Elliptic curve cryptography algorithms resistant against power analysis attacks on resource constrained devices

Hilal Houssain

numèro d'ordre : D.U: 2286

EDSPIC: 602

UNIVERSITÉ BLAISE PASCAL – CLERMONT II

ÉCOLE DOCTORALE
SCIENCES POUR L'INGÉNIEUR DE CLERMONT-FERRAND

THÈSE

Présenté par

Hilal Houssain

Pour obtenir le grade de

DOCTEUR D'UNIVERSITÉ

Discipline: Informatique

Elliptic Curve Cryptography Algorithms Resistant Against Power Analysis Attacks on Resource Constrained Devices

Président :	Michel Misson	Professeur à l'Université d'Auvergne
Rapporteurs :	Bernard Cousin	Professeur à l'Université de Rennes 1
	Pascal Urien	Professeur à Telecom Paris Tech
Directeur de theses :	Philipe Mahey	Professeur à l'Université Blaise Pascal
Co-directeurs :	Mohamed Badra	Chercheur CNRS au LIMOS
	Turki Al-Somani	Professeur à l'Université Umm Al Qura

Year: 2012

DEDICATION PAGE

To all those and all those who have contributed directly or indirectly to the completion of this work, I say simply and from the heart, thank you!

... to the memory of Dr Mohamad Khodr, and Ahlam Khodr, May God bless their souls

... to my beloved parents, and specially my loving wife and kids.

CONTENTS

List of Tables

List of Figures

List of Algorithms

Keywords

Wireless Sensor Networks, Elliptic Curve Cryptography, Side Channel Attacks, Power Analysis Attacks, Simple Power Analysis Attacks, Differential Power Analysis Attacks, and Scalar Multiplication.

Abstract

Elliptic Curve Cryptosystems (ECC) have been adopted as a standardized Public Key Cryptosystems (PKC) by IEEE, ANSI, NIST, SEC and WTLS. In comparison to traditional PKC like RSA and ElGamal, ECC offer equivalent security with smaller key sizes, in less computation time, with lower power consumption, as well as memory and bandwidth savings. Therefore, ECC have become a vital technology, more popular and considered to be particularly suitable for implementation on resource constrained devices such as the Wireless Sensor Networks (WSN). Major problem with the sensor nodes in WSN as soon as it comes to cryptographic operations is their extreme constrained resources in terms of power, space, and time delay, which limit the sensor capability to handle the additional computations required by cryptographic operations. Moreover, the current ECC implementations in WSN are particularly vulnerable to Side Channel Analysis (SCA) attacks; in particularly to the Power Analysis Attacks (PAA), due to the lack of secure physical shielding, their deployment in remote regions and it is left unattended. Thus designers of ECC cryptoprocessors on WSN strive to introduce algorithms and architectures that are not only PAA resistant, but also efficient with no any extra cost in terms of power, time delay, and area.

The contributions of this thesis to the domain of PAA aware elliptic curve cryptoprocessor for resource constrained devices are numerous. Firstly, we propose two robust and high efficient PAA aware elliptic curve cryptoprocessors architectures based on innovative algorithms for ECC core operation and envisioned at securing the elliptic curve cryptoprocessors against Simple Power Analysis (SPA) attacks on resource constrained devices such as the WSN. Secondly, we propose two additional architectures that are envisioned at securing the elliptic curve cryptoprocessors against Differential Power Analysis (DPA) attacks. Thirdly, a total of eight architectures which includes, in addition to the two SPA aware with the other two DPA aware

proposed architectures, two more architectures derived from our DPA aware proposed once, along with two other similar PAA aware architectures. The eight proposed architectures are synthesized using Field Programmable Gate Array (FPGA) technology. Fourthly, the eight proposed architectures are analyzed and evaluated by comparing their performance results. In addition, a more advanced comparison, which is done on the cost complexity level (Area, Delay, and Power), provides a framework for the architecture designers to select the appropriate design. Our results show a significant advantage of our proposed architectures for cost complexity in comparison to the other latest proposed in the research field.

Mots-clés

Les systèmes de cryptographie à base de courbe elliptique, les attaques par canaux auxiliaires, les attaques par analyse de consommation, les attaques par analyse élémentaire de consommation, les attaques par analyse différentielle de consommation, et la multiplication scalaire.

Abstrait

Les systèmes de cryptographie à base de courbe elliptique (ECC) ont été adoptés comme des systèmes standardisés de cryptographie à clé publique (PKC) par l'IEEE, ANSI, NIST, SEC et WTLS. En comparaison avec la PKC traditionnelle, comme RSA et ElGamal, l'ECC offre le même niveau de sécurité avec des clés de plus petites tailles. Cela signifie des calculs plus rapides et une consommation d'énergie plus faible ainsi que des économies de mémoire et de bande passante. Par conséquent, ECC est devenue une technologie indispensable, plus populaire et considérée comme particulièrement adaptée à l'implémentation sur les dispositifs à ressources restreintes tels que les réseaux de capteurs sans fil (WSN).

Le problème majeur avec les nœuds de capteurs chez les WSN, dès qu'il s'agit d'opérations cryptographiques, est les limitations de leurs ressources en termes de puissance, d'espace et de temps de réponse, ce qui limite la capacité du capteur à gérer les calculs supplémentaires nécessaires aux opérations cryptographiques. En outre, les mises en œuvre actuelles de l'ECC sur WSN sont particulièrement vulnérables aux attaques par canaux auxiliaires (SCA), en particulier aux attaques par analyse de consommation (PAA), en raison de l'absence de la sécurité physique par blindage, leur déploiement dans les régions éloignées et le fait qu'elles soient laissées sans surveillance. Ainsi, les concepteurs de crypto-processeurs ECC sur WSN s'efforcent d'introduire des algorithmes et des architectures qui ne sont pas seulement résistants PAA, mais également efficaces sans aucun supplément en termes de temps, puissance et espace.

Cette thèse présente plusieurs contributions dans le domaine des cryptoprocesseurs ECC conscientisés aux PAA, pour les dispositifs à ressources limitées comme le WSN. Premièrement, nous proposons deux architectures robustes et efficaces pour les ECC conscientisées au PAA. Ces architectures sont basées sur des algorithmes innovants qui assurent le fonctionnement de base des ECC et qui prévoient une sécurisation de l'ECC contre les PAA simples (SPA) sur les

dispositifs à ressources limitées tels que les WSN. Deuxièmement, nous proposons deux architectures additionnelles qui prévoient une sécurisation des ECC contre les PAA différentiels (DPA). Troisièmement, un total de huit architectures qui incluent, en plus des quatre architectures citées ci-dessus pour SPA et DPA, deux autres architectures dérivées de l'architecture DPA conscientisée, ainsi que deux architectures PAA conscientisées. Les huit architectures proposées sont synthétisées en utilisant la technologie la technologie des réseaux de portes programmables in situ (FPGA). Quatrièmement, les huit architectures sont analysées et évaluées, et leurs performances comparées. En plus, une comparaison plus avancée effectuée sur le niveau de la complexité du coût (temps, puissance, et espace), fournit un cadre pour les concepteurs d'architecture pour sélectionner la conception la plus appropriée. Nos résultats montrent un avantage significatif de nos architectures proposées par rapport à la complexité du coût, en comparaison à d'autres solutions proposées récemment dans le domaine de la recherche.

Previously Published Materials

The following papers have been published or presented, and contain material based on the content of this thesis.

1. Journal Articles

i. **Houssain. H, Al-Somani. T.F**, "Elliptic Curve Cryptoprocessor Implementation on a Nano FPGA: Interesting for Resource-Constrained Devices", International Journal of RFID Security and Cryptography (IJRFIDSC), Vol.1, Issues 1/2, p. 45 – 50, 2012

ii. **Houssain. H, Badra. M, Al-Somani. T.F**, "Comparative Study of Elliptic Curve Cryptography Hardware Implementations in Wireless Sensor Networks", International Journal of RFID Security and Cryptography (IJRFIDSC), Vol. 1, Issues 1/2, p. 67 – 73, 2012

iii. **Houssain. H, Badra. M, Al-Somani. T.F**, "Power Analysis Attacks on ECC: A Major Security Threat", International Journal of Advanced Computer Science and Applications (IJACSA) , Vol. 3, No.6, p. 90 – 96, 2012

iv. **Al-Somani. T.F, Khan. A.E, Qamar. A.M, and Houssain. H**, "Hardware/Software Co-Design Implementations of Elliptic Curve Cryptosystems", Information Technology Journal, Vol. 8, No. 4, p. 403 – 410, 2009

v. **Houssain. H, Badra. M, Al-Somani. T.F**, "Software Implementations of Elliptic Curve Cryptography in Wireless Sensor Networks", Journal of Communication and Computer, Vol. 9, No. 6, p. 712 – 720, 2012

2. Conference Papers

i. **Al-Somani. T.F, Houssain. H**, "Implementation of GF(2^m) Elliptic Curve Cryptoprocessor on Nano FPGA", Internet Technology and Secured Transactions (ICITST), 2011 International Conference for Publication, p. 7 – 12, 2011

ii. **Houssain. H, Badra. M, Al-Somani. T.F**, "Hardware implementations of Elliptic Curve Cryptography in Wireless Sensor Networks", Internet Technology and Secured Transactions (ICITST), 2011 International Conference for Publication, p. 1 – 6, 2011

List of Abbreviations Used

ADD	Addition
ASIC	Application Specific Integrated Circuits
CBA	Carry-based Attack
CMOS	Complementary Metal-Oxide Semiconductor
CPU	Central Processing Unit
DA	Doubling Attack
DoS	Denial of Service
DPA	Differential Power Analysis
ECC	Elliptic Curve Cryptosystems
ECDLP	Elliptic Curve Discrete Logarithm Problem
FPGA	Field Programmable Gate Array
FPM	Fixed point multiplication
$GF(2^m)$	Finite Field of Order 2m
MAC	Message Authentication Code
MOF	Mutual Opposite Form
MUL	Multiplication
NAF	Non-Adjacent Form
ONB	Optimal Normal Basis
PAA	Power Analysis Attacks
PADD	Point Addition
PCA	Principal Component Analysis
PDBL	Point Doubling
PKC	Public Key Cryptosystems

RAM	Random Access Memory
RFID	Radio Frequency Identity
RISC	Reduced Instruction Set Computing
ROM	Read Only Memory
RPA	Refined Power Analysis
RPM	Random point multiplication
RSA	Rivest, Shamir and Adleman
SCA	Side Channel Analysis
SeRLoC	Secure Range-Independent Localization
SPA	Simple Power Analysis
SQR	Squaring
SRAM	Static Random Access Memory
TinyECCK	Tiny Elliptic Curve Cryptosystem with Koblitz Curve
TNAF	τ - adic Non-Adjacent Form
UCLA	University of Central Lancashire
VHDL	VHSIC Hardware Description Language
VHSIC	Very High-Speed Integrated Circuit
VM	Verifiable Multilateration
WSN	Wireless Sensor Networks
ZPA	Zero Power Analysis

Acknowledgements

All praise be to Allah the Almighty who has given me knowledge, patience, and devotion to finish my PhD dissertation works. After 4 years of PhD research, I would like to thank the people who supported me along the way.

My greatest debt of gratitude is to my thesis supervisors who allowed me to conduct this work. My supervisor (Dr. Mohamad Badra) was a great help throughout these four years, and this thesis owes much to his advice, his rigor and lights that made me. My co-supervisor (Prof. Turki F. Al-Somani) profound knowledge on cryptography and electronic system design was vital to make my dissertation successfully complete and resourceful. The continuous support of Prof. Turki, together with his stimulating suggestions and encouragement helped me in all the time of research for and writing of this dissertation.

A great appreciation goes to my thesis director Prof. Philipe Mahey, for his invaluable inspiration, help, and guidance that helped me through my PhD dissertation works. I greatly valued the sincere and generous moral support he has provided me.

I extend my warmest thanks to Prof. Bernard Cousin and Prof. Pascal Urien for accepting to be members of the jury, and Prof. Michel Misson for chairing it. and for their helpful comments and suggestions.

I would like to express my utmost gratitude to to Dr. Mohammad Khodr and Ahlam Khodr (God bless their souls) for their guidance and support not only in my studies, but also in all matters of life. I am influenced by their way of thinking. They have been great teachers, friends, role models, and advisors to assist me in my career path and help me develop my professional skills.

I’m very grateful to Eng. Abdullah Al Hassani who worked with me on various pieces of this thesis and on other publications.

Acknowledgements

I cannot forget to acknowledge all my friends from Blaise Pascal University. I am particularly grateful to Ismail Mansour for his invaluable help in adjusting to France, and being wonderful friend.

Last, but not least, I would like to express my deepest thanks and appreciation to my beloved parents, and my lovely wife "Hala" for her love, moral support, patience and understanding. To my sons Mohamad and Ghayth for their love and confidence in me were the constant source of inspiration to offer the best of myself to this research.

CHAPTER 1

Introduction

Wireless Sensor Networks (WSN) [1] are ad hoc networks comprised of a large number of low-cost, low-power, and multi-functional sensor nodes and one or more base stations. The recent developments in WSN technology have led to a wide range of potential applications for this technology, such as health monitoring, industrial control, environment observation, as well as office and even military operations. In most of these applications, critical information is frequently exchanged among sensor nodes through insecure wireless channels. It is therefore crucial to add security measures to WSN using cryptography for protecting its data against threats in a way so integrity, authenticity or confidentiality can be guaranteed.

Major problem with the sensor nodes as soon as it comes to cryptographic operations is their extreme constrained resources in terms of power, space, and time delay, which limit the sensor capability to handle the additional computations required by cryptographic operations. Nevertheless, Public key cryptosystems (PKC) [2] is indeed shown to be feasible in WSN by using Elliptic Curve Cryptosystems (ECC) [3] [4]. This is because, in comparison to traditional cryptosystems like RSA [5] and ElGamal [6], ECC offers equivalent security with smaller key sizes, in less computation time, with lower power consumption, as well as memory and bandwidth savings.

The current ECC implementations in WSN [7] are particularly vulnerable to Side Channel Analysis (SCA) attacks [8]; in particularly to the Power Analysis Attacks (PAA) [9], due to the lack of secure physical shielding, their deployment in remote regions and it is left unattended. Accordingly, there should exist countermeasures to secure ECC against SCA attacks such as the Simple Power Analysis (SPA) and the Differential Power Analysis (DPA) [9] [10] attacks, but

normally these countermeasure solutions on ECC involve extra computations to be handled by the sensor. Thus designers of ECC cryptoprocessors on WSN strive to introduce algorithms and architectures that are not only PAA resistant, but also efficient with no any extra cost in terms of power, time delay, and area.

1.1. Motivation

To the extent of our knowledge, no shown effort has been made for PAA resistant ECC implementations in WSN in particular [11]. Additionally, the current PAA aware ECC architectures require extra computations to be handled by the cryptoprocessor, and thus there are not easily viable to be implemented in extremely constrained resources such as WSN.

1.2. Problem Statement

In general, approaches for PAA resistant ECC implementations in WSN correspond to extra cost in terms of energy, area, and time delay consumption for cryptographic functions. Therefore, designing ECC cryptoprocessors on WSN require the proposition of algorithms and architectures that are not only PAA resistant, but also efficient with no any extra cost in terms of power, time delay, and area. Conquering this concern, the following requirements for investigation have been identified as criterions for efficient and secure PAA resistant ECC implementations in WSN:

1. Underlying finite field, representation basis, project coordinate system, and the field arithmetic operations for ECC systems.
2. Security issues and requirements for WSN, and its current software and hardware implementation in WSN, taking into consideration the underlying finite field, representation basis, occupied chip area, consumed power, and time delay performances of these implementations.

3. Major PAA and its countermeasures on ECC.

1.3. Contributions

The contributions of this thesis to the domain of PAA aware elliptic curve cryptoprocessor for WSN are numerous.

Firstly, we propose two robust and high efficient PAA aware elliptic curve cryptoprocessors architectures for WSN. These architectures are based on innovative algorithms for ECC core operation and envisioned at securing the elliptic curve cryptoprocessors against Simple Power Analysis (SPA) [9] [10] attacks.

Secondly, we propose two additional architectures that are envisioned at securing the elliptic curve cryptoprocessors against Differential Power Analysis (DPA) [9] [10] attacks.

Thirdly, a total of eight architectures which includes, in addition to the two SPA aware with the other two DPA aware proposed architectures, two more architectures derived from our DPA aware proposed once, along with two other similar PAA aware architectures. The eight proposed architectures are synthesized using Field Programmable Gate Array (FPGA) [12] technology.

Fourthly, the eight proposed architectures are analyzed and evaluated by comparing their performance results. In addition, a more advanced comparison, which is done on the cost complexity level (Area, Delay, and Power), provides a framework for the architecture designers to select the appropriate design. Our results show a significant advantage of our proposed architectures for security level and cost complexity in comparison to the other latest proposed in the research field.

1.4. Organization of the Thesis

This thesis is organization as follows.

In chapter 2, the necessary background on ECC is provided, including the GF(2^m) finite field arithmetic, ECC arithmetics and ECC operations such as scalar multiplication, encryption, and discrete logarithm problem.

Chapter 3 presents studies on both the hardware and software recent implementations of ECC in resource constrained devices such as the WSN. These studies consider the unique characteristics of WSN nodes as resource constrained devices, and thus it cover the underlying finite field, representation basis, occupied chip area, consumed power, and time delay performances of these implementations.

Chapter 4 represents a comprehensive study for the major existing PAA on ECC and its countermeasures. In addition, we make a graphical presentation for the relation between PAA on ECC and the current countermeasures. We discuss the critical concerns to be considered in designing countermeasures against PAA on ECC particular for WSN.

Chapter 5 proposes four different robust and high efficient PAA aware elliptic curve cryptoprocessors architectures for WSN. The first two architectures are envisioned at securing the elliptic curve cryptoprocessors against SPA attacks, whereas the last two architectures are envisioned at securing the elliptic curve cryptoprocessors against DPA attacks.

Chapter 6 presents the results of synthesizing eight various cryptoprocessors, and shows a comparison study for these cryptoprocessors in terms of power, time delay and area. In addition, a more advanced comparison is done on the cost complexity level, which provides a framework for the architecture designers to select the appropriate design.

In Chapter 7, we summarize this thesis, and suggest directions for future research.

CHAPTER 2

Elliptic Curve Cryptography

This chapter provides necessary background on Elliptic Curve Cryptosystems (ECC) [3] [4], including the GF(2^m) finite field arithmetic, ECC arithmetics and ECC operations such as scalar multiplication, encryption, and discrete logarithm problem.

This chapter is organized as follows: Section 2.1 presents a brief on the finite field arithmetic, followed by the GF(2^m) arithmetics in Section 2.2. Elliptic Curve arithmetic is covered in Section 2.3. In Section 2.4, the Elliptic Curve Scalar Multiplication is discussed at length. Elliptic Curve encryption is considered in Section 2.5 and chapter summary is provided in Section 2.6.

2.1. Finite Field Arithmetic

Curve operations in elliptic curve cryptosystem are carried out using arithmetic operations in the underlying field; hence, the overall performance of this cryptosystem depends on the efficiency of the arithmetic performed in the underlying finite field.

In abstract algebra, a **finite field** or **Galois field** (so named in honor of Évariste Galois) is a field that contains only finitely numerous elements. Finite fields are vital in number theory, algebraic geometry, Galois theory, cryptography and coding theory [13] [14] [15].

G is a group that could be either a finite or infinite set of elements, and its order, represented by the symbol $|G|$, is the number of elements in the group. The group G together with a binary operation (also called group operation), ◊, collectively satisfy the following four fundamental properties:

1. **Closure:** $\forall\ a,b \in G,\ a \Diamond b \in G$.
2. **Associativity:** $\forall\ a,b,c \in G,\ (a \Diamond b) \Diamond c = a \Diamond (b \Diamond c)$.
3. **Identity:** The group contains an identity element $e \in G$ such that $\forall\ a \in G,$ $a \Diamond e = e \Diamond a = a$.
4. **Inverse:** For every element $a \in G$ there is an inverse $a^{-1} \in G$ such that $a \Diamond a^{-1} = a^{-1} \Diamond a = e$.

Abelian groups (also called commutative groups), are groups fulfilling the conditions that the result of product operation of elements is unrelated to their arrangement;

i.e., $a \Diamond b = b \Diamond a\ \ \forall\ a,b \in G$.

Cyclic groups are groups that have a generator element. A generator element $g \in G,$ is an element of the group G, if every element $a \in G$ is generated by repeatedly applying the group operation on g. Thus, $\forall\ a \in G,$

$$a = \underbrace{g \Diamond g \Diamond g \Diamond \ldots \Diamond g}_{i \text{ times}}. \qquad \text{(Equation 2.1)}$$

Additive groups are groups with the ``+'' group operator, denoted as:

$$ig = \underbrace{g + g + g + \ldots + g}_{i \text{ times}}. \qquad \text{(Equation 2.2)}$$

Equally, *multiplicative groups* are groups with the ``*'' group operator, denoted as:

$$g^i = \underbrace{g * g * g * \ldots * g}_{i \text{ times}} \qquad \text{(Equation 2.3)}$$

A *field* consists of a set of elements *F* together with two operations, addition (denoted by "+") and multiplication (denoted by "*"), that satisfy the following arithmetic properties:

1. (F, +) is an *Abelian* group with respect to the ``+'' operation, with additive identity denoted by 0.
2. (F \ {0}, *) represented by F^*, and its elements form an *Abelian* group under the " * " operation, with multiplicative identity denoted by 1, and contains all the elements in F except the additive identity 0.
3. The distribution law applies to the two binary operations; as follows:

$$\forall \ a,b,c \in F\ ,\ a*(b+c)=(a*b)+(a*c)\,.$$

As previously mentioned, if the set F is finite, then the field is said to be finite. Finite fields are represented by the symbol GF(q) and for any prime *p* and positive integer *m*, there always exists a finite field of order $q = p^m$. The prime *p* is called the characteristic of the finite field $GF(p^m)$. In addition, there are three kinds of fields that are especially adaptable for efficiently implementing elliptic curve systems are prime fields, binary fields, and optimal extension fields.

2.2. GF(2^m) Arithmetic

The finite $GF(2^m)$ field, of order 2^m, called binary fields or characteristics-two finite fields, are of particular significance in cryptography, especially in the hardware implementation of cryptosystems, since it introduces high efficiency compared to the other fields. Elements of the $GF(2^m)$ field are represented in terms of a basis. Either Normal or Polynomial Basis is usually used for the majority of the elliptic curve cryptosystem implementations. In case of hardware implementation, normal basis is more suitable than polynomial basis since its operations can be efficiently implemented in hardware, and it mainly involve rotation, shifting and exclusive-

ORing. Since for instance, one advantage of normal bases is that squaring of a field element is a simple rotation of its vector representation.

A normal basis of GF(2^m) is a basis of the form ($\beta^{2^{m-1}}$, ..., β^{2^2}, β^{2^1}, β^{2^0}), where $\beta \epsilon$ GF(2^m). In addition, an element A ϵ GF(2^m) in a normal basis can be uniquely represented in the form $A = \sum_{i=0}^{m-1} \alpha^i \beta^{2i}$, where $\alpha^i \epsilon \{0,1\}$.

GF(2^m) operations using normal basis are performed as follows:

1. **Addition**. Addition is performed by a simple bit-wise exclusive-OR (XOR) operation.

2. Squaring. Squaring is simply a rotate left operation. Thus, if

$A = (a_{m-1}, a_{m-2}, \ldots a_1, a_0)$, then $A^2 = (a_{m-2}, a_{m-3}, \ldots a_0, a_{m-1})$.

3. Multiplication. $\forall$ A, B ϵ GF(2^m), where

$$A = \sum_{i=0}^{m-1} a_i \beta^{2^i} \text{ and } B = \sum_{i=0}^{m-1} b_i \beta^{2^i}$$

The product C = A * B, is given by:

$$C = A * B = \sum_{i=0}^{m-1} c_i \beta^{2^i}$$

Multiplication is defined in terms of a set of m multiplication matrices $\lambda^{(k)}$

(k = 0,1,….,m-1),

$$c_k = \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} \lambda_{ij}^{(k)} a_i b_j \quad \forall k = 0,1,\ldots,m-1$$

Where $\lambda_{ij}^{(k)} \epsilon \{0, 1\}$

The complexity of the multiplication method and its hardware implementation is related to the number of non-zero elements in the λ matrix. For Optimal Normal Basis (ONB) [16], this value is denoted as C_N and is equal to (2m-1). An ONB is one with the minimum possible number of non-zero elements in the λ_{ij} matrix.

Values of the λ matrix elements can be derived in function of the field size m. ONB is categorized into two types, denoted by Type I and Type II [16]. An ONB of Type I is valid for a given field GF (2^m) if:

(a) m + 1 is a prime

(b) 2 is a primitive in GF (m + 1)

In the other side, an ONB of Type II is available in GF (2^m) if:

(a) 2m + 1 is prime

(b) Either 2 is a primitive in GF(2m + 1) or 2m + 1 ≡ 3 (mod 4) and the quadratic residues in GF(2m + 1) is generated by 2

An ONB is available in GF (2^m) for 23% of all possible values of m [16]. The $\lambda^{(k)}$ matrix can be formed by a *k*-fold cyclic shift to $\lambda^{(0)}$ as follows:

$$\lambda_{ij}^{(k)} = \lambda_{i-k,j-k}^{(0)} \text{ for all } 0 \leq i, j, k \leq m\text{-}1$$

The $\lambda^{(0)}$ matrix is derived differently for the two types of ONB. For the Type I ONB, $\lambda_{ij}^{(0)} = 1$ iff i and j satisfy one of the following two congruencies [17]:

(a) $2^i + 2^j \equiv 1 \bmod (m + 1)$

(b) $2^i + 2^j \equiv 0 \bmod (m + 1)$

For all Type II ONB, $\lambda_{ij}^{(k)} = 1$ iff i and j satisfy one of the following four congruencies [17]:

(a) $2^i + 2^j \equiv 2^k \bmod (2m + 1)$

(b) $2^i + 2^j \equiv -2^k \bmod (2m + 1)$

(c) $2^i - 2^j \equiv 2^k \bmod (2m + 1)$

(d) $2^i - 2^j \equiv -2^k \bmod (2m + 1)$

Therefore, $\lambda_{ij}^{(0)} = 1$ iff i and j satisfy one of the following four congruences:

$$2^i \pm 2^j \equiv \pm 1 \bmod (2m + 1)$$

4. Inversion. Inverse of a ϵ GF(2^m), denoted as a^{-1}, is defined as follows.

$$aa^{-1} \equiv 1 \bmod 2^m$$

The majority of the inversion algorithms are generated from Fermat's Little Theorem, where

$$a^{-1} = a^{2m-2}$$

for all $a \neq 0$ in GF(2^m).

In this thesis, and for its advantage in hardware implementation efficiency, ONB is chosen to represent the elements of the GF(2^m) fields in elliptic curve cryptoprocessors hardware implementations.

2.3. Elliptic Curve Arithmetic

An elliptic curve E over the finite field GF(p) defined by the parameters a, b ϵ GF(p), where p is a prime greater than 3, is the group formed by the additive identity of the group point O, known as the "point at infinity" [18], and the set of points P = (x, y), where x, y ϵ GF(p), that satisfy the elliptic curve equation (Equation 2.4)

$$y^2 = x^3 + ax + b \qquad \text{(Equation 2.4)}$$

for a, b ϵ GF(p) and $4a^3 + 27b^2 \neq 0 \bmod p$.

For every curve over a finite field GF(q), it contains a defined number of points *n* that is calculated using Hasse's theorem [14]. Adding two points on an elliptic curve E returns a third point on E which forms an *Abelian* group with the identity element 0. A cryptosystem based on the elliptic curve (elliptic curve cryptosystem) can be built using the *Abelian* group.

Point Addition (PADD) over GF(p) is best described geometrically as follows. Let P = (X_1,Y_1) and Q = (X_2,Y_2) be two distinct points on an elliptic curve E defined over GF(p) with Q ≠ −P; where –P = $(X_1,-Y_1)$ is the additive inverse of P. The resultant point R is P + Q = (X_3,Y_3) of adding P and Q is the reflection in the x-axis of the point of the elliptic curve that is intersected by the line crossing P and Q. The addition operation over GF(p) can be visualized in Figure 2.1.

Point Doubling (PDBL) operation formula can be easily derived from the PADD one, when P = Q and P ≠ −P, and the resultant point R is P + Q = 2P is the additive inverse of a third point on E intercepted by the straight line tangent to the curve at point P. The doubling operation over GF(p) is depicted in Figure 2.2.

Supersingular elliptic curves are special class of curves with some special properties that make them unstable for cryptography [19], and thus unsecure. Therefore, only non-supersingular curves over GF(2^m) are considered. Equation 2.5 defines the non-supersingular elliptic curve equation for GF(2^m) fields.

$$y^2 + xy = x^3 + ax^2 + b \qquad \text{(Equation 2.5)}$$

where $a,b \in$ GF(2^m) and $b \neq 0$

For a non-supersingular elliptic curve E defined over GF(2^m), PADD and PDBL operations are generally computed using the algebraic formulae as follows:

• Identity: P + O = O + P = P for all P ϵ E.

• Negatives: If P = (x, y) ϵ E, then (x, y) + (x, x + y) = O. The point (x, x + y) is called the negative of P, denoted as −P.

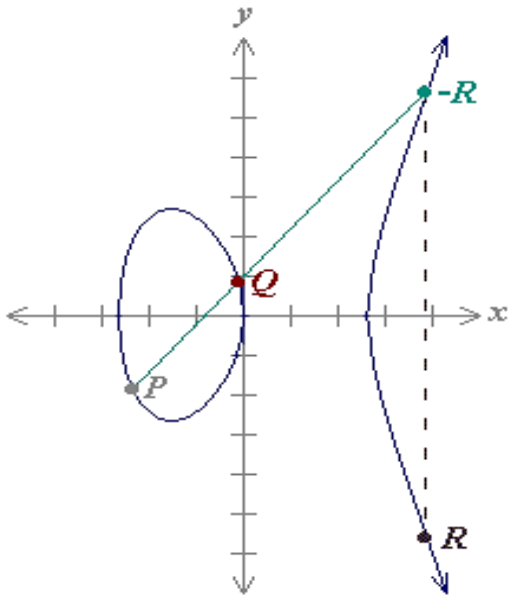

Figure 2.1: The PADD operation (R = P + Q) over GF(p).

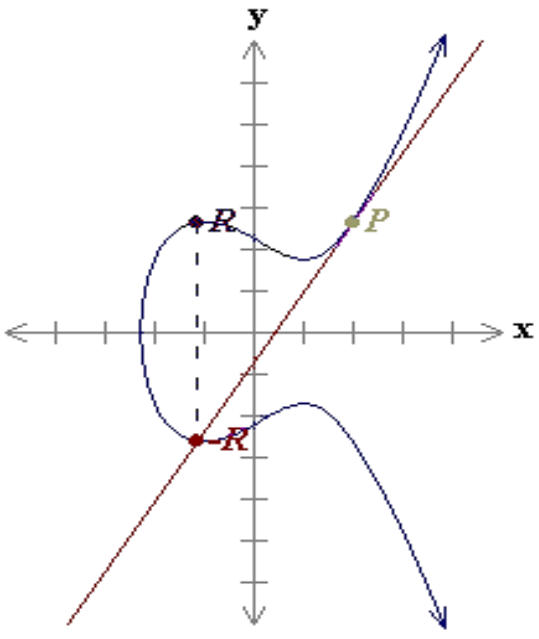

Figure 2.2: The PDBL Operation (R = 2P) Over GF(p).

• PADD: Let P = (x_1, y_1), Q = (x_2, y_2) ϵ E, P $\neq$ Q and Q $\neq$ −P, then P + Q = (x_3, y_3), where

$$x_3 = \left(\frac{y_{1+}\, y_2}{x_{1+}\, x_2}\right)^2 + \left(\frac{y_{1+}\, y_2}{x_{1+}\, x_2}\right) + x_1 + x_2 + a$$

$$y_3 = \left(\frac{y_{1+}\, y_2}{x_{1+}\, x_2}\right).(x_1 + x_3) + x_3 + y_1$$

• PDBL: If P = Q = (x_1, y_1), then 2P = P + P = (x_3, y_3), where

$$x_3 = x_1^2 + \frac{b}{x_1^2}$$

$$y_3 = x_1^2 + \left(x_1 + \frac{y_1}{x_1}\right) x_3 + x_3$$

The dominant operation of all ECC algorithms, including encryption/decryption and signature generation/verification primitives, is the point scalar multiplication k*P, where k is an integer and P is a point on the elliptic curve, represents the addition of point P k times as presented by Equation 2.6.

$$kP = \underbrace{P + P + P + \ldots + P}_{k \text{ times } P} \qquad \text{(Equation 2.6)}$$

When points on the elliptic curve E are represented in affine coordinates (x,y), it turns the PADD and PDBL operation inefficient because they contain field inversions, where inversions are the most expensive field operation and need to be largely prevented.

As mentioned in Section 2.1, the cyclic groups have a generator element g, and every element $a \in G$ is generated by repeatedly applying the group operation on g. The elliptic curve cryptosystems are based on this group, where g is represented by a base point P and n is the

number of points on the group. P is the generator of the group, and its order is n, whereas the order of any other point in the group is a finite number dividable by n.

Projective coordinates (X, Y, Z) resolve the issue of expensive inversion in the PADD and PDBL caused by the affine coordinates, by adding Z as a third coordinate in order to replace inversion field operations by other less expensive operations [19].

For elliptic curve defined over GF(2^m), many different forms of formulas may be used for PADD and PDBL in the [20] [21] [22] [23]. For instance, the Homogeneous coordinate system replaces the coordinates of an elliptic curve point (x, y) by (x, y) = (X/Z, Y/Z) [21], whereas the Jacobian coordinate system replaces these coordinates by (x, y) = (X/Z^2, Y/Z^3) [22]. Likewise, the Lopez-Dahab coordinate system takes the form (x, y) = (X/Z, Y/Z^2) [23]. In consequence, different formulas require different number of field multiplications for the point adding and doubling for each of the coordinate systems as shown in Table 2.1, Table 2.2, and Table 2.3 respectively. For instance, Lopez-Dahab [23] coordinate system is very cost effective in comparison with both Homogenous and Jacobian coordinate systems, since it only requires 14 and 5 field multiplications for PADD and PDBL respectively, whereas Homogenous requires 16 and 7 field multiplications, and Jacbian requires 15 and 7 field multiplications. Coordinate systems could be a mix of two different coordinate systems, and point operation can take each point from one of the coordinate system, and the resulting point could be given in a third coordinate system [20].

2.4. Elliptic Curve Scalar Multiplication

Scalar multiplication in the group of points of an elliptic curve is the analogous of exponentiation in the multiplicative group of integers modulo a fixed integer m.

Table 2.1: The Homogeneous Projective Coordinates System

Addition	**Multiplications**	**Doubling**	**Multiplications**
$A = X_1 Z_2$	1M	$A = X_1 Z_1$	1M
$B = X_2 Z_1$	1M	$B = bZ_1^4 + X_1^4$	1M
$C = A + B$		$C = AX_1^4$	1M
$D = Y_1 Z_2$	1M	D = Y1Z1	1M
$E = Y_2 Z_1$	1M	$E = X_1^2 + D + A$	
$F = D + E$		$Z_3 = A^3$	1M
$G = C + F$	1M	$X_3 = AB$	1M
$H = Z_1 Z_2$	5M	$Y_3 = C + BE$	1M
$I = C^3 + aHC^2 + HFG$			
$X_3 = CI$	1M		
$Z_3 = HC^3$	1M		
$Y_3 = GI + C^2[FX_1 + CY_1]$	4M		
Total	**16M**		**7M**

Table 2.2: The Jacobian projective coordinates system

Addition	**Multiplications**	**Doubling**	**Multiplications**
$A = X_1 Z_2^2$	1M	$Z_3 = X_1 Z_1^2$	1M
$B = X_2 Z_1^2$	1M	$A = bZ_1^2$	1M
$C = A + B$		$B = X_1 + A$	
$D = Y_1 Z_2^3$	2M	$X_3 = B^4$	
$E = Y_2 Z_1^3$	2M	$C = Z_1 Y_1$	1M
$F = D + E$		$D = Z_3 + X_1^2 + C$	
$G = Z_1 C$	1M	$E = DX_3$	1M
$H = FX_2 + GY_2$	2M	$Y_3 = X_1^4 Z_3 + E$	1M
$Z_3 = GZ_2$	1M		
$I = F + Z_3$			
$X_3 = aZ_3^2 + IF + C^3$	3M		
Total	**15M**		**7M**

Table 2.3: The Lopez-Dahab projective coordinates system

Addition	**Multiplications**	**Doubling**	**Multiplications**
$A_0 = Y_1^2 Z_1^2$	1M	$Z_3 = Z_1^2 X_1^2$	1M
$A_1 = Y_1 Z_2^2$	1M	$X_3 = X_1^4 + bZ_1^4$	1M
$B_0 = X_2 Z_1$	1M	$Y_3 = bZ_1^4 Z_3 + X_3(aZ_3 + Y_1^2 + bZ_1^4)$	3M
$B_1 = X_1 Z_2$	1M		
$C = A_0 + A_1$			
$D = B_0 + B_1$			

$E = Z_1 Z_2$	1M		
$F = DE$	1M		
$Z_3 = F^2$			
$G = D^2(F + aE^2)$	2M		
$H=CF$	1M		
$X_3 = C_2 + H + G$			
$I = D^2B_0E + X_3$	2M		
$J = D^2A_0 + X_3$	1M		
$Y_3=HI+Z_3J$	2M		
Total	**14M**		**5M**

Scalar multiplication is the basic and most time consuming operation in ECC; the computation of this operation includes *three* mathematical levels: scalar arithmetic, point arithmetic and field arithmetic. The mathematical hierarchy of ECC scalar multiplication is depicted in Figure 2.3.

Scalar arithmetic is at the highest level of the hierarchy, and it is for the point multiplication. Point arithmetic is for point operation such as PADD and point double, and it is at the middle level. The lowest level is of the finite field arithmetic including field multiplication, field inversion, field squaring and field addition. The cost of field addition is negligible in the finite field GF(2^m) when compared with the field inversion (equivalent cost of 10 field multiplications) and field squaring (equivalent cost of 0.2 field multiplication).

Scalar multiplication replies on the point operations over the elliptic curve. Numerous methods for scalar multiplication can be found in the literature. Good surveys have been conducted in [24] [25]. The straightforward double-and-add scalar multiplication algorithm (also called binary algorithm) is the traditional method for computing the scalar multiplication kP. The double-and-add algorithm is based on the binary expansion of the scalar k as 0's and 1's, and can be computed by scanning the bits of k =

(k_{m-1}, ..., k_0) from left to right (See Algorithm 2.1) or right to left (See Algorithm 2.2) and perform PDBL for each bit, and PADD whenever the bit value $k_i = 1$.

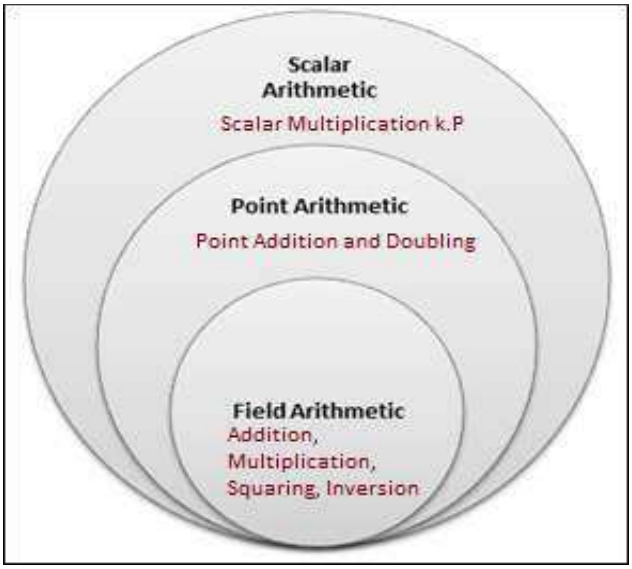

Figure 2.3: Mathematical hierarchy of ECC scalar multiplication

In Algorithm 2.1, PDBL is always performed in Step 2.1 regardless of the bit value, while PADD is only performed in Step 2.2 if the bit value $k_i = 1$. Likewise, in Algorithm 2.2, PADD is performed in Step 2.1 only if the bit value $k_i = 1$, while PDBL is always performed in Step 2.2.

Algorithm 2.1 Double-and-add elliptic curve scalar multiplication method (left-to-right)
Inputs: *P: Base Point, k: Secret key.* **Outputs**: *kP.*
1: R[0] ← P 2: for i = m-2 down to 0 do 2.1: R[0] ← 2R[0] 2.2: if $k_i = 1$ then R[0] ← R[0] + P 2.3: end for Return R[0].

Algorithm 2.2 Double-and-add elliptic curve scalar multiplication method (right-to-left).
Inputs: *P: Base Point, k: Secret key.* **Outputs**: *kP.*
1: R[0] ← O, R[1] ← P 2: for i = 0 to m-1 do 2.1: if k_i = 1 then R[0] ← R[0] + R[1] 2.2: R[1] ← 2R[1] 2.3: end for Return R[0].

2.5. Elliptic Curve Encryption

After being studies for hundred years, the practical use of the elliptic curves in public key cryptography was independently invented by Koblitz [18] and Miller [26], in the mid of 1980's. Since then, researchers proposed several approaches for the utilization of elliptic curves for encryption and decryption process, where elliptic curve Diffie-Hellman and elliptic curve ElGamal [17] are considered the most famous public key protocols relevant to elliptic curves.

2.5.1 Elliptic Curve Diffie-Hellman Protocol

Elliptic Curve Diffie-Hellman protocol is based on discrete logarithm problem, mutually invented by Diffie and Hellman in 1976 [27] as key exchange equivalent in elliptic curve cryptography. In Elliptic Curve Diffie-Hellman Protocol, if the private key of A, and its public key are denoted by are k_A and $P_A = k_AP$ respectively, the private key of B, and its public key are denoted by are k_B and $P_B = k_BP$ respectively, under a trusted public key infrastructure where P is the base point of the elliptic curve. The shared secret key S between A and B can be generated by computing k_AP_B and k_BP_A by A and B respectively. In addition, the message encryption is

performed by inserting the shared secret key into the the x-coordinate of $P_m = (x_m, y_m)$ [17]. The result cipher text point P_c, a point on the elliptic curve, is given by

$$P_c = P_m + S$$

On the other side, to message decryption process is implemented by subtracting the shared secret key from the cipher text point P_c to give the plaintext point P_m given by

$$P_m = P_c - S$$

2.5.2 Elliptic Curve ElGamal Protocol

Elliptic Curve ElGamal protocol is also based on discrete logarithm problem, invented by ElGamal in 1984 [6], as encryption and digital signature scheme. In Elliptic Curve ElGamal protocol, if B wants to encrypt and send a message point P_m to user A, B chooses a random integer l and generates the cipher text C_m which consists of the following pair of points:

$$C_m = (lP, P_m + lP_A)$$

The cipher text pair of points uses A's public key, where only user A can decrypt the plaintext using his/her private key. To decrypt the cipher text C_m, the first point in the pair of C_m, lP is multiplied by A's private key to get the point k_A (lP). This point is subtracted from the second point of C_m to produce the plaintext point P_m.

The complete decryption operations can be summarized in the following equation:

$$P_m = (P_m + lP_A) - k_A (lP) = P_m + l (k_A P) - k_A (lP)$$

2.5.3 Elliptic Curve Discrete Logarithm Problem

The security of elliptic curve cryptosystems is based on the intractability of Elliptic Curve Discrete Logarithm Problem (ECDLP). The ECDLP is best defined as follow:

Let E be an elliptic curve defined over a finite field, and P and Q are two distinct points on E, the ECDLP is the problem of finding an integer k, where $0 \leq k \leq m - 1$, such that $Q = kP$. P is the base point, and k is the elliptic curve discrete logarithm of Q with respect to P (i.e., $k = log_p (Q)$). The strength of the ECDLP is subject to the precise selection of the parameters. To date, Pollard–ρ algorithm [28] is known to be the most efficient algorithm for solving the ECDLP. Even with the ECDLP's parallelized version given by Gallant *et. al.* [29], the Pollard–ρ algorithm requires an average of $\sqrt{n}$, where n represent the number of points on the elliptic curve.

2.6. Summary

This chapter provides necessary background on ECC, including the GF(2^m) finite field arithmetic, ECC arithmetics and ECC operations such as scalar multiplication, encryption, and discrete logarithm problem.

In GF(2^m), elements are presented in different basis, where the majority are represented using (a) normal basis, or (b) polynomial basis. If ECC efficient hardware implementation is a major requirement, normal basis is a preferable option since field operations in normal basis are limited to light arithmetics such as rotation, shifting and exclusive-ORing which are known for efficient implemented in hardware.

Scalar multiplication is the basic and most time consuming operation in ECC. At the point operation level, the scalar multiplication is represented by a series of PADD and PDBL operations. At the field arithmetic level, the point operation involves field multiplication, field inversion, field squaring and field addition. Thus, efficient ECC implementation will require careful implementation at point operation and field arithmetic levels.

Several projective coordinate systems have been proposed to reduce the number of inversions in scalar multiplication to only one single inversion. Lopez-Dahab projective coordinate system

requires less number of field multiplications as compared to other existing projective coordinate systems. Accordingly, Lopez-Dahab projective coordinate system has been selected for the implementations presented in this thesis.

Being the core of elliptic curve cryptosystems security, the intractability of the elliptic curve discrete logarithm problem has been also discussed in this chapter.

CHAPTER 3

Wireless Sensor Networks

3.1 Background on WSN

Wireless sensor networks (WSN) [30] [1] are ad hoc networks consist of hundreds or even thousands of small sensor nodes with limited resources are based around a battery powered microcontroller. These nodes are equipped with a radio transceiver, and are capable to communicate with each other and with one or more sink nodes that interact with the outside world. In addition, these nodes are furnished with a set of transducers through which they acquire data about the surrounding environment, and receive commands via the sink to assign data collection, data processing and data transfer tasks. The number of nodes participating in a sensor network is mainly determined by requirements relating to network connectivity and coverage, and by the size of the area of interest. An example is illustrated in Figure 3.1.

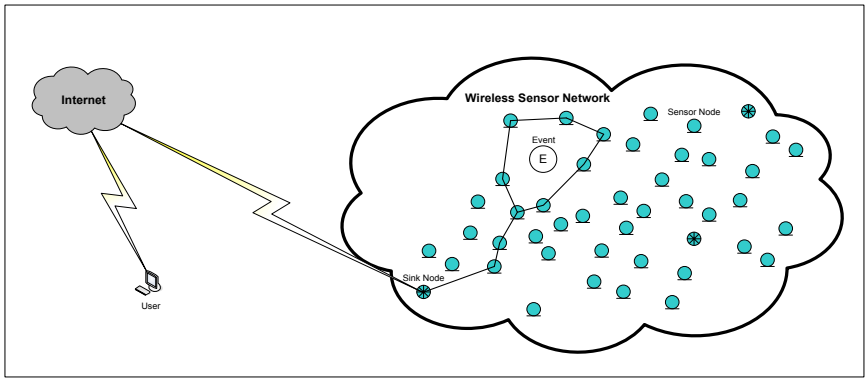

Figure 3.1: A Wireless Sensor Network

There exist a large number of different application scenarios for WSN [30]: examples are health monitoring, industrial control, environment observation, as well as office and even military applications. For example, in the health monitoring applications, WSN can be used to remotely monitor physiological parameters, such as heartbeat or blood pressure of patients, and sends a trigger alert to the concerned doctor according to a predefined threshold. In addition, sensor nodes may be deployed in several forms: at random, or installed at deliberately chosen spots.

3.1.1 Hardware Architecture of WSN nodes

A basic WSN node (also known as mote) comprises five main components (Figure 3.2) which are capable of interacting with their surrounding area through different sensors, performing data processing, and communicating data wirelessly with other nodes. The main components of the WSN node are: Controller, memory, sensors and actuators, communication device, and power supply.

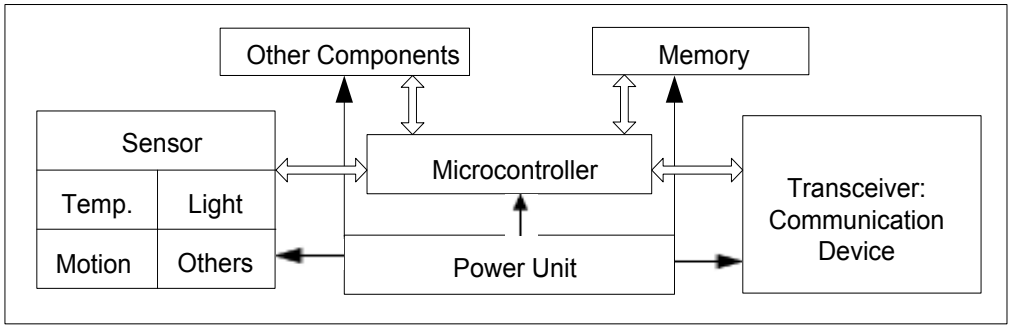

Figure 3.2: WSN Node Main Components

The controller is the core component of a WSN node. There are different options for the controller, where microcontroller is the best option that satisfies the need for general purpose processing, optimized for embedded applications, and low power consumption. Examples of microcontrollers are Texas Instruments MSP430 (16-bit RISC core, up to 4 MHz), Atmel Atmega128L (8-bit controller, larger memory than MSP430, and slower), where sensor nodes

such as Mica2 Mote, and Mica2dot use the Atmel Atmega128L microcontroller [31]. The main function of the controller is to collect and process data captured by the sensors, and most importantly decides when and where to send it. At the same time, monitoring the actuator behavior, the controller receives data from other sensor nodes.

In addition to the microcontroller, the node includes a RAM (for data) and ROM (for code) memory chips of limited capacity. The communication device of the node uses a radio transceiver to send and receive data (captured, or requests/commands) to or from other sensors or base stations. Sensors with different types can be directly connected to the node or integrated in a board and connected to the node through an extension.

Major hardware platform for WSN nodes are listed in Table 3.1. The most popular motes [31] are Mica2, MicaZ, and TelosB. The Mica2 platform is equipped with an Atmel Atmega128L and has a CC1000 transceiver. Intel has designed its own Imote that introduce various enhancements in the design over available mote, where the CPU processing power capacity is increased, together with the main memory size for on-board computing and improved radio reliability. In the Imote, a powerful ARM7TDMI core is complemented by a large main memory and non-volatile storage area; on the radio side, Bluetooth has been chosen.

TinyOS is the known operation system for WSN nodes, and it is supported by Btnode, Imote, Iris, Mica, Mica2, MicaZ, SenseNode, TelosB, T-Mote Sky, and Shimmer. Contiki, Mantis OS, SOS and Microsoft .NET Micro are other operating system supported by the nodes [31].

3.1.2 Applications of WSN

WSN are envisioned to play an important role in a wide variety of areas, such as critical military surveillance applications, forest fire monitoring, building security monitoring, child education, and micro-surgery are few examples of its applications [32]. In these networks, a large number of

sensor nodes are deployed to monitor a vast field, where the operational conditions are most often harsh or even hostile.

Table 3.1: Major Hardware Platform for WSN

Mote type	**CPU speed (MHz)**	**Prog. Mem (MB)**	**RAM (KB)**	**Radio freq (MHz)**
Mica2	16	128	4	433
MicaZ	16	128	4	2400
Cricket	16	128	4	433
TelosB/Tomte	16	48	10	2400
Imote2	13 – 416	32	256	2400

Table 3.2: WSN's Applications

Area	**Applications**
Military	- Enemy tracking and detection - Security threat detection - Military situation awareness [33] - Battlefield surveillance [34]
Environment	- Environmental data tracking - Forest fire monitoring - Fire/water detectors [35]
Habitat	- Animal tracking
Industry	- Inventory system [34] - Product quality monitoring [32]
Health	- Monitoring people locations and health conditions [34] - Sensors for: blood flow, respiratory rate, ECG (Electrocardiogram), pulse oxymeter, blood pressure, and oxygen measurement [36] - Monitor patients and assist disabled patients [32]
Smart Home/Office	- Life quality improvement
Automotive	- Coordinated vehicle tracking [33]

Same as WSN nodes can be utilized for environment monitoring; it can similarly be applied to monitor the behavior of human being. In the Smart Kindergarten project at UCLA [37], wirelessly-networked, sensor-enhanced toys and other classroom objects supervise the learning process of children and allow unremarkable monitoring by the teacher.

3.2 Security Issues in WSN

3.2.1 Constraints in WSN

WSN consists of a large number of sensor nodes which, and due to the limited energy and tiny size, have severe resources constraints in terms of processing power, storage capacity, and communication bandwidth. Because of these constraints, applying conventional security design for normal wired network becomes very challenging in WSN. To overcome this issue, and ensure a customized security measures and mechanism for WSN, it is essential to learn about these constraints and how it introduce security vulnerability or affect security measures for the WSN [38]. The major constraints of a WSN are listed below.

i. **Energy**: Energy is the main constraint for WSN, and because of the location setup of the WSN nodes, recharging nodes batteries is not always possible, and in most cases it is impractical and not feasibility. Power consumption constrains for nodes in the case of (1) sensor transducer, (2) communication among sensor nodes, and (3) microprocessor computation. Communication is more costly than computation in WSN (power consumption of transmitting one bit is equivalent to computing 800 to 1000 instructions [39]). Thus, higher security levels for WSN correspond to extra energy consumption [40].

ii. **Memory**: Memory and storage space is another constraint in WSN due to the node tiny size. In general, the memory of the node consists of flash memory (stores downloaded application code) and RAM (stores application programs, sensor data, and intermediate results of

computations). It is not always possible to run complex algorithms like public key cryptography as a security measure since the operating system and application code would use huge part of the memory. Hence the majority of the current security algorithms are infeasible in these sensors [41].

iii. **Communication**: The communication in WSN is connectionless and thus it is unreliable by default. This unreliability in its communication is a serious security threat to WSN nodes and may cause damaging or loosing communicated packets among the nodes. Some applications may not tolerate having damaged or lost packets, and thus require implementing packet recovery schemes, which involve extra cost (energy, memory, time). On the other side, in some situations, packet collision may occur due to the broadcast nature of the communication in WSN, and thus it may require retransmission of the packet [32].

3.2.2 Security Requirements in WSN

In addition to the above mentioned constraints in the WSN and since these networks are usually deployed in remote places and left unattended with no control and monitoring, these networks are vulnerable to numerous security threats that can adversely affect their proper functioning. Moreover, the characteristics of WSN are not limited to those of the conventional computer network, but it has many unique ones. In most of cases, critical information is frequently exchanged among sensor nodes through insecure wireless channels, it is therefore crucial to add security measures. Thus, in addition to the traditional security requirements such as data confidentiality, integrity, authenticity, and availability, WSN also require freshness, self-organization, secure location, and time synchronization. Brief on each security required service for WSN are listed below:

i. **Confidentiality**: Data can only be understandable by the authorized nodes. For instance, data captured by a sensor node must not be shared with unauthorized nodes [41], which require a strong key mechanism for key distribution, where these keys will be used to encrypt sensors ID, public key, location, etc. as a countermeasure against traffic analysis attacks.

ii. **Integrity**: Data is not tempered with by any unauthorized node. In some cases, an intruder intends to change the captured data by the node to introduce confusion in the decision process.

iii. **Authenticity**: Communicating node is the one that it claims to be. Also, this is applied to the received data packet be verified that have come from the known sender (as claimed) and not from an adversary. Message authentication code (MAC) is a well know technique used to ensure data authentication when communicated between two nodes. The MAC is generated using a share secret key between the nodes. Secure routing and reliable packet is major focus of authentication for WSN.

iv. **Availability**: Service is available regardless the presence of a security attack, namely the Denial of Service (DoS) attacks. The DoS attack usually refers to an adversary's attempt to disrupt, subvert, or destroy a network. However, a DoS attack can be any event that diminishes or eliminates a network capacity to perform its expected functions [42]. Approaches used to countermeasure the DoS are mainly by adding extra communication means, or introducing central control system for successful delivery insurance.

v. **Data freshness**: Data is current and no replay of old messages by adversary. In the absence of a proper secure mechanism for data freshness, an adversary in WSN may launch a replay attack using old secret shared key to assume secure message communication among the nodes. To defend against such replay attack, data packet may contain a nonce or an

incremental counter (linked to time) to validate the freshness of the communicated data packet.

vi. **Self-organization**: Due to the dynamic nature of WSN, it is not always viable to adopt a secure communication mechanism among the nodes and the base station, that relies on preinstalled shared key mechanism [43]. Nodes in a WSN should self-organize among themselves to satisfy the need of multi-hop routing protocols, and support deployment of key management schemes in the network.

vii. **Secure localization**: Accurate location of each node in a WSN must be securely communicated. In many applications for WSN, in addition to the captured data, the data packet communicated with other nodes or base station must contain information about the accurate node location. Different techniques are used for securing the node location, such as Verifiable Multilateration (VM) [44], and Secure Range-Independent Localization (SeRLoC) scheme [45].

viii. **Time synchronization**: Time synchronization is critical to most of the applications in WSN, in addition to its important role in node accurate and secure location. Time synchronization is required for collaborative data processing, signal processing techniques, and all security mechanisms for WSN.

3.2.3 Security Issues in WSN

WSN suffer from many constraints in terms of energy consumption, processing power, storage capacity, and communication bandwidth. In addition, this network uses an insecure wireless communication media, and most importantly it is vulnerable to physical attacks since it is unattended. These constraints make WSN more susceptible to various types of attacks. These attacks can be categorized as [34]:

i. **Attacks on secrecy and authentication**: Major external attacks on the secrecy and authenticity of WSN communication such as eavesdropping, packet replay attacks, and modification or spoofing of packets can be defeated by implementing standard cryptographic techniques.

ii. **Attacks on network availability**: DoS is a security attacks against the availability of WSN.

iii. **Stealthy attack against service integrity**: In a stealthy attack, the goal of the attacker is to make the network accept a false data value. For example, an attacker compromises a sensor node and injects a false data value through that sensor node. In these attacks, keeping the sensor network available for its intended use is essential. DoS attacks against WSN may permit real-world damage to the health and safety of people [42].

Moreover, since these networks are usually deployed in remote places and left unattended, it is crucial to implement security measures against physical attacks such as node capture, physical tampering, etc. A number of propositions exist in the literature for defense against physical attack on sensor nodes [42] [46] [47] [48] [49].

3.3 Implementations of ECC in WSN

Efficient computation of Public Key Cryptosystems (PKC) [2] in sensor nodes (e.g., [50] [51] [52] [53]) has been intensively investigated by researchers. Major problem with the sensor nodes as soon as it comes to cryptographic operations is their extreme constrained resources in terms of power consumption, space, and time delay, which limit the sensor capability to handle the additional computations required by cryptographic operations. Nevertheless, PKC is indeed shown to be feasible in WSN (e.g., [52] [53]) by using ECC. This is because, in comparison to traditional cryptosystems like RSA and ElGamal, ECC offers equivalent security with smaller

key sizes, in less computation time, with lower power consumption, as well as memory and bandwidth savings.

3.3.1 Hardware Implementations

This section presents a study of hardware implementations of ECC in WSN. A critical study of the underlying finite field, representation basis, occupied chip area, consumed power, and time delay performances of these implementations is conducted.

Several software implementations of ECC in WSN have been reported [52] [53] [54] [55] [56]. The advantages of software implementations include ease of use, ease of upgrade, portability, low development cost and flexibility. Their main disadvantages, on the other hand, are their lower performance and limited ability to protect private keys from disclosure compared to hardware implementations. These disadvantages have motivated many researchers to investigate efficient architectures for hardware implementations of ECC in WSN. Many hardware implementations of ECC in WSN have been reported [57] [58] [59] [60] [61] [62] [63]. Most of these implementations were for ECC defined over GF(2^m) [59] [60] [61] [62] [63], and only implementations in [57] [58] [59] were defined over GF(p).

The first hardware implementation of ECC was reported in 2005 by Gaubatz et. al. [57] [58] over GF(p). A custom-designed low power co-processor was presented in [59] [60]. The architecture of the presented co-processor occupies a chip area equivalent to 18,720 gates, using TSMC 0.13 μm CMOS standard cell technology, and consumes less than 400 μW of power at a clock frequency of 500 kHz. Field operations are implemented in a bit-serial fashion to reduce the area. Figure 3.3 shows the block diagram of the arithmetic unit used in [57] [58].

Wolkerstorfer [59] in 2005 implemented an ECC processor over dual-field performing both prime and binary field operations using polynomial basis. The presented processor has an area

complexity of around 23,000 gates implemented in 0.35 μm CMOS technology, operates at 68.5 MHz, consumes 500 μW of power and features a latency of 6.67 ms for one point multiplication. Figure 3.4 presents the architecture of the proposed processor in [59].

Batina et al. [60] in 2006 reported a low-power ECC processor over the binary field $GF(2^{131})$ using polynomial basis. The consumed power in the presented processor in [16] was less than 30 μW when the operating frequency is 500 kHz. The chip area of the presented work in [60] requires 6,718 gates using 0.13 μm CMOS technology.

Bertoni et al. [61] in 2006 proposed an efficient ECC coprocessor over $GF(2^{163})$ using polynomial basis. It computes the scalar multiplication in 17 ms at 8 MHz. The reported chip area was 11,957 gates using the 0.18 μm CMOS technology library by ST Microelectronics. The consumed power, on the other hand, was 305 μW. Figure 3.5 depicts the structure of the proposed coprocessor in [61].

Kumar and Paar [62] in 2006 reported an ECC processor over $GF(2^m)$ using polynomial basis. The word size range of the implemented processor was between 113 and 193 bits. The presented architecture in [62] consists of three units: $GF(2^m)$ addition (ADD), $GF(2^m)$ multiplication (MUL), and $GF(2^m)$ squaring (SQR) (See Figure 3.6). The area of the presented designs in [62] is between 10 k and 18 k gates on a 0.35 μm CMOS technology.

Recently, Portilla et al. [63] in 2010 reported an implementation of ECC over $GF(2^m)$ using polynomial basis on an FPGA, which incorporates a mixed solution based on an 8052 compliant microcontroller and a Xilinx XC3S200 Spartan 3 FPGA.

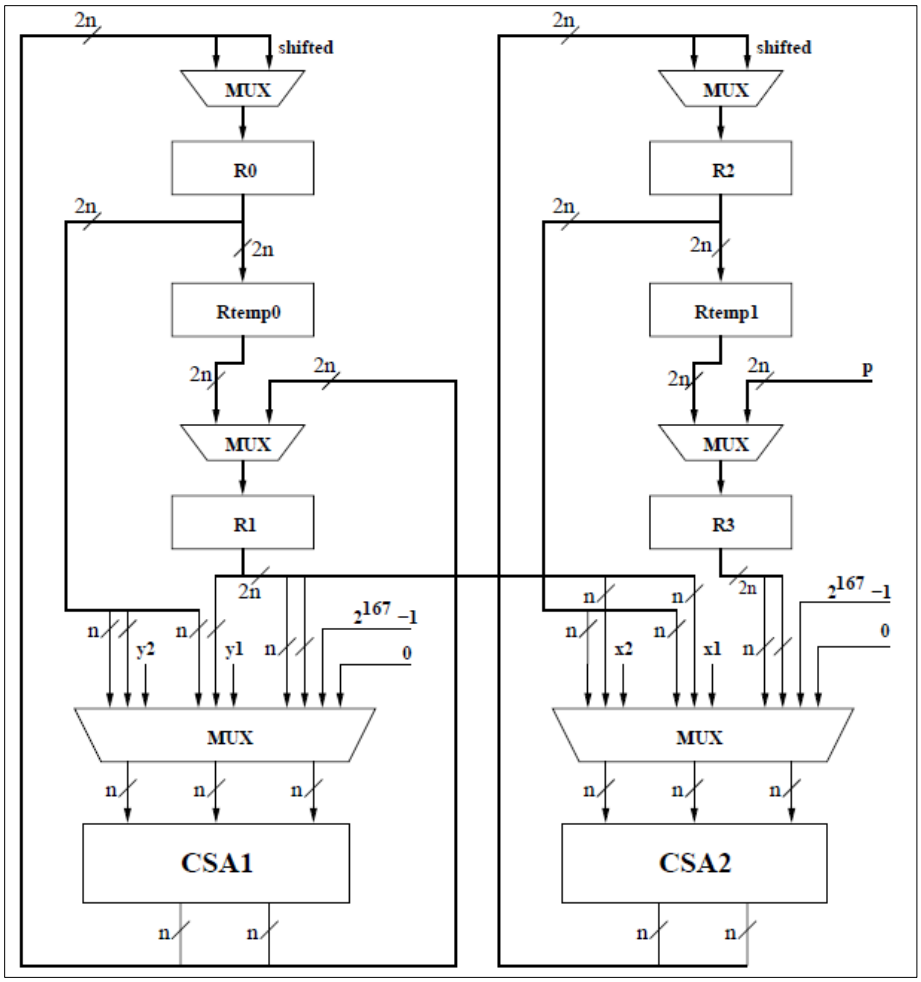

Figure 3.3: Block Diagram of the Arithmetic Unit Presented in [57] [58].

An additional XC2V2000 Virtex 2 FPGA is attached to the custom platform due to size limitations. The implemented field multiplier is generic and supports curve sizes from 163 up to 571 bits. The reported chip area is 98275 and 180317 gates for the word sizes 283 and 571 bits respectively, using the Xilinx XC2V2000 Virtex 2 FPGA. The reported power consumption, on the other hand, is 253 and 484 mA at 25 MHz for the word sizes 283 and 571 bits respectively.

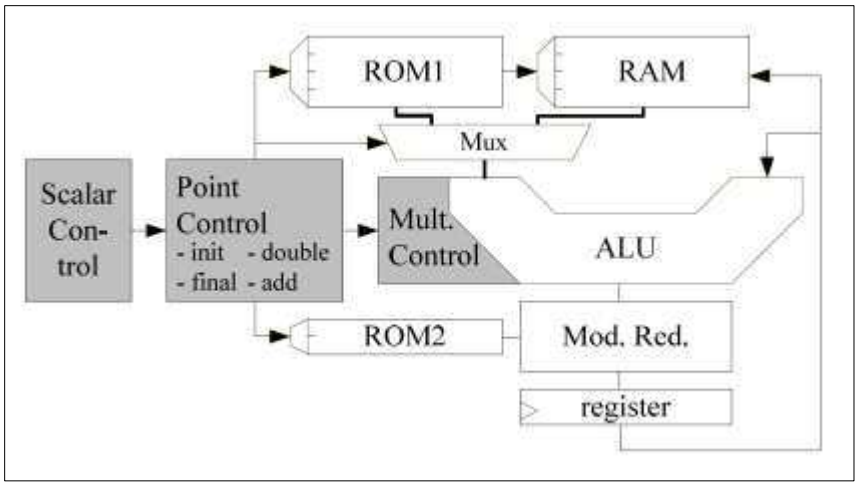

Figure 3.4: Architecture for ECC Processor in [59].

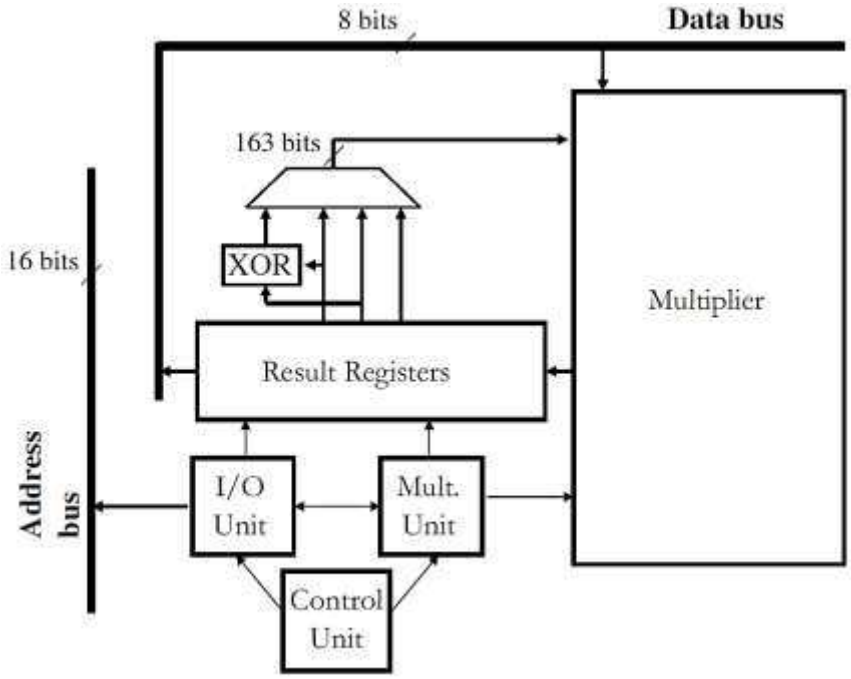

Figure 3.5: Structure of the 3-register coprocessor presented in [61].

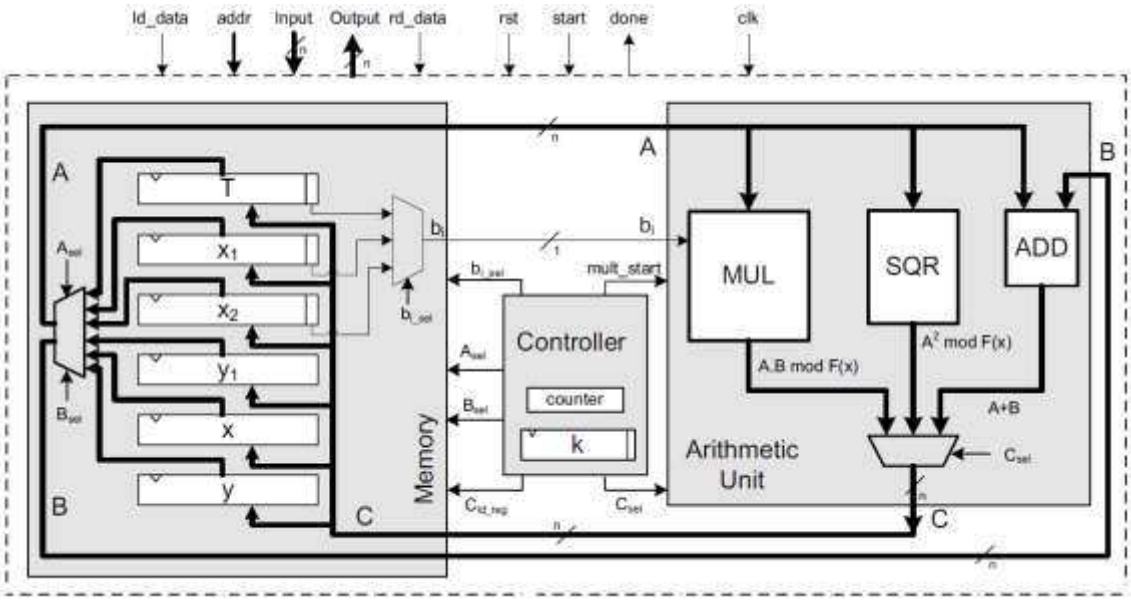

Figure 3.6: The ECC processor presented in [62].

3.3.2 Discussion on the Reviewed Hardware Implementations

The key focus of this section is in studying the hardware implementations of ECC in WSN, and emphasizing on the underlying finite field, representation basis, occupied chip area, consumed power, and time delay performances of these implementations (See Table 3.3). As shown in Table 3.3, the majority of the reported implementations used the GF(2^m) binary fields [59] [60] [61] [62] [63], and only two of these implementations used prime fields GF(p) [57] [58] [59]. This is due to the reason that GF(2^m) has shown to be best suited for cryptographic applications [25] [4]. Although it is known that normal basis representation provides more efficient hardware, Table 3.3 shows that only polynomial basis was used for all hardware implementations that used binary fields GF(2^m) [59] [60] [61] [62] [63]. This opens an opportunity to explore and inspect the performance of normal basis based ECC implementations in WSN.

Table 3.3: A Summary of hardware implementations of ECC in WSN.

Ref.	Underlying finite field	GF(2^m) Representation basis	Chip area (Gates)	Consumed power	Time performance
[57] [58]	GF(p) Word size: 100 bits		18,720 using TSMC 0.13 μm CMOS technology	Under 400 μW at 500 *kHz*	410.45 *ms* for one point multiplication at 500 *kHz*
[59]	GF(p) and GF(2^m) Word size: 192 bits	Polynomial basis	23,000 using 0.35 μm CMOS technology	500 μW at 68.5 *MHz*	6.67 *ms* for one point multiplication at 68.5 *MHz*
[60]	GF(2^m) Word size: 131 bits	Polynomial basis	6,718 using 0.13 μm CMOS technology	Less than 30 μW (when the operating frequency is 500 *kHz*)	115 *ms* for one point multiplication at 500 *kHz*
[61]	GF(2^m) Word size: 163 bits	Polynomial basis	11,957 using the 0.18 μm CMOS technology library by ST Microelectronics	305 μW at 8 *MHz*	17 *ms* for scalar multiplication at 8 *MHz*
[62]	GF(2^m) Word size [113, 131, 163, 193 bits]	Polynomial basis	Between 10,000 and 18,000 using 0.35 μm CMOS technology		[12.5, 16.8, 27.9, 38.8 ms] for scalar multiplication at 13.56 *MHz*.
[63]	GF(2^m) Word sizes [283, 571 bits]	Polynomial basis	Between 98,275 and 180,317 using Xilinx XC2V2000 Virtex 2 FPGA	253, 484 *mA* at 25 *MHz*	It computes the scalar multiplication in [750, 3600 μs] at 25 *MHz*.

Concerning the other parameters, the implementations in [59] and [62] performed ECC operation (point multiplication) in short time (6.67 ms for [59] at 68.5 MHz, and 18 ms for [62] at 13.56 MHz), but at the cost of high operating frequency and power consumption of 500 μW and an area

between 10k and 23k gates. On the other hand, implementation in [58] performed ECC operation in 410 ms at 500 kHz, consuming just less than 400 μW and occupying a chip area equivalent to 18,720 gates in 0.13 μm CMOS technology. The implementation in [60], however, is an enhancement of [58]. The presented design in [60] performed ECC operation in 115 ms at 500 kHz, consuming less than 30 μW using 8,104 gates in 0.13 μm CMOS technology. The implementation in [61], on the other hand, performed ECC in 17 ms at 8 MHz, consuming 305 μW and occupying a chip area of 11,957 using the 0.18 μm CMOS technology.

An important result of our study is found in the implementation of [63]. FPGAs were used in [63] showing that FPGAs can be used in WSN. It has been believed for a long time that FPGAs are not suitable for WSN applications because of their power consumption. However, the reported work in [63] opens the opportunity of exploring the performance of FPGAs in terms of area, time delay and power consumption.

3.3.3 Software Implementations

This section presents a study of software implementations of ECC over binary and prime fields in WSN. An analytical study of the underlying finite field, representation basis, and performance of these implementations is conducted.

Several ECC implementations in WSN have been reported [52] [53] [54] [55] [56] [57] [58] [59] [60] [61] [62] [63] [7] [64] [65] [66] [67] [68] [69] [70]. Many researchers investigated the efficient architectures for hardware implementations of ECC in WSN [57] [58] [59] [60] [61] [62] [63]. Given the advantages of software implementations include ease of use, ease of upgrade, portability, low development cost and flexibility; most of the research effort was on the software implementations of ECC in WSN [52] [53] [54] [55] [56] [7] [64] [65] [66] [67] [68] [69] [70].

The first implementation was implemented by Gura et al. [53] in 2004. They implemented elliptic curve point multiplication with 160-bit, 192-bit, and 224-bit NIST/SECG curves over GF(p) on two 8-bit microcontrollers. With assembly code and instruction set extension on an 8-bit Atmega128L processor, it took 0.81 s for ECC point multiplication on the 160-bit curve. Gura et al. [53] also proposed a new hybrid multiplication method, reducing the calculation time to 0.59 s. The presented work in [53] used mixed projective coordinates and Non-Adjacent Forms (NAFs) [71] to obtain optimized results. Inversion in [53] was implemented with the algorithm proposed by Chang Shantz [72]. The code size of the implementation in [53] is 3.682 K.

Malan et al. [52] presented the first implementation of ECC over GF(2^m) binary extension field curves for sensor networks (on 8 bits Atmega128L chip (MICA2 mote)). Inspired by the design of Dragongate Technologies Limited's Java-based jBorZoi 0.9 [73], they implemented ECC using a polynomial basis over GF(2^m), with a 163-bit key on a Koblitz curve, spending an average running time of approximately 34 s for point multiplication using just over 1 kilobyte of SRAM and 34 kilobytes of ROM, and total energy consumption of 0.816 J for public key generation. In [52], multiplication of points is achieved using Blake et al. [3] algorithm, while addition of points is achieved using L´opez and Dahab [74] algorithm. Field multiplication is implemented using L´opez and Dahab [75] algorithm, while inversion is implemented using Hankerson et al. [76] algorithm.

Blaß and Zitterbart [7] in 2005 implemented the arithmetic of GF(2^m) finite fields on the Atmels 8-bit Atmega128L microcontroller clocked at 7 MHz. The elements of the finite field of 113-bits were represented by normal bases. Random point multiplication (RPM) was implemented using an ECC version of the popular square-and-multiply algorithm for large number exponentiation as described in [71] and [77]. Fixed point multiplication (FPM) took about 6.74 s and 17.28 s for RPM, and ECDSA signature took 6.88 s and verification took 24.17 s with a total RAM of 208

Bytes and total ROM of 75.088 Kbytes. Blaß and Zitterbart [7] used offline pre-computation (of certain points), handcrafting (handcrafted optimization) as well as the Comb method and the double-and-add methods for point multiplication.

Haodong et al. [64] in 2005 implemented ECC over prime field, on TelosB mote (TPR2400) using the SECG recommended 160-bit elliptic curve: secp160r1. Haodong et al. [64] used a similar setup to the one in [63] using the hybrid multiplication method. Non-adjacent forms NAFs technique in RPM and sliding window technique [78] were adopted in this implementation. They achieved 3.13 s for FPM, and 3.51 s for RPM. For ECDSA implementation, generating a signature consumed roughly 18.09 mJ energy and verification costs 36.61 mJ. The implementation of [64] used 42.3 Kbytes ROM and 1.6 Kbytes RAM for ECDSA protocol, where the ECC Library used ROM (13.8 Kbytes), and RAM (1.3 Kbytes).

Wang and Li [55] in 2006 implemented 160-bit ECC - secp160r1 - cryptoprocessor over GF(p) on MICA mote sensors, achieved the performance 1.3 s for ECC signature generation and 2.8 s for verification, where 1.24 s for FPM and 1.35 s for RPM (signature 1.60 s and verification 3.30 s on TelosB). They adopted the hybrid multiplication method [53] in assembly language with column width d = 4. For modular reduction, the classic long division method was selected, that take advantage of pseudo-Mersenne primes specified in SECG curves, and for modular inversion an efficient Great Divide scheme [72] was adopted. Applied a mixed coordinate, and employed pre-computation using the sliding window method [78] and NAF [71]. The code of the ECC implementation is a total ROM of 75.2 Kbytes, and a total RAM of 3.06 Kbytes.

Yan and Shi [65] in 2006 implemented ECC over $F2^{163}$ and implemented the basic binary algorithm for point multiplication in 13.9 s and needs 12.412 Kbytes of memory, using fast modular reduction on an 8-bit processor at a clock rate of 8 MHz (Atmega128L). For scalar

multiplication, they implemented the basic binary algorithm that requires about m/2 additions and m doublings.

Ugus et al. [66] in 2007 presents an optimized implementation of EC-ElGamal on a MicaZ 8-bits processor mote over GF(p) with 160 bits. They used the mutual opposite form (MOF) instead of NAF. The performance of multiplications (with pre-computation) being executed with the MOF is (1.03 s as execution time), and with 2 pre-computed points takes 0.57 s. The used memory was 4.079 Kbytes.

Liu and Ning [67] in 2008 implemented TinyECC; a configurable library for ECC operations in WSN, on TinyOS with the underlying field primes p as pseudo-Mersenne primes. TinyECC [67] implementation of 192-bit ECC over GF(p) on MicaZ (Atmega128L 8-bit) mote sensors, achieved the performance of 2 s for ECC signature generation and 2.43 s for verification (Point multiplication of 2.99 s). TinyECC [67] used the weighted projective (Jacobian) representation, made use of the sliding window method (i.e. grouping a scalar k into s-bit clusters), adopted optimized modular reduction using pseudo-Mersenne prime, and used the Hybrid Multiplication to achieve computational efficiency.

Seo et al. [54] in 2008 presented TinyECCK (Tiny Elliptic Curve Cryptosystem with Koblitz curve - a kind of TinyOS package supporting elliptic curve operations) an ECC implementation over GF(2^m) on 8-bit sensor motes using ATmega128L using polynomial basis. In [54], TinyECCK with sect163k1 computed a scalar multiplication within 1.14 s on a MicaZ mote at the expense of 5,592 Bytes of ROM and 618 Bytes of RAM. Furthermore, TinyECCK with sect163k1 generated a signature and verified it in 1.37 s and 2.32 s with 13,748 Bytes of ROM and 1,004 Bytes of RAM.

Szczechowiak et al. [56] in 2008 implemented ECC on two sensor nodes platforms; the 8-bit Atmel ATmega128L processor (MICA2) and the 16-bit Texas Instruments MSP430F1611

processor (Tmote Sky). Szczechowiak et al. [56] uses the NIST k163 Koblitz curve over GF(2^{163}) binary field and over GF(p). The results show that a scalar multiplication took 2.16 s over binary field and 1.27 s over prime field on the MICA2 with energy consumption of 50.93 mJ and 30.02 mJ respectively. On the other hand, it took 1.04 s over binary field and 0.72 s over prime field for a scalar multiplication on the Tmote Sky mote with energy consumption of 10.76 mJ and 7.95 mJ. In [56], Szczechowiak et al. replaced standard C code with an assembly language specific for each platform. The Comb method for point multiplication (using additional storage to accelerate the calculations) described in [25] was used. Pre-computation was performed with window size w = 4 resulting in 16 elliptic curve points stored in ROM.

C. Lederer et al [68] in 2009, implemented a 192-bit ECC over prime field (generalized-Mersenne prime $p = 2^{192} - 2^{64} - 1$) on the MicaZ motes. Using fixed-base comb method with 14 pre-computed points, it requires 0.71 s to compute a scalar multiplication. A scalar multiplication using a random base point takes 1.67 s by applying window method with a window size of 4 (i.e. 14 pre-computed points), Based on the energy characteristics of the MicaZ mote [79], these timings translate into energy consumption of 17.04 mJ and 40.08 mJ, respectively. The implementation in [68] presented an improved version of Gura et al's [52] hybrid method for multi-precision multiplication that requires fewer single-precision additions. Also, it implemented the reduction operation as described in [80].

Khajuria et al. [69] in 2009 implemented a 163-bit ECC over GF(2^m) on 8-bit ATmega128L MicaZ platform from Crossbow. In their approach, S. Khajuria et al., in [69] used Koblitz curves and TNAF (τ - adic non-adjacent form) with partial reduction modulo and consumes 28.1 s for point multiplication, and the space consumption of this system is found to be 29.248 Kbytes in ROM and 1.070 Kbytes in RAM. For field multiplication, the right-to-left comb method was adopted.

Diego F et al. [70] in 2010 implemented a 163-bit ECC over GF(2^m) and Kobliz curves on 8-bit ATmega128L MicaZ platform. Diego F et al. [70] uses mixed addition with projective coordinates, given that the ratio of inversion to multiplication is 16. For RPM by a scalar, Solinas' τ - adic non-adjacent form (TNAF) representation with w = 4 was selected for Koblitz curves (4-TNAF method with 4 pre-computation points) and the method due to L´opez and Dahab was selected for random binary curves. For multiplying the generator, we employ the same 4-TNAF method for Koblitz curves; and for generic curves, we employ the Comb method [81] with 16 pre-computed points. Point multiplications took 0.67 s (Koblitz curves), and 1.55 s (Binary curve) for 163 bits.

3.3.4 Discussion on the Reviewed Software Implementations

The key focus of this section is in studying the software implementations of ECC over binary and prime fields in WSN, and emphasizing analytical study of the underlying finite field, representation basis, and performance of these implementations is conducted. For fair comparison, the study covers only fixed word size ECC on the same word size for processor mote. Those, implementations of 160-bit ECC over GF(p) on 8-bit processors [53] [55] [66] [56] are presented in Table 3.4 and implementations of 163-bit ECC over GF(2^m) on 8-bit processors [52] [54] [56] [65] [69] [70] are presented in Table 3.5.

In Table 3.4, the implementation in Ugus et al. [66] is significantly the fastest (0.57 s) and the implementation in Szczechowiak et al. [56] is the slowest (1.27 s) among all reported 160-bit implementations on 8-bit. The performance gain in Ugus et al. [66] implementation is primarily due to the use Mutual Opposite Form (MOF) instead of NAF and the use of the window and comb methods for scalar multiplication. Figure 3.7 illustrates the performance comparison for

these implementations. Though it was excluded from the comparison, it is worthy highlighting on the high performance of 0.71 s for 192-bit implementation in [68].

Table 3.4: 160-bits ECC over GF(p) in 8-bit processors in WSN

Ref.	Year	Performance (s)
Gura et al. [53]	2004	0.59
Wang and Li [55]	2006	1.24
Ugus et al. [66]	2007	0.57
Szczechowiak et al. [56]	2008	1.27

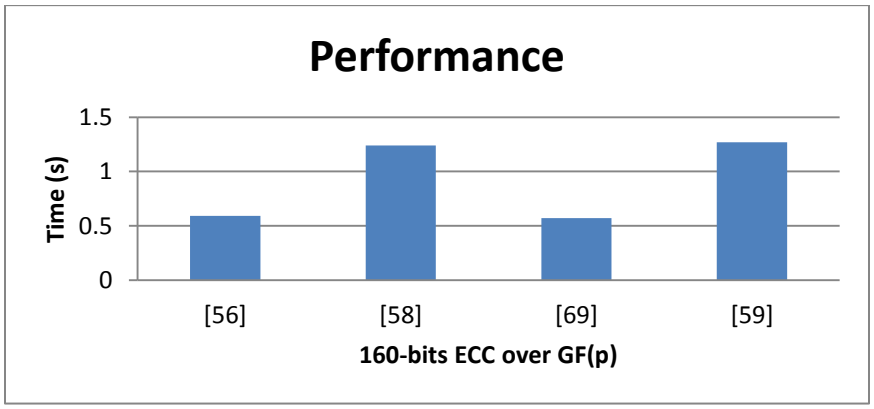

Figure 3.7: Implementation of 160-bits ECC over GF(p) in 8-bit processors in WSN

On the other side, the majority of the implementations over GF(2^m) are carried out using polynomial basis representation, expect for implementation in [7] that uses normal basis representation. In Table 3.3, the implementation in Diego F et al. [70] is slightly the fastest (0.67 s) and the implementation in Malan et. al. [52] is the slowest (34.17 s) among all reported 163-bits implementations on 8-bit. The performance gain in Diego F et al. [70] implementation is primarily due to the use of Koblitz curves, and Solinas' τ - adic non-adjacent form (TNAF) representation with w = 4. Figure 3.8 illustrates the performance comparison for these

implementations. Despite the fact that it was excluded from the comparison, it is significant stating that among the reported binary implementations; only one implementation is over normal basis (Blaß and Zitterbart [7]).

Table 3.5: GF(2^m) Polynomial basis 163-bit key 8-bit processor

Ref.	Year	Performance (s)
Malan et. al. [52]	2004	34.17
Yan and Shi [65]	2006	13.9
Seo et al. [54]	2008	1.14
Szczechowiak et al. [56]	2008	2.16
S. Khajuria et al. [65]	2009	28.1
Diego F et al. [70]	2010	0.67 (Koblitz curves)
Diego F et al. [70]	2010	1.55 (Binary curve)

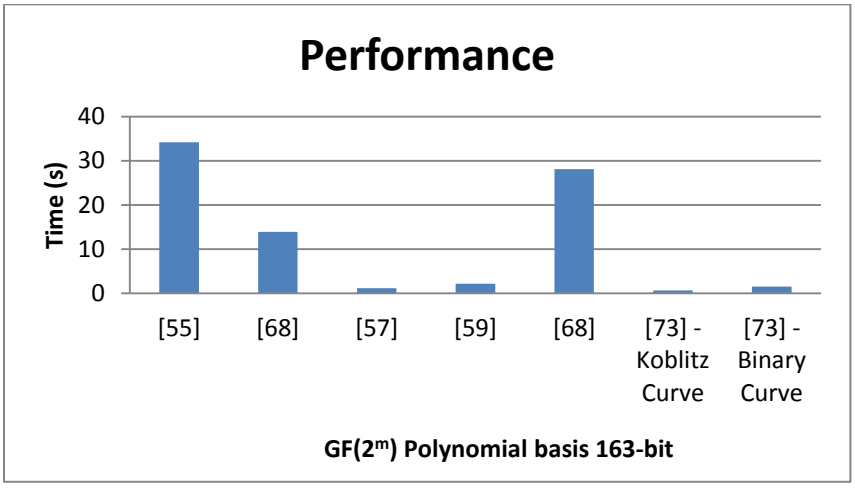

Figure 3.8: Implementation of 163-bits ECC over GF(2^m) in 8-bit processors in WSN

3.4 Summary

A study on both hardware and software implementations of ECC in WSN are presented in this chapter. The study covered the underlying finite field, representation basis, occupied chip area,

consumed power, and time delay performances of these implementations. The study shows that most of the reviewed hardware implementations were implemented on ASIC and only one was FPGA. However, it has been believed for a long time that FPGAs are not suitable for WSN applications because of their power consumption. Most of these implementations were implemented over the binary fields GF(2^m) and using polynomial basis representation. Despite that normal basis representation in GF(2^m) are more efficient in hardware implementations, all of the reviewed implementations were implemented using polynomial basis representation. This also opens an opportunity to explore the performance of ECC in WSN over GF(2^m) using normal basis representation.

For the software implementations of ECC in WSN, the study shows that the fastest prime field implementation, among all reported ones that uses 160-bit on 8-bit, took 0.57 s. As for the implementations over binary field, the study demonstrates that the majority of these implementations are carried out using polynomial basis representation, expect for one implementation that uses normal basis representation. Where the fastest binary field implementation, among all reported ones that uses 163-bit on 8-bit, took 0.67 s.

CHAPTER 4

Power Analysis Attacks on ECC in WSN and their Countermeasures

4.1 Introduction

As stated in Section 2.4 of Chapter 2, the scalar multiplication for Elliptic Curve Cryptosystems (ECC) [3] [4] is decomposed into a series of Point Additions (PADD) and Point Doublings (PDBL), and these point operations are the core for all ECC. The power and executing time requirements for PADD are different from those for PDBL on Wireless Sensor Networks (WSN) [1] nodes. In addition, the scalar multiplication algorithm performs PDBL for scalar bit value of 0, and PADD for bit value of 1, where the scalar represents the private key of the sensor mote.

Side Channel Analysis (SCA) attacks [8] exploit information leakage, such as power consumption and execution time, during the execution of an ECC protocol on WSN nodes, and thus will be able to learn about the entire private key as shown in Figure 4.1.

This chapter will provide an introduction to SCA attacks, with focus on the Power Analysis Attacks (PAA) and their countermeasures. The chapter ends by remarks on the reviewed countermeasures and a summary of the chapter.

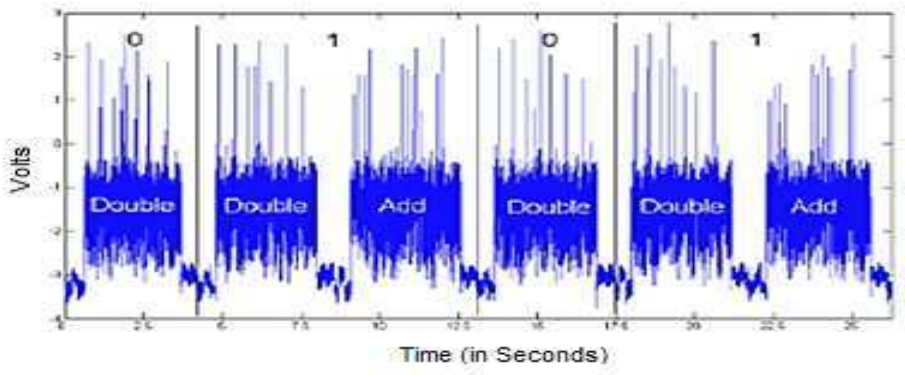

Figure 4.1: Power Traces revealing the private key of the WSN Node [82]

4.2 Power Analysis Attacks

Major nodes for WSN, such as Imote2, and MicaZ for instance, are manufactured by using CMOS (Complementary Metal-Oxide Semiconductor), where the logic inverter is its basic building block as depicted in Figure 4.2. The CMOS logic inverter [83] consists of two transistors namely P-channel and N-channel that serves as semiconductor switches and changes its status (ON or OFF) based on the input voltage V_{in}. A high voltage signal in V_{in} corresponds to logic 1 and logic 0 for low voltage signal. If the input voltage V_{in} is low, then P-channel transistor is conduction and N-channel is not conducting. In this case the current will flow from the supply voltage V_{dd} to the output and thus V_{out} is high. Therefore, the CMOS inverter logic circuit gives output 0 if the input is 1 and vice versa.

Hence, during the execution of a set of instructions, the consumed power by the device are expected to constantly change. Most importantly, measuring such consumed power during each clock cycle can be possible by using a resistor of one ohm value placed in series with the power supply and using an oscilloscope to measure the voltage change across the resistor.

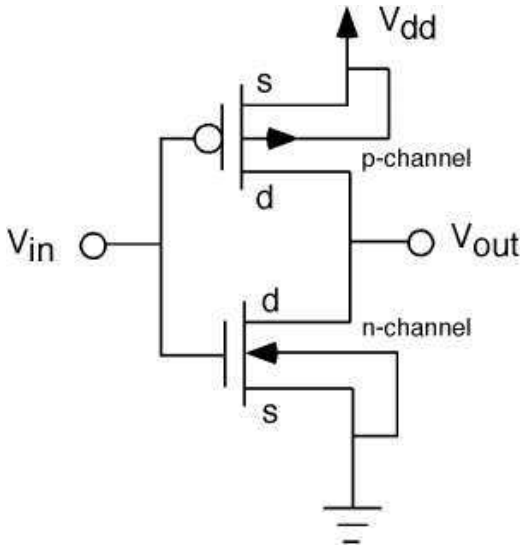

Figure 4.2: CMOS Inverter Logic Circuit [83]

In 1996, Paul Kocher introduced the power analysis procedure; then, in 1999 he introduced the PAA. These attacks have become a major threat against tamper resistant devices [84]. PAA [84] [85] allow adversaries to obtain the secret key in a cryptographic device, or partial information on it, by observing the power consumption traces. This is a serious threat especially to mobile devices such as WSN, smart cards, mobile phones, Radio Frequency Identity (RFID) [62] etc. Thus, implementers need algorithms that are not only efficient, but also PAA-resistant.

However, without adopting suitable countermeasures, an FPGA implementation is as vulnerable to power attacks as its software counterparts running on a processor. As a matter of fact, the transistors switching inside the device can leak information about the operations performed.

The following subsection presents the two main PAA techniques: 1) The Simple PAA (SPA) and 2) The Differential PAA (DPA) attacks.

4.2.1 Simple Power Analysis (SPA)

The main idea of the SPA attacks [85] is to get the secret d using the side-channel leakage information obtained through observing the power consumption from a single measurement trace.

For instance, as ECSM is the basic operation for ECC, and the most straightforward algorithm for point multiplication on an elliptic curve is the double-and-add algorithm (See Algorithm 4.1), where a PDBL is executed for each bit of the scalar and a PADD is executed only if the scalar bit is equal to one. If the power consumption trace pattern of PDBL is different from that of PADD, the side-channel leakage of the implementation reveals the presence of the PADD and thus the value of the scalar bits and attackers can easily retrieve the secret key from a single side-channel trace. Figure 4.3 shows the power trace for a sequence of PADD (represented by A) and PDBL (represented by D) operations on ECC.

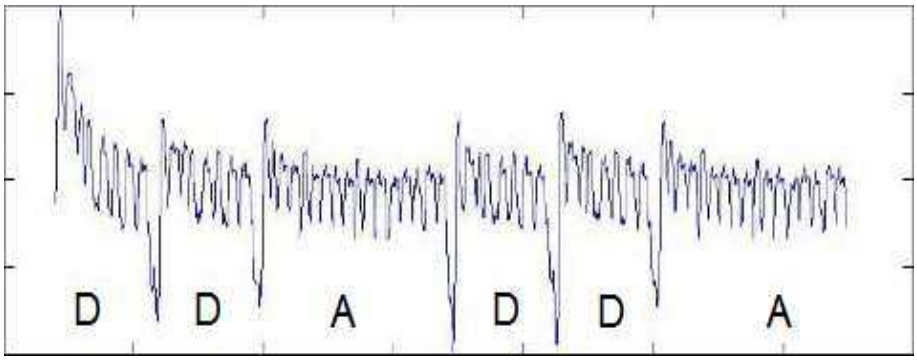

Figure 4.3: Power trace for a sequence of PADD and PDBL Operations on ECC

4.2.2 Differential Power Analysis (DPA)

In DPA attacks [85], the adversary makes use of the obvious variations in the power consumption that are caused by multiple data and operation computations, and use statistical techniques to pry the secret information. This attack uses a two round technique: data collection and data processing. A DPA attack on ECSM is described in [86].

More advanced DPA attacks techniques applicable to elliptic curve cryptosystems, such as refined power analysis (RPA) [87], zero power analysis (ZPA) [88], and doubling attacks [89] were introduced.

i. RPA (also called Goubin-type DPA) [87] attack directs its attention to the existence of a point P_0 on the elliptic curve E(K) such that one of the coordinates is 0 in K and $P_0 \neq O$. RPA could deduce the next bit of the scalar by computing power consumption of chosen message and some chosen points on the elliptic curve.

ii. ZPA attack [88] is an extension of RPA attack. This attack is based on the observation that that even if a point had no zero-value coordinate; the auxiliary register might take on a zero-value. Thus with this attack, all points with zero power consumption are noticeable.

iii. Doubling attack (DA) [89] attack is based on the two queries; one is on some input P and the other one is on 2P. The DA can detect when the same operation is done twice, i.e., exploits the similar PDBL operations for computing dP and d(2P), where d represent the scalar. There are two types of DA, normal and relative DA (relative doubling attack proposed by Yen et al. [90]), where the relative DA uses a totally different approach to derive the key bit in which the relationship between two adjacent key bits can be obtained as either $d_i = d_{i-1}$ or $d_i \neq d_{i-1}$.

iv. In addition, Template Attack [91] is very similar to DPA attack (Two rounds technique: Template building and matching), but requires access to a fully controllable device. In Template building phases (also called profiling phase), the attacker constructs a precise model of the wanted signal source, including a characterization of the noise. The matching phase comprises the actual attack.

v. Carry-based Attack (CBA) [92] is an attack that does not attack the ECSM itself but its countermeasures. This attack depends on the carry propagation occurring when long-integer additions are performed as repeated sub-word additions.

vi. Moreover, an advanced statistical technique such as Principal Component Analysis (PCA) [12] can be used by an attacker to perform PCA transformation on randomly switched PADD and PDBL (as in ECSM using Montgomery ladder) and identify the key bit.

4.3 Countermeasures

Since 1996, many research efforts [8] [9] [86] [93] [94] [95] [96] [97] [98] [99] have been made to secure ECC method implementations, in special the ECSM, against PAA. The major challenge is to avoid additional computational cost, and to develop relatively fast cryptosystems without compromising security, due to the nature of WSN as constrained devices.

4.3.1 Countermeasures for SPA

There are different strategies to resist SPA attacks. These strategies share the same objective, which is to render the power consumption traces that are caused by the data and operation computations during an ECSM independent from the secret key.

SPA attacks can be prevented by using one of the following methods:

1. Making the *group operations indistinguishable* (by processing of bits "0" and "1" of multiplier indistinguishable by inserting extra point operations). As an example, the 'Double-and-Add-Always' algorithm, introduced in [86] (As shown in Algorithm 4.1), and Montgomery ladder [94] (as shown in Algorithm 4.2) ensures that the sequence of operations appear as a PADD followed by a PDBL regularly.

'Double-and-Add-Always' algorithm [86] is highly regular, and it requires no pre-computation or prior recoding. This algorithm requires m PDBL and m PADD regardless of the value of the scalar multiplicand, and two temporary registers are needed to store the results of each iteration.

As for the Montgomery ladder [94], the execution time of the ECSM is inherently unrelated to the Hamming weight of the secret scalar, and this algorithm avoids the usage of dummy instructions. Montgomery ladder [94] resists the normal DA. However, it is attacked by the relative DA proposed by Yen et al. [90]. Moreover, recent studies have shown that processing the bits of multiplicand from left-to-right, as Montgomery ladder does, are vulnerable to certain attacks [89].

Algorithm 4.1 Double-and-Add-Always Elliptic Curve Scalar Multiplication Method
Inputs: *P: Base Point, k: Secret key.* **Outputs**: *kP.*
1: R[0] ← O 2: for i = m-1 down to 0 do 3: R[0] ← 2R[0], R[1] ← R[0] + P 4: R[0] ← R[k_i] 5: end for Return R[0]

Algorithm 4.2 Montgomery powering ladder Elliptic Curve Scalar Multiplication Method
Inputs: *P: Base Point, k: Secret key.* **Outputs**: *kP.*
1: R[0] ← P, R[1] ← 2P 2: for i = m - 2 down to 0 do 3: R[1 - k_i] ← R[0] + R[1] 4: R[k_i] ← 2R[k_i] 5: end for Return R[0]

In addition, the authors in [97] proposed secure (same security level as 'Double-and-Add-Always' method [86] and the Montgomery method [94]) and efficient ECSM method (See Algorithm 4.3) by partitioning the bit string of the scalar in half (Key splitting into half) and extracting the common substring from the two parts based on propositional logic operations. The computations for common substring are thus saved, where the computational cost is approximately (m/2) PADD + m PDBL.

2. Using of *unified formulae* for PADD and PDBL through inserting extra field operations [93] [95] [96] [9] [8] [98] [100] [101] [102], by rewriting the PADD and PDBL formulas so that their implementation provides always the same shape and duration during the ECSM.

Algorithm 4.3 Propositional Logic Operations Based Elliptic Curve Scalar Multiplication Method [97]
Inputs: *P: Base Point, k: Secret key.* $B_2 = (d_2^{m/2} \ldots d_2^e \ldots . d_2^1)_2$, $B_1 = (d_1^{m/2} \ldots d_1^e \ldots . d_1^1)_2$ **Outputs**: *kP.*
1: R[0] ← R[1] ← R[2] ← R[3] ← O 2: for i = 1 to m/2 do /* scan B1 and B2 from LSB to MSB */ 3: R[$2d_2^e + d_1^e$] ← R[$2d_2^e + d_1^e$] + P /* ADD */ 4: P ← 2P /* DBL */ 5: end for 6: R[1]← R[1] + R[3], R[2]← R[2] + R[3] 7: for i = 1 to m/2 do 8: R[2]← 2R[2] 9: end for 10: R[1]← R[2] + R[1] Return R[1].

An arithmetic was proposed in [93] and refined in [98] together with the use of Edwards coordinates for ECC as proposed by Bernstein and Lange in 2007 [103] uses the same formula to compute PADD and PDBL. In addition, Hesse [95] and Jacobi form [96] elliptic curves achieve the indistinguishability by using the same formula for both PADD and PDBL. Moreover, a method proposed by Moller [8] performs ECSM with fixed pattern of PADD and PDBL, employing a randomized initialization stage to achieve resistance against PAA. The same way, Liadet and Smart [9] have proposed to reduce information leakage by using a special point representation in some elliptic curves pertaining to a particular category, such that a single formula can be used for PADD and PDBL operations.

3. Rewriting sequence of operations as *sequences of side-channel atomic blocks* that are indistinguishable for SPA attacks [100]. The idea is to insert extra field operations and then

divide each process into atomic blocks so that it can be expressed as the repetition of instruction blocks which appear equivalent (same power trace shape and duration) by SCA. The atomic pattern proposed in [100] is composed of the following field operations: a multiplication, two additions and a negation. This choice relies on the observation that during the execution of PADD and PDBL, no more than two additions and one negation are required between two multiplications.

To reduce the cost of atomic pattern of [100], Longa proposed in his PhD thesis [101] two atomic patterns in the context of Jacobian coordinates. In [101] Longa expresses mixed affine-Jacobian PADD formula as 6 atomic patterns and fast PDBL formula as 4 atomic patterns. It allows performing an efficient left-to-right ESCM using fast PDBL and mixed affine-Jacobian addition protected with atomic patterns. In addition, the authors in [102] address the problem of protecting ECSM implementations against PAA by proposing a new atomic pattern. They maximize the use of squarings to replace multiplications and minimize the use of field additions and negations since they induce a non-negligible penalty.

4.3.2 Countermeasures for DPA

Same as in SPA attacks, there are different approaches and techniques [86] [87] [104] [81] [105] [106] used to resist DPA attacks. In general, the traditional and straightforward approach is by randomizing the intermediate data, thereby rendering the calculation of the hypothetical leakage values rather impossible.

Coron [86] suggested three countermeasures to protect against DPA attacks:

1. Blinding the scalar by adding a multiple of (#E).

   For any random number r and k' = k + r * (#E), we have k' * P = k * P since r * (#E) * P = O.

2. Blinding the point P, such that k * P becomes k * (P +R). The known value S = k * R is subtracted at the end of the computation. Blinding the point P makes RPA/ZPA more difficult.

In [89], the authors conclude that blinding the point P is vulnerable to DA since the point which blinds P is also doubled at each execution. Thereafter, in [104], the authors proposed a modification on the Coron's [86] point blinding technique to defend against the DA. The modified technique in [104] is secure against DPA attacks.

3. Randomizing the homogeneous projective coordinates (X,Y,Z) with a random $\lambda \neq 0$ to (λX, λY, λZ). The random variable λ can be updated in every execution or after each PADD or PDBL, which will makes the collection of typical templates more difficult for an attacker.

Although randomizing projective coordinates is an effective countermeasure against DPA attacks, it fails to resist the RPA as zero is not effectively randomized. Furthermore, if the device outputs the point in projective coordinates, a final randomization must be performed; otherwise [107] shows how to learn parts of the secret value.

Similar to Coron [86], Ciet and Joye [106] also suggested several similar randomization methods.

1. Random scalar splitting: $k = k_1 + k_2$ or $k = [k/r] * r + (k \bmod r)$ for a random r.

   Random scalar splitting can resist DPA attacks since it has a random scalar for each execution. In addition, it helps preventing RPA/ZPA if it is used together with Blinding the point P technique [10] [87] [108].

2. Randomized EC isomorphism.
3. Randomized field isomorphism.

In the same context, Joye and Tymen [105] proposed to execute the ECSM on an isomorphic curve and to change the intermediate representations for each execution of a complete ECSM.

In [81], the authors presented a PAA resistant ECSM algorithm, based on building a sequence of bit-strings representing the scalar k, characterized by the fact that all bit-strings are different from zero; this property will ensure a uniform computation behavior for the algorithm, and thus will make it secure against PAA attacks.

4.4 Remarks on the Reviewed Countermeasures

The main focus of this study is in highlighting on the PAA on ECC as a major security threat in the context of WSN. In a point of fact, none of the proposed countermeasures against PAA on ECC, which are suggested in literatures, have considered the case of WSN.

Given the resource constraints of WSN nodes, designing countermeasure methods against PAA seems a non-trivial problem, and it should be a matter of tradeoff between the available resources on WSN node and performance. Thus, some critical concerns need to be taken into consideration while designing such countermeasures:

1. Do not include any dummy operations (limited battery life time), and
2. Do not limit the design to particular family of curves, and thus can be implemented in any NIST standardized curves.
3. Immunity against DPA attacks may be carefully designed by combining several data randomization countermeasures and selectively change the ordering of these countermeasures with a time short enough to avoid a successful DPA attack.
4. Template attacks are serious security threats on WSN nodes especially when the template building is simple and fast.

In addition, as shown in Figure 4.4, different attacks could be thwarted by one or more countermeasures. For example, Random Projective Coordinate prevents three powerful attacks (DPA, DA, and Template attack). However, it is worthy to emphasis on the fact that finding a

countermeasure against all know attacks is extremely costly, especially in the context of constrained devices like WSN.

4.5 Summary

Taking into consideration the resource constraints of WSN nodes, its deployment in open environments make these nodes highly exposed to PAA. This chapter presented a comprehensive study of major PAA on ECC. The contributions of this chapter are as follows: First, we presented a review of the major PAA and its countermeasures on ECC. Second, we made a graphical presentation for the relation between PAA on ECC and its countermeasures. In addition, we discussed the critical concerns to be considered in designing PAA on ECC particular for WSN. Those, this chapter should trigger the need for intensive researches to be conducted in the near future on the PAA on ECC in WSN nodes, especially that ECC is considered as the most feasible PKC for WSN security.

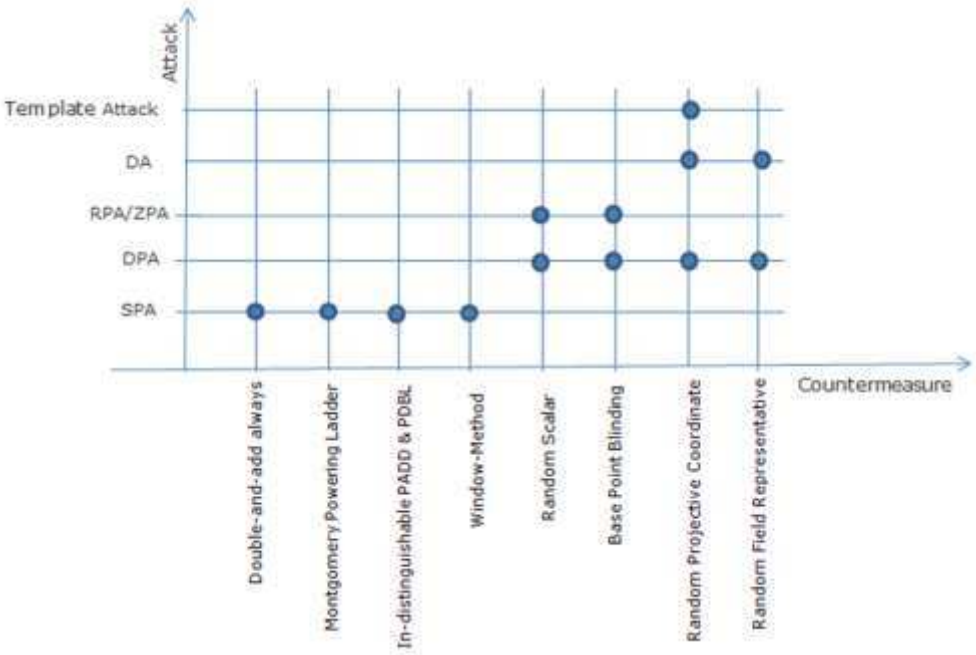

Figure 4.4: PAA vs. Countermeasures

Although attacks like PAA in WSN are normally carried out in situations where the adversary can control the target device [109], SPA attacks together with Template Attacks are still considered serious security threats, and thus a robust a cost-effect security solutions should be implementation to thwart these attacks.

CHAPTER 5

Architectures for ECC Cryptoprocessor Secure against SCA

Majority of cryptoprocessors for Elliptic Curve Cryptosystems (ECC) [3] [4] in extreme constrained resources such as sensor mote, Radio Frequency Identity (RFID) [62], and smartcards have been proposed and implemented over the binary fields GF(2^m) on Application Specific Integrated Circuits (ASIC) and only few using Field Programmable Gate Array (FPGA) [12] technology. Despite that normal basis representation in GF(2^m) are more efficient in hardware implementations, all of the reviewed implementations in this thesis were implemented using polynomial basis representation [110]. In addition, although Power Analysis Attacks (PAA) [9] are considered serious security threats on Wireless Sensor Networks (WSN) [1], none of the reported implementations provides security against all known PAA.

Thus, it is crucial to design ECC cryptoprocessor architectures (See Figure 5.1 – typical architecture for ECC coprocessor) for WSN implementations, and secure the cryptoprocessor against PAA. In this chapter, four robust, secure against PAA, and high efficient GF(2^m) elliptic curve cryptoprocessors architectures based on innovative algorithms for ECSM are proposed. The security advantages provided in these cryptoprocessors covers both the Simple Power Analysis (SPA) and Differential Power Analysis (DPA) attacks [9] [10] by applying: (i) Point Addition (PADD) operation delaying using buffer storage, (ii) Scalar splitting for cost saving and

additional complexity, and (iii) Complicated randomization technique for extra confusion to secure against DPA attacks.

The merits of these four cryptoprocessors are compared to the regular secure elliptic curve cryptoprocessor (ECC_{RG}) which is used as a reference for such comparison. The following sections and subsections provide details of the ECC_{RG} and the four proposed cryptoprocessors; namely:

1. ECC_{RG}: 'Double-and-Add'-based ECSM cryptoprocessor architecture with resistance against SPA attacks.
2. $ECC_{B\text{-}SPA}$: Buffer-based ECSM cryptoprocessor architecture with resistance against SPA attacks.
3. $ECC_{SB\text{-}SPA}$: Split Buffer-based ECSM cryptoprocessor architecture with resistance against SPA attacks.

On the other side,

4. $ECC_{RB\text{-}DPA}$: Randomized Buffer-based ECSM cryptoprocessor architecture with resistance against DPA, and
5. $ECC_{RSB\text{-}SPA}$: Randomized Split Buffer-based ECSM cryptoprocessor architecture with resistance against DPA attacks.

5.1 Architecture for regular GF(2^m) Elliptic Curve Cryptoprocessor

This section presents the architecture of a regular GF(2^m) elliptic curve cryptoprocessor, named ECC_{RG} which is based on the 'Double-and-Add' algorithm and provides security against SPA attacks. The proposed architecture is modeled using VHDL, stands for very high-speed integrated

circuit hardware description language, and is fully parameterized. The basic units of this architecture are: 1. the main controller, 2. the data embedding unit, 3. the PADD and Point Doubling (PDBL) units and 4. the field arithmetic units (adder, multiplier and inverter). In the following subsections, these units are described in details (Figure 5.1).

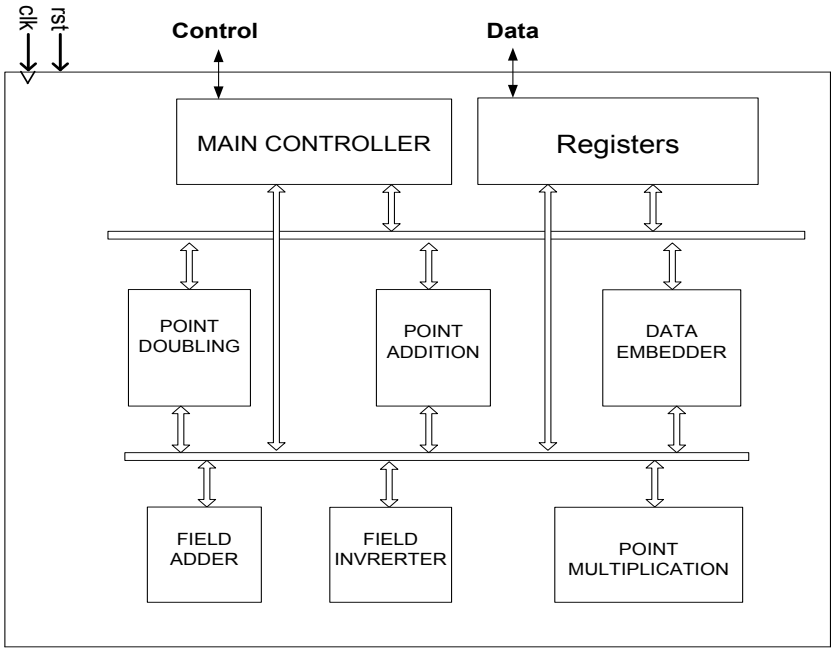

Figure 5.1: Architecture of the ECC coprocessor

5.1.1 Main Controller

The 'Double-and-Add' algorithm has been selected for scalar multiplication (Algorithm 4.1). For the encryption/decryption process, the selected encryption protocol is the elliptic curve Diffie-Hellman protocol [69]. The pseudocode of the ECC_{RG} cryptoprocessor is given in Algorithm 5.1. The input of Algorithm 5.1 are: (1) the base point P, (2) the elliptic curve parameters a,b, (3) the secret key k, (4) the encryption/decryption mode and (5) the plaintext/cipher text. The output is either the cipher text or the plaintext depending on the encryption/decryption mode.

Algorithm 5.1 Pseudocode of the ECC_{RG} Cryptoprocessor
Inputs: *P: Base Point, k*: *Secret key; a, b: Elliptic curve parameters, Plaintext/Ciphertext, Encryption/Decryption* **Outputs**: *Ciphertext/Plaintext.*
Scalar Scalar Multiplication (*kP*): 1: Algorithm 4.1(*P,k*) # Encryption/Decryption Process: 2: if (Encrypt) then 2.1: Embed the plaintext in random points on the elliptic curve 2.2: ADD (*kP*) to data points 2.3: Output (ciphertext) 3: else 3.1: ADD (− *kP*) to ciphered points 3.2: Extract the plaintext from the data points 3.3: Output (plaintext)

Referring to the cryptoprocessor pseudocode (Algorithm 5.1), scalar multiplication starts at Step 1 by executing the 'Double-and-Add-Always' ECSM algorithm (Algorithm 4.1). The encryption process starts at Step 2 by embedding the plaintext into a random point on the elliptic curve using "blinding the point" technique. The scalar multiplication result (kP) is added to this point to produce a ciphered point. The decryption process (Step 3), however, subtracts (kP) from the ciphered point.

5.1.2 Data Embedding

Data embedding is performed within the x-coordinate of a point on the elliptic curve. A random number is picked to fill the 5 most significant bits and the remaining bits will contain the data to be encrypted. If the x-coordinate is not a valid point on the elliptic curve, another random number is picked until a valid elliptic curve point is obtained.

The checking procedure is as follows [111]:

- Recall the elliptic curve equation defined over GF(2^m):

$$y^2 + xy = x^3 + ax^2 + b \quad \text{(Equation 5.1)}$$

Where a, b ϵ GF(2^m) and b ≠ 0.

- Rewrite Equation 5.1 as

$$y^2 + xy + f(x) = 0 \quad \text{(Equation 5.2)}$$

where $f(x) = x^3 + ax^2 + b$.

- Let $y = zx$, Equation 5.2 becomes:

$$z^2 + z + c = 0 \quad \text{(Equation 5.3)}$$

where

$$c = f(x).x^{-2} \quad \text{(Equation 5.4)}$$

- Find the trace of c, the trace function is simply the parity function which can be easily implemented by computing the XOR of all the bits.
- If the trace is 1, try another random number and repeat the check again. If the trace is 0, this is a valid x-coordinate and proceed to recover the y-coordinate.
- By taking the square root of Equation 5.3, it can be rewritten as:

$$z^{1/2} = z + c^{1/2} \quad \text{(Equation 5.5)}$$

which can be also rewritten as:

$$z_i = z_{i-1} + c_i \quad \text{(Equation 5.6)}$$

- Since z + 1 is actually the complement of z in a normal basis, in one of the two solutions the least significant bit will be 0 and the other one will be 1. We then further compute all the other bits one by one.
- To compute the y value, simply multiply z by x.

5.1.3 Point Addition and Doubling

PADD and PDBL are performed using Lopez-Dahab projective coordinate system which takes the form (x,y) = (X/Z, Y/Z^2) [23]. PADD and PDBL require only 14 and 5 field multiplications respectively (Table 2.3). The projective elliptic curve equation of the affine Equation 5.1 is given by

$$Y^2 + XYZ = X^3Z + aX^2Z^2 + bZ^4 \qquad \text{(Equation 5.7)}$$

If Z = 0 in Equation 5.7, then $Y^2 = 0$, i.e., Y = 0. Therefore, (1, 0, 0) is the only projective point that satisfies the equation for Z = 0. This is the point at infinity O [23]. To convert an affine point (x, y) into Lopez-Dahab projective coordinate, set X = x, Y = y, Z = 1. Similarly, to convert a projective point back to affine coordinate, we compute x = X/Z, $y = Y/Z^2$. The additive inverse of a point P = (X, Y, Z) is the point (X, XZ+Y, Z) which is used at the end of the decryption process [17].

The projective point operations formulas of Lopez-Dahab coordinate system [23] has been reported only for the most-to-least version of the scalar multiplication algorithm. Alternatively, PDBL and PADD formulas that are suitable for both versions of the scalar multiplication algorithm are proposed in Table 5.1. Clearly, the doubling formula requires only 5 field multiplications, 5 field squarings and 5 storage registers. PADD formula requires 14 field multiplications, 6 field squarings and 8 storage registers.

5.1.4 Field Operations

One key advantage of normal basis representation is the simplicity of the squaring operation. Field squaring is simply a cyclic shift operation. Field addition is a Boolean XOR operation and is implemented using an m-bit XOR unit. Thus, only one clock cycle is required to perform either of the two operations, i.e., field squaring or field addition.

Field multiplication is more complicated than addition and squaring. An efficient multiplier is highly needed and is the key for efficient finite field computations. Massey-Omura multiplier was selected for field arithmetic [112]. Since we are using FPGA as implementation technology to evaluate our proposed architectures, we have adopted for implementing the bit-serial version of the Massey-Omura multiplier to save on available FPGA resources. The Massey-Omura multiplier requires only two m-bit cyclic shift registers and combinational logic. The combinational logic consists of a set of AND and XOR logic gates (See Figure 5.2). The first implementation of the Massey-Omura multiplier was reported by Wang. et. al. [113]. The space complexity of the Massey-Omura multiplier is (2m – 1) AND gates + (2m – 2) XOR gates, while the time complexity is $T_A + (1 + \log_2 (m - 1))\ T_X$, where T_A and T_X are the delay of one AND gate and one XOR gate respectively. One advantage of the Massey-Omura multiplier is that it can be used with both types of the optimal normal basis (ONB) (Type I and Type II). Another advantage is that it is a bit-serial multiplier and hence the same circuitry used to generate c_0 can be used to generate c_i (i = 1,2, … m – 1) as shown in Figure 5.2 [114].

The encryption/decryption process requires only one inversion since we are using projective coordinate (See Equation 5.4), while an inversion per trial is required for data embedding in a valid x-coordinate. Thus, an efficient inverter is required. The selected inverter is the Itoh and Tsujii inverter [115].

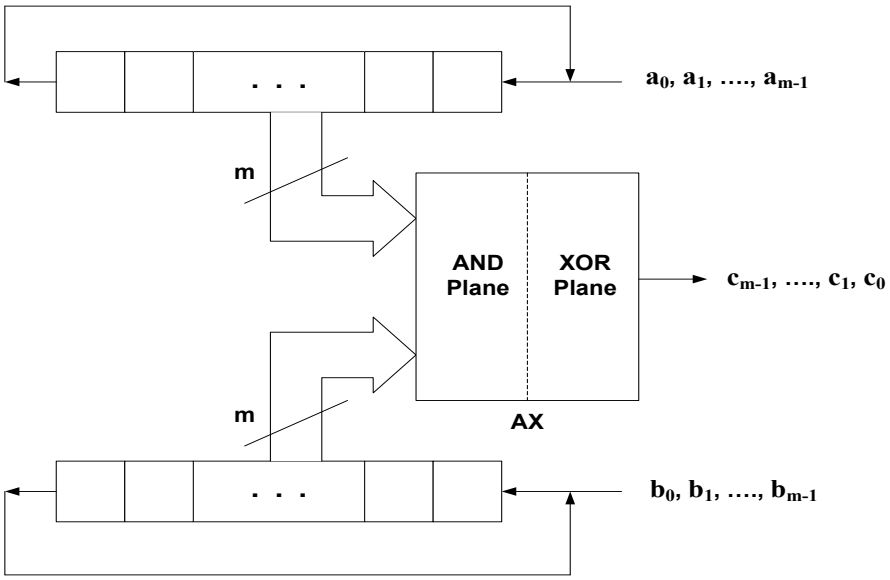

Figure 5.2: The bit-serial Massey–Omura multiplier of GF(2^m) [112].

The dataflow of the Itoh–Tsujii inverter is shown in Figure 5.3. Figure 5.3 shows that Itoh–Tsujii inverter requires three cyclic shift registers; one barrel shifter, one down counter and one multiplier (note that only one multiplier is used while two are drawn in the dataflow diagram for the purpose of clarity).

In Figure 5.3, the down counter s controls the barrel shifter r in each iteration. The barrel shifter r, accordingly, controls the required number of squarings by the cyclic shift register q. The least bit of the barrel shifter r_0, on the other hand, decides if the multiplication of the content of the cyclic shift register t by **a** is required or not. The Itoh–Tsujii Inversion algorithm is given in Algorithm 5.2. Clearly, the inverter depends a lot on the field multiplier. The Itoh–Tsujii Inversion algorithm requires only O ($\log_2$ (m)) multiplications, which is the best among other inversion algorithms reported thus far [114].

Algorithm 5.2 Itoh–Tsujii Inversion Algorithm.
Inputs: *a*. **Outputs**: $l = a^{-1}$
1: set s ← $[\log_2(m-1)] - 1$, set p ← a 3: for i = s down to 0 do 3.1: set r ← shift m − 1 to right by s bit(s) 3.2: set q ← p 3.3: rotate q to left by [r/] bit(s) 3.4: set t ← p x q 3.5: if least bit of r = 1 then 3.5.1: rotate t to left by 1 bit, p ← t x a 3.6: else 3.6.1: p ← t 3.7: s ← s – 1 4: rotate p to left by 1 bit 5: set l ← p return l

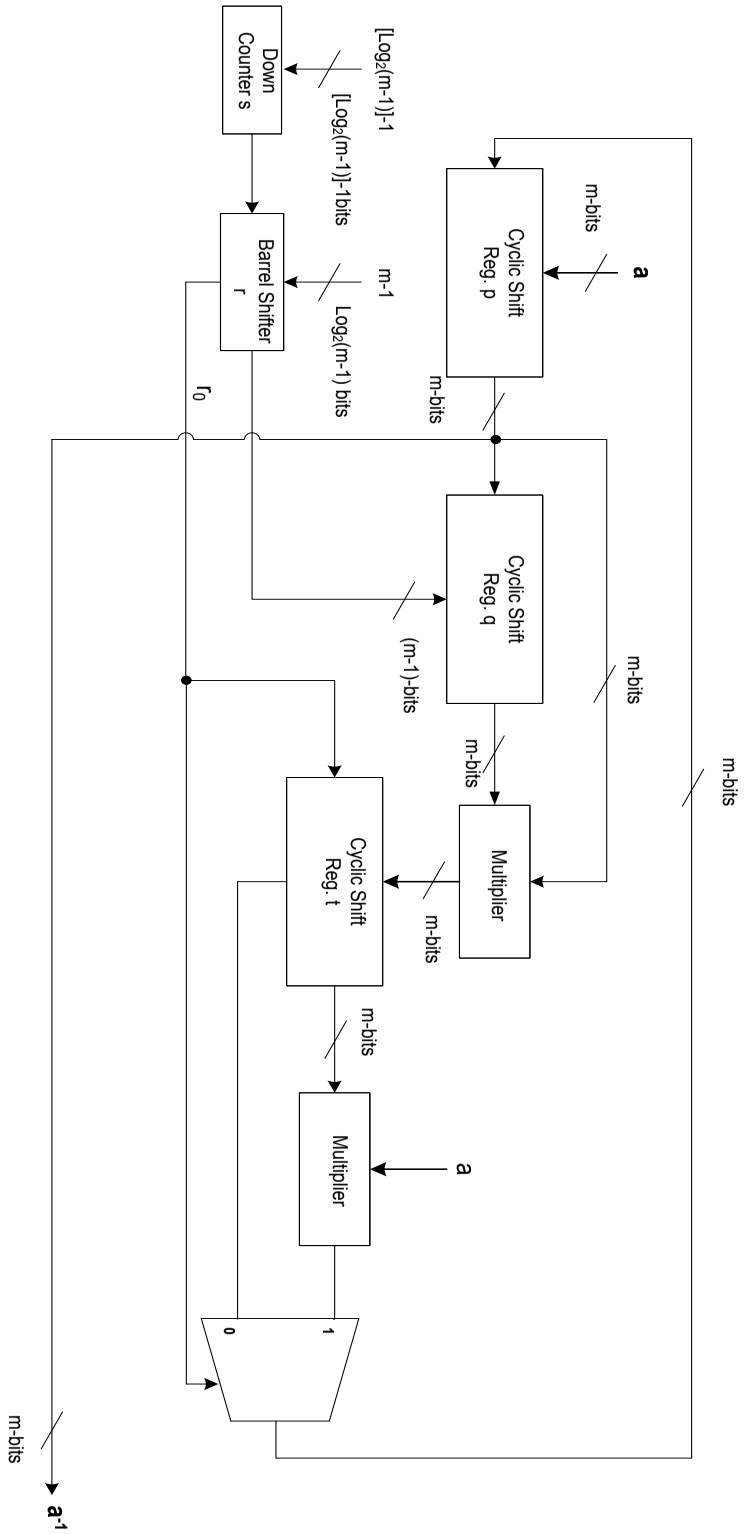

Figure 5.3: Dataflow of the Itoh and Tsujii inverter

Table 5.1: Lopez-Dahab Projective Coordinate System

PDBL	PADD
$T_1 \leftarrow X_1$ $T_2 \leftarrow Y_1$ $T_3 \leftarrow Z_1$ $T_4 \leftarrow \sqrt{b}$ $T_3 \leftarrow T_3^2$ $T_3 \leftarrow T_3 \times T_4$ $T_4 \leftarrow T_4^2$ $T_1 \leftarrow T_1^2$ $T_3 \leftarrow T_1 \times T_3 = Z_2$ $T_1 \leftarrow T_1^2$ $T_1 \leftarrow T_1 + T_4 = X_2$ $T_2 \leftarrow T_2^2$ if $a \neq 0$ then $T_5 \leftarrow a$ $T_5 \leftarrow T_3 \times T_5$ $T_2 \leftarrow T_2 + T_5$ $T_2 \leftarrow T_2 + T_4$ $T_2 \leftarrow T_1 \times T_2$ $T_4 \leftarrow T_3 \times T_4$ $T_2 \leftarrow T_2 \times T_4 = Y_2$	$T_1 \leftarrow X_0$ $T_2 \leftarrow Y_0$ $T_3 \leftarrow Z_0$ $T_4 \leftarrow X_1$ $T_5 \leftarrow Y_1$ $T_6 \leftarrow Z_1$ $T_7 \leftarrow T_3 \times T_6 = E$ $T_1 \leftarrow T_1 \times T_6 = B_1$ $T_4 \leftarrow T_3 \times T_4 = B_0$ $T_1 \leftarrow T_1 + T_4 = D$ $T_3 \leftarrow T_3^2$ $T_6 \leftarrow T_6^2$ $T_3 \leftarrow T_3 \times T_5 = A_0$ $T_6 \leftarrow T_2 \times T_6 = A_1$ $T_6 \leftarrow T_3 + T_6 = C$ $T_2 \leftarrow T_1 \times T_7 = F$ $T_1 \leftarrow T_1^2$ $T_8 \leftarrow T_7^2$ $T_8 \leftarrow a \times T_8$ $T_8 \leftarrow T_2 + T_8$ $T_5 \leftarrow T_1 + T_8 = G$ $T_8 \leftarrow T_2 + T_6 = H$ $T_6 \leftarrow T_6^2$ $T_6 \leftarrow T_6 + T_8$ $T_6 \leftarrow T_5 + T_6 = X_2$ $T_4 \leftarrow T_1 \times T_4$ $T_4 \leftarrow T_4 \times T_7$ $T_4 \leftarrow T_4 + T_6 = I$ $T_3 \leftarrow T_1 \times T_3$ $T_3 \leftarrow T_3 + T_6 = J$ $T_4 \leftarrow T_4 \times T_8$ $T_2 \leftarrow T_2^2 = Z_2$ $T_2 \leftarrow T_2 \times T_3$ $T_8 \leftarrow T_3 + T_4 = Y_2$

5.2 Proposed Architectures for ECC Secure against SPA

In this section, two proposed architectures for elliptic curve cryptoprocessors are presented; these cryptoprocessors provide resistance against SPA attacks. The first cryptoprocessor is Buffer-based, called $ECC_{B\text{-}SPA}$, and it uses an ECSM method that is based on delaying the PADD operation using buffering technique (with one buffer); whereas the second cryptoprocessor is Split Buffer-based, called $ECC_{SB\text{-}SPA}$, and it uses an ECSM method that is based on splitting the scalar into two equal length partitions and delaying the PADD operation using buffering technique (with three different buffers).

5.2.1 The $ECC_{B\text{-}SPA}$ Cryptoprocessor

This subsection introduces the Buffer-based cryptoprocessor ($ECC_{B\text{-}SPA}$), it uses a scalar multiplication method that is derived from the binary method (See Algorithm 2.2), and is based on delaying the PADD operation using buffering technique. The pseudocode of the Buffer-based ECSM method is given in (Algorithm 5.3).

5.2.1.1 Background Information on the $ECC_{B\text{-}SPA}$ Cryptoprocessor

In order to give background information on the $ECC_{B\text{-}SPA}$ cryptoprocessor, it is signification to recall that in the right-to-left version of the binary method (See Algorithm 2.2) of the ECSM, PADD is only performed if the bit value $k_i = 1$, while PDBL is always performed regardless of the bit scalar value. The mathematical equation for the binary method is given below:

$$kP = \sum_{i=0}^{m-1} 2^i k_i P \qquad \text{(Equation 5.8)}$$

where k is the scalar, P is the base point.

In Equation 5.8, the scalar multiplication result is the conditional summation of PDBL operation of P at position i of the scalar where $k^i = 1$. In addition, Equation 5.8 can be rewritten as below:

$$kP = 2^0 P|_{(k^o = 1)} + 2^1 P|_{(k^1 = 1)} + 2^2 P|_{(k^2 = 1)} + \cdots 2^{m-1} P|_{(k^{m-1} = 1)} \qquad \text{(Equation 5.9)}$$

In Equation 5.9, the scalar is divided into a number s of partitions, we call it "scalar partitioning on 1's", where each partition is associated with a computed point ($2^i P \mid k_i = 1$) to keep its significance [116]. The partition is defined as the bit string of length j and only contains one bit "1".

$$K = k^{(s-1)} \,||\, k^{(s-2)} \,||\, \ldots \,||\, k^{(1)} \,||\, k^{(0)}$$

For example, key length of 16 bits, and k = 42,395 = $(1010010110011011)_2$, can be partitioned as depicted below in Figure 5.4:

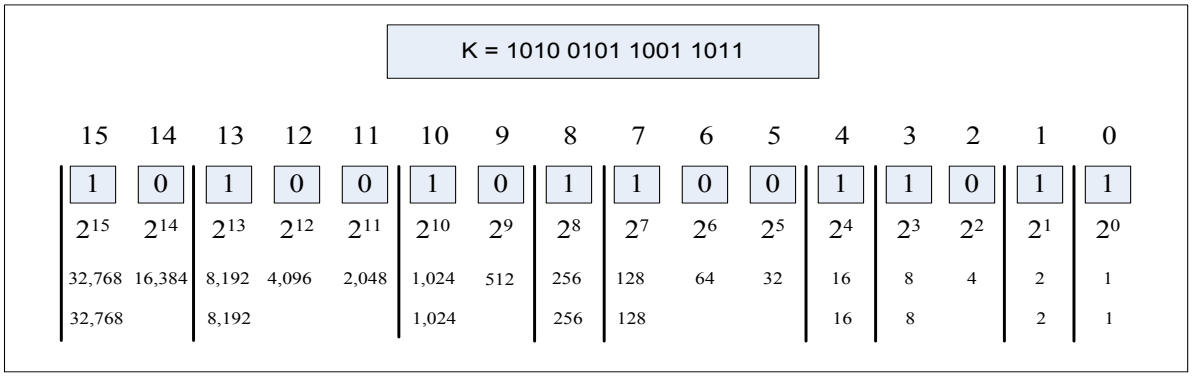

Figure 5.4: Example for "Scalar Partitioning on 1's"

5.2.1.2 Description of the $ECC_{B\text{-}SPA}$ Cryptoprocessor

To protect against PAA, the point operations (PADD and PDBL) of the ECSM must be independent of the scalar bit value k_i. In addition, since each key partition is associated with a

computed point to keep its significance, and the resulting points from processing these key partitions are accumulated to produce the scalar multiplication kP; therefore, PADD operation can be performed at a delayed time, and not necessarily at the corresponding scalar bit position. Accordingly, the proposed Buffer-based method for scalar multiplication is based on delaying the PADD operation using buffering technique, i.e., this proposed method store points into buffer and perform the PADD operations in later stage as elaborated in its algorithm (Algorithm 5.3) and shown in its dataflow (as depicted in Figure 5.5).

Algorithm 5.3 Buffer-based ECSM Method
Inputs: *P: Base Point, k: Secret key, r is capacity limit of buffer* **Outputs:** *kP.*
1: R[0] ← O, t ← 1 /* set buffer index t to 1 */ 2: for i = 0 to m-1 do 2.1: B[t] ← P /* scan k, store points in buffer */ 2.2: P ← 2P 2.3: If (t ← r) or (i ← m-1), then /* buffer reach its capacity limit or scan k is completed */ 2.3.1: for s = 1 to t do 2.3.1.1: R[0] = R[0] + B[s] 2.3.2: t ← 1 /* reset buffer index to 1 */ 2.4 else t ← t + k_i /* increment t if the bit value of k is 1 */ Return R[0]

In in Figure 5.5, the scalar is scanned from right to left and for every scalar bit value:

1. Perform a PDBL operation.

   PDBL operation keeps the significance of the point value at the scalar bit position of the scalar.

2. Write to buffer the updated value of P (result of PDBL operation)

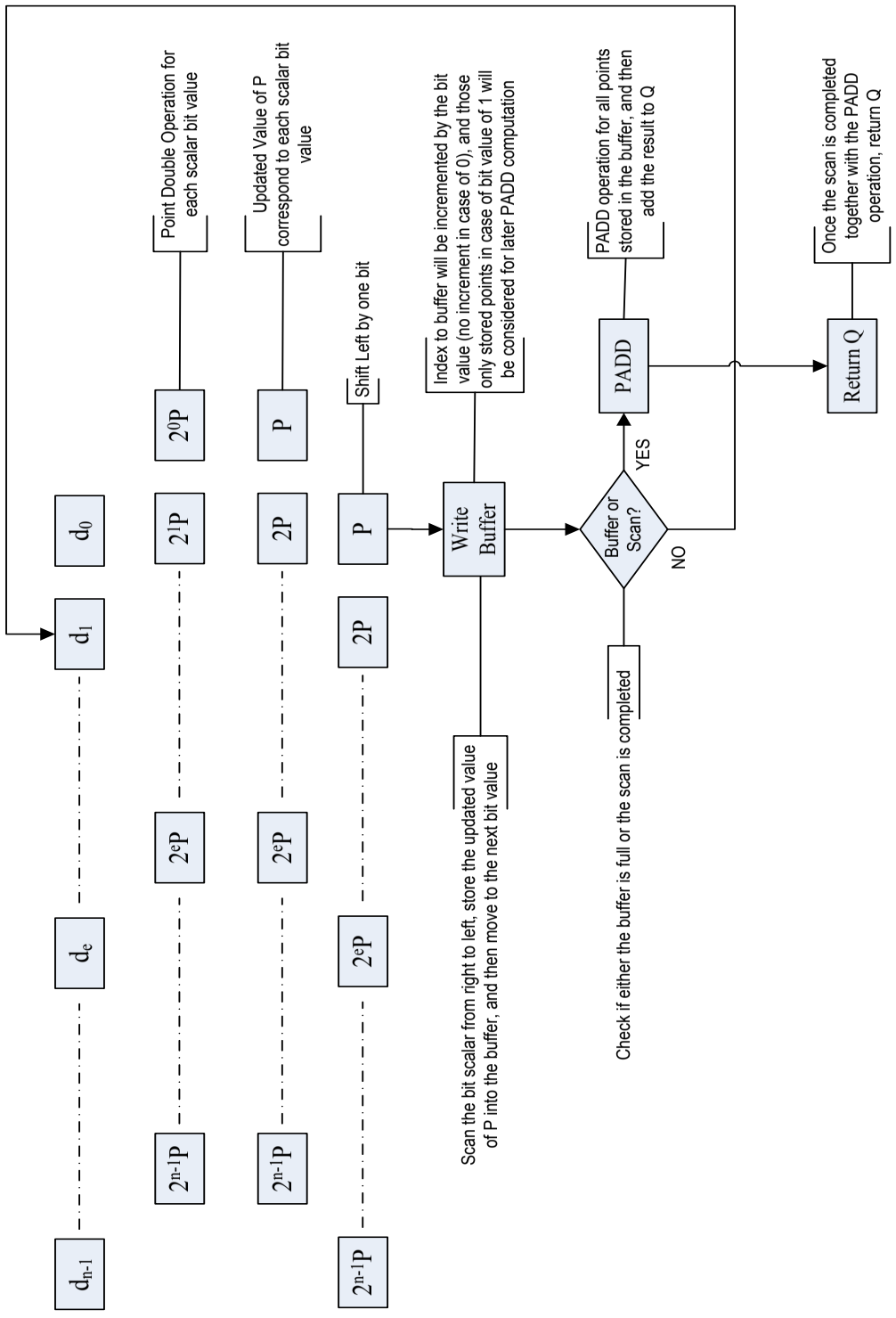

Figure 5.5: Data Flow for Buffer-based Method for Scalar Multiplication

Index to buffer is directly related to the bit scalar value; i.e., it will only increment for bit value of 1. Therefore, the buffer will only store points corresponding to bit value of 1.

3. Once the buffer is full (or the scalar scanning is completed) the PADD operation is performed on the stored points in the buffer

The scalar multiplication will be the accumulated points of the PADD operation results.

5.2.1.3 Example for the $ECC_{B\text{-}SPA}$ Cryptoprocessor

In Figure 5.6 shows an example of the Buffer-based method for ECSM. In this example, the key length is 8-bit. The key k is 186, equivalent to $(10111010)_2$ in binary, and the buffer capacity is 3. Points are stored, twice in the buffer as follow:

1. In the first round by the points (2P, 8P, 16P) because the buffer became full, and then
2. In the second round by the points (32P, 128P) since the scalar scan is completed.

These points correspond to the scalar bit positions (1,3,4) in the first round, and positions (5,7) in the second round, where in each round a PADD operation is performed on the points, and the final value of PADD is stored in Q as the result of the scalar multiplication 186P = 26P + 160P.

5.2.1.4 Performance Analysis for the $ECC_{B\text{-}SPA}$ Cryptoprocessor

In the proposed Buffer-based method (Algorithm 5.3) for ECSM, PADD is performed in later stage and only if the bit value $k_i = 1$, while PDBL is always performed regardless of the bit value k_i. In addition, this proposed method is derived from the binary method (See Algorithm 2.2); therefore, the performance required by the proposed Buffer-based method is m PDBL and an average of m/2 PADD operations, which is equivalent to the performance of the binary method, and it has a better performance in compared to the double-and-add always method. In addition, Buffer-based method requires no extra dummy computation. This can be improved to m PDBL and an average of m/3 PADD when NAF encoding is used.

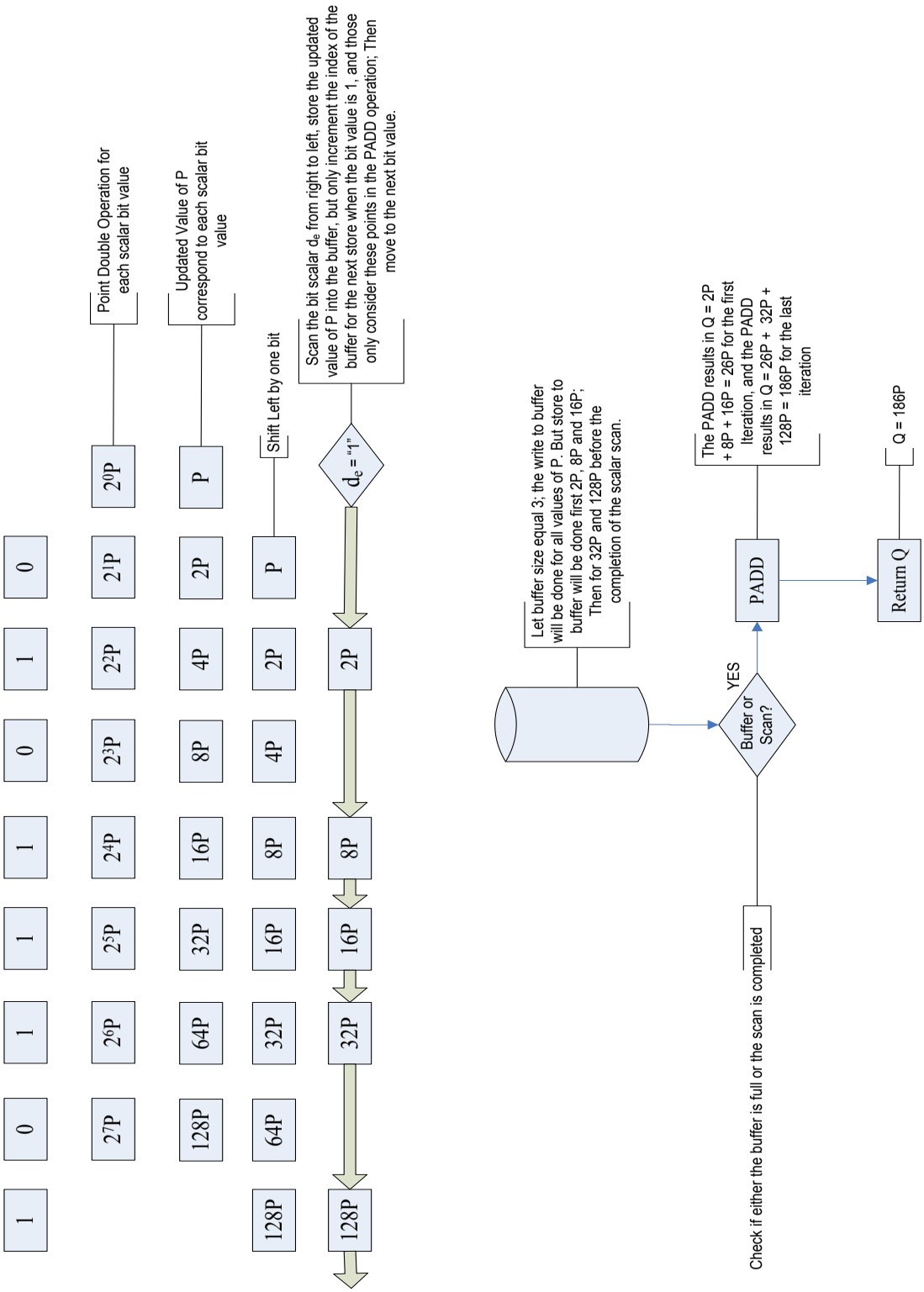

Figure 5.6: Example for Buffer-based Method for ECSM

5.2.1.5 Security Analysis for the $ECC_{B\text{-}SPA}$ Cryptoprocessor

In the proposed Buffer-based method (Algorithm 5.3) for ECSM, the PADD operation is delayed by storing points in a buffer, and a PDBL with "write to buffer" is performed for every bit value, and thus the relation between the scalar bit value and point operation is removed. Therefore, this proposed method is robust against SPA attacks since the point operations (PDBL and PADD) are independent of the bit scalar value. For instance, the power trace for the example in Section 5.2.1.3 can be simulated below (Figure 5.7).

$(K)_2$	1	0	1	1	1				0	1	0		
K (in reverse order)	0	1	0	1	1				1	0	1		
Power Trace	D	D^*	D	D^*	D^*	A	A	A	D^*	D	D^*	A	A

Figure 5.7: Example for Power Trace for the Buffer-based method

where: D stands for PDBL & 'write to buffer' operation, D^* stands for PDBL & 'write to buffer with increment of index to buffer' , and A stands for PADD operation. The key length is 8-bits, k is 186, equivalent to $(10111010)_2$ in binary, and the buffer capacity is 3.

Moreover, the security of this proposed method depends on the provided depth of confusion which is directly proportional to the size of the buffer, i.e., the smaller the buffer is, the easier to guess the number of processed bit "1" during the sequence of PDBL operations, and it will be harder when the buffer is larger. A moderate buffer size should be $log_2(m)$ to reach a confusion depth that secures ECSM against SPA attacks.

5.2.2 The $ECC_{SB\text{-}SPA}$ Cryptoprocessor

This subsection introduces the $ECC_{SB\text{-}SPA}$ cryptoprocessor with an ECSM method, called Split Buffer-based Method, that is based on splitting the scalar into two equal length partitions and

delay the PADD operation using three different buffers (See Algorithm 5.4). In addition, the scalar splitting technique is derived from the ECSM method based on propositional logic operations in [97]. In [97], their ECSM is based on partitioning the bit string of the scalar in half and extracting the common substring from the two parts based on propositional logic operations (See Algorithm 4.3).

5.2.2.1 Background Information on the $ECC_{SB\text{-}SPA}$ Cryptoprocessor

According to [97], scalar multiplication kP can be computed as:

$$kP = (K_2||K_1).P$$

$$= 2^{\frac{m}{2}}.(K_2 . P) + (K_1 . P)$$

$$= 2^{\frac{m}{2}}.(K_{XOR\_2} . P + K_{1\_AND\_2} . P) + (K_{XOR\_1} . P + K_{1\_AND\_2} . P) \qquad \text{(Equation 5.10)}$$

where $K_1 = K_{XOR\_1} + K_{1\_AND\_2}$ *and* $K_2 = K_{XOR\_2} + K_{1\_AND\_2}$

Also, K_{XOR_1} and K_{XOR_2} are K_1 and K_2 exclusive-or the common substring $K_{1\_AND\_2}$, respectively.

Splitting the scalar K into two equal partitions K_2 and K_1 is explained by example as given in Figure 5.8, where the scalar length is 16-bits such as K = $(1010\ 0101\ 1001\ 1011)_2$ = 42,395. The two partitions K_2 and K_1 are as follows:

$K_2 = (1010\ 0101)_2 = 165$ and $K_1 = (1001\ 1011)_2 = 155$

Thus, as per Equation 5.10, the scalar K can be written as

$K = 2^8 *(165) + 155 = 256*165 + 155 = 42{,}395$

Where $K_2 = (0010\ 0100) + (1000\ 0001) = 36 + 129 = 165$, and

$K_1 = (0001\ 1010) + (1000\ 0001) = 26 + 129 = 155$

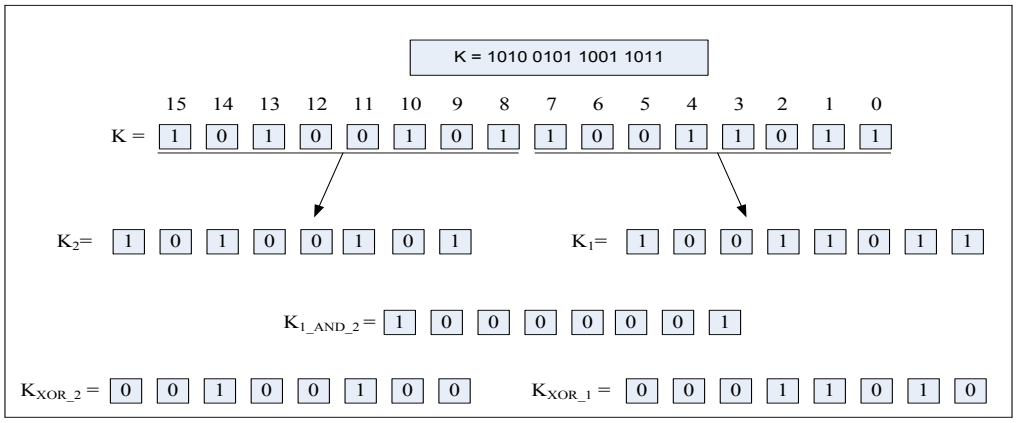

Figure 5.8: Example of Scalar Splitting with equal partitions

5.2.2.2 Description of the $ECC_{SB\text{-}SPA}$ Cryptoprocessor

Similar to the technique used in the Buffer-based Method (as described in Section 5.2.1.2), buffering technique is also used by the proposed Split Buffer-based Method for ECSM, but with splitting the bit string of the scalar k into two equal length partitions. The data flow of the Split Buffer-based Method is depicted in Step 3 and 4 will be repeated until the scan is completed, and then the PADD operation is performed on the remaining points of the buffers. The scalar multiplication will be computed as given in Equation 5.10.

In addition, the algorithm for this method is illustrated in Algorithm 5.4. Three buffers B_1, B_2, B_3 are defined with index i_1, i_2, i_3 respectively. And the different values of bits (k_2^e, k_1^e) for partitions (K_2, K_1) are defined by $n = 2k_2^e + k_1^e$, where $n \in [0,3]$ and K_{XOR_1}, K_{XOR_2}, and $K_{1\_AND\_2}$ are associated to n = 1, 2, and 3 respectively. In Step 2 of Algorithm 5.4, the bit pairs of each partition are scanned from right to left at the same bit position e, then in every iteration, 1- the new value of P is stored in B_1 (if $k_2^e = 0$, and $k_1^e = 1$), or B_2 (if $k_2^e = 1$, and $k_1^e = 0$), or B_3 (if $k_2^e = 1$, and $k_1^e = 1$), 2- the value of P is doubled; Once one of the buffers (B_1 or B_2 or B_3) is full, the PADD operation is performed on stored values of P in the buffer, and then the index of this

buffer is reset. When the scanning of the bit pairs ($k_2^{n/2}$ and $k_1^{n/2}$) is completed, the PADD operation is performed on the remaining points in the buffers.

5.2.2.3 Example for the $ECC_{SB\text{-}SPA}$ Cryptoprocessor

Figure 5.10 shows an example of the Split Buffer-based Method. In this example, the key length is 16-bit. The K = $(1010\ 0101\ 1001\ 1011)_2$ = 42,395 and the buffer capacity is 3. As mentioned in the example of scalar splitting with equal partitions (Figure 5.8), $n = 2k_2^e + k_1^e$, where $n \in [0,3]$ and K_{XOR_1}, K_{XOR_2}, and $K_{1\_AND\_2}$ are associated to n = 1, 2, and 3 respectively. Points are stored in the corresponding buffers according to the value of n, i.e., in the buffers (B_1, B_2, B_3) for n = 1, 2, 3 respectively. The points stored in B_1 are (2P, 8P, 16P), B_2 are (4P, 32P), and in B_3 are (P, 128P). Since the buffer capacity is 3, and the scalar scanning is completed in the first iteration, the buffers are only filled once.

This method uses a four-step approach:

1- The bit string of the scalar k is split into two equal length partitions

$k_2 = (k_2^{m/2} \ldots k_2^e \ \ldots k_2^1)_2$, $K_1 = (k_1^{m/2} \ldots k_1^e \ \ldots k_1^1)_2$, then

2- The partitions are scanned from right to left, and then a PDBL operation is perform for each bit pairs (k_2^e, k_1^e) of the partitions k_2 and k_1, and

3- The updated value of P (result of PDBL operation) is stored to its relevant buffer that is related to the bit pair value: B_1 for ($k_2^e = 0$, and $k_1^e = 1$), or B_2 for ($k_2^e = 1$, and $k_1^e = 0$), or B_3 for ($k_2^e = 1$, and $k_1^e = 1$).

4- The PADD operation is delayed until any of the buffers becomes full, and then it is performed on the stored points in that buffer (full). The result point of PADD operation on B_1, B_2, and B_3 represents the values of K_{XOR_1}, K_{XOR_2}, and $K_{1\_AND\_2}$ respectively.

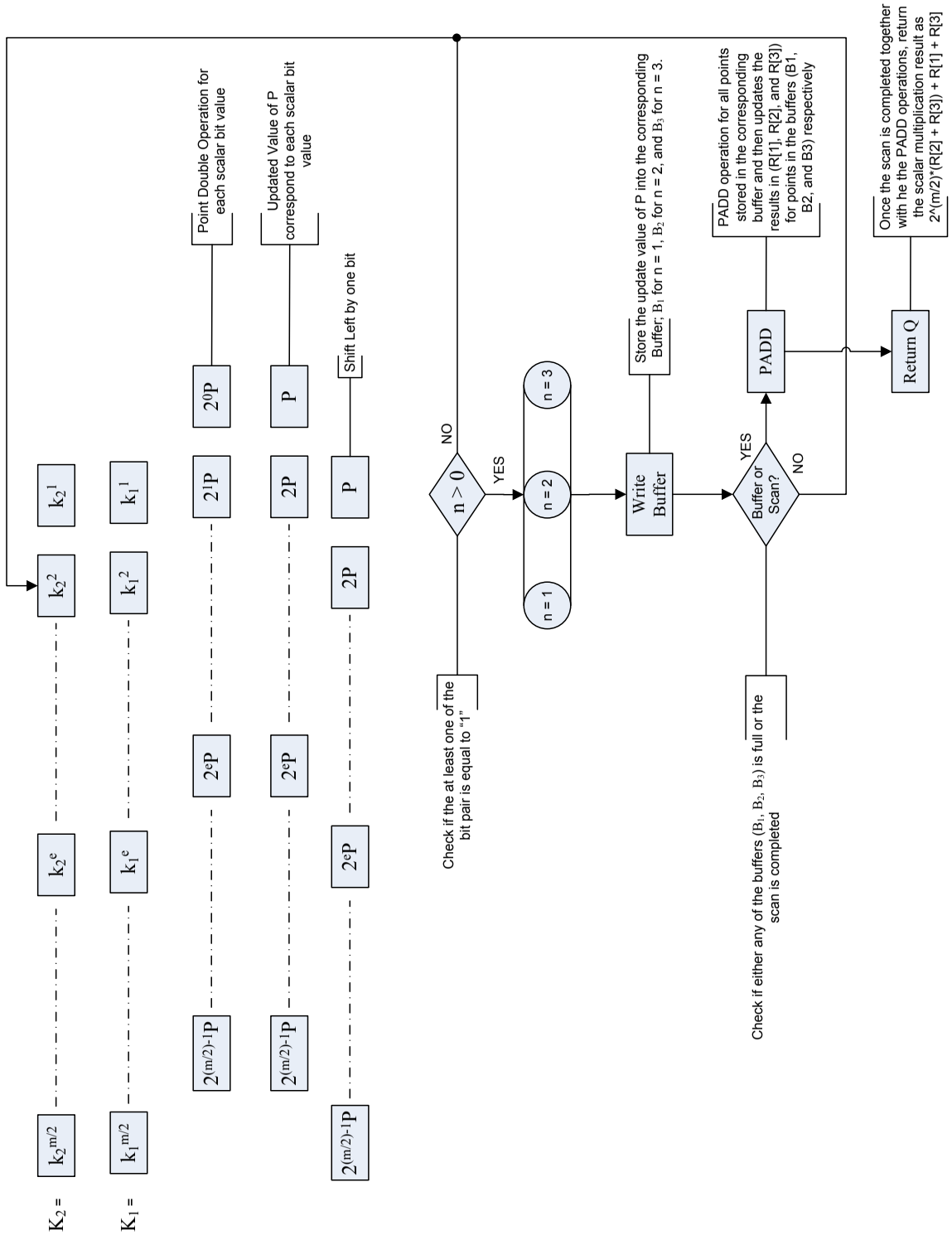

Figure 5.9: Data Flow for Split Buffer-based method for ECSM

Step 3 and 4 will be repeated until the scan is completed, and then the PADD operation is performed on the remaining points of the buffers. The scalar multiplication will be computed as given in Equation 5.10.

In addition, the algorithm for this method is illustrated in Algorithm 5.4. Three buffers B_1, B_2, B_3 are defined with index i_1, i_2, i_3 respectively. And the different values of bits (k_2^e, k_1^e) for partitions (K_2, K_1) are defined by $n = 2k_2^e + k_1^e$, where $n \in [0,3]$ and K_{XOR_1}, K_{XOR_2}, and $K_{1\_AND\_2}$ are associated to n = 1, 2, and 3 respectively. In Step 2 of Algorithm 5.4, the bit pairs of each partition are scanned from right to left at the same bit position e, then in every iteration, 1- the new value of P is stored in B_1 (if $k_2^e = 0$, and $k_1^e = 1$), or B_2 (if $k_2^e = 1$, and $k_1^e = 0$), or B_3 (if $k_2^e = 1$, and $k_1^e = 1$), 2- the value of P is doubled; Once one of the buffers (B_1 or B_2 or B_3) is full, the PADD operation is performed on stored values of P in the buffer, and then the index of this buffer is reset. When the scanning of the bit pairs ($k_2^{n/2}$ and $k_1^{n/2}$) is completed, the PADD operation is performed on the remaining points in the buffers.

5.2.2.4 Example for the $ECC_{SB\text{-}SPA}$ Cryptoprocessor

Figure 5.10 shows an example of the Split Buffer-based Method. In this example, the key length is 16-bit. The $K = (1010\ 0101\ 1001\ 1011)_2 = 42{,}395$ and the buffer capacity is 3. As mentioned in the example of scalar splitting with equal partitions (Figure 5.8), $n = 2k_2^e + k_1^e$, where $n \in [0,3]$ and K_{XOR_1}, K_{XOR_2}, and $K_{1\_AND\_2}$ are associated to n = 1, 2, and 3 respectively. Points are stored in the corresponding buffers according to the value of n, i.e., in the buffers (B_1, B_2, B_3) for n = 1, 2, 3 respectively. The points stored in B_1 are (2P, 8P, 16P), B_2 are (4P, 32P), and in B_3 are (P, 128P). Since the buffer capacity is 3, and the scalar scanning is completed in the first iteration, the buffers are only filled once.

Algorithm 5.4 Split Buffer-based ECSM Method
Inputs: *P: Base Point, k: Secret key,* $k_2 = (k_2^{m/2} \ldots k_2^e \ \ldots k_2^1)_2$, $k_1 = (k_1^{m/2} \ldots k_1^e \ \ldots k_1^1)_2$, *r is capacity limit of buffer***.** **Outputs:** *kP.*
1: R[1] ← R[2] ← R[3] ← O, $t_1 \leftarrow t_2 \leftarrow t_3 \leftarrow 1$ /* set buffers' indexes $t_1 \leftarrow t_2 \leftarrow t_3$ to 1 */
2: for e = 1 to m/2 do
2.1: n ← $2k_2^e + k_1^e$
2.2: if n > 0, then
2.2.1: $B_n[t_n]$ ←P /* scan k, store points on corresponding buffer only for bit value of 1 */
2.2.2: if t_n ← r Then
2.2.2.1: for s = 1 to t_n do
2.2.2.1.1: R[n] ← R[n] + B_n[s]
2.2.2.2: t_n ← 1 /* reset buffer index to 1 */
2.2.3: else $t_n \leftarrow t_n + 1$
2.3: P ← 2P
2.4: if e ← m/2, Then
2.4.1: for n= 1 to 3 do
2.4.1.1: if t_n >1 Then
2.4.1.1.1: for s = 1 to t_n – 1 do
2.4.1.1.1.1: R[n] ← R[n] + B_n[s]
3: R[1] ← R[1] + R[3]
4: R[2] ← R[2] + R[3]
5: for e = 1 to m/2 do
5.1: R[2] ← 2R[2]
6: R[1] ← R[2] + R[1]
Return R[1].

PADD operation is performed on the points in the buffers, and the final value of Q as the result of the scalar multiplication is: 2^8*(36P + 129P) + (26P + 129P) = 256*(165P) + (155P) = 42,395P.

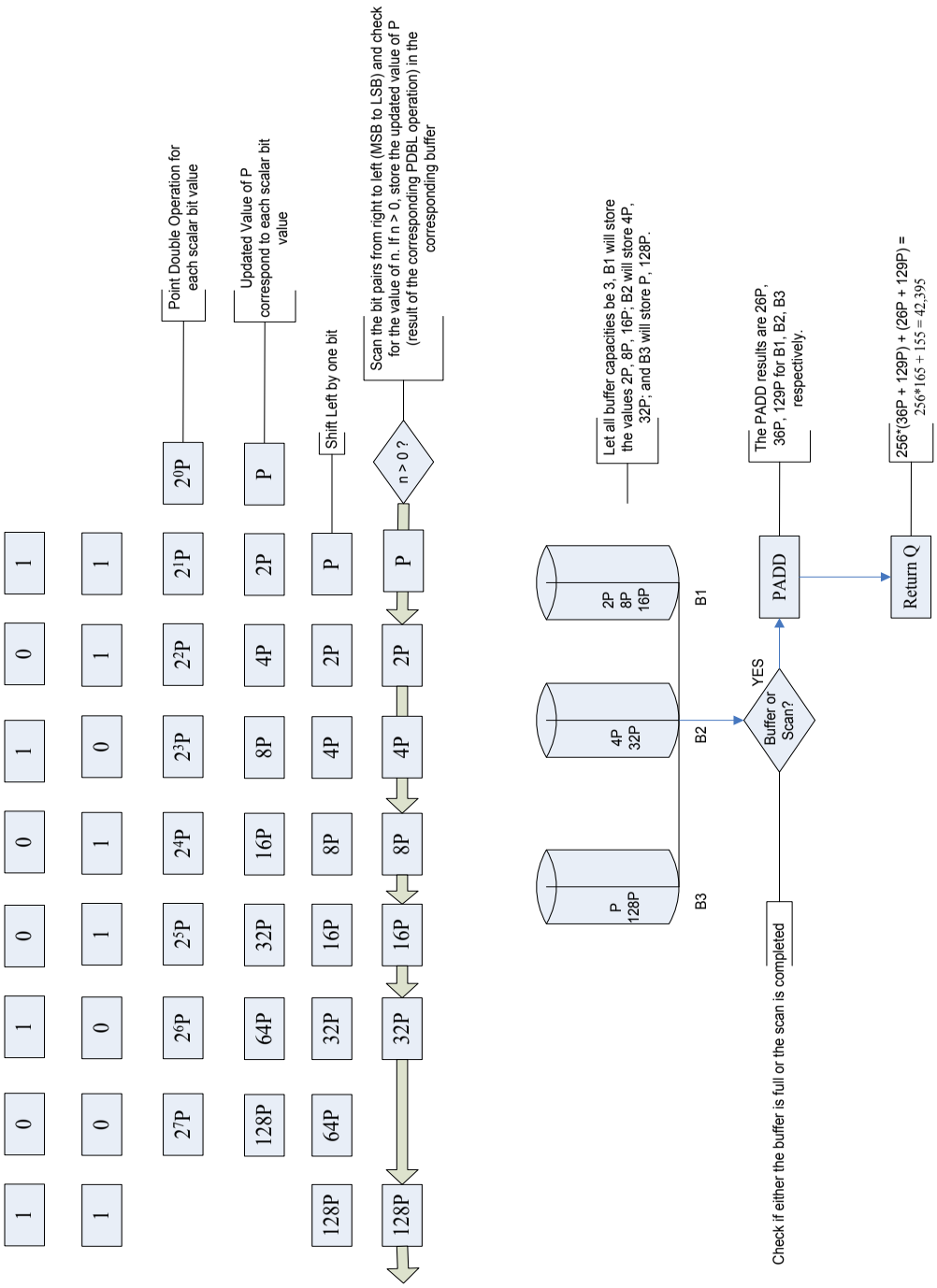

Figure 5.10: Example for Data Flow for Split Buffer-based method

5.2.2.5 Performance Analysis for the $ECC_{SB\text{-}SPA}$ Cryptoprocessor

The proposed Split Buffer-based method (Algorithm 5.4) for ECSM is derived from the binary method (See Algorithm 2.2), and thus PADD is performed in later stage and only if the bit pair value (k_2^e , k_1^e) is NOT (0,0), while PDBL is always performed regardless of the bit pair value. This method requires *m* PDBL, as proven in both Steps 2.3 and 5.1 of Algorithm 5.4, and on average ([m/2] – [m/8] = [3m/8]) PADD, as shown on Step 2.2.2.1.1 of the Algorithm 5.4, where PADD is performed for [m/2] iterations for only n > 0, i.e. PADD operation is not performed for the bit pairs (k_2^e, k_1^e) = (0 ,0), where its occurrence is with probability of [1/4], since as per Equation 5.11 the probability of (0, 0) = probability (0) * probability (0) = [1/2] * [1/2] = [1/4] . Additional number of PADD operations are performed at the end of algorithm, and these are negligible in comparison to m.

$$\text{Probability}\,(A\ and\ B) = \text{Probability}\,(A) * \text{Probability}\,(B) \qquad \text{(Equation 5.11)}$$

Therefore and to the best of our knowledge, this method outperforms all previously proposed methods in literature, including the binary method (See Algorithm 2.2) by reducing the PADD by m/8 and it only requires m PDBL and on average [3m/8] PADD. This performance improves to m PDBL and an average of [m/4] PADD when NAF encoding is used. In addition, the Split Buffer-based method requires no extra dummy computation.

5.2.2.6 Security Analysis for the $ECC_{SB\text{-}SPA}$ Cryptoprocessor

In the proposed Split Buffer-based method (Algorithm 5.4) for ECSM, the security against SPA attacks is achieved in two levels of confusion:

1. The first level is realized by inspecting bit pairs instead of a single bit of the scalar, and thus increase possible values to 4 (00, 01, 10, 11) instead of 2 (0, 1); and

2. The second level is achieved by delaying the PADD operation using buffers for interim points storage. Therefore, the relation between the scalar bit value and point operation is removed by delaying the PADD operation.

For instance, the power trace for the example in Section 5.2.2.3 can be simulated below (Figure 5.11)

$(K)_2$	1	0	1	0	0				1	0	1				
	1	0	0	1	1				0	1	1				
K (in reverse order)	1	0	1	0	0				1	0	1				
	1	1	0	1	1				0	0	1				
	D^{***}	D^{*}	D^{**}	D^{*}	D^{*}	A^{*}	A^{*}	A^{*}	D^{**}	D	D^{***}	A^{**}	A^{**}	A^{***}	A^{***}

Figure 5.11: Example for Power Trace for the Split Buffer-based method

where: D stands for PDBL operation, and D^{*}, D^{**}, D^{***} stands for PDBL operation with store in buffers (B_1, B_2, B_3) respectively; A stands for PADD operation and A^{*}, A^{**}, A^{***} stands for PADD operation on the points stored in buffers (B_1, B_2, B_3) respectively. The key length is 16-bit, k is 42,395, equivalent to $(1010\ 0101\ 1001\ 1011)_2$ in binary, and the buffer capacity is 3. Moreover, the security of this proposed method depends on the provided depth of confusion, which is directly proportional to the size of the buffer, i.e., the smaller the buffer, the easier to guess the number of processed bit pairs (01, 10, 11), and it will be harder when the buffer is larger. A moderate buffer size should be $log_2(m)$ to reach a confusion depth that secures ECSM against SPA attacks. This method requires no extra dummy computations to secure ECSM against SPA attacks.

5.3 Proposed Architecture for ECC Secure against DPA

In this section, two proposed architectures for elliptic curve cryptoprocessors are presented; these cryptoprocessors provide resistance against DPA attacks. The first cryptoprocessor is Randomized Buffer-based, called $ECC_{RB\text{-}DPA}$, and it uses an ECSM method that is based on delaying the PADD operation using a buffer and applying randomization concept; whereas the second cryptoprocessor is Randomized Split Buffer-based, called $ECC_{RSB\text{-}DPA}$, and it uses ECSM method that is based on splitting the scalar into two equal length partitions and delay the PADD operation using three different buffers and applying randomization concept.

5.3.1 The $ECC_{RB\text{-}DPA}$ Cryptoprocessor

This subsection introduces the Randomized Buffer-based $ECC_{RB\text{-}DPA}$ cryptoprocessor, it uses an ECSM method which is derived from the binary method, and is based on delaying the PADD operation by using randomized technique for points storing (in one buffer) and processing (See Algorithm 5.5).

5.3.1.1 Background Information on the $ECC_{RB\text{-}SPA}$ Cryptoprocessor

In order to give background information on the $ECC_{RB\text{-}SPA}$ cryptoprocessor, it is significant to recall that in the right-to-left version of the binary method (See Algorithm 2.2) of the scalar multiplication, PADD is only performed if the bit value $k_i = 1$, while PDBL is always performed regardless of the bit value.

Moreover, in Equation 5.9, the scalar is divided into a number s of partitions, we call it "scalar partitioning on 1's", where each partition is associated with a computed point ($2^iP \mid k_i = 1$) to keep

its significance [116]. The partition is defined as the bit string of length j and only contains one bit "1".

Algorithm 5.5 Randomized Buffer-based ECSM Method
Inputs: *P: Base Point, k: Secret key, r is capacity limit of buffer* **Outputs:** *kP.*
1: R[0] ← O, t ← 1 /* set buffer index t to 1 */
2: for i = 0 to m-1 do
2.1: r' ← RNG (0,r); /* Generate random number r', where 0 < r' < buffer capacity r
2.2: B[t] ← P /* scan k, store points on buffer */
2.3: P ← 2P
2.4: if (t ← r') then
2.4.1: j ← RNG (≤ r'); random number generator for a number less than or equal to i
2.4.2: for s = j to t do
2.4.2.1: R[0] ← R[0] + B[s]
2.4.3: t ← j; Reset buffer (to avoid calculating resident points from previous iteration)
2.5: else t ← t + k_i /* increment t if the bit value of k is 1 */
2.6: if i ← m-1 then
2.6.1: for s = 1 to t – 1 do
2.6.1.1: R[0] ← R[0] + B[s]
Return R[0]

5.3.1.2 Description of the $ECC_{RB\text{-}SPA}$ Cryptoprocessor

To protect against power analysis attacks, the point operations (PADD and PDBL) of the scalar multiplication must be independent of the scalar bit value k_i. In addition, since each key partition is associated with a computed point to keep the significance of each key partition, and the points resulting from processing these key partitions are accumulated to produce the scalar product kP, PADD operation can be performed at a delayed time in a randomized mode, and not necessarily

to be done at the corresponding key bit position. Thus, the Randomized Buffer-based method for ECSM is proposed and its dataflow is depicted in Figure 5.12.

In in Figure 5.12, the scalar is scanned from right to left and for every scalar bit value:

1. Perform a PDBL operation.

   PDBL operation keeps the significance of the point value at the scalar bit position of the scalar.

2. Write to buffer the updated value of P (result of PDBL operation)

   The buffer capacity is randomized (greater than zero, and less or equal to the initial random capacity). Index to buffer is directly related to the bit scalar value; i.e., it will only increment for bit value of 1. Therefore, the buffer will only store points corresponding to bit value of 1.

3. Once the buffer is full (i.e. the number of stored points is equal to the capacity of the buffer after applying randomization), the PADD operation is performed on a random number of points stored in the buffer.

4. When the scalar scanning is completed, the PADD operation is performed on the remaining points in the buffer.

The scalar multiplication will be the accumulated points of the PADD operation results.

5.3.1.3 Example for the $ECC_{RB\text{-}SPA}$ Cryptoprocessor

In Figure 5.13 shows an example of the Randomized Buffer-based method for ECSM. In the example, the key length is 8-bit. The key k = 186 = $(10111010)_2$ and the initial buffer capacity is 4. Points are stored to buffer, and PADD operation is performed on points from buffer in three rounds (iterations) as follow:

1) In the first round, three points (2P, 8P, 16P) are stored since the buffer becomes full (i.e. The number of stored points is equal to the capacity of the buffer – randomized as 3), but PADD

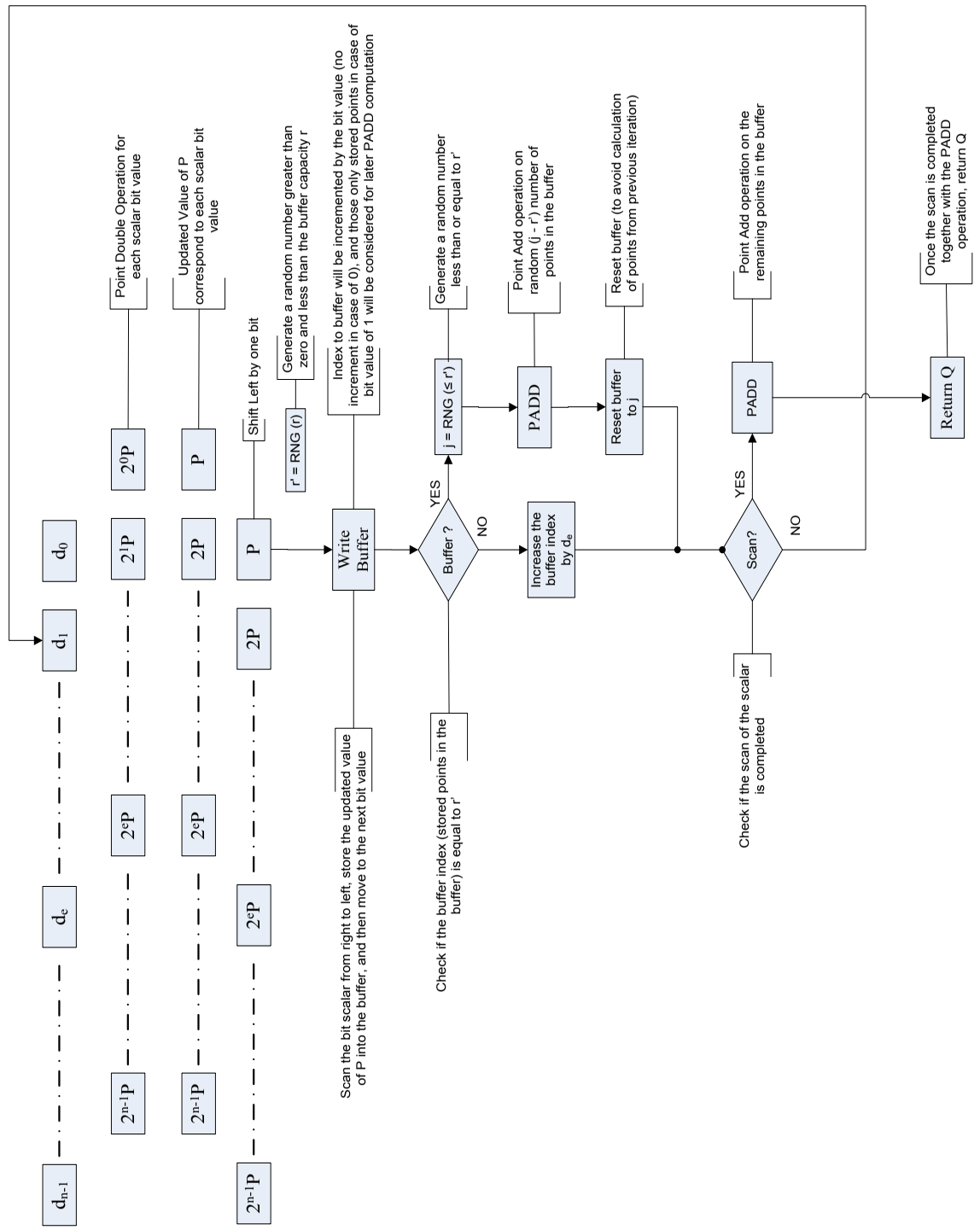

Figure 5.12: Data Flow for Randomized Buffer-based method for ECSM

operation is performed on two points only (8P, 16P) because the randomly generated number j is 2 and the processed points are the second and the third in the buffer.

2) In the second round, only one point (32P) is stored since the buffer becomes full ((i.e. The number of stored points is equal to the capacity of the buffer – randomized as 2), and PADD operation is performed on one point only (32P) because the randomly generated number j is 2 and the processed point is the second in the buffer.
3) In the third round, only one point (128P) is stored since the scalar scanning is completed, and PADD operation is performed on two points only (2P, 128P) because these are the remaining points in the buffer.
4) The result of the scalar multiplication is the final value of PADD operation on the stored in the buffer as per the below sequence:

   8P + 16P + 32P + 2P + 128P = 186P.

5.3.1.4 Performance Analysis for the $ECC_{RB\text{-}SPA}$ Cryptoprocessor

In the proposed Randomized Buffer-based method (Algorithm 5.5) for ECSM, PADD is performed in later stage and only if the bit value $k_i = 1$, while PDBL is always performed regardless of the bit value k_i. In addition, this proposed method is derived from the binary method (See Algorithm 2.2); therefore, the performance required by the proposed Buffer-based method is m PDBL and an average of m/2 PADD operations, which is equivalent to the performance of the binary method, and it has a better performance in compared to the double-and-add always method. In addition, Buffer-based method requires no extra dummy computation. This can be improved to m PDBL and an average of m/3 PADD when NAF encoding is used.

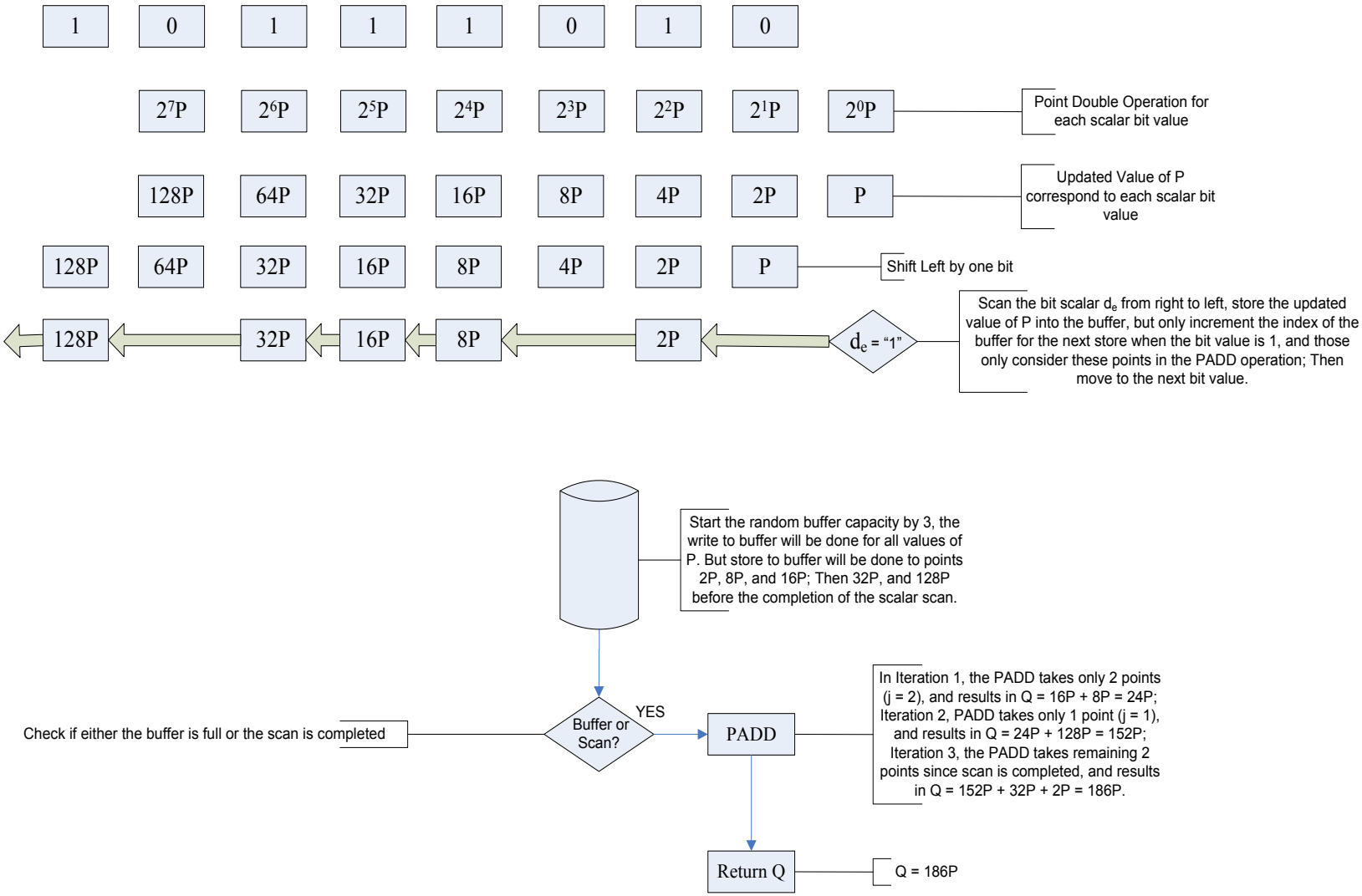

Figure 5.13: Example for Randomized Buffer-based method for ECSM

5.3.1.5 Security Analysis for the $ECC_{RB\text{-}SPA}$ Cryptoprocessor

In the proposed Randomized Buffer-based method (Algorithm 5.5) for ECSM, the PADD operation is delayed by storing points in a buffer, and a PDBL with "write to buffer" is performed for every bit value, and thus the relation between the scalar bit value and point operation is removed. In addition, randomization technique is used in number points stored in the buffer, and the number of points processed for PADD in the buffer. Therefore, this proposed method is robust against DPA attacks. For instance, the power trace for the example in Section 5.3.1.3 can be simulated below (Figure 5.14).

K	1	0	1	1	1			0	1		0		
K (in reverse order)	0	1	0	1	1			1	0		1		
Power Trace	D	D^*	D	D^*	D^*	A	A	D^*	D	A	D^*	A	A

Figure 5.14: Example for Power Trace for the Randomized Buffer-based method

where: D stands for PDBL & 'write to buffer' operation, D* stands for PDBL & 'write to buffer with increment of index to buffer', and A stands for PADD operation. The key length is 8-bits, k is 186, equivalent to $(10111010)_2$ in binary and the buffer capacity is 4.

Moreover, the security of the Randomized Buffer-based method depends on its depth of confusion which is directly proportional to:

1) The deployment of randomization technique in both the buffer capacity (being dynamic) and the processed points for PADD operation; and

2) The size of the buffer, i.e., the smaller the buffer, the easier to guess the number of processed bit "1" during the sequence of PDBL operations, and it will be harder when the buffer is larger. A moderate buffer size should be $log_2(m)$ to reach a confusion depth that secures ECSM against DPA attacks.

5.3.2 The $ECC_{RSB\text{-}DPA}$ Cryptoprocessor

This subsection introduces the $ECC_{RSB\text{-}DPA}$ cryptoprocessor with a ECSM method, called Randomized Split Buffer-based Method, that is based on splitting the scalar into two equal length partitions and delay the PADD operation using randomized technique for points storing (in three different buffers) and point processing (See Algorithm 5.6). In addition, the scalar splitting technique is derived from the ECSM method based on propositional logic operations in [97]. In [97], their ECSM method is based on partitioning the bit string of the scalar in half and extracting the common substring from the two parts based on propositional logic operations (See Algorithm 4.3).

5.3.2.1 Background Information on the $ECC_{RSB\text{-}SPA}$ Cryptoprocessor

As in**Error! Reference source not found.**, the scalar multiplication is split into two partitions (K_1 *and* K_2*,)*, where the common substring $K_{1\_AND\_2}$ is only computed once, and used in the two partitions such that

$K_1 = K_{XOR\_1} + K_{1\_AND\_2}$ *and* $K_2 = K_{XOR\_2} + K_{1\_AND\_2.}$

Splitting the scalar K into two equal partitions K_2 and K_1 is explained by example as given in Figure 5.8, and elaborated in Section 5.2.2.1.

5.3.2.2 Description of the $ECC_{RSB\text{-}SPA}$ Cryptoprocessor

Similar to the technique used in the Split Buffer-based method for ECSM (as described in Section 5.2.2.2), but the Randomized Split Buffer-based method additionally uses randomized technique for points storing (in three different buffers) and point processing (See Algorithm 5.6).

The data flow of the Randomized Split Buffer-based Method is depicted in Figure 5.15. This method uses a four-step approach:

1- The bit string of the scalar *k* is split into two equal length partitions

$K_2 = (k_2^{n/2} \ldots k_2^e \ \ldots k_2^1)_2$, and $K_1 = (k_1^{n/2} \ldots k_1^e \ \ldots k_1^1)_2$, then

Algorithm 5.6 Randomized Split Buffer-based ECSM Method
Inputs: *P: Base Point, k: Secret key,* $k_2 = (k_2^{m/2} \ldots k_2^e \ \ldots k_2^1)_2$, $k_1 = (k_1^{m/2} \ldots k_1^e \ \ldots k_1^1)_2$, *r is capacity limit of buffer***.** **Outputs:** *kP.*
1: R[1] ← R[2] ← R[3] ← O, $t_1 \leftarrow t_2 \leftarrow t_3 \leftarrow 1$ /* set buffers' indexes $t_1 \leftarrow t_2 \leftarrow t_3$ to 1 */
2: for e = 1 to m/2 do
2.1: n ← $2k_2^e + k_1^e$
2.2: if n > 0, then
2.2.1: $B_n[t_n] \leftarrow P$ /* scan k, store points on corresponding buffer only for bit value of 1 */
2.2.2: r ← RNG (<Capacity of B_n); Generate a random number less than the capacity of buffer B_n
2.2.3: if $t_n \leftarrow r$ Then
2.2.3.1: $j_n \leftarrow$ RNG ($< i_n$); random number generator for a number less than i_n
2.2.3.2: for s = 1 to t_n do
2.2.3.2.1: $R[n] \leftarrow R[n] + B_n[s]$
2.2.3.3: $t_n \leftarrow j_n$ /* reset buffer index to j_n */
2.2.3.4: else $t_n \leftarrow t_n + 1$
2.3: P ← 2P
2.4: if e ← m/2, Then
2.4.1: for n= 1 to 3 do
2.4.1.1: if t_n >1 Then
2.4.1.1.1: for s = 1 to $t_n - 1$ do
2.4.1.1.1.1: $R[n] \leftarrow R[n] + B_n[s]$
3: R[1] ← R[1] + R[3]
4: R[2] ← R[2] + R[3]
5: for e = 1 to m/2 do
5.1: R[2] ← 2R[2]
6: R[1] ← R[2] + R[1]
Return R[1].

2- The partitions are scanned from right to left, and then a PDBL operation is performed for each bit pairs (k_2^e, k_1^e) for partitions K_2 and K_1, and

3- The updated value of P (result of PDBL operation) is stored to its relevant buffer that is related to the bit pair value: B_1 for ($k_2^e = 0$, and $k_1^e = 1$), or B_2 for ($k_2^e = 1$, and $k_1^e = 0$), or B_3 for ($k_2^e = 1$, and $k_1^e = 1$).

4- The PADD operation is delayed until any of the buffers becomes dynamically full (by generating a random value for the buffer capacity), and then it is performed on a random number of stored points in that buffer (full). The result point of PADD operation on B_1, B_2, and B_3 represents the values of K_{XOR_1}, K_{XOR_2}, and $K_{1\_AND\_2}$ respectively.

Step 3 and 4 will be repeated until the scan is completed, and then the PADD operation is performed on the remaining points of the buffers. The scalar multiplication will be the accumulated points of the PADD operation results as per Equation 5.10.

According to Algorithm 5.6, three buffers B_1, B_2, B_3 are defined with index i_1, i_2, i_3 respectively. Additionally, for the bit different values of (k_2^e, k_1^e) for partitions (K_2, K_1) can be defined by $n = 2k_2^e + k_1^e$, where $n \in [0,3]$ and K_{XOR_1}, K_{XOR_2}, and $K_{1\_AND\_2}$ are associated to n = 1, 2, and 3 respectively. In Step 2 of Algorithm 5.6 the bit pairs of each partition are scanned from right to left at the same bit position e, then in every iteration, 1- the new value of P is stored in B_1 (if $k_2^e = 0$, and $k_1^e = 1$), or B_2 (if $k_2^e = 1$, and $k_1^e = 0$), or B_3 (if $k_2^e = 1$, and $k_1^e = 1$), 2- the value of P is doubled; Once any of the buffer is dynamically full (by generating a random value for the buffer capacity), the PADD operation is performed on stored values of P in the relevant buffer, and the index of this buffer is reset. When the scanning of the bit pairs ($k_2^{n/2}$ and $k_1^{n/2}$) is completed, the PADD operation is performed on the remaining points in the buffers.

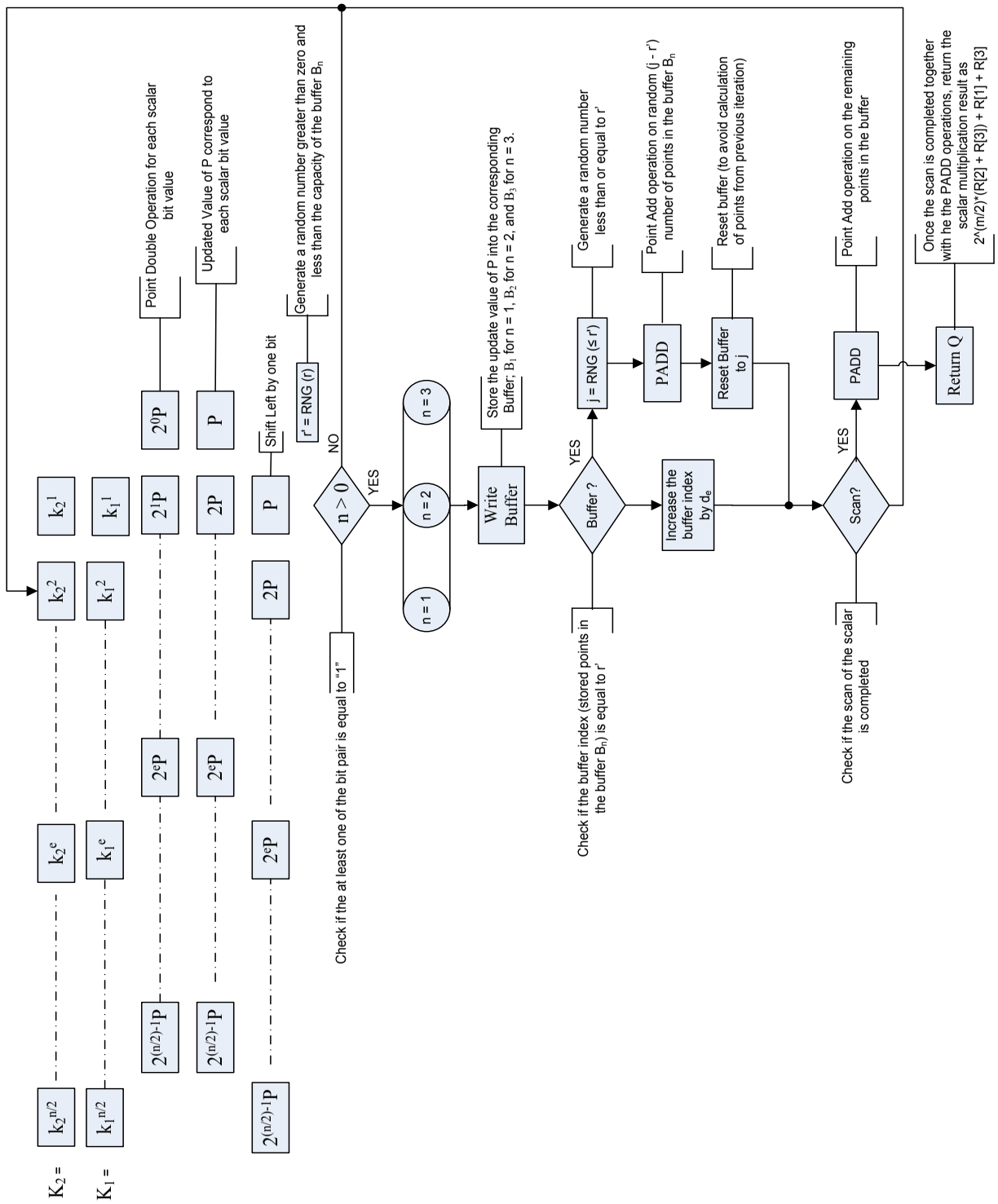

Figure 5.15: Data Flow for Randomized Split Buffer-based method for ECSM

5.3.2.3 Example for the $ECC_{RSB\text{-}SPA}$ Cryptoprocessor

Figure 5.16 shows an example of the Randomized Split Buffer-based Method. In this example, the key length is 16-bit. The K = $(1010\ 0101\ 1001\ 1011)_2$ = 42,395 and the initial buffer capacity is 4.

As mentioned in the example of scalar splitting with equal partitions (Figure 5.8), $n = 2k_2^e + k_1^e$, where n ϵ [0,3] and K_{XOR_1}, K_{XOR_2}, and $K_{1\_AND\_2}$ are associated to n = 1, 2, and 3 respectively. Points are stored in the corresponding buffers according to the value of n, i.e., in the buffers (B_1, B_2, B_3) for n = 1, 2, 3 respectively.

The randomized buffer capacity for all buffers is 2. Points are stored to buffer, and PADD operation is performed on points from buffer in three rounds (iterations) as follow:

1) In the first round, the stored points in buffers are as follow: Two points (2P, 8P) in B_1, two points (4P, 32P) in B_2, and two points (P, 128P) in B_3. In the second round, only one point (16P) is stored in B_1. All buffers are full at 2 (i.e. the number of stored points in each buffer is equal to the capacity of the buffer after applying randomization = 2), and the total number of points in this example is 7.
2) In the first round, PADD operation is performed on the points in each buffer (2P + 8P) for B_1, (4P + 32P) for B_2, and (P + 128P) for B_3, since the randomly generated number j is 1 for all buffers.
3) In the second round, one point (16P) only is processed by PADD operation for points in B_1.
4) The result of the scalar multiplication is the final value of PADD operation on the stored in the buffer as per the below sequence as per Equation 5.10:

$$2^8 * (B_2 + B_3) + (B_1 + B_3) = 2^8 * (36P + 129P) + (26P + 129P)$$
$$= 256 * (165P) + (155P)$$
$$= 42{,}395P$$

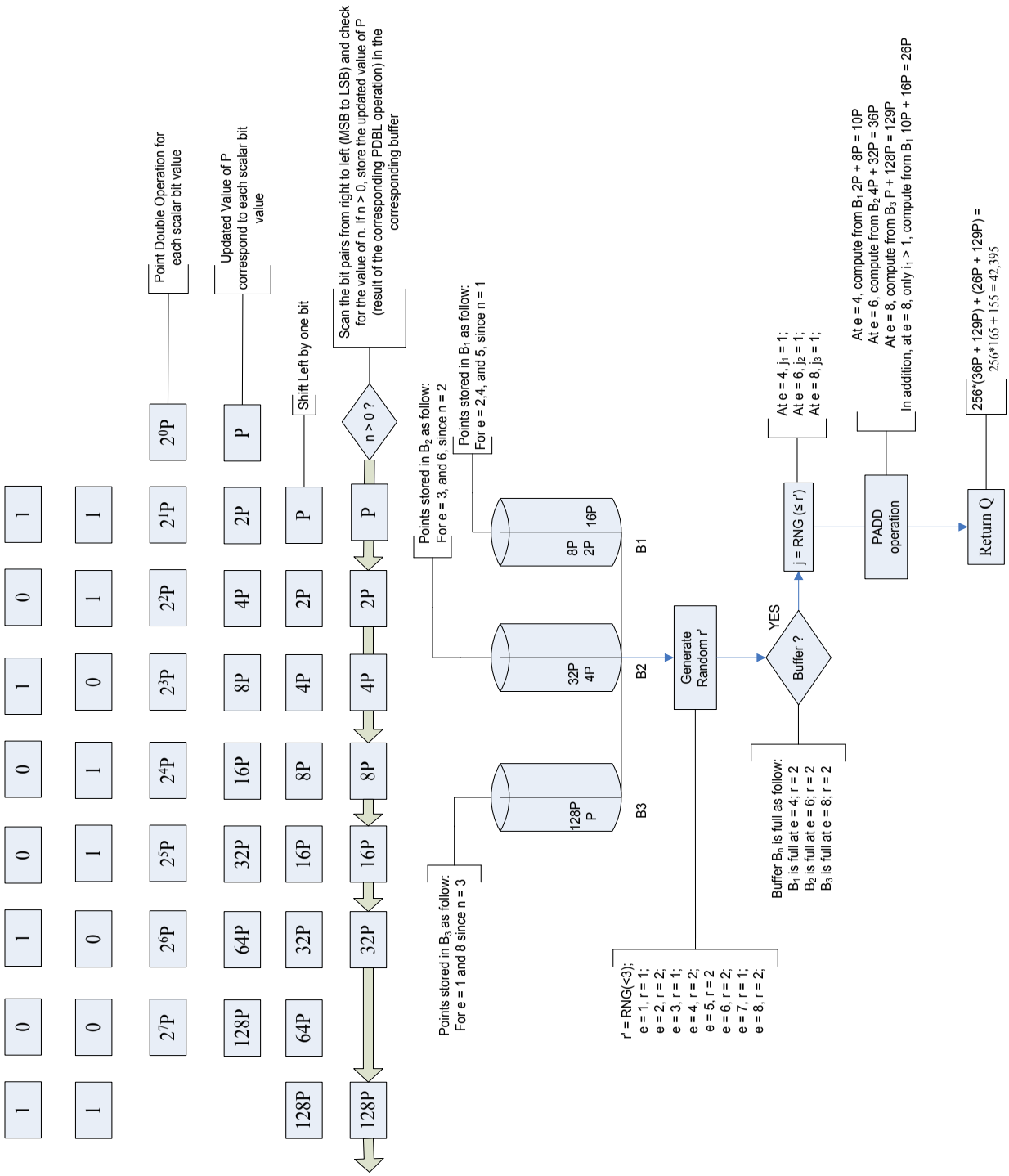

Figure 5.16: Example for Data Flow for Randomized Split Buffer-based method

5.3.2.4 Performance Analysis for the $ECC_{RSB\text{-}SPA}$ Cryptoprocessor

The proposed Randomized Split Buffer-based method (Algorithm 5.6) for ECSM is derived from the binary method (See Algorithm 2.2), and thus PADD is performed in later stage and only if the bit value $k_i = 1$, while PDBL is always performed regardless of the bit value k_i. The Randomized Split Buffer-based method requires *m* PDBL, as proven in both Steps 2.3 and 4.1 of Algorithm 5.6, and on average ([m/2] – [m/8] = [3m/8]) PADD, as shown on Step 2.2.2.1.1 of the Algorithm 5.6, where PADD is performed for [m/2] iterations for only n > 0, i.e. PADD operation is not perform for the bit pairs $(k_2^e, k_1^e) = (0, 0)$, where its occurrence is with probability of [1/4], since as per Equation 5.11 the probability of (0, 0) = probability (0) * probability (0) = [1/2] * [1/2] = [1/4]. Additional number of PADD operations are performed at the end of algorithm, and these are negligible in comparison to m.

Therefore, the Randomized Split Buffer-based method outperforms both the binary method (See Algorithm 2.2) by reducing the PADD by m/8 and it only requires m PDBL and on average [3m/8] PADD. This performance improves to m PDBL and an average of [m/4] PADD when NAF encoding is used. In addition, the Split Buffer-based method requires no extra dummy computation.

5.3.2.5 Security Analysis for the $ECCR_{SB\text{-}SPA}$ Cryptoprocessor

In the proposed Randomized Split Buffer-based method (Algorithm 5.6) for ECSM, the security against DPA attacks is achieved in two levels of confusion:

1) The first level is realized by inspecting bit pairs instead of a single bit of the scalar, and thus increase possible values to 4 (00, 01, 10, 11) instead of 2 (0, 1); and
2) The second level is achieved by delaying the PADD operation using randomization concept at both the buffers capacities levels and the number of points for PADD operation.

For instance, the power trace for the example in Section 5.2.2.3 can be simulated below (Figure 5.17).

$(K)_2$	1	0	1	0			0	1			0	1				
	1	0	0	1			1	0			1	1				
K (in reverse order)	1	0	1	0			0	1			0	1				
	1	1	0	1			1	0			0	1				
	D^{***}	D^{*}	D^{**}	D^{*}	A^{*}	A^{*}	D^{*}	D^{**}	A^{**}	A^{**}	D	D^{***}	A^{***}	A^{***}	A^{*}	

Figure 5.17: Example for Power Trace for the Randomized Split Buffer-based method

where: D stands for PDBL operation, and D^{*}, D^{**}, D^{***} stands for PDBL operation after a point store in buffers (B_1, B_2, B_3) respectively; A stands for PADD operation and A^{*}, A^{**}, A^{***} stands for PADD operation on the points stored in buffers (B_1, B_2, B_3) respectively. The key length is 16-bit, k is 42,395, equivalent to $(1010\ 0101\ 1001\ 1011)_2$ in binary, and the buffer capacity is 3. Furthermore, the depth of confusion is directly proportional to:

1) The deployment of randomization technique in both the buffer capacity (being dynamic) and the processed points for PADD operation; and
2) The size of the buffer, i.e., the smaller the buffer, the easier to guess the number of processed bit pairs (01, 10, 11), and it will be harder when the buffer is larger.

A moderate buffer size should be $log_2(m)$ to reach a confusion depth that secures ECSM against DPA attacks.

5.4 Summary

In this chapter by using the randomization concept together with the buffering and scalar splitting techniques, we propose four elliptic curve cryptoprocessor architectures for curves defined over $GF(2^m)$. The first two of these architectures are designed to provide security against SPA attacks, while the other two are designed to provide security against DPA attacks. The two proposed SPA

attack resistant cryptoprocessors are designed using ECSM methods that are based on buffering ($ECC_{B\text{-}SPA}$) and scalar splitting techniques ($ECC_{SB\text{-}SPA}$). Additional the other two proposed DPA attack resistant cryptoprocessors are designed using ECSM methods that apply randomization concept on the buffering ($ECC_{RB\text{-}DPA}$) and the scalar splitting techniques ($ECC_{RSB\text{-}DPA}$) at different levels (buffer capacity and processed points for PADD operation).

Our performance analysis shows that all four proposed cryptoprocessors need no additional computation load (and no extra dummy operation as well) compared to the double-and-add always ECSM and two of these cryptoprocessors outperform this binary method. The performance of the cryptoprocessors is as follow:

- The $ECC_{B\text{-}SPA}$ and $ECC_{RB\text{-}DPA}$ require m*PDBL + (m/2)*PADD
- The $ECC_{SB\text{-}SPA}$ and $ECC_{RSB\text{-}DPA}$ requires m*PDBL + (3*m*/8) *PADD

In term of security measurements, it is proven by examples relation between the security level and the buffer size. In addition, the countermeasures in $ECC_{SB\text{-}SPA}$ and $ECC_{RSB\text{-}DPA}$ cryptoprocessors inspect bit pairs instead of a single bit of the scalar, which introduce a new level of confusion. Finally the deployment of randomization technique in both the buffer capacity (being dynamic) and the processed points for PADD operation introduce a total confusion on the relation between the processed bits of the scalar and the performed point operation, which give advantage for the $ECC_{SB\text{-}SPA}$ and $ECC_{RSB\text{-}DPA}$ cryptoprocessors over the other proposed ones.

CHAPTER 6

Results and Discussions

To conduct an appropriate evaluation of our four proposed architectures for secure elliptic curve cryptoprocessors ($ECC_{B\text{-}SPA}$, $ECC_{SB\text{-}SPA}$, $ECC_{RB\text{-}DPA}$, and $ECC_{RSB\text{-}DPA}$), these architectures are compared to other two similar architectures; the first one is the regular secure elliptic curve reference cryptoprocessor (ECC_{RG}) which is based on 'Double-and-Add-Always' algorithm (See Algorithm 4.1), whereas the second one is a cryptoprocessor (ECC_{PLO}) based on the 'Propositional Logic Operations (PLO)' based algorithm for ECSM which was proposed in [97]. Additionally, we derive two extra architectures from our proposed architectures that are secure against the DPA attacks, where $ECC_{RB\text{-}DPA1}$ and $ECC_{RSB\text{-}DPA1}$ are designed with one level of randomization (randomizing the buffer capacity), at the same way $ECC_{RB\text{-}DPA2}$ and $ECC_{RSB\text{-}DPA2}$ with two levels of randomization (randomizing the buffer capacity, and the number of processed points for PADD operation). Therefore the evaluation covers a total of eight cryptoprocessors (ECC_{RG}, $ECC_{B\text{-}SPA}$, $ECC_{RB\text{-}DPA1}$, $ECC_{RB\text{-}DPA2}$, ECC_{PLO}, $ECC_{SB\text{-}SPA}$, $ECC_{RSB\text{-}DPA1}$, $ECC_{RSB\text{-}DPA2}$). These eight architectures were modeled using VHDL and synthesized on Altera FPGA. The developed VHDL models are parameterized to allow synthesizing the cryptoprocessors with different architectural features; additionally, these models allow for flexible definition of the following parameters:

1. The elliptic curve parameters a and b.
2. The underlying field $GF(2^m)$.
3. The base point P.

4. The secret key k.
5. The capacity of the buffer for the $ECC_{B\text{-}SPA}$ cryptoprocessor.
6. The capacity of each of the buffers for the $ECC_{RB\text{-}DPA}$ cryptoprocessor.

This chapter presents the results of synthesizing the various eight cryptoprocessors and compares these cryptoprocessors in terms of power, time delay and area. Altera Cyclone III EP3C80F780C7 FPGA has been used for prototyping. It is essential that identical FPGA chip is used with these cryptoprocessors in order to ensure that power, delay and area comparisons are done for the same technology and FPGA architecture and resources.

6.1 Comparison Methodology

The eight cryptoprocessors are designed to use the same field operation algorithms, e.g., multiplication and inversion. Thus, the performance difference between these cryptoprocessors is mainly a function of their control strategy and architectural differences independent of field operations. For example, field multiplication requires m clock cycles because of the Massey-Omura multiplier (Section 5.1).

Point Doubling (PDBL) requires 5 field multiplications, 4 field additions and 6 squarings. Each field addition and squaring requires only one clock cycle as a result of using ONB. The total number of clock cycles required for performing PDBL is 5m + 10 clock cycles. Point Addition (PADD), on the other hand, requires 14 field multiplications, 8 field additions and 6 squarings which requires 14m + 14 clock cycles.

The average time cost for point doubles, points addition and scalar multiplication required for the different algorithms for the eight cryptoprocessors are listed in Table 6.1. The results in Table 6.1 confirm the cryptoprocessors' performance analysis presented in Chapter 5, and that shows a

noticeable saving in the number of point operations (PADD) and timing for the proposed cryptoprocessors.

Table 6.1: Time Cost Comparison for the Eight ECC Cryptoprocessors

Cryptoprocessor	Time in Clock Cycles		
	Number of PDBLs	Number of PADDs	Scalar Multiplication
ECC_{RG}	m (5m + 10)	m (14m + 14)	$19m^2 + 24m$
ECC_{PLO}	m (5m + 10)	[m/2] (14m + 14)	$12m^2 + 17m$
$ECC_{B\text{-}SPA}$	m (5m + 10)	[m/2] (14m + 14)	$12m^2 + 17m$
$ECC_{RB\text{-}DPA1}$	m (5m + 10)	[m/2] (14m + 14)	$12m^2 + 17m$
$ECC_{RB\text{-}DPA2}$	m (5m + 10)	[m/2] (14m + 14)	$12m^2 + 17m$
$ECC_{SB\text{-}SPA}$	m (5m + 10)	[3m/8] (14m + 14)	$10m^2 + [1/4]\ m^2 + 15m + [1/4\ ]\ m$
$ECC_{RSB\text{-}DPA1}$	m (5m + 10)	[3m/8] (14m + 14)	$10m^2 + [1/4]\ m^2 + 15m + [1/4\ ]\ m$
$ECC_{RSB\text{-}DPA2}$	m (5m + 10)	[3m/8] (14m + 14)	$10m^2 + [1/4]\ m^2 + 15m + [1/4\ ]\ m$

6.2 Synthesis Results and Comparison

The eight ECC cryptoprocessors has been synthesized over $GF(2^{173})$, $GF(2^{191})$, and $GF(2^{230})$ for different m sizes as recommended by NIST ($m \in \{173, 191, 230\}$) on an Altera Cyclone III EP3C80F780C7 FPGA which contains 81,264 Slices. Table 6.2 lists the synthesis results for these ECC cryptoprocessors in terms of: 1) Delay measured in ms, 2) Area measured in number of slices, and 3) Power consumed measured in mW. In addition, comparison results for the Delay, Area, and Power of these cryptoprocessors are described in Figure 6.1, Figure 6.2, and Figure 6.3 respectively.

The delay comparison result (in Figure 6.1) shows that for security level of m = 173, the best time delays of 8.831 ms, 9.089 ms, and 9.536 ms are achieved by the Buffer-based cryptoprocessors (with no scalar splitting): $ECC_{B\text{-}SPA}$, $ECC_{RB\text{-}DPA1}$, and $ECC_{RB\text{-}DPA2}$ respectively; while for higher security level of m = 230, the best time delays of 23.862 ms, 24.483 ms, 24.649 ms are achieved

by the Split-Buffer-based cryptoprocessors: $ECC_{SB\text{-}SPA}$, $ECC_{RSB\text{-}DPA1}$, and $ECC_{RSB\text{-}DPA1}$ respectively.

Table 6.2: The Eight ECC Cryptoprocessor Synthesis Results.

Cryptoprocessor	m	Clock(MHz)	Delay(ms)	Area (Slices)	Area Usage	Power (mW)
ECC_{RG}	173	34.11	16.793	22,292	27%	169.70
ECC_{PLO}	173	35.80	10.114	25,977	32%	178.64
$ECC_{B\text{-}SPA}$	173	41.00	8.831	25,954	32%	173.97
$ECC_{RB\text{-}DPA1}$	173	39.84	9.089	26,137	32%	173.99
$ECC_{RB\text{-}DPA2}$	173	37.97	9.536	26,155	32%	174.00
$ECC_{SB\text{-}SPA}$	173	29.33	10.549	45,543	56%	192.92
$ECC_{RSB\text{-}DPA1}$	173	25.47	12.148	45,650	56%	193.18
$ECC_{RSB\text{-}DPA2}$	173	24.43	12.665	45,502	56%	191.60
ECC_{RG}	191	33.29	20.959	24,576	30%	177.78
ECC_{PLO}	191	25.63	17.207	28,703	35%	187.53
$ECC_{B\text{-}SPA}$	191	40.36	10.927	28,625	35%	181.01
$ECC_{RB\text{-}DPA1}$	191	33.47	13.177	29,325	37%	182.97
$ECC_{RB\text{-}DPA2}$	191	29.34	15.031	29,746	37%	183.14
$ECC_{SB\text{-}SPA}$	191	26.39	14.280	50,736	62%	197.59
$ECC_{RSB\text{-}DPA1}$	191	24.47	15.400	51,067	63%	198.29
$ECC_{RSB\text{-}DPA2}$	191	25.21	14.948	51,138	63%	198.56
ECC_{RG}	230	22.56	44.797	29,539	36%	193.09
ECC_{PLO}	230	22.76	28.063	34,483	42%	199.57
$ECC_{B\text{-}SPA}$	230	23.05	27.710	35,060	43%	196.77
$ECC_{RB\text{-}DPA1}$	230	22.90	27.891	35,949	44%	197.24
$ECC_{RB\text{-}DPA2}$	230	22.78	28.038	36,190	45%	197.54
$ECC_{SB\text{-}SPA}$	230	22.87	23.862	60,819	75%	213.05
$ECC_{RSB\text{-}DPA1}$	230	22.29	24.483	61,007	75%	213.46
$ECC_{RSB\text{-}DPA2}$	230	22.14	24.649	61,074	75%	213.84

On the other side, the area comparison result (in Figure 6.2) shows a consistency on the variation of utilized area of the eight cryptoprocessors for all values of m. Likewise, it is as expected that by using buffers and extra registers for the scalar splitting, more area are required in comparison with the other cryptoprocessors. For instance, the area utilization for the ECC_{RG} cryptoprocessor is only 29,539 slices, whereby it is almost twice (60,819, 61,007, and 61,074 slices) for the

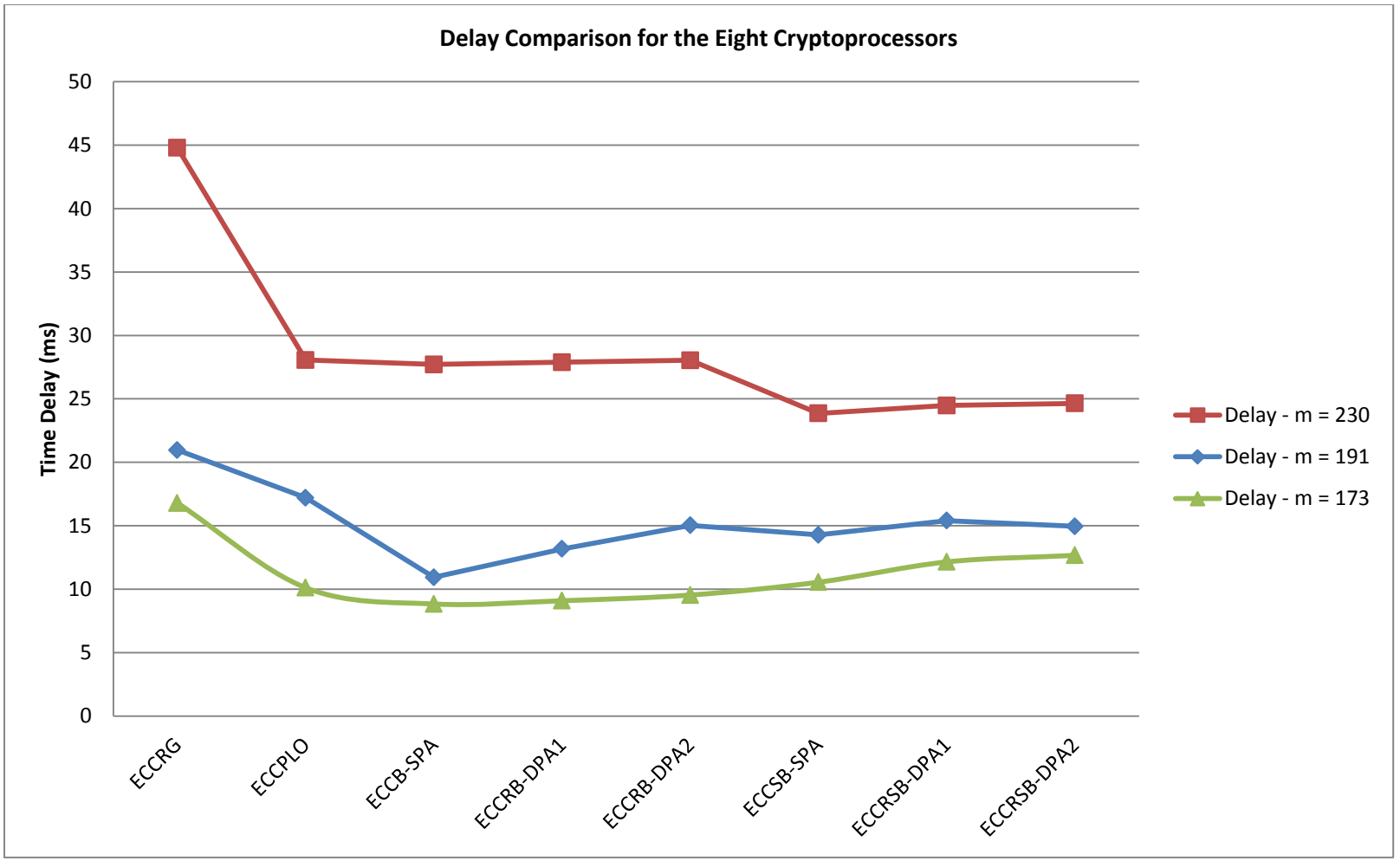

Figure 6.1: Delay Comparison for the Eight Cryptoprocessors for All Values of m (173,191,230)

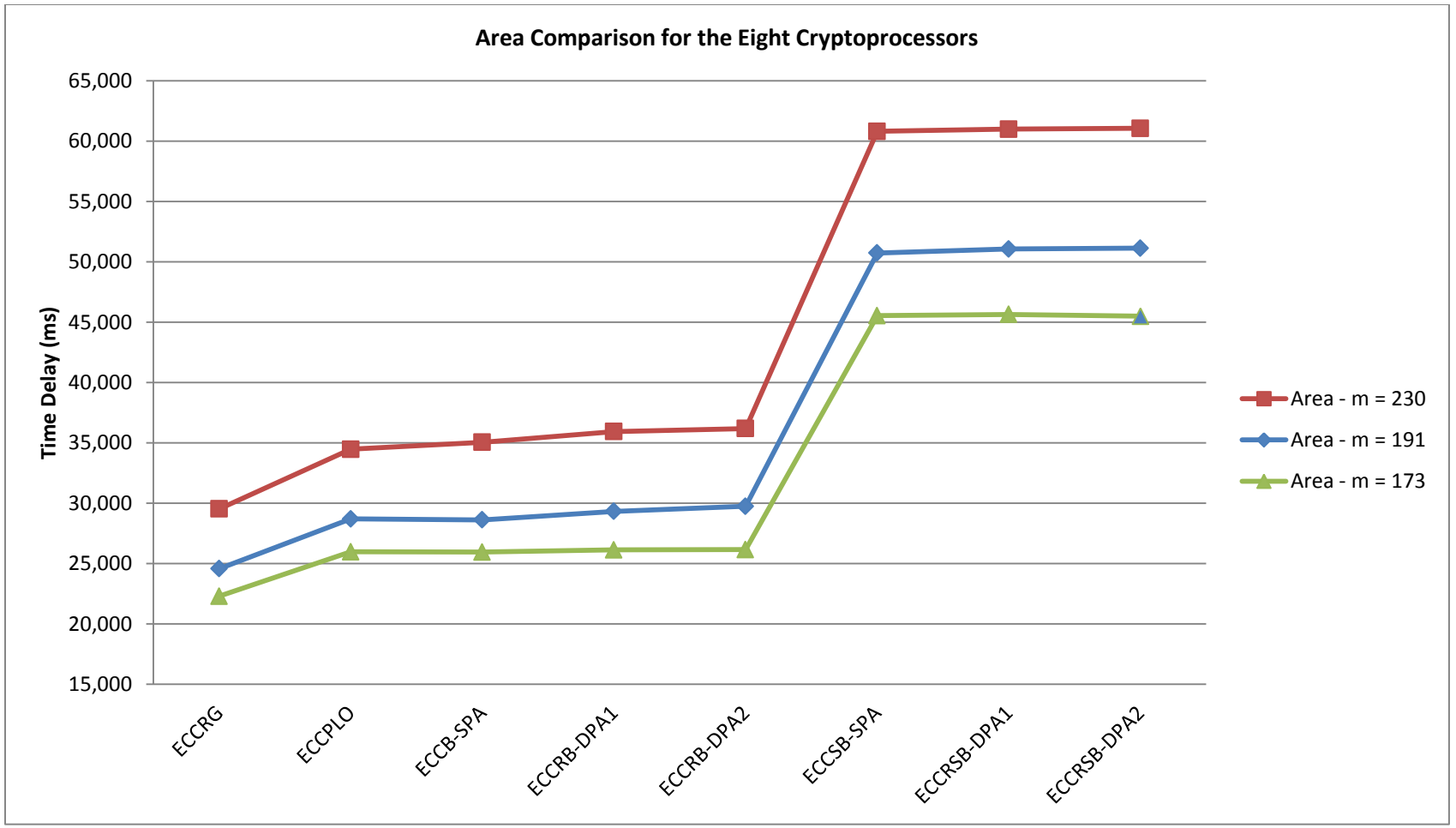

Figure 6.2: Area Comparison for the Eight Cryptoprocessors for All Values of m (173,191,230)

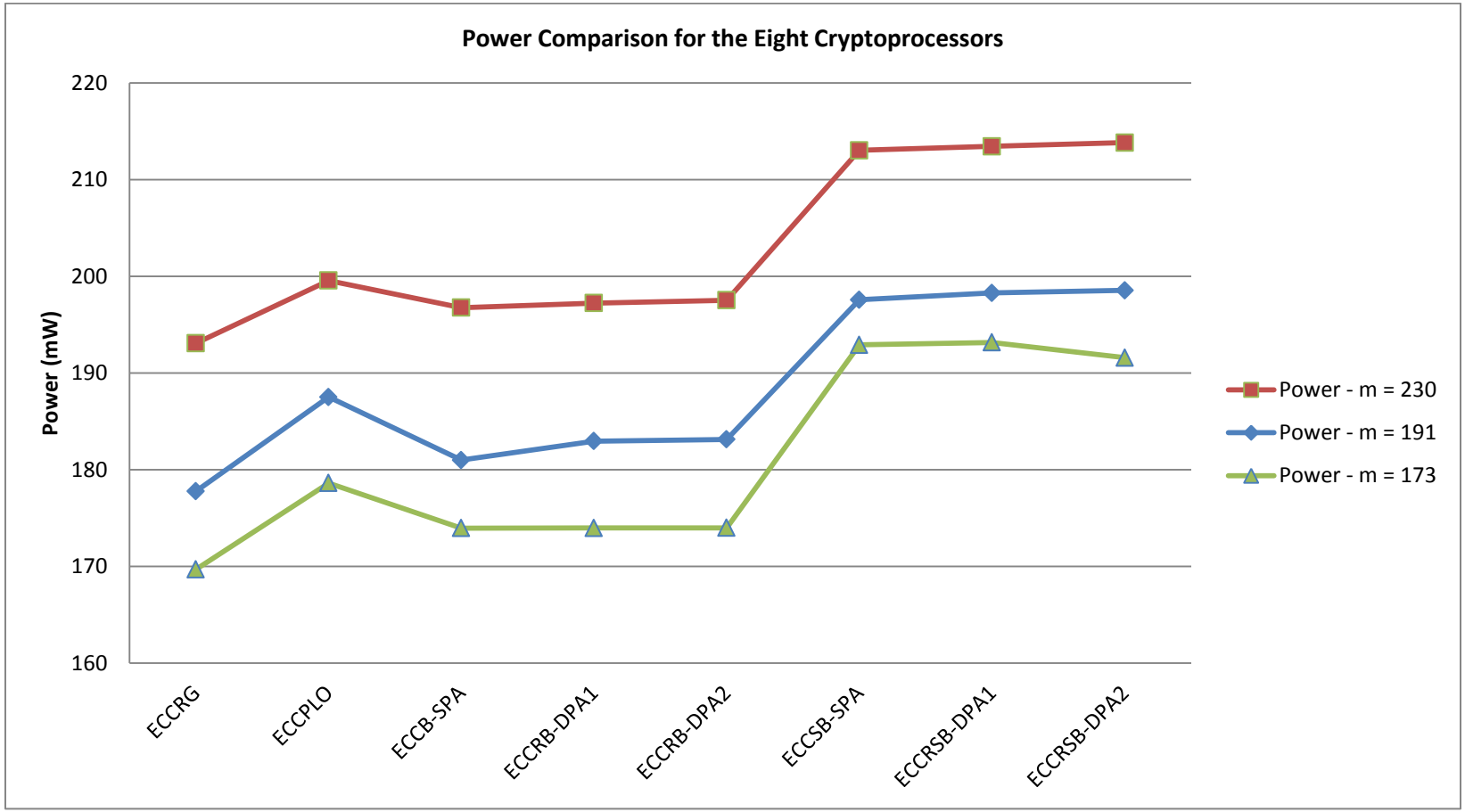

Figure 6.3: Power Comparison for the Eight Cryptoprocessors for All Values of m (173,191,230)

$ECC_{SB\text{-}SPA}$, $ECC_{RSB\text{-}DPA1}$, and $ECC_{RSB\text{-}DPA2}$ cryptoprocessors respectively.

As for the power comparison result (in Figure 6.3), it shows a harmony with the area consumption for all cryptoprocessors (in Figure 6.2). Nevertheless, although the increase in power consumption is directly proportional to the area utilized by the architectures; there is an exception for the ECC_{PLO} cryptoprocessor, because of its extra time delay which results in more power consumption. In general, the increase in power consumption is quite reasonable, since it is only in the range of 10%.

6.3 Delay, Area, and Power Cost Complexity Analysis

Architecture designers for cryptographic solutions may not have the same importance to cost factors (delay, area, and power), as this will always depend on the application requirements and constrains. For instance, in the resource constrained devices like sensor mote, or RFID, power consumption and area utilization are of more importance than delay (speed); whereas other applications might give more important to delay, but area or power may not be a concern for such. Thus, it is important to present the cost complexity in term of delay, area, and power for the eight cryptoprocessors.

The area, delay, and power will be multiplied partially or all together to generate different cost figures, which can be used by architecture designers for evaluation purpose. Any of the possible cost complexity (AT, AT^2, A^2T, ATP, ATP^2, AT^2P, A^2TP, AT^2P^2, A^2TP^2, A^2T^2P) can represent importance weighting for the cost factors instance. For instance, the cost ATP (A*T*P = multiplying A, T, and P), represent an equal importance to the application for all cost factors (time, area and power). Moreover, the general formula for cost complexity is given in Equation 6.1 as below:

$$\text{Cost Complexity} = A^x T^y P^z \qquad \text{(Equation 6.1)}$$

where x, y, z ϵ {0, 1, 2}, and the value 0 means no importance for the cost factor, and 2 means high importance for the cost factor.

As example, for applications with no importance to (not concerned about) area, but given importance to delay, but more importance to power, the cost complexity (See Equation 6.1) can be calculated as

$$\text{Cost Complexity} = A^0T^1P^2 = TP^2$$

Cost complexity of different variation of $A^xT^yP^z$ (as given in Equation 6.1) for all eight architectures are described in Table 6.3. In addition, for selective variation of $A^xT^yP^z$ (ATP, AT^2P, ATP^2, and A^2TP^2), the cost complexity for all eight architectures are plotted in (Figure 6.4, Figure 6.5, Figure 6.6, and Figure 6.7) respectively for all values of m (173, 191, and 230), where some cost complexity values are rescaled to fit in these figures.

For the ATP, ATP^2, and A^2TP^2 cost complexities comparison (in Figure 6.4, Figure 6.6, and Figure 6.7), the lowest cost results are given by the two cryptoprocessors: $ECC_{RB\text{-}DPA1}$, and $ECC_{RB\text{-}DPA2}$, while the highest cost results are given by the two cryptoprocessors: $ECC_{RSB\text{-}DPA1}$, and $ECC_{RSB\text{-}DPA2}$, which are proven to provide highest security level in compared to the other cryptoprocessors (Section 6.4).

For the AT^2P cost complexity comparison (in Figure 6.5), the lowest cost results are also given by the two cryptoprocessors: $ECC_{RB\text{-}DPA1}$, and $ECC_{RB\text{-}DPA2}$, while the highest cost results are given by the two cryptoprocessors: $ECC_{RSB\text{-}DPA1}$, and $ECC_{RSB\text{-}DPA2}$, expect for m = 230 where the highest cost result is given by the ECC_{RG} cryptoprocessor.

Table 6.3: Cost Complexity (A, D, P) measurements for all values of m

Cryptoprocessor	m	AT (10^6)	AT^2 (10^7)	A^2T (10^8)	ATP (10^8)	ATP^2 (10^{10})	AT^2P (10^9)	A^2TP (10^{10})	AT^2P^2 (10^{12})	A^2TP^2 (10^{12})	A^2T^2P (10^{11})
ECC_{RG}	173	0.374	0.629	0.083	0.635	1.078	1.067	0.142	0.181	0.240	0.238
ECC_{PLO}	173	0.248	0.236	0.064	0.443	0.791	0.422	0.115	0.075	0.205	0.110
$ECC_{B\text{-}SPA}$	173	0.263	0.266	0.068	0.457	0.794	0.462	0.119	0.080	0.206	0.120
$ECC_{RB\text{-}DPA1}$	173	0.231	0.204	0.060	0.402	0.699	0.355	0.105	0.062	0.183	0.093
$ECC_{RB\text{-}DPA2}$	173	0.238	0.216	0.062	0.414	0.720	0.376	0.108	0.065	0.188	0.098
$ECC_{SB\text{-}SPA}$	173	0.480	0.507	0.219	0.927	1.788	0.978	0.422	0.189	0.814	0.445
$ECC_{RSB\text{-}DPA1}$	173	0.555	0.674	0.253	1.071	2.070	1.301	0.489	0.251	0.945	0.594
$ECC_{RSB\text{-}DPA2}$	173	0.576	0.730	0.262	1.104	2.116	1.398	0.502	0.268	0.963	0.636
ECC_{RG}	191	0.515	1.080	0.127	0.916	1.628	1.919	0.225	0.341	0.400	0.472
ECC_{PLO}	191	0.431	0.649	0.124	0.809	1.517	1.216	0.232	0.228	0.436	0.349
$ECC_{B\text{-}SPA}$	191	0.493	0.848	0.141	0.892	1.614	1.534	0.255	0.278	0.462	0.439
$ECC_{RB\text{-}DPA1}$	191	0.320	0.350	0.094	0.586	1.073	0.641	0.172	0.117	0.315	0.188
$ECC_{RB\text{-}DPA2}$	191	0.392	0.516	0.117	0.718	1.315	0.946	0.214	0.173	0.391	0.281
$ECC_{SB\text{-}SPA}$	191	0.724	1.035	0.368	1.432	2.829	2.044	0.726	0.404	1.435	1.037
$ECC_{RSB\text{-}DPA1}$	191	0.786	1.211	0.402	1.559	3.092	2.402	0.796	0.476	1.579	1.226
$ECC_{RSB\text{-}DPA2}$	191	0.764	1.143	0.391	1.518	3.014	2.269	0.776	0.451	1.541	1.160
ECC_{RG}	230	1.323	5.928	0.391	2.555	4.934	11.446	0.755	2.210	1.457	3.381
ECC_{PLO}	230	0.967	2.711	0.333	1.930	3.851	5.410	0.665	1.080	1.328	1.866
$ECC_{B\text{-}SPA}$	230	0.984	2.761	0.345	1.936	3.809	5.433	0.679	1.069	1.336	1.905
$ECC_{RB\text{-}DPA1}$	230	0.996	2.760	0.358	1.965	3.875	5.444	0.706	1.074	1.393	1.957
$ECC_{RB\text{-}DPA2}$	230	1.009	2.815	0.365	1.994	3.939	5.561	0.722	1.099	1.425	2.013
$ECC_{SB\text{-}SPA}$	230	1.451	3.463	0.883	3.092	6.587	7.378	1.881	1.572	4.006	4.487
$ECC_{RSB\text{-}DPA1}$	230	1.494	3.657	0.911	3.188	6.806	7.806	1.945	1.666	4.152	4.762
$ECC_{RSB\text{-}DPA2}$	230	1.505	3.711	0.919	3.219	6.884	7.935	1.966	1.697	4.204	4.846

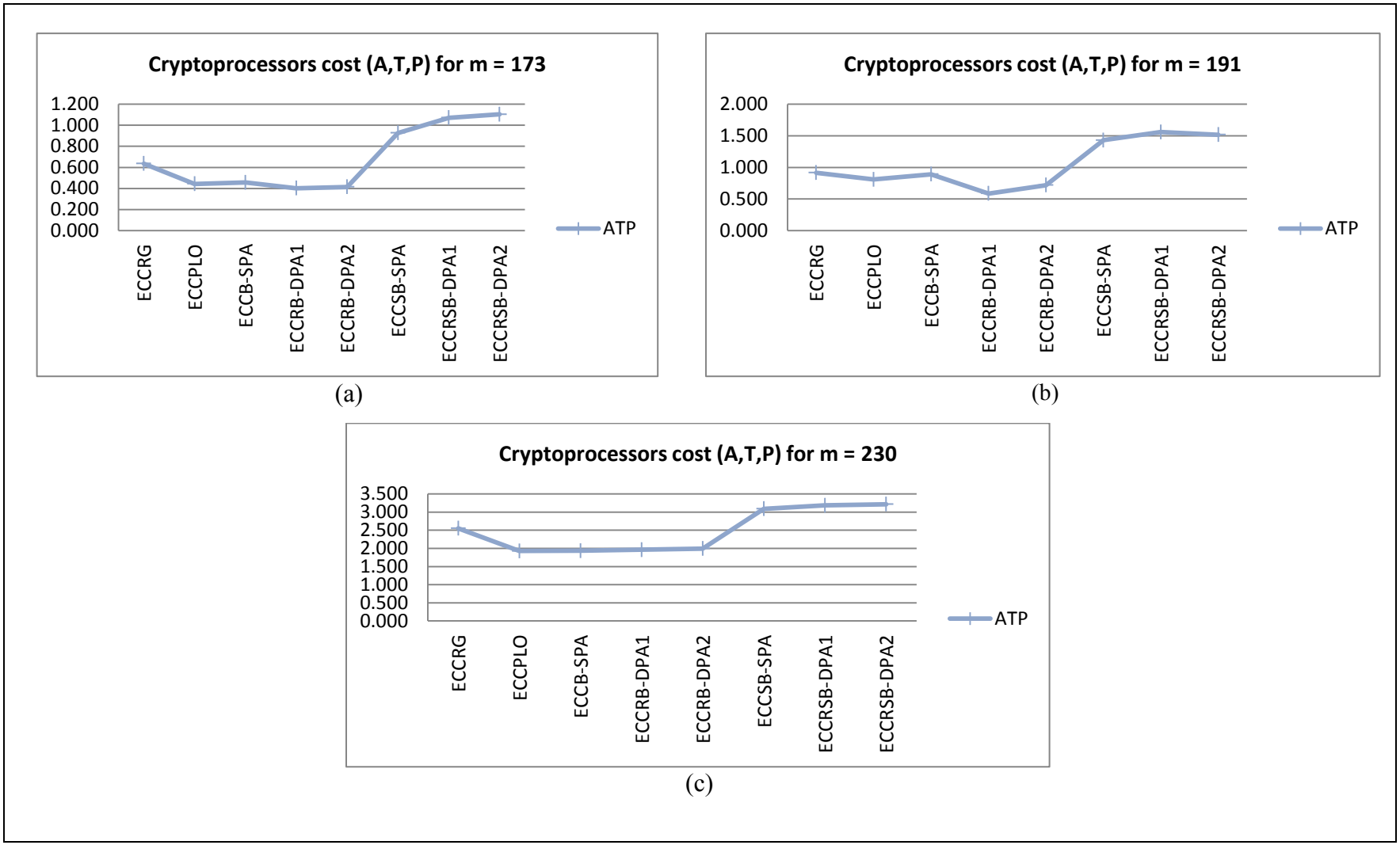

Figure 6.4: Cost Complexity (ATP) Comparison for m = 173, 191, and 230

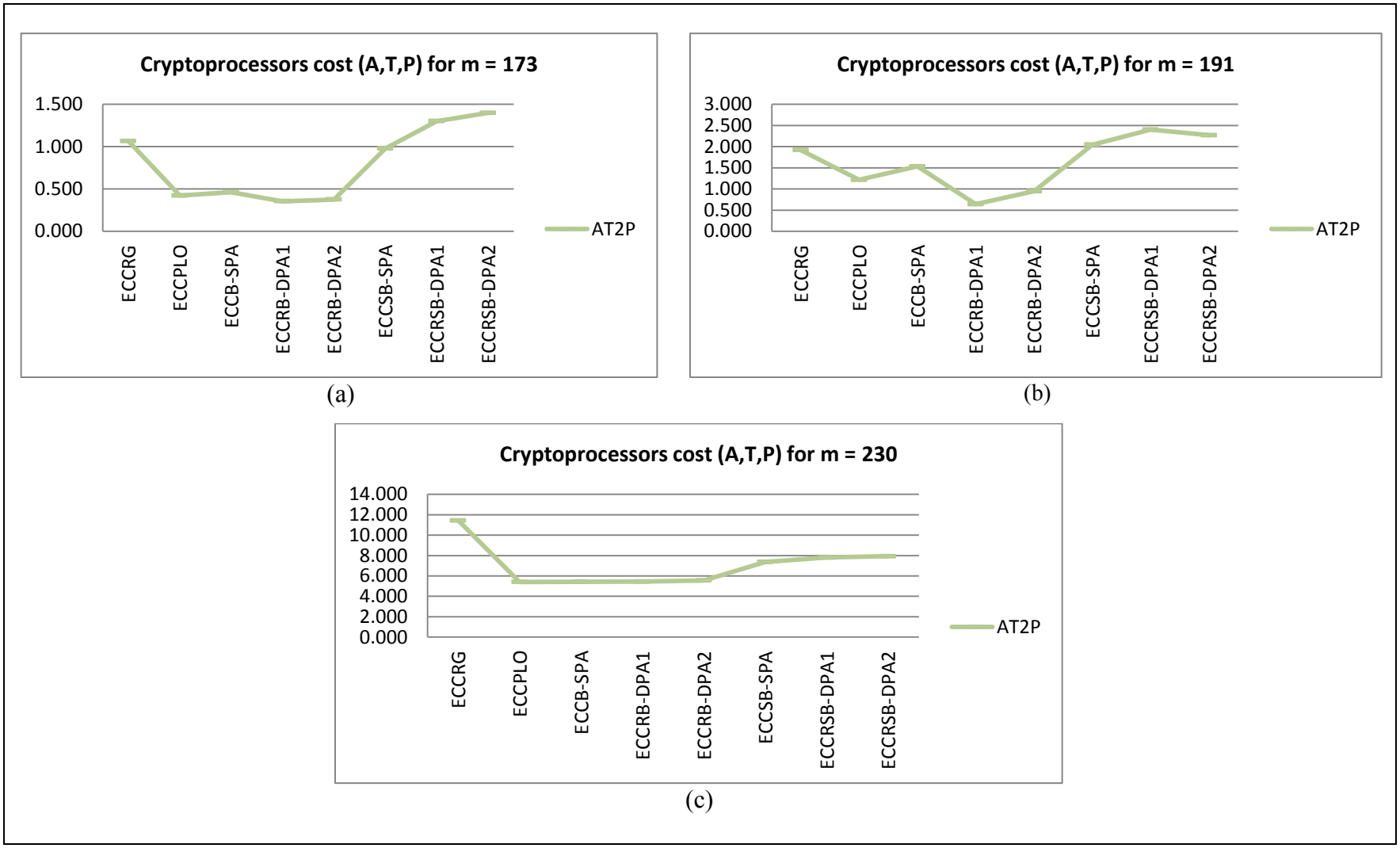

Figure 6.5: Cost Complexity (AT^2P) Comparison for m = 173, 191, and 230

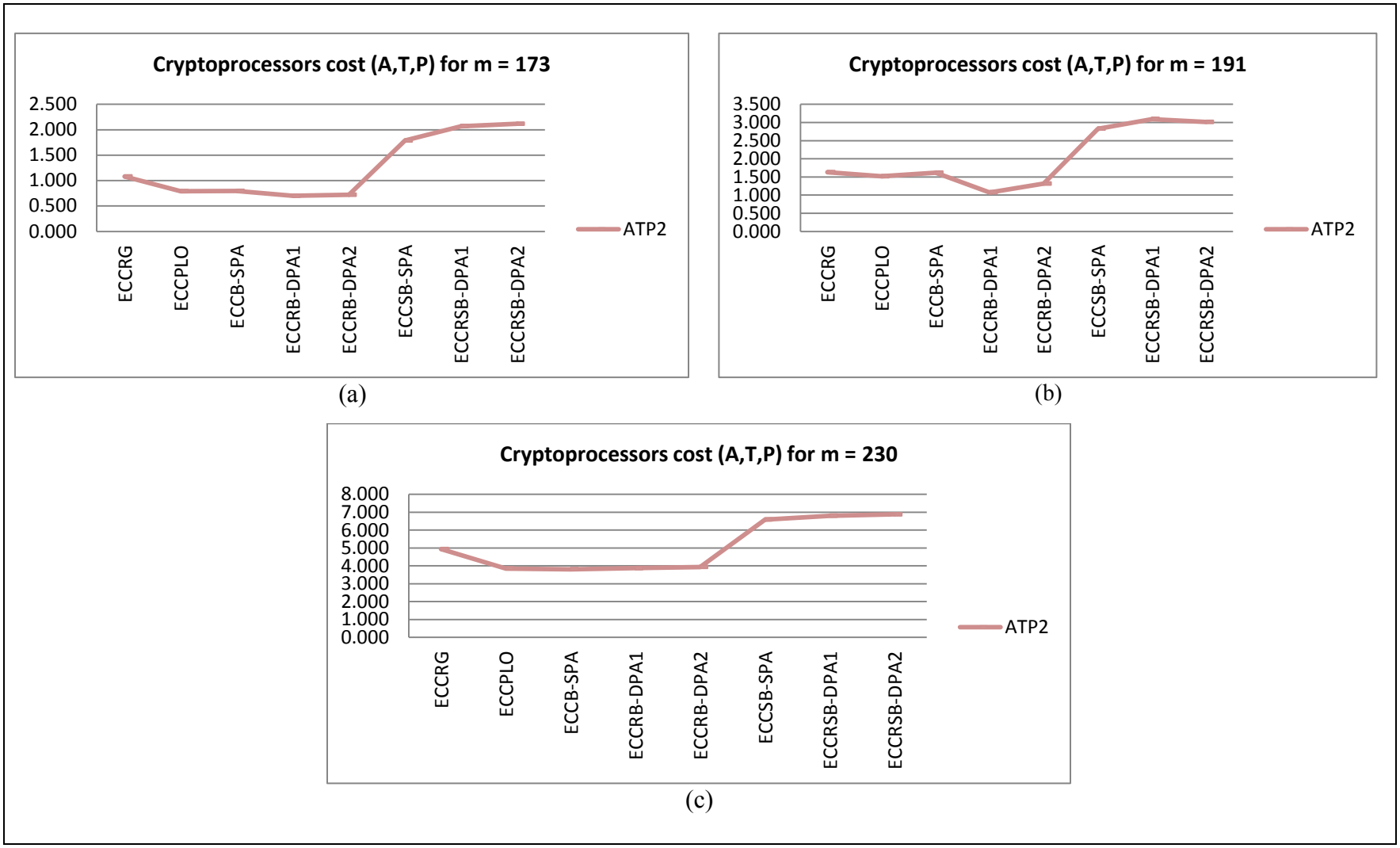

Figure 6.6: Cost Complexity (ATP^2) Comparison for m = 173, 191, and 230

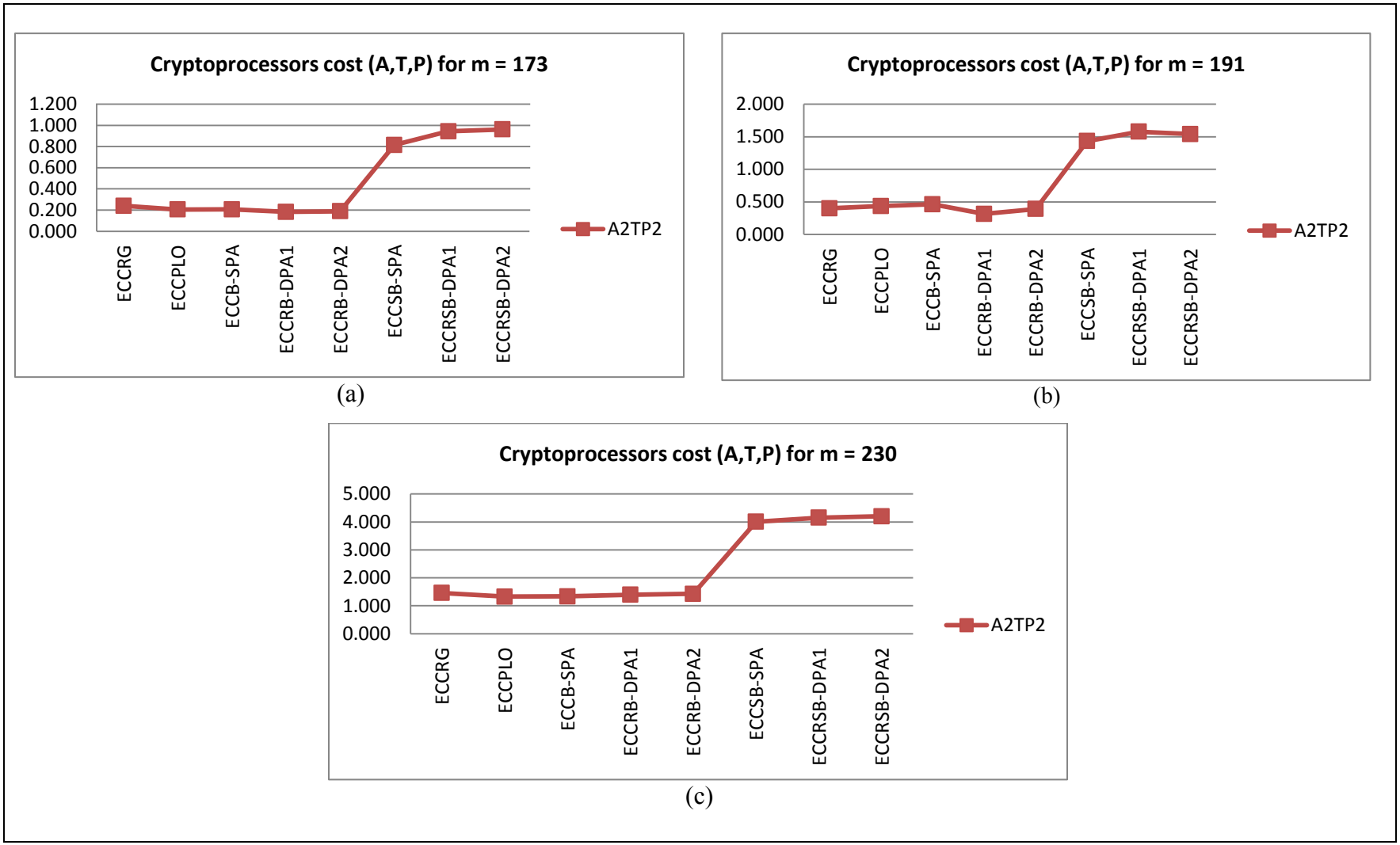

Figure 6.7: Cost Complexity (A^2TP^2) Comparison for m = 173, 191, and 230

6.4 Summary

In this chapter, we present the results of synthesizing the various cryptoprocessors and compare these eight cryptoprocessors in terms of power, time delay and area. Altera Cyclone III EP3C80F780C7 FPGA has been used for prototyping.

A delay, area, and power comparison study is conducted for the different cryptoprocessors, with different values of m. The comparison is done in details taking into consideration the randomization levels for DPA aware cryptoprocessors. In addition, a more advanced comparison is done on the cost complexity level, which provides a framework for the architecture designers to select the appropriate design.

Results showed that our proposed architectures give best cost complexity in comparison to the other latest proposed in the research field.

The presented work shows very interesting results (security level, and cost complexity) as compared to other similar work recently proposed in the research field.

CHAPTER 7

Conclusions and Future Research

In the recent few years, intense research has been focused on the efficient implementation of Elliptic Curve Cryptosystems (ECC) [3] [4] in extreme constrained resources such as the Wireless Sensor Networks (WSN) [1]. Likewise, the current ECC implementations in WSN [7] are vulnerable to Side Channel Analysis (SCA) attacks [8], in particularly to Power Analysis Attacks (PAA) [9], due to the lack of secure physical shielding, their deployment in remote regions and it is left unattended. This thesis has focused on devising algorithms and architectures for elliptic curve cryptoprocessors that are not only efficient, but also PAA resistant with no any extra cost in terms of power, time delay, and area. We proposed two cryptoprocessors ($ECC_{B\text{-}SPA}$, $ECC_{SB\text{-}SPA}$), and another two cryptoprocessors ($ECC_{RB\text{-}DPA}$, $ECC_{RSB\text{-}DPA}$) that are secure against SPA attacks and DPA attacks respectively.

A more detailed description of the contributions of this thesis follows in Section 7.1. Possible future research directions are described in Section 7.2.

1.1 Summary of Contributions

Firstly, we proposed two robust and high efficient PAA aware elliptic curve cryptoprocessors' GF(2^m) architectures ($ECC_{B\text{-}SPA}$, $ECC_{SB\text{-}SPA}$) for WSN. These architectures are based on innovative algorithms for ECC core operation and are secure against SPA attacks.

Secondly, we proposed two additional cryptoprocessors' GF(2^m) architectures ($ECC_{RB\text{-}DPA}$, $ECC_{RSB\text{-}DPA)}$ that are secured against DPA attacks.

The security advantages provided in these four cryptoprocessors covers both the SPA and DPA attacks by applying: (i) PADD operation delaying using buffer storage, (ii) Scalar splitting for cost saving and additional complexity, and (iii) Complicated randomization technique for extra confusion to secure

against DPA attacks.

Thirdly, a total of eight architectures which includes, in addition to the two SPA aware with the other two DPA aware proposed architectures, two more architectures derived from our DPA aware proposed once, along with two other similar PAA aware architectures. The eight proposed architectures are synthesized for GF(2^{173}), GF(2^{191}), and GF(2^{230}) on an Altera Cyclone III EP3C80F780C7 FPGA.

The time delay performance results of these four cryptoprocessors in number of Point Doubling (PDBL) and Point Addition (PADD) are as follow:

- The $ECC_{B\text{-}SPA}$ and $ECC_{RB\text{-}DPA}$ require m*PDBL + (m/2)*PADD
- The $ECC_{SB\text{-}SPA}$ and $ECC_{RSB\text{-}DPA}$ requires m*PDBL + (3*m*/8) *PADD

In term of security level, it is directly related to the buffer size. In addition, the countermeasures in $ECC_{SB\text{-}SPA}$ and $ECC_{RSB\text{-}DPA}$ cryptoprocessors inspect bit pairs instead of a single bit of the scalar, which introduce a new level of confusion. Finally the deployment of randomization technique in both the buffer capacity (being dynamic) and the processed points for PADD operation introduce a total confusion on the relation between the processed bits of the scalar and the performed point operation, which give advantage for the $ECC_{SB\text{-}SPA}$ and $ECC_{RSB\text{-}DPA}$ cryptoprocessors over the other proposed ones.

These results in the time delay and security level have a practical impact in the area and power consumption of these cryptoprocessors. For instance, these results may directly increase the area space as the buffer size increase, which leads to more processing effort, and thus more power consumption. Most remarkably, as different application might give different importance to critical factors such as power, area, and time delay, a careful selection of cryptoprocessor with the best cost complexity results can lead to the realization of record-breaking implementations of ECC in resource constrained devices for the targeted application.

Fourthly, the eight proposed architectures are analyzed and evaluated by comparing their performance results. In addition, a more advanced comparison, which is done on the cost complexity level (Area,

Delay, and Power), provides a framework for the architecture designers to select the appropriate design. Our results show a significant advantage of our proposed architectures for cost complexity in comparison to the other latest proposed in the research field.

For the ATP, ATP^2, and A^2TP^2 cost complexities comparison (in Figure 6.4, Figure 6.6, and Figure 6.7) for all eight cryptoprocessors have been done and evaluated. The results show that the lowest cost results are given by the two cryptoprocessors: $ECC_{RB\text{-}DPA1}$, and $ECC_{RB\text{-}DPA2}$, while the highest cost results are given by the two cryptoprocessors: $ECC_{RSB\text{-}DPA1}$, and $ECC_{RSB\text{-}DPA2}$, which are proven to provide highest security level in compared to the other cryptoprocessors (Section 6.4).

For the AT^2P cost complexity comparison (in Figure 6.5), the lowest cost results are also given by the two cryptoprocessors: $ECC_{RB\text{-}DPA1}$, and $ECC_{RB\text{-}DPA2}$, while the highest cost results are given by the two cryptoprocessors: $ECC_{RSB\text{-}DPA1}$, and $ECC_{RSB\text{-}DPA2}$, expect for m = 230 where the highest cost result is given by the ECC_{RG} cryptoprocessor.

1.2 Future Work

Future potential research may further investigate the following:

1. Exploring the hardware/software co-design of PAA aware ECC architecture for WSN.
2. Developing a mechanism for accurately evaluating the security level of PAA aware cryptoprocessors, and
3. Rebuilding our framework for the architecture designers to include security level (S-Level) as a fourth dimension in addition to (Area, Delay, and Power).
4. Evaluating the four architectures on other ASIC platforms (e.g. chip-based payment card for banking financial transactions).

BIBLIOGRAPHY

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci, "Wireless Sensor Networks: a survey," *Computer Networks,* 15 March vol. 38, issue. 4, p. 393-422, 2002.

[2] ANSI X9.62, "Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)," 1998.

[3] I. Blake, G. Seroussi, N. Smart, "Elliptic Curves in Cryptography," *Cambridge University Press, Cambridge, 1999.*

[4] H. Cohen, G. Frey, R. Avanzi, C. Doche, T. Lange, K. Nguyen and F. Vercauteren, "Handbook of Elliptic and Hyperelliptic Curve Cryptography. Discrete Mathematics and Its Applications," Vol. 34, Chapman and Hall, CRC, USA, 2005.

[5] C. K. Koc, "High-speed rsa implementation," In RSA Laboratories TR201, Nov. 1994.

[6] T. Elgamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *Information Theory, IEEE Transactions on,* vol. 31, p. 469 – 472, July 1985.

[7] E.-O. Blaß and M. Zitterbart, "Efficient Implementation of Elliptic Curve Cryptography for Wireless Sensor Networks," Telematics Technical Reports, March, 2005.

[8] B. Möller, "Parallelizable elliptic curve point multiplication method with resistance against side-channel attacks," in *Int. Conf. on Information Security (ISC 2002)*, Sao Paulo, Brazil, vol. 2433, p. 402–413, 2002.

[9] P.-Y. Liardet, N.P. Smart, "Preventing SPA/DPA in ECC systems using the Jacobi form," in *Workshop on Cryptographic Hardware and Embedded Systems (CHES 2001)*, Paris, France, vol. 2162, p. 391– 401, 2001.

[10] J. Ha, J. Park, S. Moon and S. Yen, "Provably Secure Countermeasure Resistant to Several Types of Power Attack for ECC," *in Information Security Applications (WISA),* vol. 4867. Springer, p. 333 – 344, 2007.

[11] H. Houssain, M. Badra and T. Al-Somani, "Power Analysis Attacks on ECC: A Major Security Threat," *International Journal of Advanced Computer Science and Applications (IJACSA),* vol. 3, issue. 6 ,p. 90 - 96, 2012.

[12] L. Batina, J. Hogenboom, N. Mentens, J. Moelans and J. Vliegen, "Side-channel evaluation of FPGA implementations of binary Edwards curves," in *in International Conference on Electronics, Circuits and Systems 2010*, p. 1255-1258, Athens, Greece, Dec. 12-15, 2010.

[13] N. Biggs, Discrete Mathematics, New York: Oxford University Press, 1985.

[14] R. McEliece, Finite Fields for Computer Scientists and Engineers, Kluwer Academic Publishers, 1987.

[15] R. Lidl and H. Niederreiter, "Introduction to finite fields and their applications", revised edition ed., Cambridge, UK: Cambridge University Press, 1994.

[16] R. Mullin, . I. Onyszchuk, S. Vanstone and R. Wilson, "Optimal normal bases in GF(pm)". Discrete Appl. Math, vol. 22, p. 149–161, 1988/1989.

[17] M. Rosing, Implementing Elliptic Curve Cryptography, Manning Publications Company, 1999.

[18] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation,* vol. 48, p. 203–209, 1987.

[19] A. Menezes, Elliptic Curve Public Key Cryptosystems, Kluwer Academic Publishers, 1993.

[20] H. Cohen, T. Ono and A. Miyaji, "Efficient elliptic curve exponentiation using mixed coordinates," in *In Advances in Cryptology ASIACRYPT'98*, 1998.

[21] K. Koyama and Y. Tsutuoka, "Speeding up elliptic cryptosystems by using signed binary window method," in *Advances in Cryptology Proc. Of Crypto'92, LNCS 740*, 1993.

[22] H. Cohen, A. Miyaji and T. Ono, "Efficient elliptic curve exponentiation.," in *Advances in Cryptology-Proc. Of ICICS'97, LNCS 1334*, 1997.

[23] J. Lopez and R. Dahab, "Improved Algorithms for Elliptic Curve Arithmetic in GF(2n)," in *Selected Areas in Crpytography.*, LNCS 1556. Springer-Verlag. p. 201 - 212, 1999.

[24] D. Gordon, "A Survey of Fast Exponentiation Methods," *Journal of Algorithms,* p. 129 – 146, 1998.

[25] D. Hankerson, A. Menezes and S. Vanstone, Guide to Elliptic Curve Cryptography, Springer-Verlag, 2004.

[26] V. S. Miller, "Use of elliptic curves in cryptography," in *CRYPTO '85: Proceedings of the Advances in cryptology*, New York, NY, USA, 1986.

[27] W. Diffie and M. Hellman, "New directions in cryptography," *Information Theory, IEEE Transactions on,* vol. 22, p. 644–654, 1976.

[28] J. Pollard, "Monte Carlo methods for index computation mod p," *Mathematics of Computation,* vol. 32, p. 918–924, 1978.

[29] R. Gallant, . R. Lambert and . S. Vanstone, "Improving the parallelized Pollard lambda search on binary anomalous curves," *Math. Comp.,* vol. 69, issue. 232, p. 1699-1705, 2000.

[30] E. D., G. R. , H. J. and K. S. , "Next Century Challenges: Scalable Coordination in Sensor Networks," in *Mobile Computing and Networking (MobiCom'99)*, Seattle, WA USA, 1999.

[31] M. Healy, T. Newe and E. Lewis, "Wireless Sensor Node Hardware: A Review," p. 621-624, 2008.

[32] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci, "A Survey on Sensor Networks," *IEEE Communications Magazine,* p. 102-114, 2002.

[33] C.-C. Shen, C. Srisathapornphat and C. Jaikaeo, "Sensor Information Networking Architecture and Applications," *IEEE Personal Communications,* August p. 52 - 59, 2001.

[34] E. Shi and A. Perrig, "Designing Secure Sensor Networks," *IEEE Wireless Communications,* December p. 38 - 43, 2004.

[35] S. S. Doumit and D. P. Agrawal, "Self-Organizing and Energy-Efficient Network of Sensors," *IEEE,* p. 1 - 6, 2002.

[36] W. B. Heinzelman, A. L. Murphy, H. S. Carvalho and M. A. Perillo, "Middleware to Support Sensor Network Applications," *IEEE Network,* p. 6-14, January/February, 2004.

[37] M. Srivastava, R. Muntz and M. Potkonjak, "Smart kindergarten: Sensor-based wireless networks for smart developmental problem-solving enviroments," in *in Proceedings of the 7th Annual International Conference on Mobile Computing and Networking (MobiCom'01)*, Rome, Italy, 2001.

[38] D. Carman, P. Krus and B. Matt, "Constraints and approaches for distributed sensor network security," NAI Labs, Network Associates Inc., Glenwood, MD, 2000.

[39] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler and K. Pister, "System architecture directions for networked sensors," in *Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems*, New York, 2000.

[40] L. Yuan and G. Qu, "Design space exploration for energy efficient secure sensor networks," in *Proceedings of IEEE International Conference on Application-Specific Systems, Architectures, and Processors*, p. 88-100, July 2002.

[41] A. Perrig, R. Szewczyk, V. Wen, D. Culler and J. Tygar, "SPINS: Security protocols for sensor networks," *Wireless Networks,* September vol. 8, issue. 5, p. 521-534, 2002.

[42] A. D. Wood and J. A. Stankovic, "Denial of service in sensor networks," *IEEE Computer,* vol. 35, issue. 10, p. 54-62, 2002.

[43] L. Eschenauer and V. Gligor, "A key-management scheme for distributed sensor networks," in *Proceedings of the 9th ACM Conference on Computer and Networking, p. 41-47*, Nov 2002.

[44] S. Capkun and J.-P. Hubaux, "Secure positioning in wireless networks," *IEEE Journal on Selected Areas in Communications,* vol. 24, issue. 2, p. 221-232, 2006.

[45] L. Lazos and R. Poovendran, "SERLOC: Robust localization for wireless sensor networks," *ACM Transactions on Sensor Networks,* vol.1, issue. 1, p. 73-100, 2005.

[46] R. Anderson and M. Kuhn, "Low cost attacks on tamper resistant devices," In Proceedings of the 5th International Workshop on Security Protocols, Bruce Christianson, Bruno Crispo, T. Mark A. Lomas, and Michael Roe (Eds.). Springer-Verlag, London, UK, p. 125-136 , 1997.

[47] C. Hartung, J. Balasalle and R. Han, "Node compromise in sensor networks: The need for secure systems," Technical Report CU-CS-988-04, Department of Computer Science, University of Colorado at Boulder, 2004.

[48] X. Wang, W. Gu, S. Chellappan, K. Schoseck and D. Xuan, "Lifetime optimization of sensor networks under physical attacks," in *Proceedings of IEEE International Conference on Communications*, May 2005.

[49] X. Wang, W. Gu, S. Chellappan, D. Xuan and T. H. Laii, "Search-based physical attacks in sensor networks: Modeling and defense," Technical report, Department of Computer Science and Engineering, Ohio State University, February 2005.

[50] R. Watro, D. Kong, S. Cuti, C. Gardiner, C. Lynn and P. Kruus, "TinyPK: securing sensor networks with public key technology," In SASN '04: Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks, p. 59–64, 2004.

[51] L. Oliveira, D. Aranha, E. Morais, F. Daguano, J. L´opez and R. Dahab, "TinyTate: Computing the TinyTate in resource-constrained nodes," in *6th IEEE International Symposium on Network Computing and Applications*, Cambridge,MA, 2007.

[52] D. Malan, M. Welsh and M. Smith, "A public-key infrastructure for key distribution in TinyOS based on elliptic curve cryptography," in *Proc. of the 1st IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON '04), p. 71–80*, Santa Clara, Calif, USA, 2004.

[53] N. Gura, A. Patel, A. S. Wander, H. Eberle and S. Chang Shantz, "Comparing elliptic curve cryptography and RSA on 8-bit CPUs," *in Cryptographic Hardware and Embedded Systems — CHES 2004,* vol. 3156 of LNCS, p. 119–132, Springer Verlag, 2004.

[54] S. Seo, D.-G. Han, H. Kim and S. Hong, "TinyECCK: Efficient Elliptic Curve Cryptography Implementation over GF(2m) on 8-bit MICAz Mote," *IEICE Transactions on Info and Systems E91-D(5),* p. 1338-1347, 2008.

[55] H. Wang and Q. Li, "Efficient implementation of public key cryptosystems on mote sensors," *in Information and Communications Security — ICICS 2006,* vol. 4307 of LNCS, p. 519–528, 2006.

[56] P. Szczechowiak, L. B. Oliveira, M. Scott, M. Collier and R. Dahab, "NanoECC: Testing the limits of elliptic curve cryptography in sensor networks," *in Wireless Sensor Networks — EWSN 2008,* vol. 4913 of LNCS, p. 305–320, 2008.

[57] E. Ozturk, B. Sunar and E. Savas, "Low-power elliptic curve cryptography using scaled modular arithmetic," *in Cryptographic Hardware and Embedded Systems - CHES 2004,* vol. 3156 of LNCS, p. 92–106, 2004.

[58] G. Gaubatz, J.-P. Kaps, E. Öztürk and B. Sunar, "State of the art in ultra-low power public key cryptography for wireless sensor networks," *Third IEEE International Conference on Pervasive Computing and Communications Workshops, Workshop on Pervasive Computing and Communications Security–PerSec'05, IEEE Computer Society,* p. 146–150, Mar, 2005.

[59] J. Wolkerstorfer, "Scaling ECC Hardware to a Minimum," *in ECRYPT workshop - Cryptographic Advances in Secure Hardware - CRASH 2005,* September 6-7, 2005. Invited Talk.

[60] L. Batina, N. Mentens, K. Sakiyama, B. Preneel and I. Verbauwhede, "Low-Cost Elliptic Curve Cryptography for Wireless Sensor Networks," in *Proc. ESAS'06, p.6-17*, 2006.

[61] L. Breveglieri, G. Bertoni, and M. Venturi, "Power Aware Design of an Elliptic Curve Coprocessor for 8 bit Platforms," in *Proc. of PERCOMW'06, p. 337*, 2006.

[62] S. Kumar and C. Paar, "Are standards compliant elliptic curve cryptosystems feasible on RFID?," in *Proc. of Workshop on RFID Security*, Graz, Austria, July 2006.

[63] J. Portilla, A. Marnotes, E. de la Torre, T. Riesgo, O. Stecklina, S. Peter and P. Langendörfer, "Adaptable Security in Wireless Sensor Networks by Using Reconfigurable ECC Hardware Coprocessors," *in International Journal of Distributed Sensor Networks,* 2010.

[64] "TelosB Implementation of Elliptic Curve Cryptography over Primary Field," WM-CS Technical Report, 2005.

[65] H. Yan and Z. Shi, "Studying software implementations of elliptic curve cryptography," in *Third International Conference on Information Technology: New Generations (ITNG 2006), p. 78-83*, 2006.

[66] O. Ugus, D. Westhoff, R. Laue, A. Shoufan, and S.A. Huss, "Optimized Implementation of Elliptic Curve based Additive Homomorphic Encryption for Wireless Sensor Networks," in *Workshop on Embedded Systems Security (WESS 2007), p. 11–16*, 2007.

[67] A. Liu and P. Ning, "TinyECC: A configurable library for elliptic curve cryptography in wireless sensor networks," in *Proc. 7th International Conference on Information Processing in Sensor Networks (IPSN 2008), p. 245–256*, 2008.

[68] C. Lederer, R. Mader, M. Koschuch, J. Großschädl, A. Szekely and S. Tillich, "Energy-Efficient Implementation of ECDH Key Exchange for Wireless Sensor Networks," in *in WISTP 2009*.

[69] S. Khajuria and H. Tange, "Implementation of diffie-Hellman key exchange on wireless sensor using elliptic curve cryptography," in *1st International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace and Electronic Systems Technology (Wireless VITAE '09), p. 772–776*, May 2009.

[70] D. F. Aranha, R. Dahab, J. López and L. B. Oliveira, "Efficient implementation of elliptic curve cryptography in wireless sensors," *Advances in Mathematics of Communications, vol 4, issue 2, p. 169 - 187,* 2010.

[71] F. Morain and J. Olivos, "Speeding up the computations on an elliptic curve using addition-subtraction chains," *Theoretical Informatics and Applications,* 24, p. 531–543, 1990.

[72] S. C. Shantz, From euclid's gcd to montgomery multiplication to the great divide, Sun Microsystems Laboratories TR-2001-95, June 2001.

[73] "Dragongate Technologies Limited, "jBorZoi 0.9," http://dragongate-technologies.com/products.html," August 2003. [Online].

[74] J. L´opez and R. Dahab, "High-Speed Software Multiplication in F2m," Institute of Computing, Sate University of Campinas, S˜ao Paulo, Brazil, Tech. Rep., May 2000.

[75] D. Hankerson, J. L. Hemandez and A. Menezes, "Software Implementation of Elliptic Curve Cryptography over Binary Fields," *LNCS, vol. 1965, 2001.*

[76] N. Koblitz, " CM-Curves with Good Cryptographic Properties," in *11th Annual International Cryptology Conference on Advances in Cryptology*, 1991.

[77] J. A. Solinas, "An Improved Algorithm for Arithmetic on a Family of Elliptic Curves," in *17th Annual International Cryptology Conference on Advances in Cryptology*, 1997.

[78] E. K. Reddy, "Elliptic Curve Cryptosystems and Side-channel Attacks," *International Journal of Network Security,* vol.12, issue.3, p.151-158, May 2011.

[79] G. d. Meulenaer, . F. Gosset, F.-X. Standaert and O. Pereira, "On the energy cost of communication and cryptography in wireless sensor networks," in *Proceedings of the 4th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WIMOB 2008)*, 2008.

[80] J. Großsch¨adl and s. E. Sava, "Instruction set extensions for fast arithmetic in finite fields GF(p) and GF(2m)," *In Cryptographic Hardware and Embedded Systems — CHES 2004,* vol. 3156 of LNCS, p. 133–147, 2004.

[81] M. Hedabou, P. Pinel and L. Bénéteau, "A comb method to render ECC resistant against Side Channel Attacks," IACR Cryptology ePrint Archive 2004, p. 342, 2004.

[82] D. E. Knuth, The Art of Computer Programming , Volume 2: Seminumerical Algorithms, San Francisco: Addison-Wesley, 1998.

[83] ""http://bwrc.eecs.berkeley.edu/Classes/icdesign/ee141_f99/Notes/lecture3.pdf."," 2003. [Online].

[84] P. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems," in *Advances in Cryptology, Proc. CRYPTO '96*, N. Koblitz, ed., p. 104-113, 1996.

[85] P. Kocher, J. Jaffe and B. Jun, "Differential power analysis," in *Proc. Adv. Cryptology – CRYPTO'99*, Santa Barbara, CA, vol. 1666, p. 388–397, 1999.

[86] J. S. Coron, "Resistance against differential power analysis for elliptic curve cryptosystems," in *Cryptographic Hardware and Embedded Systems – CHES 1999*, Worcester, MA: Springer, vol. 1717, p. 292–302, 1999.

[87] L. Goubin, "A refined power-analysis attack on elliptic curve cryptosystems," in *Proceedings of PKC 2003, LNCS 2567*, p. 199-211. Springer Berlin / Heidelberg, 2003.

[88] T. Akishita and T. Takagi, "Zero-value register attack on elliptic curve cryptosystem," IEICE Transactions, 88-A(1): p. 132–139, 2005.

[89] P. Fouque and F. Valette, "The doubling attack– why upwards is better than downwards," in *Proc. CHES'03*, vol. 2779, p. 269–280, 2003.

[90] S. M. Yen, L. C. Ko, S. J. Moon and J. C. Ha, "Relative doubling attack against montgomery ladder," in *Proc. ICISC'05*, vol. 3935, p. 117–128, 2006.

[91] S. Chari, J. R. Rao and P. Rohatgi, "Template Attacks," *in Cryptographic Hardware and Embedded Systems, CHES, ser. LNCS,* vol. 2523, p. 13–28, 2002.

[92] P. Fouque, D. R´eal, F. Valette and M. Drissi, "The Carry Leakage on the Randomized Exponent Countermeasure," *in Cryptographic Hardware and Embedded Systems - CHES, ser. LNCS,* vol. 5154. Springer, p. 198 – 213, 2008.

[93] E. Brier and M. Joye, "Weierstraß elliptic curves and side-channel attacks," in *David Naccache and Pascal Paillier (Eds.), Public Key Cryptography, vol. 2274 of LNCS*, p. 335 – 345. Springer, Berlin / Heidelberg, 2002.

[94] P. Montgomery, "Speeding up the Pollard and elliptic curve methods of factorization," *Mathematics of Computation,* vol. 48, issue. 177, p. 243 – 264, 1987.

[95] M. Joye and J. Quisquater, "Hessian elliptic curves and side-channel attacks," *Cryptographic Hardware and Embedded Systems CHES 2001, LNCS 2162,* Springer-Verlag, p.402 – 410, 2001.

[96] O. Billet and M. Joye, "The Jacobi model of an elliptic curve and side-channel analysis," *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes 2003, LNCS 2643,* Springer- Verlag, p.34 – 42, 2003.

[97] W. Keke, L. Huiun, Z. Dingju and Y. Fengqi, "Efficient Solution to Secure ECC Against Side-channel Attacks," 20 (CJE-3): p. 471 - 475, 2011.

[98] É. Brier, I. Déchène and M. Joye, "Unified PADDition formulæ for elliptic curve cryptosystems," *In Embedded Cryptographic Hardware: Methodologies & Architectures.,* Nova Science Publishers, 2004.

[99] T. F. Al-Somani and A. A. Amin, "High Performance Elliptic Curve Scalar Multiplication with Resistance against Power Analysis Attacks," *Journal of Applied Sciences,* vol. 8 (24), p. 4587 - 4594, 2008.

[100] B. Chevallier-Mames, M. Ciet and M. Joye, "Low cost solutions for preventing simple side-channel analysis: Side channel atomicity," *IEEE Trans. Computers,* 53(6): p. 760 – 768, 2004.

[101] P. Longa, *Accelerating the Scalar Multiplication on Elliptic Curve Cryptosystems over Prime Fields.,* PhD thesis, School of Information Technology and Engineering, University of Ottawa, 2007.

[102] C. Giraud and V. Verneuil, "Atomicity Improvement for Elliptic Curve Scalar Multiplication," CARDIS 2010: 80441.

[103] D. Bernstein and T. Lange, "Faster Addition and Doubling on Elliptic Curves," *Advances in Cryptology - ASIACRYPT, K. Kurosawa (ed.),* vol. 4833 of LNCS, p. 29-50, Springer, 2007.

[104] S. Ghosh, D. Mukhopadhyay and D. R. Chowdhury, "Petrel: Power and Timing Attack Resistant Elliptic Curve Scalar Multiplier Based on Programmable GF(p) Arithmetic Unit," *IEEE Trans. on Circuits and Systems 58-I(8),* p. 1798-1812, 2011.

[105] M. Joye and C. Tymen, "Protections against differential analysis for elliptic curve cryptography," *In: [cKKNP01] Cryptographic Hardware and Embedded Systems – CHES 2001, LNCS,* Vol. 2162, p. 377.

[106] M. Ciet and M. Joye, "(Virtually) Free Randomization Techniques for Elliptic Curve Cryptography," *in Information and Communications Security (ICICS2006), LNCS 2836,* Springer, 2003, p. 348–359.

[107] D. Naccache, N. P. Smart and J. Stern, "Projective Coordinates Leak," *In: Advances in Cryptology - EuroCrypt 2004, LNCS,* Vol. 3027, p. 257–267. Springer, Berlin / Heidelberg, 2004.

[108] T. Akishita and T. Takagi, "Zero-Value Point Attacks on Elliptic Curve Cryptosystem," vol. 2851, p. 218 – 233, 2003.

[109] G. d. Meulenaer and F.-X. Standaert, "Stealthy Compromise of Wireless Sensor Nodes with Power Analysis Attacks," MOBILIGHT, p. 229-242, 2010.

[110] H. Houssain, M. Badra and T. F. Al Somani, "Hardware Implementations of Elliptic Curve Cryptography in Wireless Sensor Networks," in *Proc. 6th International Conf. on Internet Technology and Secured Transactions (ICITST 2011)*, Abu Dhabi, UAE, p. 1 - 6, Dec 2011.

[111] N. Koblitz, A. Menezes and S. Vanstone, "The State of Elliptic Curve Cryptography," *Designs, Codes and Cryptography,* pp. Vol. 19, Issue. 2–3, p. 173–193, 2000.

[112] J. L. Massey and J. K. Omura, "Computational method and apparatus for finite field arithmetic". US Patent No. 4587627 1986.

[113] C. C. Wang, T. Truong, H. M. Shao, L. J. Deutsch, J. K. Omura and . I. S. Reed, "VLSI architectures for computing multiplications and inverses in GF(2m)," *IEEE Trans. Comput.,* Vol. 34, Issue. 8, p. 709-716. 1985.

[114] T. F. Al-Somani and A. Amin, "Hardware implementations of GF(2m) arithmetic using normal basis," *J. Appl. Sci,* Vol. 6, Issue 6, p. 1362–1372, 2006.

[115] T. Itoh and S. Tsujii, "A fast algorithm for computing multiplicative inverses in GF(2m) using normal bases," *Info. Comput.,* Vol. 78, Issue.3, p. 171–177, 1988.

[116] C. Lim and P. Lee, "More Flexibility Exponentiation with Precomputation," *Advances in Cryptology - Crypto'94,* LNCS 839, p. 95–107, 1994.

[117] R. Kling, "Intel Mote: An Enhanced Sensor Network Node," in *International Workshop on Advanced Sensors, Structural Health Monitoring and Smart Structures at Keio University*, Tokyo, Japan, 2003.

[118] J. L´opez and R. Dahab, "An Overview of Elliptic Curve Cryptography," Institute of Computing, Sate University of Campinas, S˜ao Paulo, Brazil, Tech. Rep., May 2000.

[119] C. Hartung, J. Balasalle and R. Han, "Node compromise in sensor networks: The need for secure systems," Technical Report CUCS-990-05, Department of Computer Science, University of Colorado at Boulder, Jaunary 2005.

[120] IEEE P1363, "Standard Specifications for Public-Key Cryptography," 2000.

[121] National Institute of Standards and Technology, Recommended Elliptic Curves for Federal Government Use, in the appendix of FIPS 186-2.

[122] Standards for Efficient Cryptography Group (SECG), Specification of Standards for Efficient Cryptography.