutive distinguishable samples - to identify trends. Specifically, we used Automated Simple SCA to compare the amplitudes of the synchronized samples in both sets, as well as the amplitudes of the neighboring samples within a small window of samples.

Our analysis revealed no significant differences in execution time or the shape of the atomic block sub-traces, likely due to high noise levels, an insufficient number of samples per clock cycle, long execution time of the kP implementation using FLECC library and the large file size of the EM trace. Based on the knowledge of the limitations described in this thesis, further investigation is required to conclusively determine whether Longa's atomic patterns in the kP algorithm are vulnerable to single-trace SCA attacks, particularly horizontal address-bit SCA.

Chapter 8

Appendix

8.1 Results of Literature Overview

The 63 papers referenced in the column "Reference" in Table 8.1 below all truly cited Longa's master thesis [2].

The column "Theme" represents the topic of discussion of the reference paper.

The column "State-of-the-art" represents whether the reference paper considers Longa's MNAMNAA atomic pattern as an example of atomic patterns in the context of state-of-the-art countermeasure against simple SCA attacks.

"Basis for improvement" column shows whether the author(s) of the reference paper used Longa's MNAMNAA-pattern(s) for their theoretical investigations or in practical experiments.

The column "Analysed" shows whether the reference paper analysed the efficiency or SCA-resistance of Longa's MNAMNAA-pattern(s) theoretically, or made claims to Longa's pattern(s) based on analysis done on other similar atomic patterns.

The column "Implemented" represents whether the author(s) actually implemented Longa's MNAMNAA atomic pattern(s) on hardware in their work.

The column "Used for *kP*" represents whether the reference paper has made an improvement on *kP* algorithm without any change on atomic patterns.

The column "Remarks" describes in which context [2] was referred to in the reference paper.

Reference	Theme	State-of-the-art	Basis for improvement	Analysed	Implemented	Used for kP	Remarks
[28]	SCA	Y	Y	Y	N	Y	Introduced general doubling and readdition algorithms using Longa's atomic patterns.
[31]	SCA	Y	Y	Y	N	Y	Used Longa's MNAMNAA pattern to build readdition and general doubling algorithms
[30]	SCA	Y	Y	Y	N	Y	Provided a general framework that can accommodate Longa's base-{2,3} number systems, <i>wmbNAF</i> representations and atomic patterns.
[23]	SCA	Y	Y	Y	N	N	Compared the performance of his new atomic pattern algorithm with Longa's.
[12]	SCA	Y	Y	Y	N	N	Theoretically applied their novel attack with input randomization against Longa's atomic implementation.
[32]	SCA	Y	N	Y	N	N	Showed vulnerabilities in Longa's atomic pattern against HCCA.
[33]	SCA	Y	N	N	N	N	Mentioned Longa's atomic pattern as state-of-the-art scalar multiplication algorithm.
[3, 21, 27, 34]	SCA	Y	N	N	N	N	Mentioned Longa's atomic pattern as state-of-the-art SCA countermeasure.
[35]	EC kP efficiency	Y	N	N	N	Y	Evaluated costs of point doubling and addition in doubling-addition $2P + Q$ formula from [2].
[37]	SCA	Y	N	N	N	Y	Mentioned Longa's atomic pattern as state-of-the-art scalar multiplication algorithm.
[38-40]	SCA	Y	N	N	N	N	Mentioned Longa's atomic patterns as state-of-the-art SPA countermeasure.
[36]	EC kP efficiency	Y	N	N	N	Y	Compared the costs of Longa's point tripling and special point addition formulae with others.
[84]	EC kP efficiency	N	N	N	N	Y	Mentioned Longa's <i>mbNAF</i> method.

Reference	Theme	State-of-the-art	Basis for improvement	Analysed	Implemented	Used for kP	Remarks
[73, 75]	EC kP efficiency	N	N	N	N	Y	Mentioned Longa's generic multibase scalar representations.
[85]	EC kP efficiency	N	N	N	N	Y	Mentioned Longa's <i>mbNAF</i> and <i>wmbNAF</i> representations and compared them with the proposed representation.
[86]	EC kP efficiency	N	N	N	N	Y	Mentioned Longa's doubling-addition as state-of-the-art formula and implemented it to compare point multiplication costs.
[74]	EC kP efficiency	N	N	N	N	Y	Implemented Longa's doubling-addition algorithm.
[76]	EC kP efficiency	N	N	N	N	Y	Mentioned Longa's multibase single scalar multiplication, as it is very similar to the authors' work.
[41-72], [77-83]	Misc.	N	N	N	N	Misc.	Not relevant to our goal of study in this thesis.

TABLE 8.1: Literature overview: aspects relevant for this thesis were taken as the basis for classification of papers referring to [2]

8.2 Revision of Longa's Atomic Pattern Algorithms

We reviewed all of Longa's atomic pattern and SSCA-protected algorithms (Appendices B1-B10, C1-C3 and E1-E3 in [2]). Below are the atomic patterns in which we have found errors (Appendices B2, B3, B4, B5, B6, B7 and B8 in [2]). The corrections are shown in bold text with yellow highlights in the following.

8.2.1 Longa's Appendix B2 algorithm revised

The MNAMNAA-based atomic point doubling algorithm taken from [2](Appendix B2) is revised as follows:

Input: $P = (X_1, Y_1, Z_1)$

Output: $2P = (X_3, Y_3, Z_3)$

$T_1 \leftarrow X_1, T_2 \leftarrow Y_1, T_3 \leftarrow Z_1$

$\Delta 1$	$\Delta 2$
$T_4 = T_3^2$	$T_5 = T_4 \times T_5$ $(A.B)$
*	*
$T_5 = T_1 + T_4$ $(A = X_1 + Z_1^2)$	$T_4 = T_5 + T_5$ $(2A.B)$
$T_6 = T_2^2$	$T_3 = T_2 \times T_3$ (Z_3)
$T_4 = -T_4$	*
$T_2 = T_2 + T_2$	$T_4 = T_4 + T_5$ (α)
$T_4 = T_1 + T_4$ $(B = X_1 - Z_1^2)$	$T_2 = T_6 + T_6$ $(2Y_1^2)$
$\Delta 3$	$\Delta 4$
$T_5 = T_4^2$	$T_2 = T_2^2$ $(4Y_1^4)$
*	$T_5 = -T_1$ $(-X_3)$
$T_6 = T_2 + T_2$	$T_5 = T_5 + T_6$ $(\beta - X_3)$
$T_6 = T_1 \times T_6$	$T_5 = T_4 \times T_5$ $(\alpha(\beta - X_3))$
$T_1 = -T_6$	$T_2 = -T_2$ $(-4Y_1^4)$
$T_1 = T_1 + T_1$	$T_2 = T_2 + T_2$ $(-8Y_1^4)$
$T_1 = T_1 + T_5$	$T_2 = T_2 + \textcolor{yellow}{T}_5$ (Y_3)

TABLE 8.2: Revised MNAMNAA-based atomic point doubling. The newly revised register is shown in bold with yellow highlight.

8.2.2 Longa's Appendix B3 algorithm revised

The MANA-based atomic mixed addition algorithm taken from [2](Appendix B3) is revised as follows:

Input: $P = (X_1, Y_1, Z_1)$ and $Q = (X_2, Y_2)$

Output: $P + Q = (X_3, Y_3, Z_3, X'_1, Y'_1)$

$T_1 \leftarrow X_1, T_2 \leftarrow Y_1, T_3 \leftarrow Z_1, T_x \leftarrow X_2, T_y \leftarrow Y_2$

$\Delta 1$	$\Delta 2$
$T_4 = T_3^2$ * * *	$T_5 = T_x \times T_4$ * $T_6 = -T_1$ $T_5 = T_5 + T_6$ ($A = Z_1^2 X_2 - X_1$)
$\Delta 3$	$\Delta 4$
$T_6 = T_5^2$ * * *	$T_7 = T_1 \times T_6$ $T_8 = T_1 + T_1$ * *
$\Delta 5$	$\Delta 6$
$T_9 = T_5 \times T_6$ $T_8 = T_8 + T_9$ * *	$T_4 = T_3 \times T_4$ * * *
$\Delta 7$	$\Delta 8$
$T_4 = T_y \times T_4$ * $T_{10} = -T_2$ $T_4 = T_4 + T_{10}$ ($B = Z_1^3 Y_2 - Y_1$)	$T_{10} = T_4^2$ * $T_8 = -T_8$ $T_1 = \mathbf{T}_4 + T_8$ (X_3)
$\Delta 9$	$\Delta 10$
$T_8 = T_2 \times T_9$ * $T_6 = -T_1$ $T_6 = T_6 + T_7$ ($X'_1 - X_3$)	$T_6 = T_6 \times \mathbf{T}_4$ * $T_9 = -T_8$ $T_2 = T_6 + T_9$ ($-Y'_1$) (Y_3)
$\Delta 11$	
$T_3 = T_3 \times T_5$ * $T_4 = -T_7$ $T_4 = T_1 + T_4$ ($X_3 - X'_1$)	

TABLE 8.3: Revised MANA-based atomic mixed addition. The newly revised registers are shown in bold with yellow highlight.

8.2.3 Longa's Appendix B4 algorithm revised

The MNAMNAA-based atomic mixed addition algorithm taken from [2](Appendix B4) is revised as follows:

Input: $P = (X_1, Y_1, Z_1)$ and $Q = (X_2, Y_2)$

Output: $P + Q = (X_3, Y_3, Z_3, X'_1, Y'_1)$

$T_1 \leftarrow X_1, T_2 \leftarrow Y_1, T_3 \leftarrow Z_1, T_x \leftarrow X_2, T_y \leftarrow Y_2$

$\Delta 1$	$\Delta 2$		
$T_4 = T_3^2$ * * $T_5 = T_x \times T_4$ $T_6 = -T_1$ $T_5 = T_5 + T_6$ *	(Z_1^2) $(Z_1^2 X_2)$ $(-X_1)$ $(A = Z_1^2 X_2 - X_1)$ *	$T_6 = T_5^2$ * * $T_7 = T_1 \times T_6$ * $T_8 = \mathbf{T}_7 + \mathbf{T}_7$ *	(A^2) (X'_1) $(2X'_1)$
$\Delta 3$	$\Delta 4$		
$T_9 = T_5 \times T_6$ * $T_8 = T_8 + T_9$ $T_4 = T_3 \times T_4$ * * *	(A^3) $(A^3 + 2X'_1)$ (Z_1^3) * * *	$T_4 = T_y \times T_4$ $T_{10} = -T_2$ $T_4 = T_4 + T_{10}$ $T_{10} = T_4^2$ $T_8 = -T_8$ $T_1 = \mathbf{T}_{10} + T_8$ *	$(Z_1^3 Y_2)$ $(-Y_1)$ $(B = Z_1^3 Y_2 - Y_1)$ (B^2) $(-A^3 - 2X'_1)$ (X_3)
$\Delta 5$	$\Delta 6$		
$T_8 = T_2 \times T_9$ $T_6 = -T_1$ $T_6 = T_6 + T_7$ $T_6 = T_6 \times \mathbf{T}_4$ $T_9 = -T_8$ $T_2 = T_6 + T_9$ *	(Y'_1) $(-X_3)$ $(X'_1 - X_3)$ $(B(X'_1 - X_3))$ $(-Y'_1)$ (Y_3) *	$T_3 = T_3 \times T_5$ $T_4 = -T_7$ $T_4 = T_1 + T_4$ $T_5 = T_4^2$ $T_6 = -T_8$ $T_6 = T_2 + T_6$ *	(Z_3) $(-X'_1)^{(a)}$ $(X_3 - X'_1)^{(a)}$ $(A^2)^{(a)}$ $(-Y'_1)^{(a)}$ $(B)^{(a)}$

TABLE 8.4: Revised MNAMNAA-based atomic mixed addition. The newly revised registers are shown in bold with yellow highlight.

(a) Field operations in $\Delta 6$ if a special addition follows.

8.2.4 Longa's Appendix B5 algorithm revised

The MANA-based atomic special addition with identical Z-coordinate algorithm taken from [2](Appendix B5) is revised as follows. The variables X'_1 and Y'_1 are outputs from the MANA-based atomic mixed addition algorithm.

Input: $P = (X_1, Y_1, Z)$ and $Q = (X_2, Y_2, Z)$

Output: $P + Q = (X_3, Y_3, Z_3)$

$T_1 \leftarrow \mathbf{X}_2, T_2 \leftarrow \mathbf{Y}_2, T_3 \leftarrow Z, T_7 \leftarrow \mathbf{X}'_1, T_8 \leftarrow \mathbf{Y}'_1$

$\Delta 1$	$\Delta 2$
*	$T_5 = T_4 \times T_4 \quad (A^2)$
*	*
$T_4 = -T_7 \quad (-X'_1)$	$T_6 = -T_8 \quad (-Y'_1)$
$T_4 = T_1 + T_4 \quad (A = X_2 - X'_1)$	$T_6 = T_2 + T_6 \quad (B = Y_2 - Y'_1)$
$\Delta 3$	$\Delta 4$
$T_7 = T_5 \times T_7 \quad (X'_1 A^2)$	$T_5 = T_4 \times T_5 \quad (A^3)$
$T_9 = T_7 + T_7 \quad (2X'_1 A^2)$	$T_9 = T_5 + T_9 \quad (A^3 + 2X'_1 A^2)$
*	*
*	*
$\Delta 5$	$\Delta 6$
$T_1 = T_6 \times T_6 \quad (B^2)$	$T_2 = T_5 \times T_8 \quad (Y'_1 A^3)$
*	*
$T_9 = -T_9 \quad (-A^3 - 2X'_1 A^2)$	$T_5 = -T_1 \quad (-X_3)$
$T_1 = T_1 + T_9 \quad (X_3)$	$T_5 = T_5 + T_7 \quad (C = X'_1 A^2 - X_3)$
$\Delta 7$	$\Delta 8$
$T_5 = T_5 \times T_6 \quad (B.C)$	$T_3 = T_3 \times T_4 \quad (Z_3)$
*	*
$T_2 = -T_2 \quad (-Y'_1 A^3)$	*
$T_2 = T_2 + T_5 \quad (Y_3)$	*

TABLE 8.5: Revised MANA-based atomic special addition with identical Z-coordinate. The newly revised registers are shown in bold with yellow highlight.

8.2.5 Longa's Appendix B6 algorithm revised

The MNAMNAA-based atomic special addition algorithm with identical Z-coordinate taken from [2](Appendix B6) is revised as follows. The variables X'_1 and Y'_1 are outputs from the MNAMNAA-based atomic mixed addition algorithm.

Input: $P = (X_1, Y_1, Z)$ and $Q = (X_2, Y_2, Z)$

Output: $P + Q = (X_3, Y_3, Z_3)$

$T_1 \leftarrow \mathbf{X}_2, T_2 \leftarrow \mathbf{Y}_2, T_3 \leftarrow Z, T_7 \leftarrow \mathbf{X}'_1, T_8 \leftarrow \mathbf{Y}'_1$

$\Delta 1$	$\Delta 2$
*	
$T_4 = -T_7$	$T_7 = T_5 \times T_7 \quad (X'_1 A^2)$
$T_4 = T_1 + T_4 \quad (A = X_2 - X'_1)$	*
$T_5 = T_4^2 \quad (A^2)$	$T_9 = T_7 + T_7 \quad (2X'_1 A^2)$
$T_6 = -T_8 \quad (-Y'_1)$	$T_5 = T_4 \times T_5 \quad (A^3)$
$T_6 = T_2 + T_6 \quad (B = Y_2 - Y'_1)$	*
*	$T_9 = T_5 + T_9 \quad (A^3 + 2X'_1 A^2)$
	*
$\Delta 3$	$\Delta 4$
$T_1 = T_6^2 \quad (B^2)$	$T_5 = T_5 \times T_6 \quad (B.C)$
$T_9 = -T_9 \quad (-A^3 - 2X'_1 A^2)$	$T_2 = -T_2 \quad (-Y'_1 A^3)$
$T_1 = T_1 + T_9 \quad (X_3)$	$T_2 = T_2 + T_5 \quad (Y_3)$
$T_2 = T_5 \times T_8 \quad (Y'_1 A^3)$	$T_3 = T_3 \times T_4 \quad (Z_3)$
$T_5 = -T_1 \quad (-X_3)$	*
$T_5 = T_5 + T_7 \quad (C = X'_1 A^2 - X_3)$	*
*	*

TABLE 8.6: Revised MNAMNAA-based atomic special addition with identical Z-coordinate. The newly revised registers are shown in bold with yellow highlight.

8.2.6 Longa's Appendix B7 algorithm revised

The MANA-based atomic point tripling algorithm taken from [2](Appendix B7) is revised as follows.

Input: $P = (X_1, Y_1, Z_1)$

Output: $3P = (X_3, Y_3, Z_3, X_2, Y_2)$

$T_1 \leftarrow X_1, T_2 \leftarrow Y_1, T_3 \leftarrow Z_1$

$\Delta 1$	$\Delta 2$
$T_4 = T_3^2 \quad (Z_1^2)$	$T_3 = T_2 \times T_3 \quad (Z'_1)$
$T_3 = T_3 + T_3 \quad (2Z_1)$	$T_4 = T_1 + T_4 \quad (A = X_1 + Z_1^2)$
$T_5 = -T_4 \quad (-Z_1^2)$	*
*	$T_5 = T_1 + T_5 \quad (B = X_1 - Z_1^2)$
$\Delta 3$	$\Delta 4$
$T_4 = T_4 \times T_5 \quad (A.B)$	$T_2 = T_2^2 \quad (Y_1^2)$
$T_5 = T_4 + T_4 \quad (2A.B)$	$T_4 = T_4 + T_5 \quad (C = 3A.B)$
*	*
*	$T_2 = T_2 + T_2 \quad (2Y_1^2)$
$\Delta 5$	$\Delta 6$
$T_5 = T_4^2 \quad (C^2)$	$T_6 = T_1 \times T_2 \quad (X'_1)$
$T_1 = T_1 + T_1 \quad (2X_1)$	*
*	$T_1 = -T_6 \quad (-X'_1)$
*	$T_1 = T_1 + T_1 \quad (-2X'_1)$
$\Delta 7$	$\Delta 8$
$T_2 = T_2^2 \quad (4Y_1^4)$	$T_4 = T_1 \times T_4 \quad (C.D)$
$T_7 = T_1 + T_5 \quad (X_2)$	*
$T_1 = -T_7 \quad (-X_2)$	$T_2 = -T_2 \quad (-4Y_1^4)$
$T_1 = T_1 + T_6 \quad (D = X'_1 - X_2)$	$T_2 = T_2 + T_2 \quad (-Y'_1)$
$\Delta 9$	$\Delta 10$
$T_5 = T_1^2 \quad (D^2)$	$T_6 = T_5 \times T_6 \quad (F = X'_1 D^2)$
$T_8 = T_2 + T_4 \quad (Y_2)$	*
*	$T_9 = -T_1 \quad (-D)$
$T_4 = T_2 + T_8 \quad (E = Y_2 - Y'_1)$	*
$\Delta 11$	$\Delta 12$
$T_5 = \mathbf{T_9} \times T_5 \quad (-D^3)$	$T_{10} = T_4^2 \quad (E^2)$
$T_1 = T_6 + T_6 \quad (2F)$	$T_1 = T_1 + T_{10} \quad (X_3)$
$T_1 = -T_1 \quad (-2F)$	$T_{10} = -T_1 \quad (-X_3)$
$T_1 = T_1 + T_5 \quad (-D^3 - 2F)$	*
$\Delta 13$	$\Delta 14$
$T_5 = T_2 \times T_5 \quad (Y'_1 D^3)$	$T_2 = T_2 \times T_4 \quad (E(F - X_3))$
$T_2 = T_6 + T_{10} \quad (F - X_3)$	*
*	$T_5 = -T_5 \quad (-Y'_1 D^3)$
*	$T_2 = T_2 + T_5 \quad (Y_3)$
$\Delta 15$	
$T_3 = T_3 \times T_9 \quad (Z_3)$	
*	
$T_4 = -T_7 \quad (-X_2)^{(a)}$	
$T_4 = T_1 + T_4 \quad (X_3 - X_2)^{(a)}$	

TABLE 8.7: Revised MANA-based atomic point tripling. The newly revised register is shown in bold with yellow highlight.

(a) Field operations in $\Delta 15$ if a special addition follows.

8.2.7 Longa's Appendix B8 algorithm revised

The MNAMNAA-based atomic point tripling algorithm taken from [2](Appendix B8) is revised as follows.

Input: $P = (X_1, Y_1, Z_1)$

Output: $3P = (X_3, Y_3, Z_3, X_2, Y_2)$

$T_1 \leftarrow X_1, T_2 \leftarrow Y_1, T_3 \leftarrow Z_1$

$\Delta 1$	$\Delta 2$
$T_4 = T_3^2 \quad (Z_1^2)$ *	$T_4 = T_4 \times T_5 \quad (A.B)$ *
$T_3 = T_3 + T_3 \quad (2Z_1)$ $T_3 = T_2 \times T_3 \quad (Z'_1)$ $T_5 = -T_4 \quad (-Z_1^2)$ $T_4 = T_1 + T_4 \quad (A = X_1 + Z_1^2)$ $T_5 = T_1 + T_5 \quad (B = X_1 - Z_1^2)$	$T_5 = T_4 + T_4 \quad (2A.B)$ $T_2 = T_2^2 \quad (Y_1^2)$ *
$\Delta 3$	$\Delta 4$
$T_5 = T_4^2 \quad (C^2)$ *	$T_2 = T_2^2 \quad (4Y_1^4)$ $T_1 = -T_7 \quad (-X_2)$
$T_1 = T_1 + T_1 \quad (2X_1)$ $T_6 = T_1 \times T_2 \quad (X'_1)$ $T_1 = -T_6 \quad (-X'_1)$ $T_1 = T_1 + T_1 \quad (-2X'_1)$ $T_7 = T_1 + T_5 \quad (X_2)$	$T_1 = T_1 + T_6 \quad (D = X'_1 - X_2)$ $T_4 = T_1 \times T_4 \quad (C.D)$ $T_2 = -T_2 \quad (-4Y_1^4)$ $T_2 = T_2 + T_2 \quad (-Y'_1)$ $T_8 = T_2 + T_4 \quad (Y_2)$
$\Delta 5$	$\Delta 6$
$T_5 = T_1^2 \quad (D^2)$ *	$T_5 = \mathbf{T}_9 \times T_5 \quad (-D^3)$ *
$T_4 = T_2 + T_8 \quad (E = Y_2 - Y'_1)$ $T_6 = T_5 \times T_6 \quad (F = X'_1 D^2)$ $T_9 = -T_1 \quad (-D)$ *	$T_1 = T_6 + T_6 \quad (2F)$ $T_{10} = T_4^2 \quad (E^2)$ $T_1 = -T_1 \quad (-2F)$ $T_1 = T_1 + T_5 \quad (-D^3 - 2F)$ $T_1 = T_1 + T_{10} \quad (X_3)$
$\Delta 7$	$\Delta 8$
$T_5 = T_2 \times T_5 \quad (Y'_1 D^3)$ $T_{10} = -T_1 \quad (-X_3)$ $T_2 = T_6 + T_{10} \quad (F - X_3)$ $T_2 = T_2 \times T_4 \quad (E(F - X_3))$ $T_5 = -T_5 \quad (-Y'_1 D^3)$ $T_2 = T_2 + T_5 \quad (Y_3)$ *	$T_3 = T_3 \times T_9 \quad (Z_3)$ $T_4 = -T_7 \quad (-X_2)^{(a)}$ $T_4 = T_1 + T_4 \quad (X_3 - X_2)^{(a)}$ $T_5 = T_4^2 \quad (A^2)^{(a)}$ $T_6 = -T_8 \quad (-Y_2)^{(a)}$ $T_6 = T_2 + T_6 \quad (B)^{(a)}$ *

TABLE 8.8: Revised MNAMNAA-based atomic point tripling. The newly revised register is shown in bold with yellow highlight.

(a) Field operations in $\Delta 8$ if a special addition follows.

8.3 Distinguishability of Doubling 1 and Doubling 2

By focusing on the first atomic block (with about 800,000 samples) from the sub-traces of Doubling 1 and Doubling 2, we overlay the synchronized sub-traces, so they have matching patterns in the beginning of $\Delta 1$. At 20 μs (20,000 samples) after where we have initially synchronized the patterns, the traces become non-synchronous, as shown in Figure 8.1 (a). The red trace (Doubling 2) has a delay of about 50 ns (50 samples, i.e., 5 clock cycles) when compared to the blue trace (Doubling 1). The discrepancy is then corrected as shown in Figure 8.1 (b).

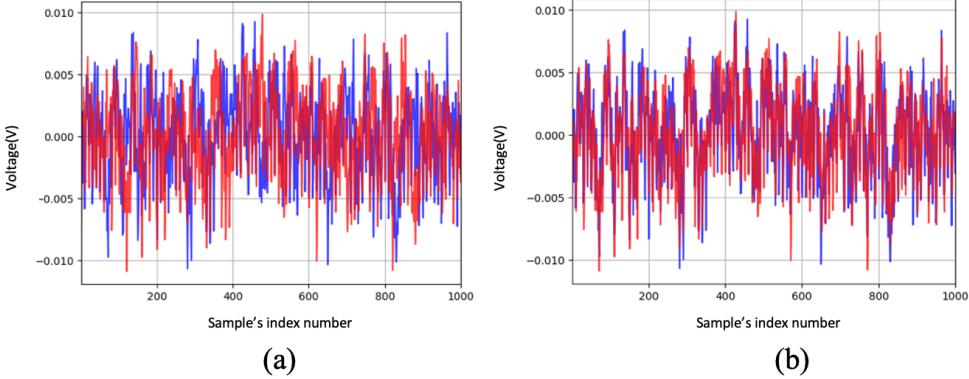


FIGURE 8.1: (a) Showing Doubling 1 (blue) and Doubling 2 (red) are non-synchronous at 20,000 samples after the previous synchronization, (b) synchronous again after shifting Doubling 2 by 50 samples left.

Since we are unable to identify and separate each arithmetic field operation within an atomic block in the sub-traces exactly, as we did not place any NOPs in between, we studied the execution time of the FLECC constant-time field operations using breakpoints. It was found that the constant-time Montgomery modular multiplication, negation and addition operations from FLECC are quite constant-time, according to our extensive study in Chapter 6.2.2. Despite of that, the 5-clock cycle difference causing the delay in the red line in Figure 8.1 (above) aligns with the duration difference between Doubling 1 and Doubling 2 measured using breakpoints (see Table 6.7 in Chapter 6.3.1). Further investigation is necessary to determine the reasons for this delay.

8.4 Execution Time of No Operations

2000 NOPs	
Measured between	No. of clock cycles
Doubling 3 $\Delta 1 \& \Delta 2$	27996
Doubling 3 $\Delta 2 \& \Delta 3$	29995
Doubling 3 $\Delta 3 \& \Delta 4$	29995
Addition 1 $\Delta 1 \& \Delta 2$	30006
Addition 1 $\Delta 2 \& \Delta 3$	28007
Addition 1 $\Delta 3 \& \Delta 4$	30006
Addition 1 $\Delta 4 \& \Delta 5$	28007
Addition 1 $\Delta 5 \& \Delta 6$	30006

TABLE 8.9: Measured execution time of 2,000 NOPs.

5000 NOPs	
Measured between	No. of clock cycles
Doubling 2 & Doubling 3	69996
Doubling 3 & Addition 1	70012
Addition 1 & Doubling 4	69996

TABLE 8.10: Measured execution time of 5,000 NOPs.

8.5 Attack Scenario

Assume an attacker can differentiate atomic blocks in the kP trace measured, whereby the duration of the first atomic block $\Delta 1$ of EC point addition (PA) is (slightly) longer than that of EC point doubling (PD).

Consider a left-to-right kP algorithm implemented using Longa's MNAMNAA atomic patterns, the first point operation is a point doubling. Since a point doubling operation consists of 4 atomic blocks, the next $\Delta 1$ in the kP trace will be at atomic block number $i = 1 + 4 = 5$, with duration $t_{\Delta 1}(x)$, where x is either PD or PA at the 5th atomic block, the attacker does not know yet. An attacker can compare his observed $t_{\Delta 1}(x)$ with $t_{\Delta 1}(PD)$ to reveal the operation executed:

- If $t_{\Delta 1}(x) \approx t_{\Delta 1}(PD)$, x is the atomic pattern of point doubling. The attacker will then take the ($i = i + 4$)th atomic block in the kP trace as the next $\Delta 1$ to analyse.
- Otherwise, if $t_{\Delta 1}(x) > t_{\Delta 1}(PD)$, x is the atomic pattern of point addition. The attacker will then take the ($i = i + 6$)th atomic block in the kP trace as the next $\Delta 1$ to analyse.

Therefore, the attacker can step-by-step reveal all point doublings and point additions from the measured kP trace, based on the assumption that the attacker can effectively separate each atomic block within the kP trace.

Similarly, this algorithm applies when $t_{\Delta 1}(PA) < t_{\Delta 1}(PD)$. The key condition is that $t_{\Delta 1}(PA) \neq t_{\Delta 1}(PD)$. However, further analysis of the kP trace is required to identify additional indicators for distinguishing the atomic patterns. Applying multiple indicators enhances the likelihood of a successful attack.