國立交通大學

電子工程學系　電子研究所碩士班

碩 士 論 文

抵抗能量攻擊法的雙域橢圓曲線密碼運算單元之設計與實現

Design and Implementation of a Dual-Field Elliptic Curve Cryptographic Processor with Power Analysis Countermeasures

學生：陳耀琳

指導教授：李鎮宜 教授

中華民國九十九年七月

抵抗能量攻擊法的雙域橢圓曲線密碼運算單元之設計與實現

Design and Implementation of a Dual-Field Elliptic Curve Cryptographic Processor with Power Analysis Countermeasures

研 究 生：陳耀琳　　Student：Yao-Lin Chen

指導教授：李鎮宜教授　　Advisor：Chen-Yi Lee

國 立 交 通 大 學

電子工程學系 電子研究所 碩士班

碩 士 論 文

A Thesis
Submitted to Department of Electronics Engineering & Institute Electronics
College of Electrical and Computer Engineering
National Chiao Tung University
In Partial Fulfillment of the Requirements
for the Degree of
Master of Science
in

Electronics Engineering
July 2009
Hsinchu, Taiwan, Republic of China

中華民國九十九年七月

抵抗能量攻擊法的雙域橢圓曲線密碼運算單元之設計與實現

學生：陳耀琳　　　　　　　　　　　　　　　　　　指導教授：李鎮宜

國立交通大學電子工程學系 電子研究所碩士班

摘　　　要

在這篇論文中，我們提出了一個可支援雙域有限域運算以及可支援任意橢圓曲線運算的雙域橢圓曲線密碼運算單元。透過我們提出的通用演算法，這個運算單元的執行週期數大幅的降低。藉由我們提出的面積共用方法以及梯子選擇法，我們 160 位元以及 256 位元的雙域橢圓曲線密碼運算單元的面積在聯電 90 奈米製程下只須 0.29mm^2 和 0.45mm^2。此外，運算單元的操作面積也可以透過我們提出的指數判定器以及資料路徑分離法可大幅的提升。我們也提出一個可以對抗能量攻擊法的雙域橢圓曲線密碼運算單元。透過我們提出的通用亂數演算法，我們面積的損失僅僅 8.4%。

Design and Implementation of a Dual-Field Elliptic Curve Cryptographic Processor with Power Analysis Countermeasures

student：Yao-Lin Chen Advisors：Chen-Yi Lee

Department of Electronics Engineering & Institute of Electronics
National Chiao Tung University

ABSTRACT

In this thesis, we propose a high-performance dual-field elliptic curve cryptographic processor (DECP) architecture that can support all finite field operations and elliptic curve (EC) functions with arbitrary field and curve. Based on our proposed fast unified division algorithm, the operation cycles can be significantly reduced. Compared with previous works using high radix multiplication in projective coordinate, our 160-bit and 256-bit DECPs can achieve competitive performance in terms of execution cycles with only 0.29mm^2 and 0.45mm^2 silicon area in UMC 90nm CMOS technology by exploiting hardware sharing and ladder selection techniques. In addition, the operating frequency in prime field and binary field can be increased due to the proposed data-path separation and degree checker. To resist power analysis attack, we propose a DECP with power analysis countermeasures architecture based on the proposed unified random algorithms with only 8.4% area overhead.

誌　　　　謝

研究生活，剎那間的消逝。兩年中，常和實驗室的大家一起研究到早晨、一起唱歌、一起談心。在 Si2 和 STAR 遇到許多很棒的人，是我一輩子想珍藏的。嘴砲的義澤、很娘聲音很高的明諭、老到掉渣又很寶的浩民、很照顧大家的均宸、貼心又窩心的建辰、一直把咩的佳龍、給我很多方向以及建議的 STAR 領導人柏均、常常關心我的義閔、幫我改英文的人偉、好夥伴勇志、很熱心的曜哥、很八卦的欣儒、人很好的如宏、嘴巴很機車但心地不錯的靜瑜、投球很猛的廷聿、幫我改機的歐陽、說話不算話的昱帆、我的女朋友憲平、嘴巴功力真是一流的智翔、給我很多幫助的小馬、很黑但人很 nice 的書餘、又壯又很厲害的偉豪、衝浪少年芳年、大發慈悲給我研究費的好媽媽子菁和伶霞、看起來很靦腆的易蓁、比我小就當助理的桂綸鎂立瑜、問不到我名字又被我叫純美的婷美、講話很斯文的美玲。遇到他們真的是我一輩子的福氣，真的很謝謝他們。一路上，交大的老師們給我許多的指導和照顧。指導老師李鎮宜老師每次在會議中，都會給我許多大方向的建議，讓我對我的研究更加有目標、STAR 的指導老師張錫嘉老師，對我們都很照顧，關心我們的生活以及研究、系上老師江蕙如老師，在研究方向上給我許多鼓勵，透過她的推薦，讓我在研究所考試得到很棒的成績，老師人真的很好很好。

最後，我要感謝我的家人。謝謝他們一直以來的鼓勵，從幼稚園、小學、國中、高中、大學、研究所，無怨無悔的照顧，真的很謝謝他們。在大學求學的這段時間，很謝謝佩蓉的陪伴，在我快樂時候陪我分享喜悅，在我失意的時候陪我分享悲傷。也很謝謝一路上陪我成長的朋友們，幼幼社的麻吉們以及交大電子系的大家。真的很謝謝你們，因為有你們，才有今天的陳耀琳，謝謝你們，謝謝。

Contents

List of Figures

List of Tables

List of Notations

Table 1: Mathematical symbols

Definition	Description	Definition	Description
$E()$	Elliptic curve	L, K	Fields
P, Q, G	Points in EC	O	A point at infinity
p	Prime	m	Field size
n	Maximum field size	i, j, l	Variables
k	Key	A	Affine
J	Jacobian	J_M	Modified Jacobian
J_C	Chudnovsky Jacobian	L_P	Lopez projective
L_M	L_P with Montgomery ladder algorithm	L_{MC}	L_M with common Z projective coordinate
X, Y	Integer	U, V, R, S, T, D	Operands
r	Random number	$H()$	Hash function
x	x-coordinate of a point	y	y-coordinate of a point
M	Message	N	Curve order
w	Word-length	B,C	$l \times 1$ matrices

Table 2: Abbreviations

Definition	Description	Definition	Description
DECP	Dual-field ECC processor	EC	Elliptic curve
ECSM	EC scalar point multiplication	MD	Modular division
MMD	Montgomery MD	MI	Modular inversion
MMI	Montgomery MI	MM	Modular multiplication
MMM	Montgomery MM	MA	Modular addition
MS	Modular subtraction	PA	Power analysis
ECDBL	EC point doubling	ECADD	EC point addition
ECSUB	EC point subtraction	D	Division
M	Multiplication	SQ	Squaring
A	Addition	S	Subtraction
UD	Unified division	R2-UD	Radix-2 UD
R4-UD	Radix-4 UD	UM	Unified multiplication
R2-UM	Radix-2 UM	R4-UM	Radix-4 UM
GFAU	Galois field arithmetic unit	R2-GFAU	Radix-2 GFAU
R4-GFAU	Radix-4 GFAU	R2-DECP	Radix-2 DECP
R4-DECP	Radix-4 DECP	PAC	PA countermeasures
R2-GFAUPAC	R2-GFAU with PAC	R2-DECPAC	R2-DECP with PAC
FLT-UMI	Unified MI based on FLT	FLT-UMMI	Unified MMI based on FLT
K-UI	Kaliski's unified inversion	T-UMD	Takagi's unified MD
L-UD	Liu's UD	MSQ	Modular SQ
FT-UD	UD based on T-UMD	w-UM/D	Word-based multiplication /division architecture
AT	Gates $\times$ time		

Chapter 1

Introduction

1.1 Elliptic Curve Cryptography

To ensure the data security of network communication, public-key encryption algorithms have been widely adopted. Elliptic curve cryptography (ECC) [1–6] can provide the same security level as the Rivest, Shamir and Adleman (RSA) [7] algorithm with much reduced key-size. In ECC scheme, the major operation is the elliptic curve point scalar multiplication (ECSM). To reduce the execution time in software implementation [8], several accelerating hardware processors are proposed. Many ECC designs have been published over specified finite field, either $GF(p)$ [9–11] or $GF(2^m)$ [12–22]. The designs over $GF(2^m)$ usually target at area constrainted applications such as smart cards or RFID cards due to the carry-free propagation and fixed irreducible polynomial in specific ECs.

However, to support higher security level, both arbitrary key-size and field operations are essentially required. Some dual-field ECC processors (DECP), in which the coordinate is transformed to the projective coordinate to avoid inversion operations in ECSM, have been proposed up to now [23–25]. Satoh and Takano [23] exploit a $r \times r$-bit multipliers to speed up the ECSM in the Jacobian's projective coordinate, and Lai and Huang [24, 25] present a parallel architecture based on [23] to enhance the throughput. However, operations in the projective coordinate are more complicated than that in the affine coordinate, and the inversion is still needed in coordinate transformation before and after the ECSM in projective coordinate. To reduce the execution cycles of ECSM and coordinate transformation, the size r of multipliers or the number of parallel units are increased, which usually results in high hardware cost.

The traditional approach of inversion operation is based on the Fermat's little theorem (FLT) [26]. It can be realized by repeating squaring and multiplication operation but results in longer execution time [23–25]. In 1995, Kaliski [27] proposed a unified inversion algorithm to accomplish the inversion operations. Several later algorithms and architectures are based on this algorithm [28–34]. Furthermore, to directly reduce the execution cycles of ECSM in affine coordinate, many architectures and algorithms [35–37] are based on Takagi's modular division (MD) algorithm [38].

To solve the overhead of inversion and the following multiplication operation in ECSM and coordinate transformation, we propose a fast unified division algorithm supporting Montgomery modular division (MMD) and MD operations over dual fields. Note that the "unified" means the algorithm is able to handle dual-field operations. In addition, we apply hardware sharing method, data-path separation, and degree checker into a DECP to reduce the hardware cost and increase the operating frequency. Our DECP supports ECSM and finite field operations with arbitrary curves and parameters over dual fields. The implementation result shows our DECP outperforms relative works in functionality, hardware efficiency, execution time, and power consumption.

1.2 Power Analysis

Physical attacks on cryptographic devices using side-channel information are attracting extensive attention [39–41]. In order to reveal secret parameters, the power dissipation, electromagnetic radiation, or operating times (i.e. timing attack [42]) as correlated to internal operation are measured. Simple power analysis (SPA) [43] and differential power analysis (DPA) [44, 45] are known as basic and powerful side-channel attacks, which have been discussed in several literatures [4, 46–51].

To resist the power analysis attack, we use the masking techniques to randomize the operating data. We propose a unified random division and a unified random multiplication algorithm to make the total random numbers equal 2^m, where m is the field length. Compared with the proposed unified algorithms, the implementation of the unified random algorithms increases little hardware cost to resist DPA attack. In addition, the SPA attack is resisted by well known double-and-add/sub-always method.

1.3 Organization

In this thesis, we propose the unified algorithms and ECC architectures to support the operations in elliptic curve cryptography, and propose the unified random algorithms and architectures to resist power analysis in cryptographic processor. In Chapter 2, the preliminaries of ECC cryptosystem is introduced. In Chapter 3, we propose the unified algorithms to accomplish the division and multiplication operations. In Chapter 4, we propose the Galois field arithmetic units and dual-field ECC processors to support finite field operations, EC functions, or power analysis countermeasures. In Chapter 5, we show the implementation results of our proposed architectures. In the last Chapter, we give a brief conclusion and discussion.

Chapter 2

Preliminary of Elliptic Curve Cryptography Cryptosystem

Elliptic curve cryptography cryptosystem, which is based on the arithmetic on elliptic curves (ECs) over finite field, has been widely adopted in recent years. The arithmetic on ECs is the EC point scalar multiplication (ECSM) which is computed in Galois field. In addition, the applications of ECC are the EC data en/decryption and EC based protocols which are composed of ECSM, random number generator, hash function, and Galois field arithmetic. The relationship is given in a hierarchical organization as shown in Figure 2.1.

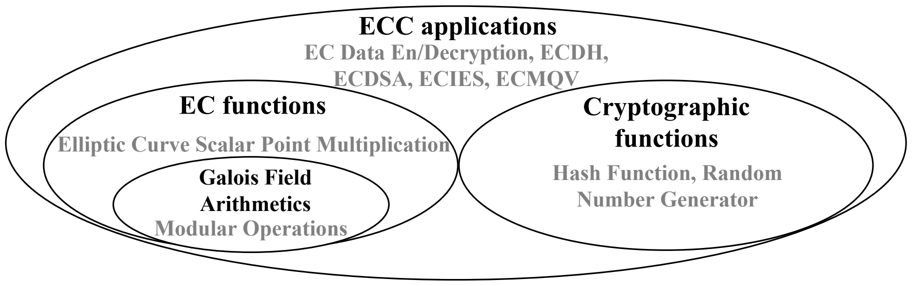

ECC applications
EC Data En/Decryption, ECDH, ECDSA, ECIES, ECMQV
EC functions
Elliptic Curve Scalar Point Multiplication
Galois Field Arithmetics
Modular Operations
Cryptographic functions
Hash Function, Random Number Generator

Figure 2.1: Hierarchical organization of EC protocol

The ECSM operation consists of four parts, which are operating field, coordinate, point multiplication method, and Galois field arithmetic, shown in Table 2.2. Furthermore, power analysis on ECC is discussed nowadays. By measuring power traces of ECC devices, the secret informations can be extracted. We will introduce some power analysis methods and countermeasures to attack or resist them, respectively.

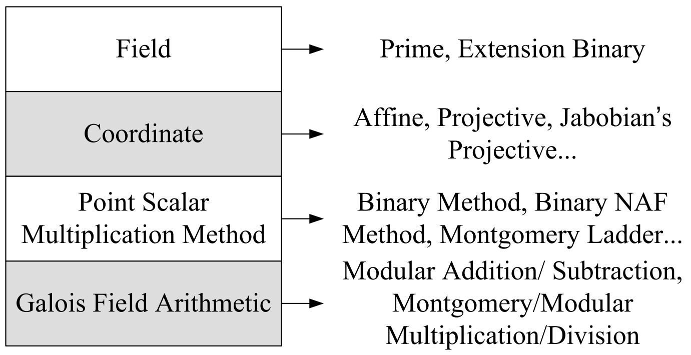

Field
Prime, Extension Binary
Coordinate
Affine, Projective, Jabobian's Projective...
Point Scalar Multiplication Method
Binary Method, Binary NAF Method, Montgomery Ladder...
Galois Field Arithmetic
Modular Addition/ Subtraction, Montgomery/Modular Multiplication/Division

Figure 2.2: The component of ECSM operation

2.1 Point Addition and Doubling over Finite Fields

If L and K are two fields, $L \supseteq K$, the general elliptic curve E defined over K is an equation of the form (also called Weierstrass equation)

$$E(L) : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \tag{2.1}$$

where a_1,..., $a_6 \in K$ are constants and $(x, y) \in L \times L$ is the set of points along with a point O at infinity. The point O at infinity is defined as the identity element, i.e., $P + O = O + P = P$ for all $P \in E(L)$. Note that if $P = (x, y)$, then the negation of P, denoted by $-P$, is defined as $(x, -a_1x - a_3 - y)$.

However, it is more practical to specify what kind of finite field of set x, y, a_1, ..., and a_6 belong to in equation 2.1. Most of ECC designs are implemented over $GF(p)$ or $GF(2^m)$, where p is a prime integer and m is the field size determined by the key length. An equation form of the non-singular EC $E(GF(p))$ is given by

$$E(GF(p)) : y^2 = x^3 + ax + b \pmod p, \tag{2.2}$$

where a, $b \in GF(p)$ and $4a^3 + 27b^2 \neq 0 \pmod p$. For two distinct points $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ with $P \neq \pm Q$, the formulas of the EC point addition (ECADD) $P + Q = (x_3, y_3)$ and EC point doubling (ECDBL) in affine coordinates are shown in Table 2.1. The ECDBL is that the point P adds itself, i.e., $2P = (x_3, y_3)$, but $P \neq -P$. For the existence of inverses, it is easy to return the value as O because $P + (-P) = O$.

For $GF(2^m)$, the non-singular EC is an equation of this form

$$E(GF(2^m)) : y^2 + xy = x^3 + ax^2 + b \pmod p, \tag{2.3}$$

where a, $b \in GF(2^m)$ and $b \neq 0 \pmod p$ and p is an irreducible polynomial of degree m. Table 2.1 lists all the formulas of the point addition over $GF(2^m)$ and the point doubling in affine coordinates. Note that the EC point subtraction (ECSUB) $Q-P$ with $P = (x, y)$ can be computed by ECADD $Q+(-P)$, where the coordinates of $-P$ are given by $(x, -y)$ over $GF(p)$ and $(x, x + y)$ over $GF(2^m)$.

Table 2.1: ECDBL and ECADD.

Field	Doubling$(x_3,y_3)=2(x_1,y_1)$	Addition$(x_3,y_3)=(x_1,y_1)+(x_2,y_2)$
$GF(p)$	$\lambda = \frac{3x_1^2+a}{2y_1} \pmod p$	$\lambda = \frac{y_2-y_1}{x_2-x_1} \pmod p$
	$x_3 = \lambda^2 - 2x_1 \pmod p$	$x_3 = \lambda^2 - x_1 - x_2 \pmod p$
	$y_3 = \lambda(x_1 - x_3) - y_1 \pmod p$	$y_3 = \lambda(x_1 - x_3) - y_1 \pmod p$
$GF(2^m)$	$\lambda = x_1 + \frac{y_1}{x_1} \pmod p$	$\lambda = \frac{y_2+y_1}{x_2+x_1} \pmod p$
	$x_3 = \lambda^2 + \lambda + a \pmod p$	$x_3 = \lambda^2 + \lambda + x_1 + x_2 + a \pmod p$
	$y_3 = \lambda(x_1 + x_3) + x_3 + y_1 \pmod p$	$y_3 = \lambda(x_1 + x_3) + x_3 + y_1 \pmod p$

2.2 Analysis of Point Addition and Doubling in Different Coordinates

Traditionally, ECSM is operated in affine coordinate. To avoid the inversion operation which is more expensive than multiplication, many coordinates has been proposed, such as Jacobian projective coordinate [1] and López projective coordinate [52], etc. Table 2.2 and 2.3 show the analysis of EC point doubling/addition in different coordinates [53]. The execution cycle of the ECSM in affine coordinate is dominated by the division operation. To outperform other coordinates, the execution cycle of division must be less than $5.2Mc$, where Mc means the cycle of multiplication.

Table 2.2: ECDBL and ECADD for various coordinates over $GF(p)$.

ECDBL over $GF(p)$				
Function	$A \leftarrow 2A$	$J \leftarrow 2J$	$J_M \leftarrow 2J_M$	$J_C \leftarrow 2J_C$
Major Operation	1D+1M+2SQ	4M+6SQ	4M+4SQ	5M+6SQ
ECADD over $GF(p)$				
Function	$A \leftarrow A + A$	$J \leftarrow J + A$	$J_M \leftarrow J_M + A$	$J_C \leftarrow J_C + A$
Major Operation	1D+1M+1SQ	12M+4SQ	13M+6SQ	11M+3SQ

Table 2.3: ECDBL and ECADD for various coordinates over $GF(2^m)$.

ECDBL over $GF(2^m)$				
Function	$A \leftarrow 2A$	$L_P \leftarrow 2L_P$	$L_M \leftarrow 2L_M$	$L_{MC} \leftarrow 2L_{MC}$
Major Operation	1D+1M+1SQ	7M+5SQ[a]	4M+1SQ[a]	5M+1SQ[a]
ECADD over $GF(2^m)$				
Function	$A \leftarrow A + A$	$L_P \leftarrow L_P + A$	$L_M \leftarrow L_M + L_M$	$L_{MC} \leftarrow L_{MC} + L_{MC}$
Major Operation	1D+1M+1SQ	10M+4SQ[b]	2M+4SQ[b]	2M+3SQ[b]

[a]: Including the operation cycles of extra step.

2.3 Elliptic Curve Point Scalar Multiplication Methods

Intuitively, the ECSM operation, i.e. $kP = P + P ... + P$, requires $(k-1)$ iterative point addition to accomplish. To reduce the execution cycle, many methods such as binary method and window method were proposed [1]. Considering the hardware efficiency and operation cycles, we adopt the binary Non-adjacent form (NAF) method shown in algorithm 2.1. Prior to this algorithm, the secret key must be transformed to the NAF form. The details of the NAF is illustrated in [2].

Algorithm 2.1. ***(Binary NAF method for point multiplication.)***

Input: *P and k, where $P \in E(L)$, k is an integer with NAF form and $k_{m-1} = 1$.*

Output: *$Q = [k]P$.*

```
1. $Q = P$
2. for $i = m - 2$ to $1$ by $-1$ do
3.     $Q = [2]Q$
4.     if $k_i = 1$, then
5.         $Q = Q + P$
6.     else if $k_i = -1$, then
7.         $Q = Q - P$
8.     end if
9. end for
```

2.4 Galois Field Arithmetic

Galois field arithmetic is very important not only in ECSM operations but also in EC protocols. Eight different modular operations over finite field are commonly used and details of these modular operations and their abbreviations are listed in Table 2.4. The division and multiplication are more complicated than addition and subtraction, so many approaches have been proposed to enhance the performance of division and multiplication.

Table 2.4: Galois field arithmetic.

Operations	
Modular addition	$MA(X, Y) = X + Y \pmod{p}$
Modular subtraction	$MS(X, Y) = X - Y \pmod{p}$
Modular multiplication	$MM(X, Y) = X \cdot Y \pmod{p}$
Montgomery modular multiplication	$MMM(X, Y) = X \cdot Y \cdot 2^{-m} \pmod{p}$
Modular inversion	$MI(X) = X^{-1} \pmod{p}$
Montgomery modular inversion	$MMI(X) = X^{-1} \cdot 2^{-m} \pmod{p}$
Modular division	$MD(X, Y) = X \cdot Y^{-1} \pmod{p}$
Montgomery modular division	$MMD(X, Y) = X \cdot Y^{-1} \cdot 2^{m} \pmod{p}$

2.4.1 Unified Multiplication Algorithms

Unified Modular Multiplication Algorithm

The unified MM computes $R \equiv X \cdot Y \pmod p$, where $0 \le X, Y < p$ or $0 \le deg(X), deg(Y) < deg(p)$ over prime field or binary field, respectively. MM operation can be realized in two methods, left-to-right MM and right-to-left MM. The implementation of these two methods are similar. Algorithm 2.2 shows the left-to-right unified MM (UMM) algorithm. Note that the addition/subtraction operations mean XOR gates in binary field, and the operation "$2\cdot$" represents "$x\cdot$" in binary field.

Algorithm 2.2. ***(Left-to-right unified modular multiplication.)***
Input: *X, Y, and p, where X, Y are n-bit integer over $GF(p)$ or $GF(2^m)$ and p is the prime or irreducible polynomial.*
Output: $R \equiv X \cdot Y \pmod p$.

```
1. $R = 0$, $S = Y$
2. for $i$ from $0$ to $m-1$ by $+1$ do
3.     $R = (R + X_i \cdot S) \pmod p$
4.     $S = 2 \cdot S \pmod p$
5. endfor
```

High Radix Unified Montgomery Modular Multiplication Algorithm

The well known Montgomery multiplication algorithm, proposed by P. L. Montgomery [54], is commonly used to compute the modular multiplication without trial division. The concept of the MMM is to turn the MM into iterative operations with both addition and logic level shifting. Hence the MMM is quite appropriate for software or hardware implementation. The additional overhead is the pre-/post-processing stages of the domain transformation for the input/output. In the pre-processing stage, the data is transformed from integer domain, $X \cdot 2^0$, to Montgomery domain, $X \cdot 2^m$. And in the post-processing stage, the data is transformed back to integer domain.

Algorithm 2.3 [54] computes $R \equiv X \cdot Y \cdot 2^{-m} \pmod p$, where $0 \le X, Y < p$ or $0 \le deg(X), deg(Y) < deg(p)$ over prime field or binary field, respectively. Because the X and Y are in the Montgomery domain, the MMM computes $X \cdot Y \cdot 2^{-m} \pmod p$ to make the output R still in the Montgomery domain. In order to use $n \cdot r$-bit multiplier, an n-bit number needs to be divided into l r-bit blocks (i.e., $n = l \cdot r$). The operand X can be represented by r-bit words X_i as $X = X_{l-1} \cdot 2^{r(l-1)} + ... + X_1 \cdot 2^r + X_0$. The T_i operation is used to make the least significant word of accumulated operand R be zero. The proof is shown as follows:

$$R + X_i \cdot Y + T_i \cdot p \pmod{2^r} = R + X_i \cdot Y + (R_0 + X_i \cdot Y) \cdot q \cdot p \pmod{2^r} = 0 \quad (2.4)$$

Therefore, the division operation is easily achieved by shifting r bit. Moreover, the operand R may excess p during the MMM iteration over prime field, so a reduction step after the last iteration is required. On the other hand, since $deg(R) \geq m$ would not occur in binary field operation, the recovery step is not required. Traditionally, the r is commonly set to 1 for low-cost design. The algorithm is shown in algorithm 2.4.

There is a variety of hardware architectures to implement the MMM. Both the systolic architecture [55] and the word-level architecture [56] exploit the pipelining techniques to shorten the critical path. Beside, Satoh and Takano proposed double loop method [23] to apply into MMM operation. Compared with architectures with $n \times r$-bit multipliers, Satoh and Takano's work just needs one $r \times r$-bit multiplier to improve the MMM operation. Besides, the operation cycle increases from $m+1$ and m cycles to $2l^2+4l+1$ and $2l^2+3l+1$ over prime field and binary field [23], respectively.

Algorithm 2.3. *(**Radix-r unified Montgomery multiplication algorithm.**)*
Input: *X, Y, q, and p, where X, Y are n-bit integer over $GF(p)$ or $GF(2^m)$, $q = -p^{-1} \pmod{2^r}$, and p is the prime or irreducible polynomial.*
Output: $R \equiv X \cdot Y \cdot 2^{-m} \pmod{p}$.

```
1. $R = 0$
2. for $i = 0$ to $l - 1$ by $+1$ do
3.     $T = (R_0 + X_i \cdot Y) \cdot q \pmod{2^r}$
4.     $R = \frac{R_i + X_i \cdot Y + T \cdot p}{2^r}$
5. endfor
6. if $R \geq p$ and the operating field is prime, then: $R = R - p$
```

Algorithm 2.4. ***(Radix-2 unified Montgomery multiplication algorithm.)***

Input: *X, Y, and p, where X, Y are in $GF(p)$ or $GF(2^m)$ and p is the prime or irreducible polynomial.*

Output: *$R \equiv X \cdot Y \cdot 2^{-m} \pmod p$.*

```
1. $R = 0$
2. for $i$ from $0$ to $m-1$ by $+1$ do
3.     $T = (R + X_i \cdot Y)$
4.     $R = \frac{(T + T_0 \cdot p)}{2}$
5. endfor
6. if $R \geq p$ and the operating field is prime, then: $R = R - p$
```

2.4.2 Unified Inversion and Division Algorithms

Unified Inversion Algorithms based on Fermat's Little Theorem

Based on Fermat's Little Theorem (FLT), $X^{p-1} = 1 \pmod p$, the inversion operation is easily achieved by $X^{-1} = X^{p-2} \pmod p$. FLT is commonly used in projective operation, because of low cost and high integration with radix-r MMM. Algorithm 2.5 shows the unified MMI algorithm based on FLT (FLT-UMMI), and the execution cycle of FLT-UMMI is about $m^2 \sim 2m^2$, where m is the execution cycle of MMM. Besides, the FLT can also be used to accomplish MI operation shown in algorithm 2.6.

Algorithm 2.5. ***(Unified MMI algorithm based on FLT.)***

Input: *$X \cdot 2^m$ and p, where X is in $GF(p)$ or $GF(2^m)$ and p is the prime or irreducible polynomial.*

Output: *$R \equiv X^{-1} \cdot 2^m \pmod p$.*

```
1. if the operating field is prime, then: $T = p - 2$
2. else: $T = 2^m - 2$
3. $R = X \cdot 2^m$
4. for $i$ from $m-2$ to $0$ by $-1$ do
5.     $R = MMM(R, R)$
6.     if $T_i = 1$, then: $R = MMM(R, X \cdot 2^m)$
7. endfor
```

Algorithm 2.6. ***(Unified MI algorithm based on FLT (FLT-UMI).)***

Input: *X and p, where X is in $GF(p)$ or $GF(2^m)$ and p is the prime or irreducible polynomial.*

Output: $R \equiv X^{-1} \pmod p$.

```
1. if the operating field is prime, then: $T = p - 2$
2. else: $T = 2^m - 2$
3. $R = X$
4. for $i$ from $m - 2$ to $0$ by $-1$ do
5.     $R = R \cdot R \pmod p$
6.     if $T_i = 1$, then: $R = R \cdot X \pmod p$
7. endfor
```

Kaliski's Unified Inversion Algorithm

Algorithm 2.7 shows the unified inversion algorithm proposed by Kalisiki (K-UI) [27]. This algorithm supports the MI and MMI operation over dual fields. This algorithm calculates $R = X^{-1} \cdot 2^m \pmod p$, where the operand R is defined as the Montgomery representation of modular inverse, m is the bit-length of p, and X ($\neq 0$) be the elements of the field. Similarly, the $R = X^{-1} \pmod p$ can also be obtained from this algorithm, where the operand R is defined as the integer representation of modular inverse. The inversion is computed by intertwining the procedure for finding the modular quotient with that for calculating $gcd(X, p)$. The algorithm requires four operands, U, V, R, and S. U and V are used for calculating $gcd(X, p)$ and the operands R and S are used for calculating modular inverse. The operands U and V are initialized to Y and p, respectively, and the properties shown in Table 2.5 are applied iteratively to calculate $gcd(X, p)$. For example, U can be replaced by $U/2$ according to the property $gcd(U, V) = gcd(U/2, V)$, when U is even. In addition, R and S are initialized to the values of X and 0, respectively. Besides, the corresponding R, S operations are determined by the following invariants:

$$\begin{cases} X \cdot R \equiv -U \cdot 2^i \pmod p \\ X \cdot S \equiv V \cdot 2^i \pmod p \end{cases} \tag{2.5}$$

During the phase 1 operation which means the operating steps are 2∼8, the domain value i is increased by 1 every cycle. Table 2.5 shows the detail operations of U, V, R, and S

based on the properties and invariants. For instance, if U is even, the algorithm changes value U to $U/2$ and the value i is increased to $i+1$ for obeying the equivalence 2.5. To increase the value i to $i+1$ in the second equivalence, the operand S must be multiplied by 2.

At the end of the while loop, the value U and V would be 1 and 0 which means $R = -X^{-1} \cdot 2^i \pmod{p}$ with $m \leq i \leq 2m$ and $S = 0 \pmod{p}$. Then in phase 2 which contains step 10 to 14, the value of i is reduced to m. This can be done by either iteratively halving modulo p or multiplication modulo p [28]. After phase 2, the value R would be $-X^{-1} \cdot 2^m \pmod{p}$ or $-X^{-1} \pmod{p}$, and in the prime field R should be reduced to within the range $[0, p-1]$ by $p - R$ operation. Finally, it has been proved that the cycle number needed to complete MMD and MD operations are $m \sim 3m$ and $2m \sim 4m$ if X and p are co-prime [27].

Algorithm 2.7. *(Kaliski's unified inversion algorithm.)*

Input: *X, and p, where X are in $GF(p)$ or $GF(2^m)$ and p is the prime integer or irreducible polynomial.*

Output: $\begin{cases} \text{If operation is MMI, then } R \equiv X^{-1} \cdot 2^m \pmod{p}. \\ \text{If operation is MI, then } R \equiv X^{-1} \pmod{p}. \end{cases}$

```
1.  $U = p$, $V = X$, $R = 0$, $S = 1$, $i = 0$
2.  while $V > 0$ do
3.      if $U$ is even, then: $U = \frac{U}{2}$, $S = 2 \cdot S$
4.      else if $V$ is even, then: $V = \frac{V}{2}$, $R = 2 \cdot R$
5.      else if $U - V > 0$, then: $U = \frac{U-V}{2}$, $R = R + S$, $S = 2 \cdot S$
6.      else if $V - U \geq 0$, then: $V = \frac{V-U}{2}$, $S = S + R$, $R = 2 \cdot R$
7.      $i = i + 1$
8.  endwhile
9.  if $R \geq P$, then: $R = R - p$
10. while $i \neq \begin{cases} m \\ 0 \end{cases}$ do
11.     if $R$ is even: $R = R/2$
12.     else: $R = (R + p)/2$
13.     $i = i - 1$
14. endwhile
15. if the operating field is prime, then: $R = p - R$
```

Takagi's Unified Modular Division Algorithm

In 1998, Takagi proposed a unified modular division algorithm (T-UMD) [38] based on the extended binary GCD algorithm [57]. The algorithm calculates $S = X \cdot Y^{-1} \pmod{p}$ by finding the value $gcd(Y, p)$ and the corresponding modular quotient, where X and Y are the elements of the field with odd prime (or irreducible polynomial) p.

This algorithm requires four operands, U, V, R, and S. U and V are used for calculating $gcd(Y, p)$ and the operands R and S are used for calculating modular quotient. The operands U and V are initialized to Y and p, respectively, and the properties shown in Table 2.6 are applied repeatedly to calculate $gcd(Y, p)$. The operands R and S are initialized to the values of X and 0, respectively. Then, the same operations that are

Table 2.5: The properties of Kaliski's unified inversion algorithm.

Initial	$X \cdot (0) \equiv -(p) \cdot 2^0 \pmod p$ $X \cdot (1) \equiv (X) \cdot 2^0 \pmod p$
End of MMI operation	$X \cdot (-X^{-1} \cdot 2^m) \equiv -(1) \cdot 2^m \pmod p$ $X \cdot (0) \equiv (0) \cdot 2^\rho \pmod p$
End of MI operation	$X \cdot (-X^{-1}) \equiv -(1) \pmod p$ $X \cdot (0) \equiv (0) \cdot 2^\rho \pmod p$

	Properties	Invariants
U is even	$gcd(U,V) = gcd(\frac{U}{2}, V)$	$X \cdot R \equiv -U/2 \cdot 2^{i+1} \pmod p$ $X \cdot 2 \cdot S \equiv V \cdot 2^{i+1} \pmod p$
V is even	$gcd(U,V) = gcd(U, \frac{V}{2})$	$X \cdot 2 \cdot R \equiv -U \cdot 2^{i+1} \pmod p$ $X \cdot S \equiv V/2 \cdot 2^{i+1} \pmod p$
$U > V$	$gcd(U,V) = gcd(\frac{U-V}{2}, V)$	$X \cdot \frac{R+S}{2} \equiv -\frac{U-V}{2} \cdot 2^{i+1} \pmod p$ $X \cdot 2 \cdot S \equiv V \cdot 2^{i+1} \pmod p$
$U \leq V$	$gcd(U,V) = gcd(U, \frac{V-U}{2})$	$X \cdot 2 \cdot R \equiv -U \cdot 2^{i+1} \pmod p$ $X \cdot \frac{R+S}{2} \equiv \frac{V-U}{2} \cdot 2^{i+1} \pmod p$
phase 2	–	$X \cdot \frac{R}{2} \equiv -(1) \cdot 2^{i-1} \pmod p$ $X \cdot (0) \equiv (0) \cdot 2^\rho \pmod p$

ρ is equal to the value i in the last iteration of phase 1.

performed to the operands U and V are applied to the operands R and S for calculating the modular quotient by reducing U and V value. Furthermore, the operands U and V are integers and are allowed to be negative. δ represents $\alpha - \beta$, where α and β are values such that 2^α and 2^β indicate the upper bounds of $|U|$ and $|V|$, respectively. The value $\delta = 0$ is introduced to represent $min(\alpha, \beta)$. For correctness, we do some modification on the condition of *while* loop in the original algorithm.

This algorithm is based on the following invariants:

$$\begin{cases} X^{-1} \cdot Y \cdot R \equiv U \pmod p \\ X^{-1} \cdot Y \cdot S \equiv V \pmod p \end{cases} \tag{2.6}$$

It can easily be shown that the equivalences always hold in Table 2.6. Since $gcd(Y,p) = 1$, the operands $U = 0$ and V is 1 or -1 in the last iteration. Hence, in the final step of algorithm, the equivalence $X^{-1} \cdot Y \cdot S = 1 \pmod p$ holds and S is equal to $X \cdot Y^{-1} \pmod p$. Moreover, the number of iterations needed to complete the algorithm is at least m and at most $2m$ cycles if Y and p are co-prime.

Algorithm 2.8. ***(Takagi's unified modular division algorithm.)***

Input: *X, Y, and p, where X, Y are in $GF(p)$ or $GF(2^m)$ and p is the prime integer or irreducible polynomial.*

Output: $S \equiv X \cdot Y^{-1} \pmod{p}$.

```
1.  $U = Y,\ V = p,\ R = X,\ S = 0,\ \delta = 0$
2.  while $U \neq 0$ do
3.      if $U$ is even, then: $U = U/2,\ R = R/2 \pmod{p},\ \delta = \delta - 1$
4.      else
5.          if $\delta < 0$, then: $swap(U, V),\ swap(R, S),\ \delta = -\delta$
6.          if $(U + V) \pmod{4} = 0$, then: $q = -1$
7.          else: $q = 1$
8.          $U = \frac{U+q\cdot V}{4},\ \delta = \delta - 1$
9.          $R = \frac{R+q\cdot S}{4} \pmod{p}$
10.     endif
11. endwhile
12. if $V = -1$, then: $S = P - S$
```

Table 2.6: The properties of Takagi's unified modular division algorithm.

Initial	$X^{-1} \cdot Y \cdot (X) \equiv (Y) \pmod{p}$ $X^{-1} \cdot Y \cdot (0) \equiv (p) \pmod{p}$
End	$X^{-1} \cdot Y \cdot (0) \equiv (0) \pmod{p}$ $X^{-1} \cdot Y \cdot (\pm X \cdot Y^{-1}) \equiv (\pm 1) \pmod{p}$

	Properties	Invariants
U is even and V is odd	$gcd(U, V) = gcd(\frac{U}{2}, V)$	$X^{-1} \cdot Y \cdot R \equiv U/2 \pmod{p}$ $X^{-1} \cdot Y \cdot S \equiv V \pmod{p}$
$U + V$ is divisible by 4	$gcd(U, V) = gcd(\frac{U+V}{4}, V)$	$X^{-1} \cdot Y \cdot \frac{R+S}{4} \equiv \frac{U+V}{4} \pmod{p}$ $X^{-1} \cdot Y \cdot S \equiv V \pmod{p}$
$U - V$ is divisible by 4	$gcd(U, V) = gcd(\frac{U+V}{4}, V)$	$X^{-1} \cdot Y \cdot \frac{R-S}{4} \equiv \frac{U-V}{4} \pmod{p}$ $X^{-1} \cdot Y \cdot S \equiv V \pmod{p}$

Liu's Unified Division Algorithm

In algorithm 2.9, the Liu's unified division algorithm (L-UD) is proposed in [31, 33]. The initial value of U, V, R, and S are set to p, Y, 0, and X, respectively, and the equivalences are shown as follows:

$$\begin{cases} X^{-1} \cdot Y \cdot R \equiv -U \cdot 2^i \pmod{p} \\ X^{-1} \cdot Y \cdot S \equiv V \cdot 2^i \pmod{p} \end{cases} \tag{2.7}$$

The execution cycle of L-UD algorithm is the same as K-UI algorithm, but it can support MMD and MD operations.

Algorithm 2.9. ***(Liu's unified division algorithm.)***

Input: *X, Y, and p, where X, Y are in $GF(p)$ or $GF(2^m)$ and p is the prime integer or irreducible polynomial.*

Output: $\begin{cases} \text{If operation is MMD, then } R \equiv X \cdot Y^{-1} \cdot 2^m \pmod{p}. \\ \text{If operation is MD, then } R \equiv X \cdot Y^{-1} \pmod{p}. \end{cases}$

```
 1. $U = p$, $V = X$, $R = 0$, $S = Y$, $i = 0$
 2. while $V > 0$ do
 3.     if $U$ is even, then: $U = \frac{U}{2}$, $S = 2 \cdot S$
 4.     else if $V$ is even, then: $V = \frac{V}{2}$, $R = 2 \cdot R$
 5.     else if $U - V > 0$, then: $U = \frac{U-V}{2}$, $R = R + S$, $S = 2 \cdot S$
 6.     else if $V - U \geq 0$, then: $V = \frac{V-U}{2}$, $S = S + R$, $R = 2 \cdot R$
 7.     if $R \geq P$, then: $R = R - p$
 8.     if $S \geq P$, then: $S = S - p$
 9.     $i = i + 1$
10. endwhile
11. while $i \neq \begin{cases} m \\ 0 \end{cases}$ do
12.     if $R$ is even: $R = R/2$
13.     else: $R = (R + p)/2$
14.     $i = i - 1$
15. endwhile
16. if the operating field is prime, then: $R = p - R$
```

2.5 Elliptic Curve Cryptographic Applications

ECC can be used to achieve data en/decryption, signature, and authentication [2, 58, 59]. Among them, the major operations are ECSM or the modular division operation.

2.5.1 Elliptic Curve Data En/Decryption

By ECSM operation, the EC data en/decryption [58] can be easy accomplished. We assume that Alice wants to send a message M to Bob. The en/decryption flow is shown in algorithm 2.10.

Algorithm 2.10. ***(Elliptic Curve Data En/Decryption.)***

1. *Bob chooses a m-bit random number k as the private key.*
2. *Bob computes $[k]P$, then send to Alice.*
3. *Alice chooses a m-bit random number r.*
4. *Alice computes $\{R, S\} = \{[r]P, M + [r]([k]P)\}$, then send to Bob.*
5. *Bob gets the message M by computing $S - [k]R$.*

2.5.2 Elliptic Curve Based Protocols

Many EC based protocols [2, 59], such as elliptic curve digital signature algorithm (ECDSA), EC Menezes-Qu-Vanstone (ECMQV), and EC Diffie-Hellman (ECDH), are used for different applications. For the ECDSA, the domain parameters are given by (H, L, E, N, P), where H is a hash function, G is a point on the curve of prime order N. Algorithm 2.11 and 2.12 show the ECDSA signing and verification, respectively. Among these two algorithms, the ECSM and MD operations are the most critical.

Algorithm 2.11. *(ECDSA signing.)*

Input: *M, and x, where M is message and k is secret key*

Output: *(R, S), where (R, S) is a signature on M.*

```
1. Choose $r \in \{1, ..., N-1\}$
2. $T = [r]P$
3. $R = x_T \pmod N$, where $x_T$ means the $x$-coordinate of point $T$
4. if $R = 0$, then: goto step 1
5. $V = H(M)$
6. $S = (V + kR)/r \pmod N$
7. if $s = 0$, then: goto step 1
```

Algorithm 2.12. *(ECDSA verification.)*

Input: *M, Y, G, and (R, S) where M is message, G is public key, and (R, S) is a signature*

Output: *OUT = Reject or Accept.*

```
1. if $R, S \notin \{1, ..., N-1\}$, then: $OUT$ = Reject
2. $V = H(M)$
3. $U_1 = V/S \pmod N$
4. $U_2 = R/S \pmod N$
5. $T = [U_1]P + [U_2]G$
6. if $R = x_T$, then: $OUT$ = Accept
7. else: $OUT$ = Reject
```

2.6 Power Analysis Attacks and Countermeasures

2.6.1 Simple Power Analysis

In most of implementations, the standard sequence of field operations in point addition differs from that in point doubling [4]. The SPA attacks use this difference to reveal the secret key value. In Figure 2.3(a) and 2.3(b), the power traces of the ECDBL and ECADD are shown, respectively. By using the difference between these two trace, the secret key can be revealed. From the power trace of the ECSM operation with secret key in Figure 2.3(c), the key value can be observed.

Several SPA countermeasure methods have been proposed, including unified equation [46], double-and-add-always method [4], and Montgomery ladder [47]. In this work, we adopt double-and-add/sub-always method to accomplish the ECSM operation, which is shown in algorithm 2.13, since the method is easier than the other.

Algorithm 2.13. ***(Double-and-add/sub-always method.)***

Input: *P and k, where k is an integer with NAF form and P is a point.*

Output: $Q = [k]P$.

```
1. $Q = P$
2. for $i$ from $m-2$ to $0$ by $-1$
3.     $Q = [2]Q$
4.     if $k_i = 0$, then: $R = Q + P$
5.     else if $k_i = 1$, then: $Q = Q + P$
6.     else: $Q = Q - P$
7. enfor
```

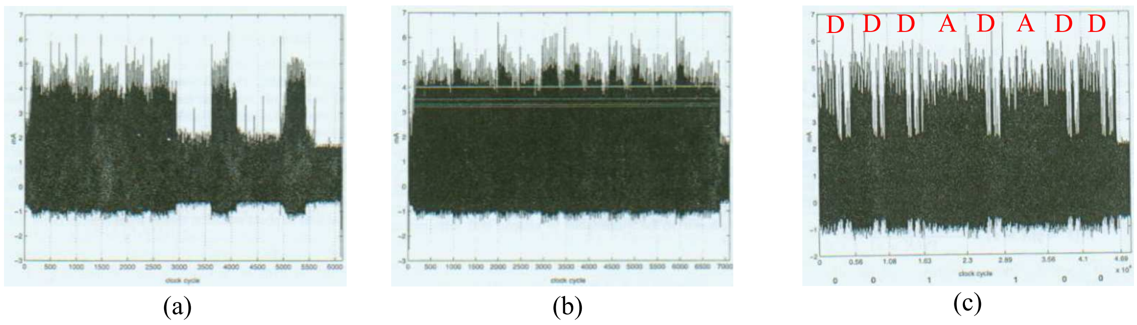


Figure 2.3: (a) A ECADD power trace. (b) A ECDBL power trace. (c) A ECSM power trace.

2.6.2 Differential Power Analysis

In the DPA attacks, an attacker records the power consumption of the cryptographic devices and analyzes the collected power traces by statistical calculation to extract the secret key. Several variations of the DPA attacks have been proposed, such as DPA attack [4], doubling attack [48], address-bit attack [49], refined power analysis [50], and zero-value point attack [51].

Figure 2.4 shows a simple DPA attack flow [33]. The DPA attack assumes that the attacker can perform the ECSM operation with different keys, EC parameters, and EC points, and has knowledge about all the implementation details of the attacked device. For a given secret power trace of ECSM, the attacker reveals the key bit-by-bit. We suppose parts of the secret key $[k_{n-1}, k_{n-2}...k_{i+1}]$ is recognized by the attacker, and the next attacked bit is k_i. Next, we input the key-value $[k_{n-1}, k_{n-2}...k_{i+1}, k_i = 0, ...]$ and $[k_{n-1}, k_{n-2}...k_{i+1}, k_i = 1, ...]$ into the device to obtain two power traces. Then, we cut the traces of $[k_i = 0]$ and $[k_i = 1]$ from the obtained traces to do further correlation with original power trace. The correlation formula is shown below:

$$\rho(B, C) = \frac{\sum_{i=1}^{l}(B_i - \bar{B})(C_i - \bar{C})}{\sqrt{\sum_{i=1}^{l}(B_i - \bar{B})^2 \sum_{i=1}^{l}(C_i - \bar{C})^2}} \tag{2.8}$$

The parameters B, C mean the $l \times 1$ matrices, and the $\rho(B, C)$ represents the correlation value of B, C, where $-1 \leq \rho \leq 1$. If the correlation between $[k_i = 0]$ and original power trace is higher, then the attacker can disclose $k_i = 0$. On the other hand, k_i would be 1 if the correlation between the trace for $[k_i = 1]$ and that for the original key is higher.

To resist DPA attack, Coron [60] proposed three methods. The Coron's first countermeasure is to randomize the private exponent, such as $k' = k + r \cdot \#E(L)$. Note that the r is a random number, and $\#E(L)$ is the curve order. In addition, the second countermeasure is to blind the base point to compute further ECSM, $Q = [k]P' - S$, where $P' = P + R$, $S = [k]R$, and R is a random point. The last countermeasure is to randomize a point in projective coordinate. The method changes the original point (x, y, z) to (rx, ry, rz) and performs ECSM in projective coordinate. The security level of a device can be enhanced by increasing the size of random r. Beside, these random methods were classified as the masking DPA countermeasures in [39]. By randomizing the intermediate values that are processed by the cryptographic device, masking method makes the power consumption of a cryptographic device independent of the intermediate values of the cryptographic algorithm to resist the DPA attack.

In [33], a simple DPA attack scheme is used to reveal the secret key from the two chips. The first chip is a 521-bit DECP and the second chip is a 521-bit DECP with PA countermeasure. Coron's first countermeasure is adopted for DPA countermeasure, since the second method is hard to implement and the third method is only adopted in the projective coordinate. Figure 2.5(a) shows the correlation coefficient trace, and we can reveal the key value “0”, “0”, “1”, and “0” because the spikes appear in the right locations. On the other hand, in Figure 2.5(b), since there have no spikes in the right locations, the secret key can’t be revealed.

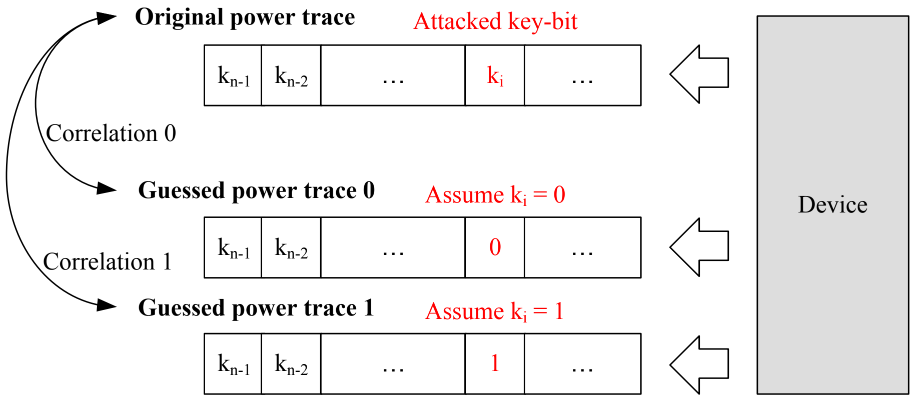


Figure 2.4: Simple DPA attack flow

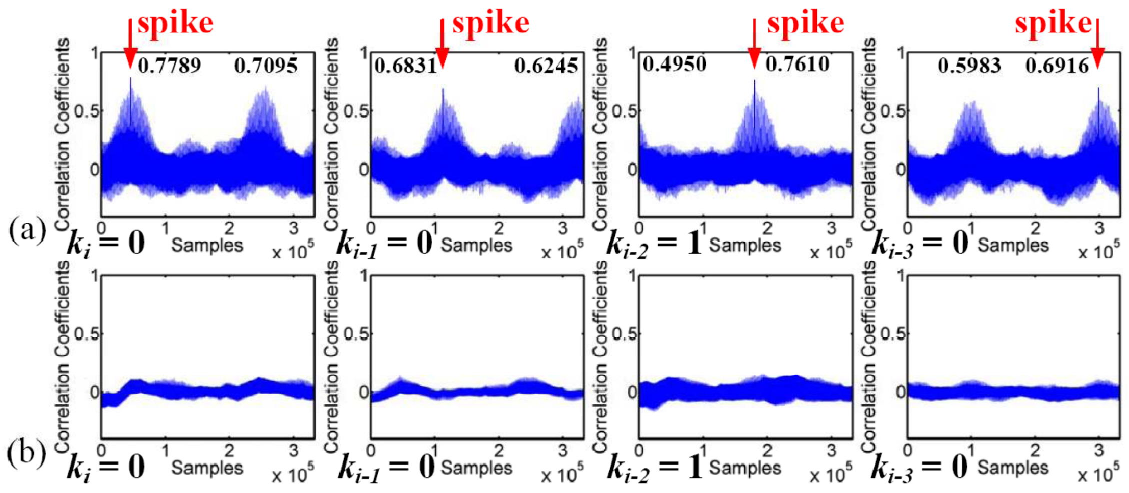


Figure 2.5: Correlation coefficients of key value,$[k_i = 0, k_{i-1} = 0, k_{i-1} = 1, k_{i-1} = 0]$, for (a) unprotected chip (b) protected chip.

Chapter 3

Proposed Unified Algorithms

In this chapter, we propose many unified algorithms to support division or multiplication operations. In addition, to resist the power analysis attack, the unified random algorithms are proposed.

3.1 Unified Division Algorithm

Traditionally, the FLT is used to achieve the inversion operation in the coordinate and domain transformation. However, the execution cycle is too huge to have the same time complexity with ECSM. Moreover, the K-UI and T-UMD need extra multiplication to achieve the MMD/MD operation. Based on K-UI, we propose a radix-2 unified division (R2-UD) algorithm and a radix-4 unified division (R4-UD) algorithm to reduce numerous execution cycles of division operation.

Table 3.1: The properties of R2-UD.

Conditions	Properties
$U \pmod 2 = 0$	$gcd(U,V) = gcd(\frac{U}{2}, V)$
$V \pmod 2 = 0$	$gcd(U,V) = gcd(U, \frac{V}{2})$
$U > V$	$gcd(U,V) = gcd(\frac{U-V}{2}, V)$
$U \leq V$	$gcd(U,V) = gcd(U, \frac{V-U}{2})$

Our proposed R2-UD is shown in algorithm 3.1. Followings are the invariant equiva-

Table 3.2: The invariant equivalences of the proposed R2-UD algorithm for MMD operation.

Initial	$Y \cdot X^{-1} \cdot (0) \equiv (p) \cdot 2^0 \pmod p$ $Y \cdot X^{-1} \cdot (X) \equiv (Y) \cdot 2^0 \pmod p$
End	$Y \cdot X^{-1} \cdot (X \cdot Y^{-1} \cdot 2^m) \equiv (1) \cdot 2^m \pmod p$ $Y \cdot X^{-1} \cdot (0) \equiv (0) \cdot 2^m \pmod p$

	$i < m$	$i \geq m$
U is even	$Y \cdot X^{-1} \cdot R \equiv \frac{U}{2} \cdot 2^{i+1} \pmod p$ $Y \cdot X^{-1} \cdot 2 \cdot S \equiv V \cdot 2^{i+1} \pmod p$	$Y \cdot X^{-1} \cdot \frac{R}{2} \equiv \frac{U}{2} \cdot 2^m \pmod p$ $Y \cdot X^{-1} \cdot S \equiv V \cdot 2^m \pmod p$
V is even	$Y \cdot X^{-1} \cdot 2 \cdot R \equiv U \cdot 2^{i+1} \pmod p$ $Y \cdot X^{-1} \cdot S \equiv \frac{V}{2} \cdot 2^{i+1} \pmod p$	$Y \cdot X^{-1} \cdot R \equiv U \cdot 2^m \pmod p$ $Y \cdot X^{-1} \cdot \frac{S}{2} \equiv \frac{V}{2} \cdot 2^m \pmod p$
$U > V$	$Y \cdot X^{-1} \cdot (R - S) \equiv \frac{U-V}{2} \cdot 2^{i+1} \pmod p$ $Y \cdot X^{-1} \cdot 2 \cdot S \equiv V \cdot 2^{i+1} \pmod p$	$Y \cdot X^{-1} \cdot \frac{R-S}{2} \equiv \frac{U-V}{2} \cdot 2^m \pmod p$ $Y \cdot X^{-1} \cdot S \equiv V \cdot 2^m \pmod p$
$U \leq V$	$Y \cdot X^{-1} \cdot 2 \cdot R \equiv U \cdot 2^{i+1} \pmod p$ $Y \cdot X^{-1} \cdot (S - R) \equiv \frac{V-U}{2} \cdot 2^{i+1} \pmod p$	$Y \cdot X^{-1} \cdot R \equiv U \cdot 2^m \pmod p$ $Y \cdot X^{-1} \cdot \frac{S-R}{2} \equiv \frac{V-U}{2} \cdot 2^m \pmod p$

lences obeyed in our proposed algorithm.

$$X^{-1} \cdot Y \cdot R \equiv U \cdot 2^i \pmod p \tag{3.1}$$

$$X^{-1} \cdot Y \cdot S \equiv V \cdot 2^i \pmod p \tag{3.2}$$

For the initialization of R2-UD, the operands U, V, R, and S are set to the values p, Y, 0, and X, respectively. The operations of UV in algorithm 3.1 are based on the binary greatest common divisor (GCD) operation, which is proven in Table 3.1. Note that the addition/subtraction can be implemented by XOR gates in binary field operation and the "$\frac{1}{2}$" represents "$\frac{1}{x}$". In each iteration, the valid value of operands U or V is reduced by 1 bits. Because of $gcd(U, V) = gcd(p, Y) = 1$, the values of U and V are 1 and 0 after the last iteration. And the values of R and S are $X \cdot Y^{-1} \cdot 2^i \pmod p$ and $0 \pmod p$, respectively. In addition, the operands R and S are transformed into Montgomery or integer domain due to the MMD or MD operation, respectively. In the beginning of MMD operation, the RS operations are executed to add i by 1. For instance, if the operating step is 19, the equivalences are $X^{-1} \cdot Y \cdot (2 \cdot R) \equiv U \cdot 2^{i+1} \pmod p$ and $X^{-1} \cdot Y \cdot (S - R) \equiv (\frac{V-U}{2}) \cdot 2^{i+1} \pmod p$. When $i \geq m$, the operations keep the operands R and S in Montgomery domain. In the end of this algorithm, the value R is equal to $X \cdot Y^{-1} \cdot 2^m \pmod p$. On the other

Table 3.3: The invariant equivalences of the proposed R2-UD algorithm for MD operation.

Initial	$Y \cdot X^{-1} \cdot (0) \equiv (p) \pmod p$
	$Y \cdot X^{-1} \cdot (X) \equiv (Y) \pmod p$
End	$Y \cdot X^{-1} \cdot (X \cdot Y^{-1}) \equiv (1) \pmod p$
	$Y \cdot X^{-1} \cdot (0) \equiv (0) \pmod p$
	$i = 0$
U is even	$Y \cdot X^{-1} \cdot \frac{R}{2} \equiv \frac{U}{2} \pmod p$
	$Y \cdot X^{-1} \cdot S \equiv V \pmod p$
V is even	$Y \cdot X^{-1} \cdot R \equiv U \pmod p$
	$Y \cdot X^{-1} \cdot \frac{S}{2} \equiv \frac{V}{2} \cdot 2^m \pmod p$
$U > V$	$Y \cdot X^{-1} \cdot \frac{R-S}{2} \equiv (U - V) \pmod p$
	$Y \cdot X^{-1} \cdot S \equiv V \pmod p$
$U \leq V$	$Y \cdot X^{-1} \cdot R \equiv U \pmod p$
	$Y \cdot X^{-1} \cdot \frac{S-R}{2} \equiv (V - U) \pmod p$

hand, if the operation is set to MD, the data are operated in the integer domain. Then the output value of R is equal to $X \cdot Y^{-1} \pmod p$. The detail explanation of the invariant equivalences during the R2-UD is shown in tables 3.2 and 3.3. Besides, Table 3.4 gives an example.

Algorithm 3.2 and 3.3 show the proposed R4-UD and the properties of R4-UD are shown in tabel 3.5. The execution cycle of R4-UD is $0.56m$ $\sim$$1.12m$, since there has a condition reducing just one bit with the probability $\frac{1}{8}$. Consequently, the execution cycle

Table 3.4: The example of the proposed R2-UD.

	MMD					MD				
iteration	U	V	R	S	i	U	V	R	S	i
1	13	7	0	9	0	13	7	0	9	0
2	3	7	4	5	1	3	7	2	9	0
3	3	2	8	1	2	3	2	2	10	0
4	3	1	3	1	3	3	1	2	5	0
5	1	1	2	2	4	1	1	5	5	0
6	1	0	2	0	4	1	0	5	0	0

Table 3.5: The properties of the proposed R4-UD.

$U \pmod 4$	$V \pmod 4$	Properties
0	0, 1, 2, or 3	$gcd(U,V) = gcd(\frac{U}{4}, V)$
1, 2, or 3	0	$gcd(U,V) = gcd(U, \frac{V}{4})$
equivalence		$gcd(U,V) = gcd(\frac{U-V}{4}, V) = gcd(U, \frac{V-U}{4})$
2	1 or 3	$gcd(U,V) = gcd(\frac{\frac{U}{2}-V}{2}, V) = gcd(\frac{U}{2}, \frac{V-\frac{U}{2}}{2})$
1 or 3	2	$gcd(U,V) = gcd(\frac{U-\frac{V}{2}}{2}, \frac{V}{2}) = gcd(U, \frac{\frac{V}{2}-U}{2})$
other		$gcd(U,V) = gcd(\frac{U-V}{2}, V) = gcd(U, \frac{V-U}{2})$

is $7/8(m/2 \sim m)+1/8(m \sim 2m) = 0.56m \sim 1.12m$. The execution cycle of this algorithm is about half the cycles of algorithm 3.1, but the hardware cost is almost two times larger because the total number of RS operations increases from 8 to 21. Consequently, it is an area-time trade-off design.

Compared with previous works, such as FLT-UMMI [26], K-UI [27], and T-UMD [38], our algorithm has fewer execution cycle in division operation without using extra multiplication operation and pre-computed value, 2^{2m}. The MMD/MD operating steps and performance analysis is shown in tables 3.6 and 3.7. Moreover, the MMD and MD operations are used many times in ECSM and EC protocols [1], so our design can significantly outperform previous works.

Algorithm 3.1. *(**Proposed R2-UD algorithm.**)*

Input: *X, Y and p, where X, Y in $GF(p)$ or $GF(2^m)$ and p is the prime integer or irreducible polynomial.*

Output: $\begin{cases} \text{If operation is MMD, then } R \equiv X \cdot Y^{-1} \cdot 2^m \pmod{p}. \\ \text{If operation is MD, then } R \equiv X \cdot Y^{-1} \pmod{p}. \end{cases}$

```
 1. $U = p$, $V = Y$, $R = 0$, $S = X$
 2. if Operation is MMD, then: $j = 0$
 3. else: $j = 1$
 4. while $V > 0$ do
 5.     if $U$ is even, then
 6.         $U = \frac{U}{2}$
 7.         if $i < m$ and $j = 0$, then: $S = 2 \cdot S \pmod{p}$, $i = i + 1$
 8.         else: $R = \frac{R}{2} \pmod{p}$
 9.     else if $V$ is even, then
10.         $V = \frac{V}{2}$
11.         if $i < m$ and $j = 0$, then: $R = 2 \cdot R \pmod{p}$, $i = i + 1$
12.         else: $S = \frac{S}{2} \pmod{p}$
13.     else if $U > V$, then
14.         $U = \frac{U-V}{2}$
15.         if $i < m$ and $j = 0$, then: $R = R - S \pmod{p}$, $S = 2 \cdot S \pmod{p}$, $i = i + 1$
16.         else: $R = \frac{R-S}{2} \pmod{p}$
17.     else
18.         $V = \frac{V-U}{2}$
19.         if $i < m$ and $j = 0$, then: $S = S - R \pmod{p}$, $R = 2 \cdot R \pmod{p}$, $i = i + 1$
20.         else $S = \frac{S-R}{2} \pmod{p}$
21.     endif
22. endwhile
```

Algorithm 3.2. *(Proposed R4-UD algorithm.)*

Input: *X, Y and p, where X, Y in $GF(p)$ or $GF(2^m)$ and p is the prime integer or irreducible polynomial.*

Output: $\begin{cases} \textit{If operation is MMD, then } R \equiv X \cdot Y^{-1} \cdot 2^m \pmod{p}. \\ \textit{If operation is MD, then } R \equiv X \cdot Y^{-1} \pmod{p}. \end{cases}$

```
 1. $U = p$, $V = Y$, $R = 0$, $S = X$
 2. while ($V > 0$) do
 3. $c = U \pmod 4$, $d = V \pmod 4$, $j = i$
 4.     if $i = m - 1$, then: $ctrl = 1$, $i = i + 1$
 5.     else if $c = 0$, then: $U = \frac{U}{4}$, $ctrl = 2$, $i = i + 2$
 6.     else if $d = 0$, then: $V = \frac{V}{4}$, $ctrl = 3$, $i = i + 2$
 7.     else if $c = d$, then
 8.         $i = i + 2$
 9.         if $U > V$, then: $U = \frac{U-V}{4}$, $ctrl = 4$
10.         else: $V = \frac{V-U}{4}$, $ctrl = 5$
11.     else if $c = 2$, then
12.         $i = i + 2$
13.         if $\frac{U}{2} > V$, then: $U = \frac{\frac{U}{2}-V}{2}$, $ctrl = 6$
14.         else: $V = \frac{V-\frac{U}{2}}{2}$, $U = \frac{U}{2}$, $ctrl = 8$
15.     else if $d = 2$, then
16.         $i = i + 2$
17.         if $U > \frac{V}{2}$, then: $U = \frac{U-\frac{V}{2}}{2}$, $V = \frac{V}{2}$, $ctrl = 9$
18.         else: $V = \frac{\frac{V}{2}-U}{2}$, $ctrl = 7$
19.     else
20.         $i = i + 1$
21.         if $U > V$, then: $U = \frac{U-V}{2}$, $ctrl = 10$
22.         else: $V = \frac{V-U}{2}$, $ctrl = 11$
23.     endif
24.     $(R, S) = OP\_RS(R, S, ctrl, j, p)$.
25. endwhile
```

Algorithm 3.3. *(**Operations for operands** R **and** S **(OP_RS).**)*

Input: *R, S, ctrl, j and p, where p is an m-bit prime or irreducible poly..*

Output: *R, S*

```
 1. if $j < m$ and operation is MMD, then
 2.     switch ctrl
 3.         case 1: $R = 2 \cdot R \pmod{p}$, $S = 2 \cdot S \pmod{p}$
 4.         case 2: $R = 4 \cdot R \pmod{p}$
 5.         case 3: $S = 4 \cdot S \pmod{p}$
 6.         case 4: $R = R - S \pmod{p}$, $S = 4 \cdot S \pmod{p}$
 7.         case 5: $S = S - R \pmod{p}$, $R = 4 \cdot R \pmod{p}$
 8.         case 6: $R = R - 2 \cdot S \pmod{p}$, $S = 4 \cdot S \pmod{p}$
 9.         case 7: $S = S - 2 \cdot R \pmod{p}$, $R = 4 \cdot R \pmod{p}$
10.         case 8: $R = 2 \cdot R - S \pmod{p}$, $S = 4 \cdot S \pmod{p}$
11.         case 9: $S = 2 \cdot S - R \pmod{p}$, $R = 4 \cdot R \pmod{p}$
12.         case 10: $R = R - S \pmod{p}$, $S = 2 \cdot S \pmod{p}$
13.         case 11: $S = S - R \pmod{p}$, $R = 2 \cdot R \pmod{p}$
14.     endswitch
15. else
16.     switch ctrl
17.         case 2: $R = \frac{R}{4} \pmod{p}$
18.         case 3: $S = \frac{S}{4} \pmod{p}$
19.         case 4: $R = \frac{R-S}{4} \pmod{p}$
20.         case 5: $S = \frac{S-R}{4} \pmod{p}$
21.         case 6: $R = \frac{\frac{R}{2}-S}{2} \pmod{p}$
22.         case 7: $S = \frac{\frac{S}{2}-R}{2} \pmod{p}$
23.         case 8: $R = \frac{R-\frac{S}{2}}{2} \pmod{p}$, $S = \frac{S}{2} \pmod{p}$
24.         case 9: $S = \frac{S-\frac{R}{2}}{2} \pmod{p}$, $R = \frac{R}{2} \pmod{p}$
25.         case 10: $R = \frac{R-S}{2} \pmod{p}$
26.         case 11: $S = \frac{S-R}{2} \pmod{p}$
27.     endswitch
28. endif
```

3.2 Unified Multiplication Algorithm

The traditional method of MMM (i.e. algorithm 2.4) over $GF(p)$ needs one step to recover the value to the range $[0, p-1]$, but the step is not required in $GF(2^m)$ operation. We remove the step by confirming the accumulated operand R always satisfies within the range $[0, p-1]$. The overhead is one subtraction. By combining with the proposed UD, the extra units can be shared. In addition, after removing the recover step, the steps of MMM over prime field are similar with that over binary field. Since the proposed UD and MMM are bit-level algorithm, we can combine MM with them to enhance the functionality. Consequently, we propose a radix-2 unified multiplication (R2-UM) algorithm and a radix-4 unified multiplication (R4-UM) algorithm shown in algorithms 3.4 and 3.5.

Algorithm 3.4. ***(Proposed R2-UM.)***

Input: *X, Y and p, where X, Y are in $GF(p)$ or $GF(2^m)$ and p is the prime or irreducible polynomial.*

Output: $\begin{cases} \text{If Operation is MMM, then } R \equiv X \cdot Y \cdot 2^m \pmod{p}. \\ \text{If Operation is MM, then } R \equiv X \cdot Y \pmod{p}. \end{cases}$

```
1. $R = 0$, $S = Y$
2. for $i$ from $0$ to $m-1$ by $+1$ do
3.     $R = R + X_i \cdot S \pmod{p}$
4.     if operation is MMM, then: $R = \frac{R}{2} \pmod{p}$
5.     else: $S = 2 \cdot S \pmod{p}$
6. endfor
```

Algorithm 3.5. ***(Proposed R4-UM.)***

Input: *X, Y and p, where X, Y are in $GF(p)$ or $GF(2^m)$ and p is the prime or irreducible polynomial.*

Output: $\begin{cases} \textit{If Operation is MMM, then } R \equiv X \cdot Y \cdot 2^m \pmod{p}. \\ \textit{If Operation is MM, then } R \equiv X \cdot Y \pmod{p}. \end{cases}$

```
1. $R = 0$, $S = Y$
2. for $i$ from $0$ to $\frac{m-1}{2}$ by $+1$ do
3.      if $m \pmod{2} = 1$ and $i = \frac{m-1}{2}$, then: $R = R + X_{2\cdot i} \cdot S \pmod{p}$
4.      else: $R = R + X_{2\cdot i} \cdot S + X_{2\cdot i+1} \cdot 2 \cdot S \pmod{p}$
5.      if operation is MMM, $m \pmod{2} = 1$ and $i = \frac{m-1}{2}$ , then: $R = \frac{R}{2} \pmod{p}$
6.      else if operation is MMM, then: $R = \frac{R}{4} \pmod{p}$
7.      else: $S = 4 \cdot S \pmod{p}$
8. endfor
```

3.3 Unified Random Algorithms

Based on the masking method, we propose two unified random algorithms to resist DPA attack. We use a m-bit random number r when the modular operations are executed each time. The algorithms have two modes to execute the data operations. The first mode would increase the domain value, and the other would not increase the value. The modes are changed depending on the one in r. When r_i is equal to one, the algorithm would execute the first mode, where the value i means the iteration number. Otherwise, the second mode is executed. Consequently, the intermediate value is randomized by the random value r, so the DPA attack can be resisted.

3.3.1 Unified Random Division Algorithm

In algorithm 3.6, the unified random division algorithm (URD) is proposed to support the division operation in random domain, 2^λ, where $0 \le \lambda \le m$. Note that the value λ is equal to the total number of ones in r. The algorithm computes $X \cdot Y^{-1} \cdot 2^\lambda \pmod{p}$, and have two modes, MMD and MD, to achieve the random domain operation. If $r_i = 1$, the mode is set to MMD to increase the domain value of the operands R and S by 1.

Otherwise, the mode is set to MD which does not increase the domain value. At the end of the algorithm, the output data is in the random domain 2^λ.

Algorithm 3.6. ***(Proposed URD algorithm.)***

Input: *X, Y, r, and p, where X, Y in $GF(p)$ or $GF(2^m)$, r is a random number, and p is the prime integer or irreducible polynomial.*

Output: $R = X \cdot Y^{-1} \cdot 2^\lambda$.

```
1.  $U = p$, $V = Y$, $R = 0$, $S = X$, $\lambda = 0$
2.  while $V > 0$ do
3.      if $U$ is even, then
4.          $U = \frac{U}{2}$
5.          if $r_i = 1$, then: $S = 2 \cdot S \pmod{p}$, $\lambda = \lambda + 1$
6.          else: $R = \frac{R}{2} \pmod{p}$
7.      else if $V$ is even, then
8.          $V = \frac{V}{2}$
9.          if $r_i = 1$, then: $R = 2 \cdot R \pmod{p}$, $\lambda = \lambda + 1$
10.         else: $S = \frac{S}{2} \pmod{p}$
11.     else if $U > V$, then
12.         $U = \frac{U-V}{2}$,
13.         if $r_i = 1$, then
14.             $R = R - S \pmod{p}$, $S = 2 \cdot S \pmod{p}$, $\lambda = \lambda + 1$
15.         else: $R = \frac{R-S}{2} \pmod{p}$
16.         endif
17.     else
18.         $V = \frac{V-U}{2}$
19.         if $r_i = 1$, then
20.             $R = 2 \cdot R \pmod{p}$, $S = S - R \pmod{p}$, $\lambda = \lambda + 1$
21.         else $S = \frac{S-R}{2} \pmod{p}$
22.         endif
23.     endif
24. endwhile
```

3.3.2 Unified Random Multiplication Algorithm

Algorithm 3.7 shows the proposed unified random multiplication algorithm (URM) which combines MMM and MM operations to support the multiplication in random domain, 2^{λ}, where $0 \leq \lambda \leq m$. The step 4 is the MMM mode which increases the domain value by 1. And the step 5 is the MM mode which does not change the domain value. Consequently, the value λ is equal to the total number of ones in r.

Algorithm 3.7. ***(Proposed unified random multiplication algorithm.)***

Input: *X, Y, p and r, where X, Y are in $GF(p)$ or $GF(2^m)$, r is a random number, and p is the prime or irreducible polynomial.*

Output: $R \equiv X \cdot Y \cdot 2^{-\lambda} \pmod{p}$.

```
1. $R = 0$, $S = Y$, $\lambda = 0$
2. for $i$ from $0$ to $m-1$ by $+1$ do
3.     $R = R + X_i \cdot S \pmod{p}$
4.     if $r_i = 1$, then: $R = \frac{R}{2} \pmod{p}$, $\lambda = \lambda + 1$
5.     else: $S = 2 \cdot S \pmod{p}$
6. endfor
```

Table 3.6: Operating steps of MMD/MD operations over dual fields of previous works.

	FLT	
Step	MMD	MD
1	FLT-UMMI$(Y \cdot 2^m) = Y^{-1} \cdot 2^m \pmod p$	FLT-UMI$(Y) = Y^{-1} \pmod p$
2	MMM$(Y^{-1} \cdot 2^m, X \cdot 2^m)$ $= X \cdot Y^{-1} \cdot 2^m \pmod p$	MM(Y^{-1}, X) $= X \cdot Y^{-1} \pmod p$
	K-UI	
Step	MMD	MD
1	K-UI$(Y \cdot 2^m) = Y^{-1} \pmod p$	K-UI$(Y) = Y^{-1} \cdot 2^m \pmod p$
2	MMM$(Y^{-1}, X \cdot 2^m)$ $= X \cdot Y^{-1} \pmod p$	MMM$(Y^{-1} \cdot 2^m, X)$ $= X \cdot Y^{-1} \pmod p$
3	MMM$(X \cdot Y^{-1}, 2^{2m})$ $= X \cdot Y^{-1} \cdot 2^m \pmod p$	—
	T-UMD	
Step	MMD	MD
1	T-UMD$(X \cdot 2^m, Y \cdot 2^m)$ $= X \cdot Y^{-1} \pmod p$	T-UMD$(X, Y) = X \cdot Y^{-1} \pmod p$ $= X \cdot Y^{-1} \pmod p$
2	MMM$(X \cdot Y^{-1}, 2^{2m})$ $= X \cdot Y^{-1} \cdot 2^m \pmod p$	—

Table 3.7: Performance Analysis of Division Operation.

	Operation	[26]	[27]	[38]	[31]	R2-UD	R4-UD
Execution Time	MMD	m^2+m$\sim$2m^2+m	3m$\sim$5m	2m$\sim$3m	m$\sim$3m	m$\sim$2m	0.56m$\sim$1.12m
	MD	m^2+m$\sim$2m^2+m	2m$\sim$4m	m$\sim$2m	2m$\sim$4m	m$\sim$2m	0.56m$\sim$1.12m
Multiplication Operation/ Pre-computation		Yes/No	Yes/Yes	Yes/Yes	No/No	No/No	No/No

Chapter 4

Proposed Architectures

In this chapter, we propose a DECP supporting all the arithmetic functions on elliptic curves over dual fields. Besides, to resist the power analysis attacks, such as SPA, and DPA attacks, we propose a DECP with power analysis countermeasures (DECPAC).

4.1 Galois Field Arithmetic Unit

In this section, the radix-2 Galois field arithmetic unit (R2-GFAU) and radix-4 Galois field arithmetic unit (R4-GFAU) are proposed based on the proposed R2-UD/M and R4-UD/M, respectively. These two architectures support all finite field operations, such as MA, MS, MMM, MM, MD, and MMD over dual fields. To increase the operating frequency and reduce the hardware cost, many techniques had been presented. Figures 4.1 and 4.2 are the architectures of R2-GFAU and R4-GFAU. Since the architectures are very similar, only the details of R4-GFAU are illustrated in the following.

In Figure 4.2, the R4-GFAU is controlled by inputs to accomplish the dual-field modular operations. In R4-GFAU, the UV data-path is used to execute the UV operations, and the R, S data-path are used to finish the R, S operations. The following shows an example about the data flow of R4-GFAU. Initially, we set the operation is MMD over $GF(p)$. During the operations, the UV data-path cell compares the two operands (U',V')= ($\frac{U}{2}$,V) when the operating step is 11. Suppose the decision results are $\frac{U}{2} > V$ and $i < m$, and then the (R',S',P',P'') is set to ($2 \cdot R$,$-S$,$+p$,$-p$) in R data-path and (S'',P''') is ($4 \cdot S$,$-p$) in S data-path to compute the next R, S values. The result of R

is selected from $2R - S + p$, $2R - S$, $2R - S - p$, and $2R - S - 2p$ in R data-path by deciding whose range is within $[0, p-1]$. And the result of S is selected from $4S$, $4S - p$, $4S - 2p$, and $4S - 3p$ in S data-path.

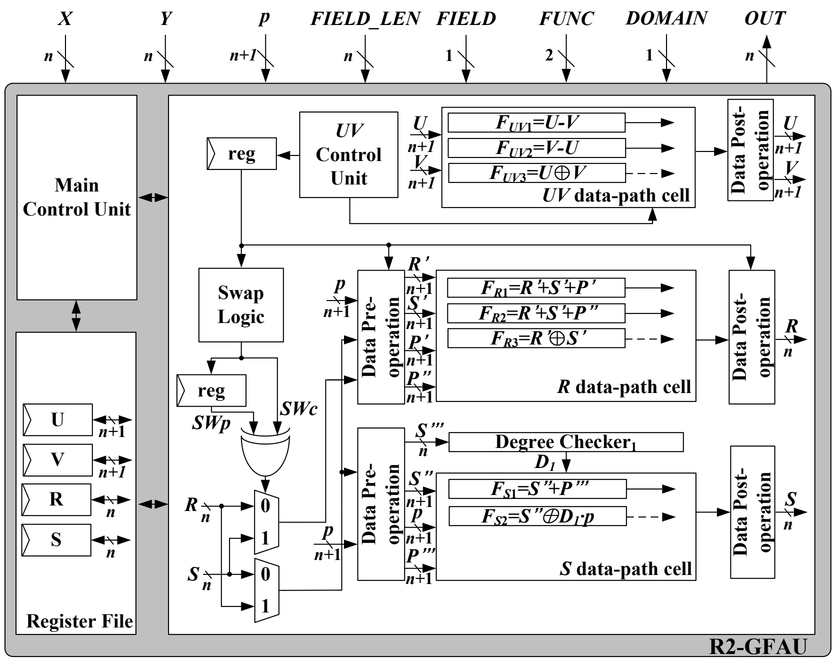


Figure 4.1: Architecture of R2-GFAU.

4.1.1 Data-path Separation

As the critical path in the proposed R4-UD is from UV data-path to R, S data-path, a data-path separation method is presented to separate it. The control signal from the UV data-path is stored and sent to RS data-path in the next cycle. Although this approach increases one cycle, the critical path can be reduced from two adders to one adder without considering the data pre-/post-operation. Figure 4.3 shows the detailed flow of the proposed method. Firstly, the UV data-path is executed. Then, the RS data-path is executed in the next cycle. We can clearly see the path is separated and the cycle count is increased by 1.

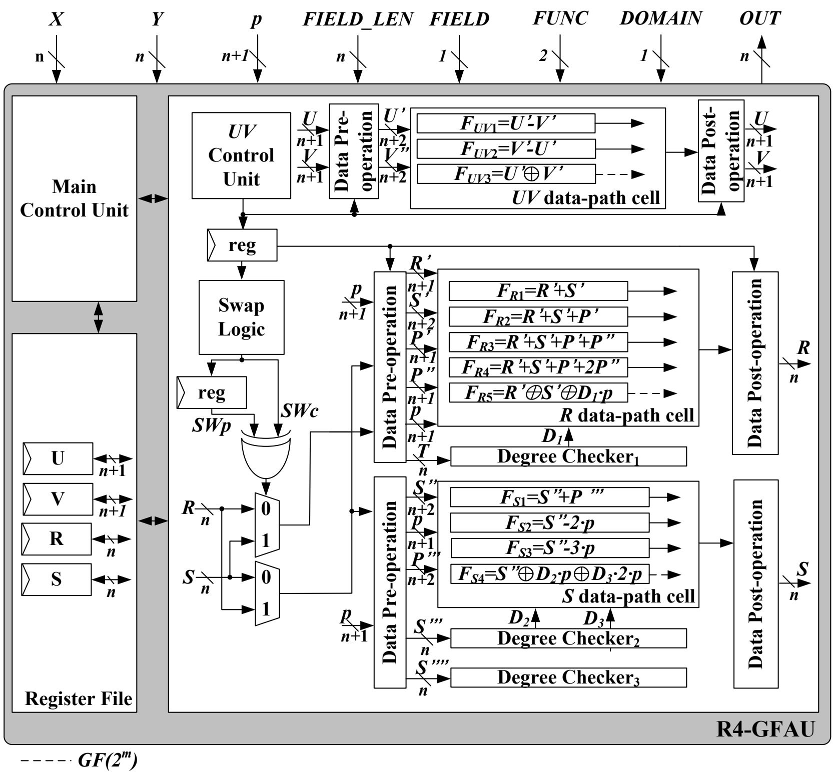


Figure 4.2: Architecture of R4-GFAU.

4.1.2 Hardware Sharing

Since both carry-propagation adder and XOR gate are the kernel arithmetic units of every modular operation, we can reuse these addition units to reduce the cost. The detailed hardware sharing method is shown in tables 4.1 and 4.2. The MMD and MM operations require the most adder units in UV, R, and S data-path. And the MA, MS, and MMM operations require only R data-path.

Besides, the division operation requires 21 different operations in R, S data-path. To reduce the hardware complexity, we propose a swap logic circuit. In algorithm 3.3, the operations of value R, S have some common arithmetic operations, such as $R = R - 2 \cdot S \pmod{p}$ and $S = S - 2 \cdot R \pmod{p}$ in step 15 and 22. We exploit a swap logic circuit to decide the R, S values are swapped or not in the beginning. The swap operation is decided by the previous and current value of swap signal, SW_p and SW_c. Note that when the operating step is 3, 4, 6, 8, 10, or 12 in algorithm 3.3, the swap signal is set to 1. Otherwise, the signal value is set to 0. The two operands R, S are swapped when the previous and current swap signals have different values. All the operations of this

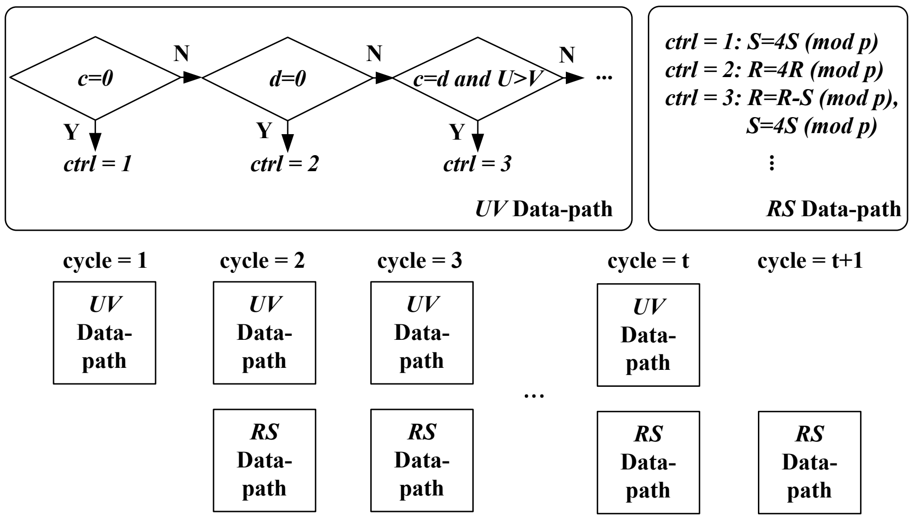


Figure 4.3: Data-path separation method.

algorithm are paired, such as operating steps 4 and 5, 6 and 7, and 8 and 9. By swap logic circuit, the similar operations can be shared and then the number of operations are reduced to 11 types.

In addition, the proposed R4-UD has some common controlled signals between dual fields (e.g., $j < m$, $c = 0$, $d = 0$.), so we can share them to reduce the complexity of controller.

Table 4.1: Details of hardware sharing method in R2-GFAU.

Field	Operation	F_{UV1}	F_{UV2}	F_{UV3}	F_{R1}	F_{R2}	F_{R3}	F_{S1}	F_{S2}
$GF(p)$	MA/MS				✓	✓			
	MMM				✓	✓			
	MM	✓	✓		✓	✓		✓	
	MMD	✓	✓		✓	✓		✓	
	MD	✓	✓		✓	✓			
$GF(2^m)$	MA						✓		
	MMM						✓		
	MM						✓		✓
	MMD			✓			✓		✓
	MD			✓			✓		

Table 4.2: Details of hardware sharing method in R4-GFAU.

Field	Operation	F_{UV1}	F_{UV2}	F_{UV3}	F_{R1}	F_{R2}	F_{R3}	F_{R4}	F_{R5}	F_{S1}	F_{S2}	F_{S3}	F_{S4}
$GF(p)$	MA/MS				✓	✓							
	MMM				✓	✓							
	MM				✓	✓	✓	✓		✓	✓	✓	
	MMD	✓	✓		✓	✓	✓			✓	✓	✓	
	MD	✓	✓		✓	✓	✓						
$GF(2^m)$	MA								✓				
	MMM								✓				
	MM								✓				✓
	MMD			✓					✓				✓
	MD			✓					✓				

4.1.3 Degree Checker

Intuitively, the degree-check operations in $GF(2^m)$, such as $2 \cdot S \pmod p$ and $4 \cdot S \pmod p$, are implemented by using huge multiplexers shown in Figure 4.4(a), but this method results in a long critical path. The critical path is log_2nAND+log_2nOR. Figure 4.4(b) shows the proposed degree checker, which requires only n 2-to-1 AND gates and 1 n-to-1 OR gate to finish the degree checking operation. The critical path lies in AND+log_2nOR. With this approach, it can compare the degree of the input value D_{in} with the field length. Note that the m-th bit of field length register is set to 1 and others are set to zero. If the input degree is smaller than field length, the output D_{out} is 0. Otherwise, the output is 1. This approach can also be used in the counter operation $j < m$ by setting $D_{in} = i$ and $i = \{i_{n-2}, i_{n-3}, ..., i_0, 1'b1\}$ every cycle, where $i = \{i_{n-1} = 0, ..., i_j = 0, i_{j-1} = 1, ..., i_0 = 1\}$. When $D_{out} = 1$, variable j is equal to m.

4.1.4 Ladder Selection

In R4-GFAU, the selection in the data post-operation of RS data-path are more complex than R2-GFAU. Intuitively, the post-operation architecture is used a lot of multiplexers to implement, which is shown in Figure 4.5. In each state, the data should be selected by multiplexer, so the number of multiplexers is equal to the total number of states. To reduce the selection complexity, we propose a ladder selection architecture shown in Fig-

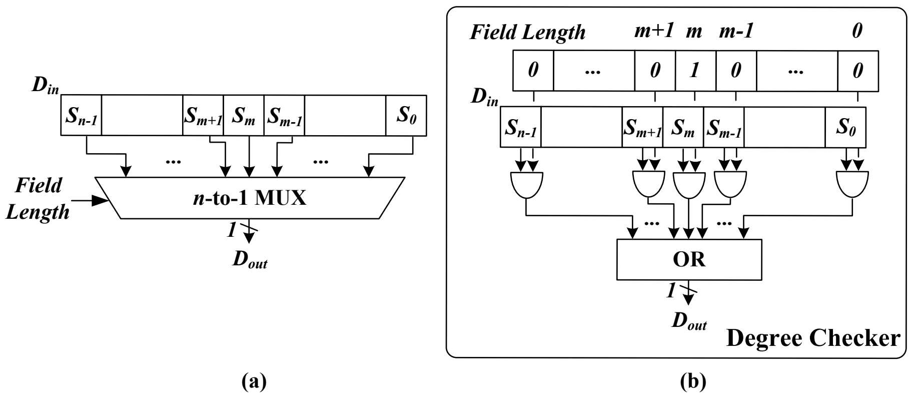


Figure 4.4: (a) The degree checking architecture by intuitive implementation. (b) Architecture of degree checker.

ure 4.6. The output value of RS data-path is decided by a fixed order. For example, if the operating operation is $S = 4S \pmod{p}$, the operands $\{S'', P'''\} = \{4S, -p\}$. With the order, which is from $F_{S3} < 0$ to $F_{S2} < 0$, the correct value is decided. Consequently, the output value is within the range $[0, p-1]$. The data selection hardware cost in the post-data operation block is reduced by this approach.

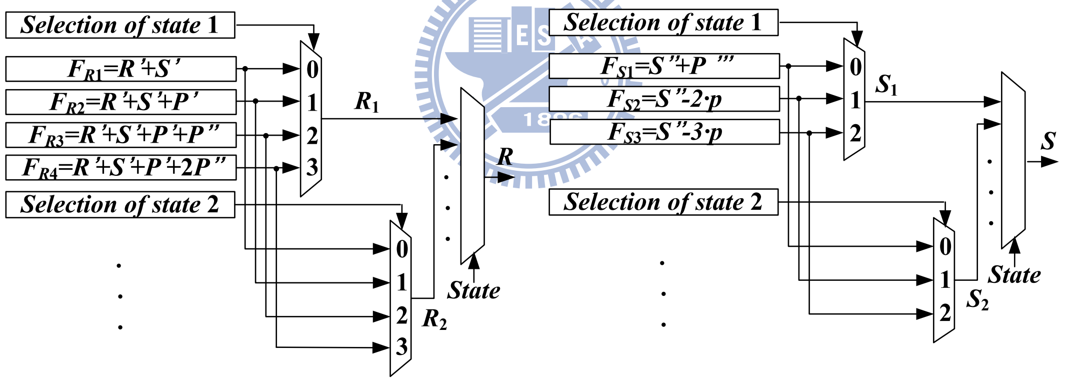


Figure 4.5: The data post-operation by intuitive implementation.

4.2 Dual-Field Elliptic Curve Cryptography Processor

Figure 4.7 shows the overall block diagram of our proposed DECP with a standard advanced microcontroller bus architecture (AMBA) high-performance bus (AHB) interface.

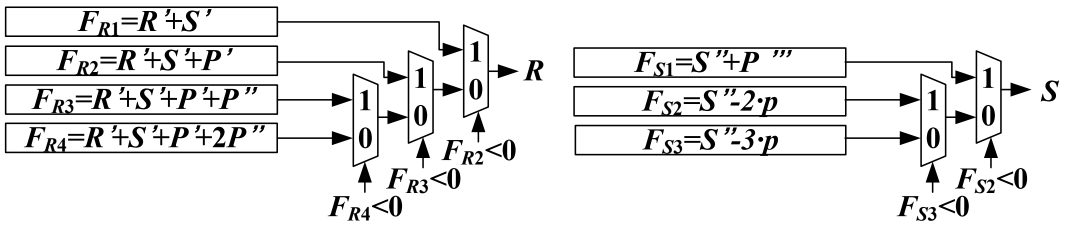


Figure 4.6: Architecture of ladder selection.

The ECSM with modular operations over dual fields, required for the ECC schemes such as signature, authentication, and key exchange defined in IEEE 1363 [1], can be calculated through the Galois field arithmetic unit (GFAU). The inputs are the user public/private-key, EC coordinates, EC parameters and protocol instructions. To perform these contents in real-time, the instruction decoding and pre-/post-processing satges are combined in our processor. After the instruction decoding, the pre-processing stage is to convert the EC coordinates and parameters into the Montgomery domain. Before returning the calculation results, the EC coordinates are converted back to the integer domain at post-processing stage. All the operands are stored in register file and transmitted to GFAU controlled by EC controller. Furthermore, to reduce the host CPU loading, the pre-/post-process stage can be achieved by the MMD and MMM operations. To convert an input value X between integer domain and Montgomery domain, it can be simply achieved through $\mathrm{MMD}(X, 1) \equiv X \cdot 2^m \pmod p$ and $\mathrm{MMM}(X \cdot 2^m, 1) \equiv X \pmod p$.

In Figure 4.8, the ECSM is based on the double-and-add/sub always method to achieve. To save one register, the point would be inversed and recovered, when the ECSUB is executed.

The performance analysis of R2-DECP and R4-DECP are shown in Tables 4.3 and 4.4. The execution cycle of R4-DECP is about two times better than the cycle of R2-DECP. The R4-DECP offers higher throughout with some area overhead. The overhead is reduced much by our proposed techniques, such as swap logic and ladder selection.

Table 4.3: Performance Analysis of R2-DECP.

Operations	Execution Cycles	
	$GF(p)$	$GF(2^m)$
ECDBL	$1 \cdot \mathrm{D} + 3 \cdot \mathrm{M} + 4 \cdot \mathrm{A/S}$	$1 \cdot \mathrm{D} + 2 \cdot \mathrm{M} + 8 \cdot \mathrm{A}$
	$= 4m + 4 \sim 5m + 4$	$= 3m + 8 \sim 4m + 8$
ECADD	$1 \cdot \mathrm{D} + 2 \cdot \mathrm{M} + 6 \cdot \mathrm{S}$	$1 \cdot \mathrm{D} + 2 \cdot \mathrm{M} + 9 \cdot \mathrm{A}$
	$= 3m + 6 \sim 4m + 6$	$3m + 9 - 4m + 9$
Domain Tran. (DT)	$3 \cdot \mathrm{D} + 2 \cdot \mathrm{M}$	$3 \cdot \mathrm{D} + 2 \cdot \mathrm{M}$
	$= 5m \sim 8m$	$= 5m \sim 8m$
ECSM	$m \cdot$ ECDBL $+ \frac{m \cdot \mathrm{ECADD}}{3} + \mathrm{DT}$	$m \cdot$ ECDBL $+ \frac{m \cdot \mathrm{ECADD}}{3} + \mathrm{DT}$
	$= 5m^2 + 11m \sim 6.33m^2 + 14m$	$= 4m^2 + 14m \sim 5.33m^2 + 17m$
Operation	**Critical Path Complexity**	
	$GF(p)$	$GF(2^m)$
ECSM	m	$\log_2 m$

4.3 Dual-Field Elliptic Curve Cryptography Processor with Power Analysis Countermeasures

To avoid the power analysis on our operating secret key, we randomize our operating domain. Generally the operating data have a factor 2^m, that means the data is operated in the Montgomery domain. In addition, if the factor is 2^0, that means operating domain is the integer domain. We define data has a factor 2^λ, that represents the data operated in random domain, where $0 \leq \lambda \leq m$. By the masking method, the domain value is changed in each ECSM operation. However, the total random numbers are just $m + 1$, which is too small, so we exploit the proposed URD and URM algorithms to increase it. Before operating ECSM, we choose a random number r to decide the random domain, 2^λ. Note that the number of ones in r is equal to the λ.

However, the random domain method can only randomize the first m cycle in division operation, but the next m cycle should be protected by another method. Since the S data-path is not used in the next m cycle, we set the input S'' to a random number to randomize the power consumption. Moreover, the total random number of MA and MS is still equal to $m+1$. Because the two operations are only accomplished by the R data-path,

Table 4.4: Performance Analysis of R4-DECP.

Operations	Execution Cycles	
	$GF(p)$	$GF(2^m)$
ECDBL	$1 \cdot \mathrm{D} + 3 \cdot \mathrm{M} + 4 \cdot \mathrm{A/S}$ $= 2.06m + 4 \sim 2.62m + 4$	$1 \cdot \mathrm{D} + 2 \cdot \mathrm{M} + 8 \cdot \mathrm{A}$ $= 1.56m + 8 \sim 2.12m + 8$
ECADD	$1 \cdot \mathrm{D} + 2 \cdot \mathrm{M} + 6 \cdot \mathrm{S}$ $= 1.56m + 6 \sim 2.12m + 6$	$1 \cdot \mathrm{D} + 2 \cdot \mathrm{M} + 9 \cdot \mathrm{A}$ $1.56m + 9 - 2.12m + 9$
Domain Tran. (DT)	$3 \cdot \mathrm{D} + 2 \cdot \mathrm{M}$ $= 2.68m \sim 4.36m$	$3 \cdot \mathrm{D} + 2 \cdot \mathrm{M}$ $= 2.68m \sim 4.36m$
ECSM	$m\cdot$ ECDBL $+\frac{m\cdot\mathrm{ECADD}}{3}+$DT $= 2.58m^2 + 8.68m \sim 3.32m^2 + 10.36m$	$m\cdot$ ECDBL $+\frac{m\cdot\mathrm{ECADD}}{3}+$DT $= 2.08m^2 + 13.68m \sim 2.82m^2 + 15.36m$

we also set the input of the S data-path to a random number to randomize the power consumption. By this approach, the secret information can be masked. In addition, we use the double and add/sub always method to resist SPA attack. The detail operating follow is shown in Figure 4.10.

By applying the above idea, the R2-DECP with power analysis countermeasures (R2-DECPAC) is proposed and shown in Figure 4.9. The R2-DECPAC is based on a R2-GFAU with power analysis countermeasure (R2-GFAUPAC) to support the random domain operations. We do three step modification from R2-DECP. First, the architecture of R2-GFAUPAC is based on the proposed URD and URM to implement. Second, we include a $|r|$-bit chaos-based pseudo number generator [61] into our R2-DECPAC to generate a m-bit random number in each ECSM operation. This approach can prevent the PA attack on the input of random numbers. Finally, the ECSM stage in EC controller is based on double-and-add/sub always method. The performance analysis is shown in Table 4.5.

Compared with our previous work [33] shown in Table 4.6, our approach requires lower area overhead and includes a pseudo random number generator. In addition, this SPA countermeasure has 50% execution cycle increase, but does not have any area degreation.

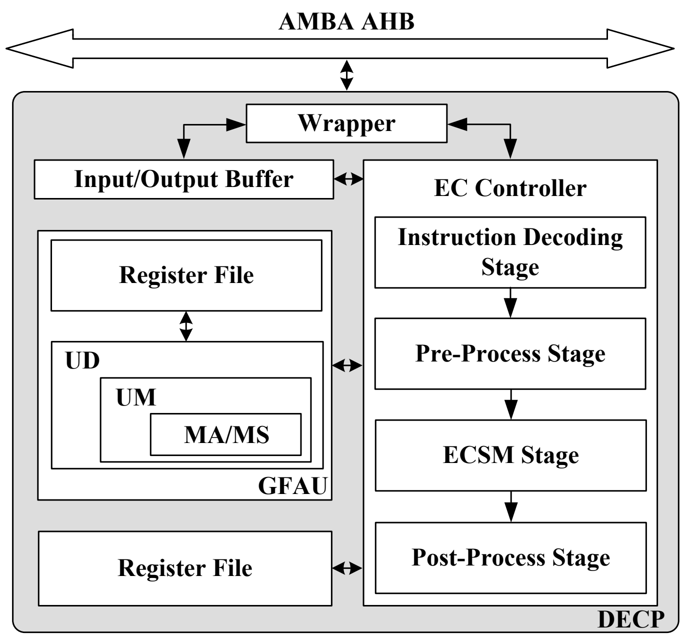


Figure 4.7: Architecture of DECP.

Table 4.5: Performance Analysis of R2-DECPAC.

Operations	Execution Cycles	
	$GF(p)$	$GF(2^m)$
ECSM	$m\cdot$ ECDBL $+m\cdot$ ECADD+DT $= 7m^2+15m \sim 9m^2+18m$	$m\cdot$ ECDBL $+m\cdot$ ECADD+DT $= 6m^2+22m \sim 8m^2+25m$

Table 4.6: Comparison with our previous work.

	R2-DECPAC	ESSCIRC'10 [33]
SPA countermeasure	Double-and-add/sub-always method	Double-and-add/sub-always method
DPA countermeasure	Random-domain method	Random scalar method
Additional arithmetic unit	—	$(n+\lvert r\rvert)$-bit adder $(n+\lvert r\rvert)$-bit multiplier
Random number generator	$\lvert r\rvert$-bit chaos-based pseudo number generator	—
Execution cycle increase by SPA countermeasure	50%	50%
Execution cycle increase by DPA countermeasure	0%	$100\lvert r\rvert/\lvert k\rvert\%$

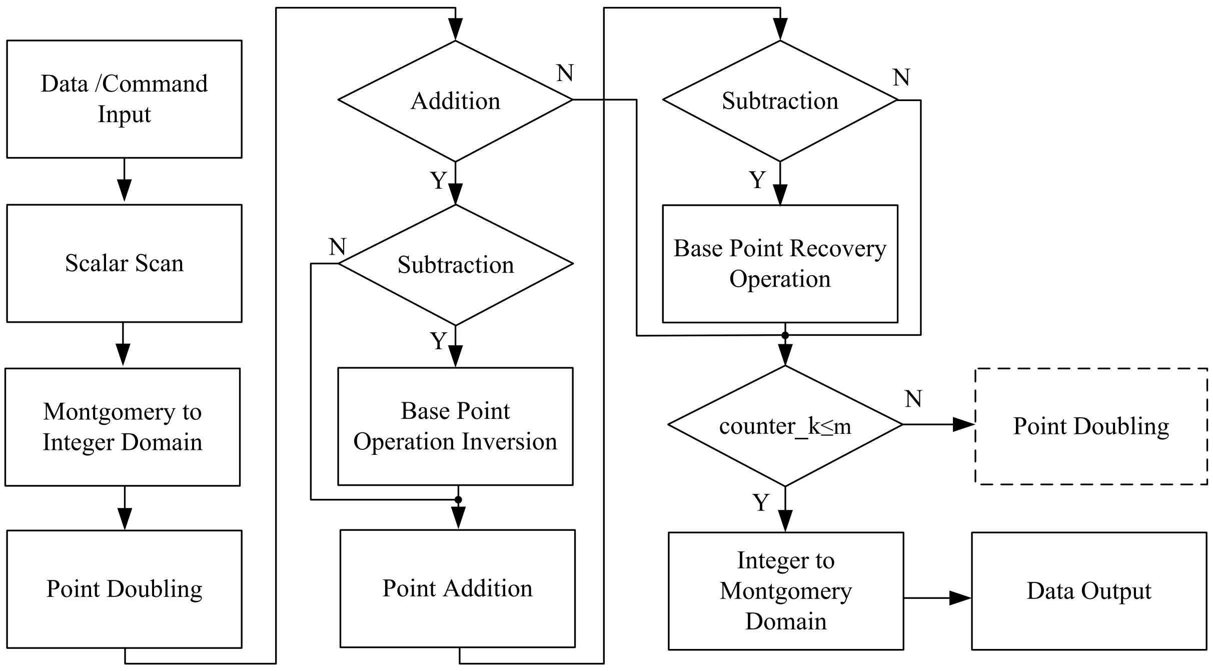


Figure 4.8: The flow chart of DECP.

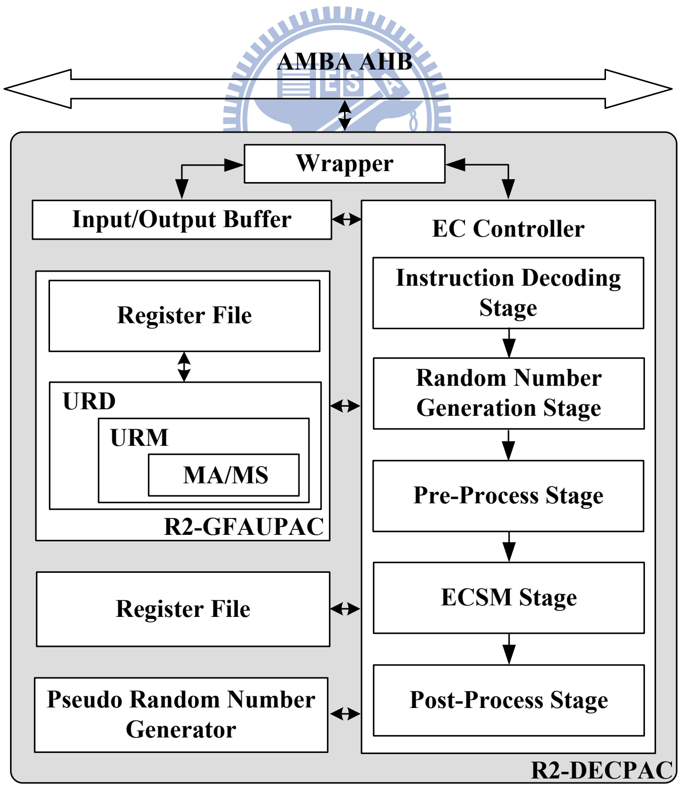


Figure 4.9: Architecture of R2-DECPAC.

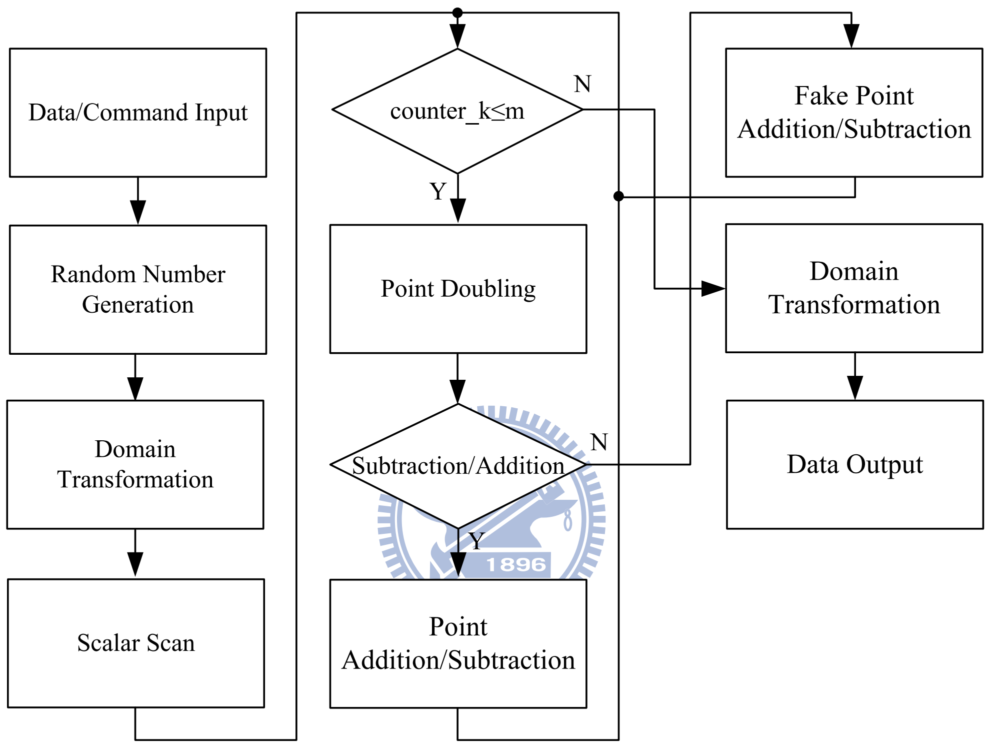


Figure 4.10: Flow Chart of R2-DECPAC.

Chapter 5

Implementation Results

In this chapter, we show the implementation results of our proposed GFAU, DECP, GFAUPAC, and DECPAC. The comparison tables show our designs outperform relative works.

5.1 Galois Field Arithmetic Unit

Tables 5.1 and 5.2 show the implementation result of the proposed R2-GFAU and R4-GFAU. The proposed R4-GFAU requires about half operation cycles of the R2-GFAU, but results in two times hardware cost. The AT product, gates $\times$ execution time, of R2-GFAU is 1.3 and 1.4 times better than R4-GFAU. This product ratio would decrease, since the GFAU is part of th DECP. Without including the GFAU, the architecture of R2-DECP and R4-DECP are similar. Besides, due to the proposed R2-UD and R4-UD, our proposed designs reduce about 2$\sim$3 times operation cycle compared with Chen's work [36] and Kaihara and Takagi's work [35] based on T-UMD. Compared with Tseng's work [32] and Liu's works [31] based on L-UD, our GFAU is 1$\sim$3 times better in execution cycles. By the proposed data-path separation technique, the operating frequency of our GFAU is better than them. In [30], this work is based on word-based architecture and the K-UI algorithm, but results in larger execution time. Besides, because our design supports more modular operations, the hardware cost is larger than previous works. But the execution time of our designs are faster due to the fast UD algorithm. The AT product of our GFAU is 2.3$\sim$91.5 better than previous works.

Table 5.1: Comparisons among 256-bit finite field designs over $GF(p)$.

	Tech.	Gates	Key Size	Function	Cycles	Time(s)@ $f_{\max}$(MHz)	AT
R2-GFAU^{*,1}	90nm	56.7K, 32Kcells	256	MMD/MD	316	0.79μ@400.0	1
				MMM/MM	257	0.64μ@400.0	–
				MA/MS	2	5.0n@400.0	–
R4-GFAU^{*,1}	90nm	100.5K, 42Kcells	256	MMD/MD	191	0.58μ@327.8	1.3
				MMM/MM	129	0.39μ@327.8	–
				MA/MS	2	7.1n@327.8	–
MT'08 [32]^{*,1,a}	0.18μm	47.4K	256	MMD	376	3.76μ@100.0	4.0
				MD	632	6.32μ@100.0	6.7
MT'07 [31]^{*,1}	0.18μm	42.1K	256	MMD	376	3.76μ@100.0	3.5
				MD	632	6.32μ@100.0	5.9
CHES'02 [30]^{*,1}	0.5μm	41.0K	256	MMI	–	100.0μ@ –	91.5
TC'07 [36]^{1}	0.35μm	33Kcells	256	MD	624	1.76μ@354.6	2.3
TC'05 [35]^{1}	0.35μm	27Kcells	256	MD	517	4.53μ@114.2	4.8
				MMM	175	1.53μ@114.2	–

* Dual-field design. ^{1} Synthesis result. ^{a} Supporting MMM, MA, and MS.

5.2 Dual-Field Elliptic Curve Cryptography Processor

The implementation results of R2-DECP and R4-DECP are shown in Tables 5.3 and 5.4. The results are verified by the NIST recommended ECs [2, 59]. The AT product of these two designs are almost the same, and the proposed R4-DECP can achieve higher throughput.

In Tables 5.5 and 5.6, compared with our previous work [33], our R2-DECP is 1.4 and 1.6 better due to the proposed R2-UD and degree checker. Based on R2-UD, our design reduces 27% execution cycle compared with [33]. In addition, the operating frequency increases 10% in binary field operation, because of the proposed degree checker. Due to

Table 5.2: Implementation results of proposed 256-bit GFAU over $GF(2^m)$.

	Tech.	Gates	Key Size	Function	Cycles	Time(s)@ $f_{\max}$(MHz)	AT
R2-GFAU^{*,1}	90nm	56.7K, 32Kcells	256	MMD/MD	362	0.65μ@555.6	1
				MMM/MM	257	0.46μ@555.6	–
				MA/MS	2	3.6n@555.6	–
R4-GFAU^{*,1}	90nm	100.5K, 42Kcells	256	MMD/MD	216	0.56μ@384.6	1.4
				MMM/MM	129	0.33μ@384.6	–
				MA/MS	2	5.2n@384.6	–

the proposed hardware sharing methods, our design is smaller than [33].

Our proposed 160-bit and 256-bit R4-DECP are implemented in UMC 90nm CMOS technology. Figure 5.1 shows the physical view of the DECP, which has core area of 0.29mm^2 and 0.45mm^2, and the post-layout simulation results are shown in Tables 5.7, 5.8, 5.9, and 5.10.

The comparison with previous works is given in Tables 5.7, 5.8, 5.9, 5.10, and 5.11. Our design supports all EC functions including point addition, point doubling, point scalar multiplication, domain transformation, and finite-field operations. In [13], Chen adopts T-UMD and systolic array to accomplish ECSM, but is three times slower than us in execution cycle. Furthermore, our design achieves competitive execution cycles with Satoh and Takano's work [23] and Lai and Huang's work [24] using 1 64-bit and 4 32-bit multiplier. Both [24] and [15] exploit parallel architecture technique to reduce the execution cycle but substantially increase the hardware cost. Consequently, the area of our DECP is about 2 times smaller than theirs. In [9], the work uses systolic array to achieve the highest operating frequency but is about 3 times slower than our design in execution cycle. Compared with the 160-bit and 256-bit designs in [24], our DECP is about 4 and 2 times better in AT product. From the table, our DECP outperforms other EC processor designs in terms of functionality, hardware efficiency, execution time, and power consumption.

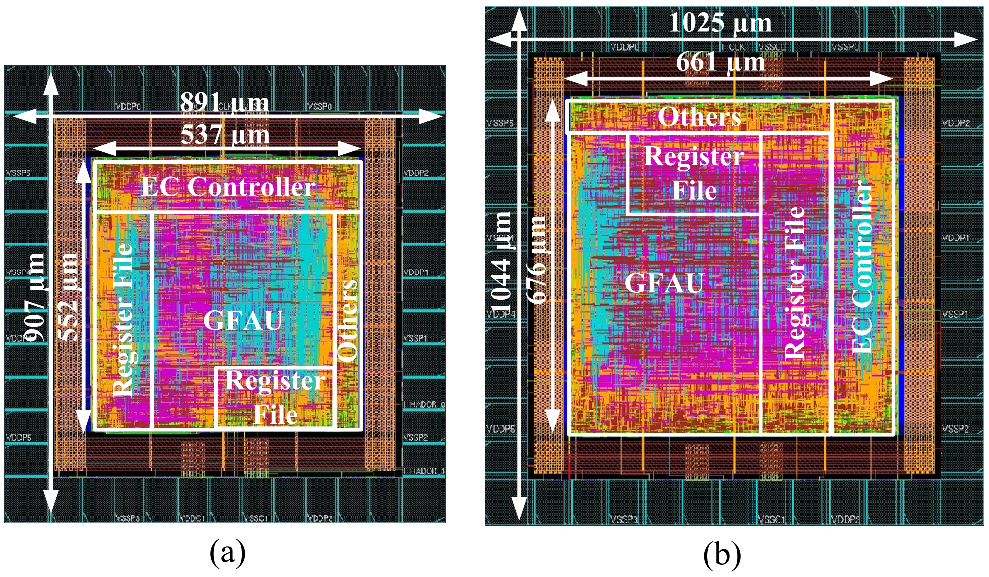


Figure 5.1: (a) Layout of 160-bit R4-DECP chip. (b) Layout of 256-bit R4-DECP chip.

Table 5.3: Implementation results of 256-bit R2/4-DECP over $GF(p)$.

	Tech.	Gates(K)	Key Size	Cycles	Time(ms)@ $f_{\max}$(MHz)	AT
R2-DECP^{*,1}	90nm	82.0	256	347,266	0.86@400.0	1
R4-DECP^{*,1}	90nm	134.3	256	193,386	0.51@333.3	1.0

5.3 Dual-Field Elliptic Curve Cryptography Processor with Power Analysis Countermeasures

Table 5.12 shows the implementation results of R2-GFAUPAC. The AT product between R2-GFAU and R2-GFAUPAC are similar, since the algorithms and number of arithmetic units are almost the same. To compare with our previous work, we implement the proposed DECPAC with maximum field size 521 bit. The 521-bit R2-DECPAC adopts a 32-bit chaos-based pseudo number generator which passes the random tests [62] shown in Figure 5.2. The implementation results of R2-DECPAC is shown in Table 5.13. Compared with R2-DECP, the R2-DECPAC requires 1.55 and 1.65 times execution cycles over dual fields, respectively, due to the double-and-add/sub always method. Moreover,

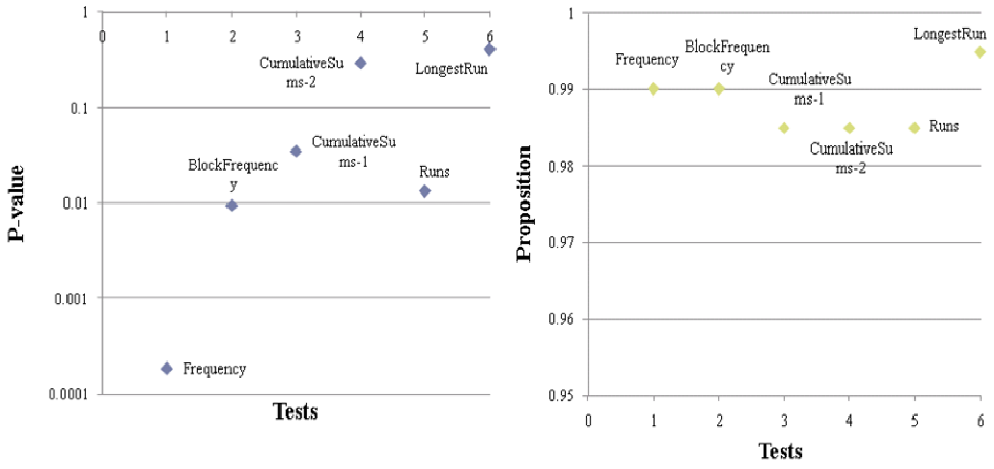


Figure 5.2: Random test on a 32-bit pseudo number generator.

Table 5.4: Implementation results of 256-bit R2/4-DECP over $GF(2^m)$.

	Tech.	Gates(K)	Key Size	Cycles	Time(ms)@ $f_{\max}$(MHz)	AT
R2-DECP^{*,1}	90nm	82.0	256	298,210	0.54@555.6	1
R4-DECP^{*,1}	90nm	134.3	256	165,354	0.44@377.3	1.3

the area degradation is just 8.4%, and the AT product is 1.7 and 1.8 times worse than R2-DECP. In Tables 5.14 and 5.15, compared with [33] based on L-UD, our approach is 1.3 times better in execution cycles due to the proposed R2-UD. In addition, [32] used sclar spliting to resist SPA attcak, but is 1.8 times slower than ours in execution cycles. The implemetation results show our approach is advantageous in system speed and hardware cost.

Table 5.5: Comparisons among 521-bit ECC designs over $GF(p)$.

	Tech.	Gates(K)	Key Size	Cycles	Time(ms)@ $f_{\max}$(MHz)	AT	Power (mW)
R2-DECP$^{\bigstar,1}$	90nm	165.9	521	1,438,637	3.88@370.3	1	–
ESSCIRC'10 [33]$^{\bigstar,1}$	90nm	170.7	521	1,967,982	5.31@370.3	1.4	–
MT'08 [32]$^{\bigstar,1,a}$	0.18μm	225.0	512	1,824,522	13.7@133.0	4.8	–

^{a} 512-bit DECP.

Table 5.6: Comparisons among 521-bit ECC designs over $GF(2^m)$.

	Tech.	Gates(K)	Key Size	Cycles	Time(ms)@ $f_{\max}$(MHz)	AT	Power (mW)
R2-DECP$^{\bigstar,1}$	90nm	165.9	409	769,492	1.38@555.6	1	–
ESSCIRC'10 [33]$^{\bigstar,1}$	90nm	170.7	409	1,165,672	2.23@500.0	1.6	–

Table 5.7: Comparisons among 160-bit ECC designs over $GF(p)$.

	Tech.	Core(mm^2) /Gates(K)	Key Size	Cycles	Time(ms)@ $f_{\max}$(MHz)	AT	Power (mW)
R4-DECP$^{\bigstar,2}$	90nm	0.29/82.8	160	79,528	0.31@256.4	1	22.5
TCAS-2'09 [24]$^{\bigstar,3}$	0.13μm	1.44/169.4	160	74,021	0.61@121.0	4.0	70.0
TVLSI'08 [25]$^{\bigstar,2}$	0.13μm	1.06/150.6	160	74,021	0.34@217.0	2.0	–
TC'03 [23]$^{\bigstar,1}$	0.13μm	– /117.5	160	153,000	1.21@137.7	5.5	–

^{2} Post-simulation result. ^{3} Measurement result.

Table 5.8: Comparisons with 160-bit ECC designs over $GF(2^m)$.

	Tech.	Core(mm^2) /Gates(K)	Key Size	Cycles	Time(ms)@ $f_{\max}$(MHz)	AT	Power (mW)
R4-DECP$^{\bigstar,2}$	90nm	0.29/82.8	160	56,506	0.19@289.9	1	25.9
TCAS-2'09 [24]$^{\bigstar,3}$	0.13μm	1.44/169.4	160	54,319	0.37@146.0	4.0	82.1
TVLSI'08 [25]$^{\bigstar,2}$	0.13μm	1.06/150.6	160	54,319	0.16@350.0	1.5	–
TC'03 [23]$^{\bigstar,1}$	0.13μm	– /117.5	160	86,000	0.19@510.2	1.4	–
DATE'07 [21]^{1,a}	0.25μm	– / –	163	9,251	0.08@111.1	–	154.2

^{a} 163-bit ECC processor.

Table 5.9: Comparisons among 256-bit ECC designs over $GF(p)$.

	Tech.	Core(mm^2) /Gates(K)	Key Size	Cycles	Time(ms)@ $f_{\max}$(MHz)	AT	Power (mW)
R4-DECP$^{\bigstar,2}$	90nm	0.45/122.0	160	79,720	0.32@250.0	–	–
			256	193,386	0.77@250.0	1	31.0
TCAS-2'09 [24]$^{\bigstar,1}$	0.13μm	– /197.0	256	252,067	1.21@208.0	2.5	–
ISCAS'08 [55]$^{\bigstar,2}$	0.18μm	17.8/ –	160	28,000	0.12@233.0	–	~10
			256	70,457	0.30@233.0	–	~10
MT'07 [31]$^{\bigstar,1}$	0.18μm	– /292.5	256	439,746	5.86@75.0	18.2	–
TC'03 [23]$^{\bigstar,1}$	0.13μm	– /120.2	256	369,000	2.69@137.0	3.4	–
TCAS-2'07 [9]^{1}	0.13μm	– /122.0	256	562,000	1.01@556.0	1.3	–

Table 5.10: Comparisons among 256-bit ECC designs over $GF(2^m)$.

	Tech.	Core(mm^2) /Gates(K)	Field	Cycles	Time(ms)@ $f_{\max}$(MHz)	AT	Power (mW)
R4-DECP$^{\bigstar,2}$	90nm	0.45/122.0	160	56,698	0.20@277.8	–	–
			256	165,354	0.59@277.8	1	35.6
TCAS-2'09 [24]$^{\bigstar,1}$	0.13μm	– /197.0	256	195,714	0.74@263.0	2.0	–
ISCAS'08 [55]$^{\bigstar,2}$	0.18μm	17.8/ –	160	22,000	0.095@233.0	–	∼10
			256	56,050	0.24@233.0	–	∼10
TC'03 [23]$^{\bigstar,1}$	0.13μm	– /120.2	256	230,000	0.45@510.0	0.6	–
DATE'08 [20]^{1,a}	90nm	– /1494.7	233	3,077	0.015@200.0	0.3	64.64
JSSC'01 [12]^{3,b}	0.25μm	– /880.0	256	725,000	14.5@50.0	177.2	–

^{a} 233-bit ECC processor. ^{b} Including modular exponentiation hardware.

Table 5.11: Comparisons among 571-bit ECC designs over $GF(2^m)$.

	Tech.	Core(mm^2) /Gates(K)	Key Size	Cycles	Time(ms)@ $f_{\max}$(MHz)	AT	Power (mW)
R4-DECP$^{\bigstar,1}$	90nm	– /308.2	571	719,659	2.09@344.8	1	–
TVLSI'09 [13]^{2,a}	0.13μm	2.34/331.7	571	2,033,500	4.9@415.0	2.5	277.6
TC'07 [15]^{1,b}	0.13μm	– /343.0	571	407,048	1.39@292.0	0.7	–
		– /244.0	571	451,140	1.55@292.0	0.6	–
DATE'07 [21]^1	0.25μm	– / –	571	322,275	0.48@53.3	–	396.1
CHES'00 [18]^1	0.25μm	– /165.0	571	1,452,000	22.0@66.0	5.6	–

^{a} Including modular exponentiation hardware. ^{b} This work can perform Hyper-ECC.

Table 5.12: Implementation results of 256-bit R2-GFAUPAR.

	Tech.	Gates(K)	Field	Function	Cycles	Time(ms)@ $f_{\max}$(MHz)	AT
R2-GFAUPAC^{*,1}	90nm	56.4	$GF(p_{256})$	RD	316	0.79μ@400.0	1[a]
				RM	257	0.64μ@400.0	–
				MA/MS	2	5.0n@400.0	–
			$GF(2^{256})$	RD	427	0.77μ@555.6	1.2[b]
				RM	257	0.46μ@555.6	–
				MA/MS	2	3.6n@555.6	–

[a] Compared with 256-bit R2-GFAU for $GF(p_{256})$ division operation.

[b] Compared with 256-bit R2-GFAU for $GF(2^{256})$ division operation.

Table 5.13: Implementation results of 256-bit R2-DECPAR.

	Tech.	Gates(K)	Field	Cycles	Time(s)@ $f_{\max}$(MHz)	AT
R2-DECPAC^{*,1}	90nm	88.8	$GF(p_{256})$	539,134	1.37@392.1	1.7[a]
			$GF(2^{256})$	494,196	0.89@555.6	1.8[b]

[a] Compared with 256-bit R2-DECP for $GF(p_{256})$ ECSM operation.

[b] Compared with 256-bit R2-DECP for $GF(2^{256})$ ECSM operation.

Table 5.14: Comparisons among 521-bit ECC designs over $GF(p)$.

	Tech.	Gates(K)/Area Degradation	Key Size	Cycles	Time(ms)@ $f_{\max}$(MHz)	AT	Time Increase
R2-DECPAC^{*,1}	90nm	179.9/8.4%	521	2,020,494	5.46@370.3	1	39.4%
ESSCIRC'10 [33]^{*,1,a}	90nm	185.1/8.9%	521	2,534,400 ~2,690,063	6.84~7.26 @370.3	1.3~ 1.4	37.2~ 45.6%
MT'08 [32]^{*,1,b}	0.18μm	277.0/23.1%	512	3,649,044	27.4@133.0	7.7	62.5%

[a] PA-resistant DECP. [b] 512-bit SPA-resistant DECP.

Table 5.15: Comparisons among 521-bit ECC designs over $GF(2^m)$.

	Tech.	Gates(K)/Area Degradation	Key Size	Cycles	Time(ms)@ $f_{\max}$(MHz)	AT	Time Increase
R2-DECPAC$^{\bigstar,1}$	90nm	179.9/6.9%	409	1,224,496	2.20@555.6	1	59.1%
ESSCIRC'10 [33]$^{\bigstar,1,a}$	90nm	185.1/8.9%	409	1,748,502∼ 1,852,862	3.5∼3.7 @500.0	1.6∼ 1.7	50.0∼ 59.3%

^{a} PA-resistant DECP.

Chapter 6

Conclusion and Discussion

In this thesis, we propose the unified algorithms to reduce the execution cycles of the division or multiplication operations. By these approaches and the proposed data-path separation and degree checker, our ECC processor achieves better performance in execution time. Besides, due to the proposed hardware sharing methods and ladder selection, our processor has smaller area compared with relative works. Our 160-bit dual field ECC processor is implemented in UMC 90nm CMOS technology and can execute one elliptic curve scalar multiplication in 310μs at 256.4MHz over GF(p) and 194μs at 289.9MHz over $GF(2^m)$, respectively, with core area 0.29mm^2 and power consumption at most 25.9mW. In addition, our 256-bit dual field ECC processor can execute one elliptic curve scalar multiplication in 770μs at 250.0MHz over GF(p) and 590μs at 277.8MHz over $GF(2^m)$, respectively, with core area 0.45mm^2 and power consumption at most 35.6mW.

To resist the DPA attacks, unified random algorithms are proposed. The total random numbers of these algorithms are up to 2^m. Our proposed R2-DECPAC is based on these algorithms with only 8.4% area degradation. The proposed 521-bit R2-DECPAC can execute one 521-bit ECSM in 5.46ms over GF(p) and one 409-bit ECSM in 2.2ms over $GF(2^m)$. Moreover, this approach should be further proven by real power analysis. Our proposal can be served as an soft-IP for those applications demanding cost-effective security solutions.

Appendix A

Appendix

A.1 Duality of Multiplication and Division

By our observation, the multiplication and division have some duality. The MMM and MD can be implemented in word-based architecture, since their operations can be executed word-by-word. In each iteration, the operands do not excess $2p$ in $GF(p)$, because the div 2 and div 4 operations are executed after addition and subtraction, such as $(R+qS)/4 \pmod p$ in T-UMD algorithm and $(R+X_iY)/2 \pmod p$ in radix-2 MMM algorithm. In addition, in table 4.1, we can find the MM and MMD operations require more arithmetic units than MMM and MD operations, separately. Consequently, the four operations have a duality relation shown in table A.1.

Table A.1: Duality of division and multiplication.

	MMM	MM	MMD	MD
Word-based architecture	✓			✓
More complicated		✓	✓	

A.2 Power Analysis Attack on The Dual-Field Elliptic Curve Cryptographic Processors

In [33], we proposed the DECP and DECPAC architectures. The DECP architecture is based on the L-UD algorithm, circular shift register, and proposed data-path separation method to implement. Moreover, based on the double-and-add/sub always and random scalar method, the DECPAC is implemented to resist the PA attacks. The countermeasures are already proven and the analysis environment is shown in figure A.1 and table A.2.

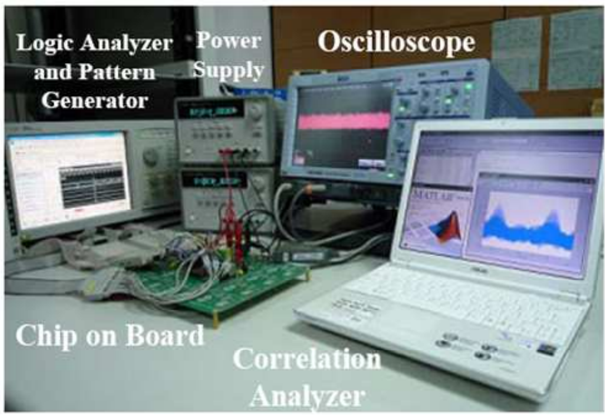


Figure A.1: The environment of PA attacks.

Table A.2: The detail environment of PA attacks.

Power Supply	Core: 1V Pad: 2.5V
Operating Frequency	10MHz
LA and Pattern Generator	Agilent 16902A
Oscilloscope	LeCroy SDA 400A
Differential Probe	LeCroy AP0341GHz
Sample Rate	1G Sample/sec
Resistance	22Ohm

A.3 Unified Division Algorithm Based on Takagi's Algorithm

In Section 3.1, we proposed the UD algorithms based on K-UI. The major modification is based on the proposed free recovery method which eliminates the phase 2 operation.

We use the same method to apply into T-UMD. Algorithm A.1 shows the proposed UD based on T-UMD (T-UD), which supports MMD and MD operations over dual fields with only $m \sim 2m$ execution cycles. Table A.3 shows the implementation results of the $GF(p)$ division algorithms. As a result, the performance of R2-UD is better than FT-UD, so we implement the ECSM operation based on this.

Algorithm A.1. *(Unified division algorithm based on Takagi's algorithm.)*
Input: *X, Y, and p, where X, Y are in $GF(p)$ or $GF(2^m)$ and p is the prime integer or irreducible polynomial.*

Output: $\begin{cases} \textit{If operation is MMD, then } S \equiv X \cdot Y^{-1} \cdot 2^m \pmod{p}. \\ \textit{If operation is MD, then } S \equiv X \cdot Y^{-1} \pmod{p}. \end{cases}$

```
 1. $U = Y$, $V = p$, $R = X$, $S = 0$, $\delta = 0$, $i = 0$
 2. while $U \neq 0$ do
 3.     if $U$ is even, then: $U = U/2$, $\delta = \delta - 1$
 4.         if operation = MMD, then: $S = 2S \pmod{p}$, $i = i + 1$
 5.         else: $R = R/2 \pmod{p}$
 6.     else
 7.         if $\delta < 0$, then: swap$(U, V)$, swap$(R, S)$, $\delta = -\delta$
 8.         if $(U + V) \pmod{4} = 0$, then: $q = -1$
 9.         else: $q = 1$
10.         $U = \frac{U + q \cdot V}{4}$, $\delta = \delta - 1$
11.         if operation = MMD and $i < m - 1$, then
12.             $R = R + q \cdot S \pmod{p}$, $S = 4S \pmod{p}$, $i = i + 2$
13.         else if operation = MMD and $i = m - 1$, then
14.             $R = \frac{R + q \cdot S}{2} \pmod{p}$, $S = 2S \pmod{p}$, $i = i + 1$
15.         else: $R = \frac{R + q \cdot S}{4} \pmod{p}$
16.         endif
17.     endif
18. endwhile
19. if $V = -1$, then: $S = P - S$
```

Table A.3: Implementation results of 256-bit $GF(p)$ division algorithms.

	Tech.	Gates(K)	Field	Function	Cycles	Time(μs)@ $f_{\max}$(MHz)
w-UM/D	0.13μm	48.6	$GF(p_{256})$	MMD	669	1.94@344.8
				MD	397	1.06@344.8
T-UD	0.13μm	78.9	$GF(p_{256})$	MMD	381	1.6@238.1
				MD	381	1.6@238.1
L-UD	0.13μm	54.5	$GF(p_{256})$	MMD	375	1.2@312.5
				MD	634	2.0@312.5
R2-UD	0.13μm	61.2	$GF(p_{256})$	MMD	316	1.1@285.7
				MD	316	1.1@285.7

A.4 Word-based Unified Multiplication/Division Architecture

During the T-UMD algorithm, the operations of U can be implemented by word-based architecture, due to any word of U does not depend on the other words. And the original operations of R, $R = R/2 \pmod p$ and $R = (R + q \cdot S)/4 \pmod p$, can be changed to $R = (R + R_0 \cdot p)/2$ and $R = (R + q \cdot S + (R + q \cdot S)_0 \cdot p + (R + q \cdot S + (R + q \cdot S)_0 \cdot p)_1 \cdot 2p)/4$ to eliminate the dependency. After the last iteration, the value of S should be reduced to within $[0,p-1]$. Combined with the MMM algorithm, we propose a word-based multiplication/division architecture (w-UM/D) to support MMM and MD operations, which is shown in figure A.2. The UV data-path is used to accomplish the UV operations, $U/2$ and $(U + qV)/4$. And the RS data-path computes the RS operations, $R/2 \pmod p$ and $(R + qS)/4 \pmod p$. Besides, the predicted logic computes the value $U[w+1:w]$ and $V[w+1:w]$ in the current state, due to the original $U[w+3:w+1]$ and $V[w+3:w+1]$ are the data in previous state. And the concatenated logic is used to concatenate $U[w+1:w]$ and $U[w-1:0]$ into $U[w+1:0]$. In addition, we choose three word lengths 16, 32, and 64 to implement a 256-bit w-UM/D, and the implementation results are shown in table A.3 and figure A.3. The MMD operation of the w-UM/D is achieved by MD and MMM operations as following:

$$MD(X,Y) = X \cdot Y^{-1} \pmod{p} \rightarrow MMM(X \cdot Y^{-1}, 2^{2m}) = X \cdot Y^{-1} \cdot 2^{m} \pmod{p} \tag{A.1}$$

Moreover, when the field size is equal to 1024-bit, the overall performance of w-UM/D is better than R2-UD because the operating frequency of w-UM/D is almost two times faster than that of R2-UD.

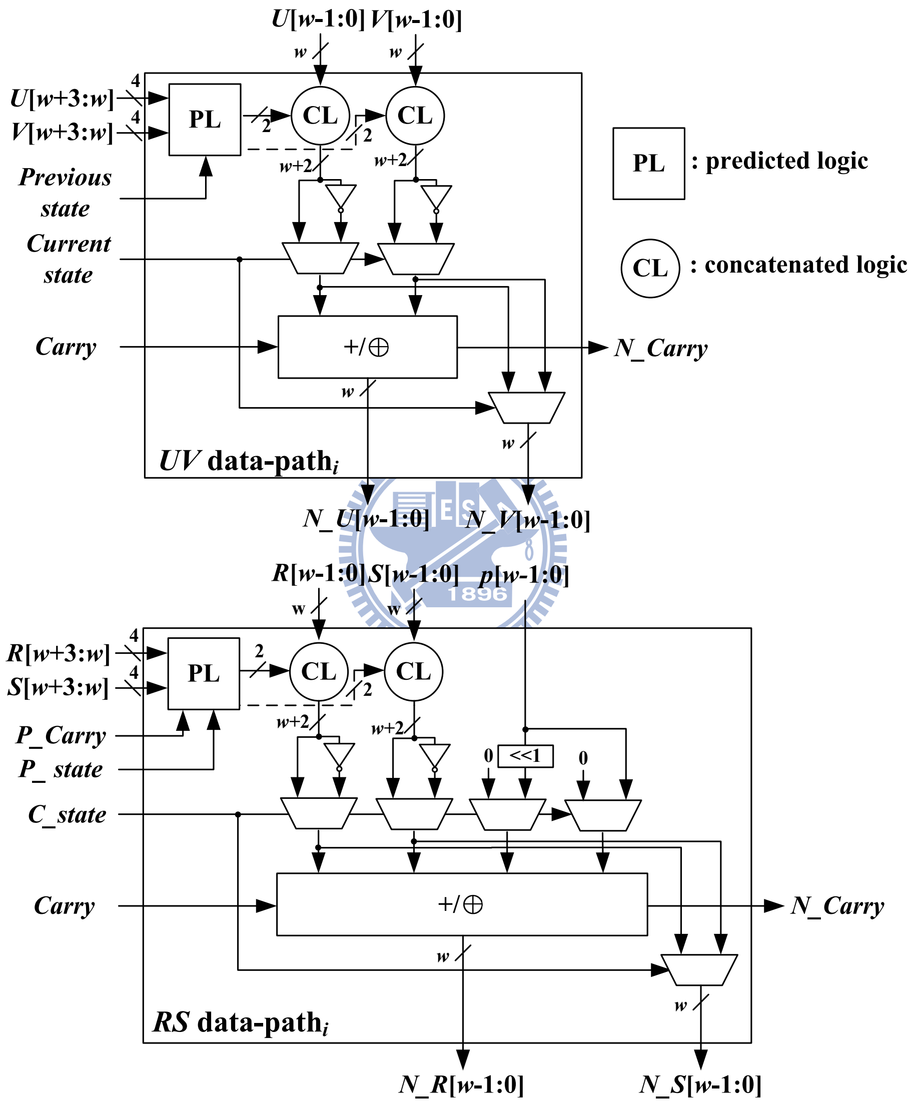


Figure A.2: The $GF(p)$ architecture of proposed w-UM/D.

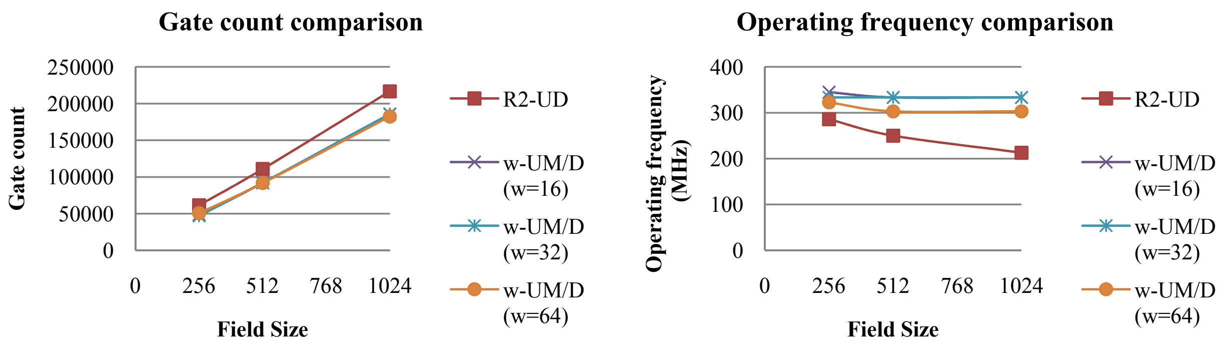


Figure A.3: The implementation result of unified division algorithms.

A.5 ECC Processor for RFID systems

To satisfy the requirement of radio frequency identification (RFID) tag [63, 64], the single field ECC processor is proposed [14, 22]. In the requirement of [63], the response time of RFID is 250ms and the energy received by th tag is 100μW. In ISO/IEC 18000-3(13.65MHz) [64], the power must be less than 15μW. To achieve the targets, we propose a low-cost ECC processor. We choose the L_{MC} method [14] to finish the ECSM operation, because the method requires fewer registers, which is proven in [14]. The algorithm of L_{MC} requires the MA, MM, and modular squaring (MSQ) operations. We use the parallel method [14] to implement the MM operation shown in figure A.4. Followings are the corresponding formula:

$$X \cdot Y \pmod p = \textstyle\sum_{i=0}^{n/d} X \cdot Y_{(i+1)\cdot d-1:i\cdot d} \pmod p \tag{A.2}$$

Note that the value of d represents the digit size. The execution cycle of MM operation is $(n/d) + 1$. In addition, we apply the fast squaring method [22] to finish the MSQ operation. Due to this method, the MSQ operation requires only 1 cycle. Combining the above architectures with MA, the single field ECC processor is proposed in figure A.5. In table A.4, the implementation results show our work outperforms the relative works in gates $\times$ cycles ratio. And the power consumption is slower than 15μW when the digit size is bigger than 7, which is shown in table A.5. These results show our work satisfies the requirement of ISO/IEC 18000-3(13.65MHz).

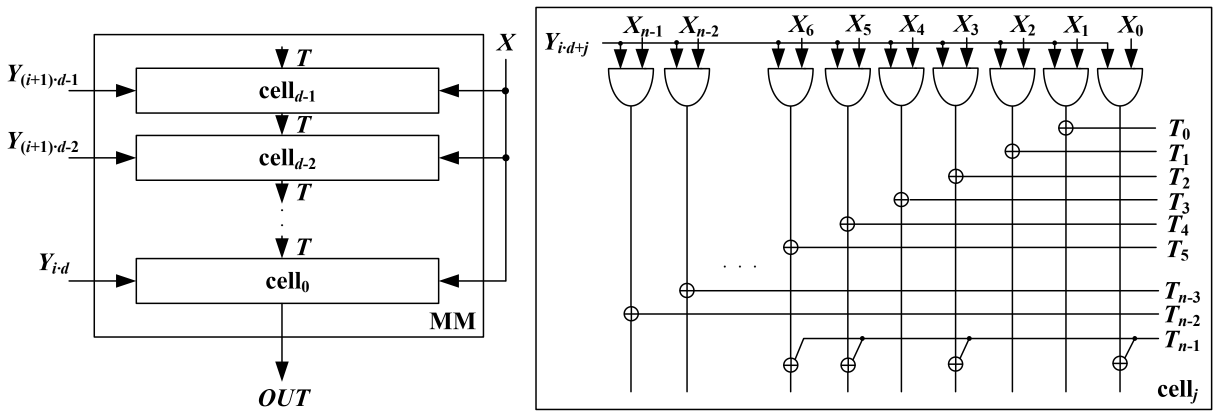
Table A.4: Implementation results of ECC processors over $GF(2^m)$.

	Tech.	Digit Size	Gates(K)	Field	Cycles	Gates × Cycles
Proposed	0.13μm	1	14.9	$GF(2^{163})$	214,168	6.7
		7	19.4		33,885	1.4
		14	26.1		18,321	1
[14]	0.13μm	1	10.1	$GF(2^{163})$	486,738[a]	10.3
		5	13.5		337,931[a]	9.5
[22]	0.35μm	–	15.1	$GF(2^{163})$	430,654	13.6
			16.2		376,864	17.2

[a]: By our modification.

Table A.5: Implementation results of proposed ECC processor.

	Tech.	Digit Size	Gates(K)	Field	Cycles	Time(ms)@ f_{max}(kHz)	Power(μW)
Proposed	90nm	1	11.0	$GF(2^{163})$	214,168	250@858.7	16.2
		7	19.4		33,885	250@135.5	14.2
		14	26.1		18,321	250@73.3	14.6



$Y_{(i+1)\cdot d-1}$
$Y_{(i+1)\cdot d-2}$
$Y_{i\cdot d}$
T
cell$_{d-1}$
cell$_{d-2}$
cell$_0$
X
MM
OUT
$Y_{i\cdot d+j}$
X_{n-1} X_{n-2} X_6 X_5 X_4 X_3 X_2 X_1 X_0
T_0 T_1 T_2 T_3 T_4 T_5 T_{n-3} T_{n-2} T_{n-1}
cell$_j$

Figure A.4: The architecture of MM operation over $GF(2^m)$.

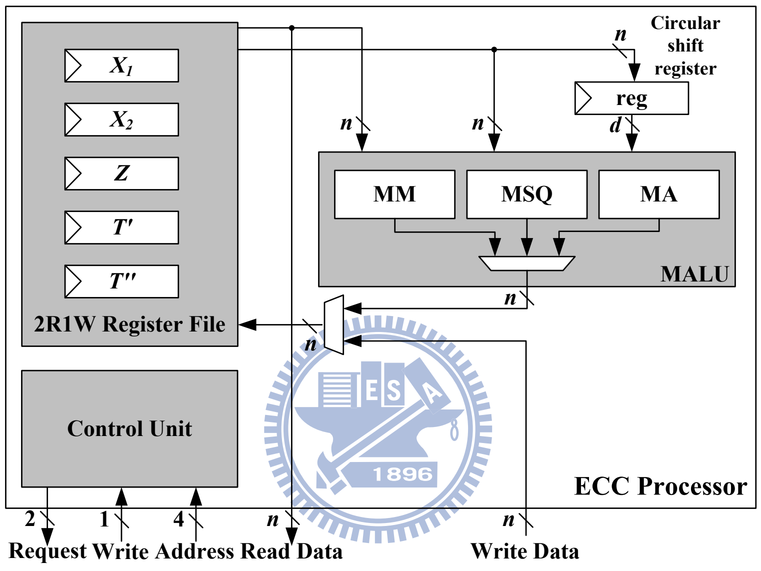


Figure A.5: The architecture of the proposed ECC processor.

Bibliography

[1] *Standard Specifications for Public-Key Cryptography: Additional Techniques*, IEEE Std. 1363A, 2000.

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

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

[4] ——, *Advances in Elliptic Curve Cryptography.* Cambridge University Press, 2005.

[5] H. Cohen and G. Frey, *Handbook of Elliptic and Hyperelliptic Curve Cryptography.* Chapman and Hall/CRC, 2006.

[6] W. Trappe and L. C. Washington, *Introduction to Cryptography with Coding Theory*, 2nd ed. Pearson Education International, 2006.

[7] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[8] H. Yan and Z. J. Shi, "Studying software implementations of elliptic curve cryptography," in *3rd International Conference on Information Technology: New Generations (ITNG)*, Las Vegas, Nevada, USA, 2006.

[9] G. Chen, G. Bai, and H. Chen, "A high-performance elliptic curve cryptographic processor for general curves over $GF(p)$ based on a systolic arithmetic unit," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 54, no. 5, pp. 412–416, May 2007.

[10] C. J. McIvor, M. McLoone, and J. V. McCanny, “Hardware elliptic curve cryptographic processor over $GF(p)$,” *IEEE Trans. Circuits Syst. I, Regular*, vol. 53, no. 9, pp. 1946–1957, Sept. 2006.

[11] P. Longa and A. Miri, “Fast and flexible elliptic curve point arithmetic over prime fields,” *IEEE Transactions on Computers*, vol. 57, no. 3, pp. 289–302, Mar. 2008.

[12] J. Goodman and A. P. Chandrakasan, “An energy-efficient reconfigurable public-key cryptography processor,” *IEEE J. Solid-State Circuits*, vol. 36, no. 11, pp. 1808–1820, Nov. 2001.

[13] J.-H. Chen, M.-D. Shieh, and W.-C. Lin, “A high-performance unified-field reconfigurable cryptographic processor,” *IEEE Transactions on Computers*, pp. 1–14, Nov. 2009.

[14] Y. K. Lee, K. Sakiyama, L. Batina, and I. Verbauwhede, “Elliptic-curve-based security processor for RFID,” *IEEE Transactions on Computers*, vol. 57, no. 11, pp. 1514–1527, Nov. 2008.

[15] K. Sakiyama, L. Batina, B. Preneel, and I. Verbauwhede, “Multicore curve-based cryptoprocessor with reconfigurable modular arithmetic logic units over $GF(2^n)$,” *IEEE Transactions on Computers*, vol. 56, no. 9, pp. 1269–1282, Sept. 2007.

[16] K. Jarvinen and J. Skytta, “On parallelization of high-speed processors for elliptic curve cryptography,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 16, no. 9, pp. 1162–1175, Sept. 2008.

[17] B. Ansari and M. A. Hasan, “High-performance architecture of elliptic curve scalar multiplication,” *IEEE Transactions on Computers*, vol. 57, no. 11, pp. 1443–1453, Nov. 2008.

[18] S. Okada, N. Torii, K. Itoh, and M. Takenaka, “Implementation of elliptic curve cryptographic coprocessor over $GF(2^m)$ on an FPGA,” in *Proc. Cryptographic Hardware and Embedded Systems (CHES’00)*, Worcester, MA, USA, Aug. 2000.

[19] Y. Eslami, A. Sheikholeslami, P. G. Gulak, S. Masui, and K. Mukaida, "An area-efficient universal cryptography processor for smart cards," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 14, no. 1, pp. 43–56, Jan. 2006.

[20] M. Purnprajna, C. Puttmann, and M. Porrmann, "Power aware reconfigurable multiprocessor for elliptic curve cryptography," in *Design, Automation and Test in Europe Conference and Exhibition*, ICM, Munich, Germany, Mar. 2008, pp. 1462–1467.

[21] S. Peter, P. Langendörfer, and K. Piotrowski, "Flexible hardware reduction for elliptic curve cryptography in GF(2^m)," in *Design, Automation and Test in Europe Conference and Exhibition*, Nice Acropolis, France, Apr. 2007, pp. 1–6.

[22] S. Kumar and C. Paar, "Are standards compliant elliptic curve cryptosystems feasible on RFID?" in *Workshop Record of the ECRYPT Workshop RFID Security*, 2006.

[23] A. Satoh and K. Takano, "A scalable dual-field elliptic curve cryptographic processor," *IEEE Trans. Comput.*, vol. 52, no. 4, pp. 449–460, 2003.

[24] J.-Y. Lai and C.-T. Huang, "Elixir: High-throughput cost-effective dual field processors and the design framework for elliptic curve cryptography," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 16, no. 11, pp. 1567–1580, Nov. 2008.

[25] ——, "A highly efficient cipher processor for dual-field elliptic curve cryptography," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 56, no. 5, pp. 394–398, May 2009.

[26] E. W. Weisstein, "Fermat's little theorem," in *MathWorld–A Wolfram Web Resource*, http://mathworld.wolfram.com/FermatsLittleTheorem.html.

[27] B. S. K. Jr., "The Montgomery inverse and its applications," *IEEE Transactions on Computers*, vol. 44, no. 8, pp. 1064–1065, Aug. 1995.

[28] E. Savaş and Ç. K. Koç, "The Montgomery modular inverse - revisited," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 49, no. 7, pp. 763–766, July 2000.

[29] R. Deng and Y. Zhou, "Improvement to Montgomery modular inverse algorithm," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 55, no. 9, pp. 1207–1210, Sept. 2006.

[30] A. A.-A. Gutub, A. F. Tenca, and Ç. K. Koç, "Scalable and unified hardware to compute Montgomery inverse in $GF(p)$ and $GF(2^n)$," in *CHES'02*, Redwood Shores, CA, USA, 2003.

[31] Y.-J. Liu, "An implementation of universal dual-field scalar multiplication on elliptic curve cryptosystems," Master's thesis, National Chiao Tung University, 2007.

[32] C.-Y. Tseng, "Design and implementation of an SPA-resistance dual-field elliptic curve arithmetic unit," Master's thesis, National Chiao Tung University, 2008.

[33] J.-W. Lee, Y.-L. Chen, C.-Y. Tseng, H.-C. Chang, and C.-Y. Lee, "A 521-bit dual-field elliptic curve cryptographic processor with power analysis resistance," in *European Solid-State Circuits Conference (ESSCIRC)*, Seville, Spain, Sept. 2010 (to appear).

[34] E. Savaş, "A carry-free architecture for Montgomery inversion," *IEEE Transactions on Computers*, vol. 54, no. 12, pp. 1508–1519, Dec. 2005.

[35] M. E. Kaihara and N. Takagi, "A hardware algorithm for modular multiplication/division," *IEEE Transactions on Computers*, vol. 54, no. 1, pp. 12–21, January 2005.

[36] G. Chen and H. Chen, "A new systolic architecture for modular division," *IEEE Transactions on Computers*, vol. 56, no. 2, pp. 282–286, Feb. 2007.

[37] G. M. d. Dormale, P. Bulens, and J.-J. Quisquater, "Efficient modular division implementation," in *FPL 2004, LNCS 3203*, Leuven, Belgium, 2004, pp. 231–240.

[38] N. Takagi, "A VLSI algorithm for modular division based on the binary GCD algorithm," *IEICE Trans. Fundamentals*, vol. E81-A, no. 5, pp. 724–728, May 1998.

[39] S. Mangard, E. Oswald, and T. Popp, *Power analysis Attacks-Revealing the Secrets of Smart Cards*. Springer, 2006.

[40] A. Miyamoto, N. Homma, T. Aoki, and A. Satoh, "SPA aganist an FPGA-based RSA implementation with a high-radix Montgomery multiplier," in *IEEE Int. Symp. Circuit Sust. (ISCAS)*, New Orleans, USA, May 2007, pp. 1847–1850.

[41] ——, “Chosen-message SPA attacks against FPGA-based RSA hardware implementation,” in *Int. Conf. on Field Programmable Logic and Applications (FPL)*, Heidelberg, Germany, Sept. 2008.

[42] P. C. Kocher, “Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems,” in *Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology*, 1996, pp. 104–113.

[43] P. C. Kocher, R. Lee, and G. McGraw, “Security as a new dimension in embedded system design,” in *Proceedings of the 41th Annual Conference on Design Automation*, 2004, pp. 753–760.

[44] P. C. Kocher, J. Jaffe, and B. Jun, “Differential power analysis,” in *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, 1999, pp. 388–397.

[45] R. Muresan and S. Gregori, “Protection circuit against differential power analysis attacks for smart cards,” *IEEE Transactions on Computers*, vol. 57, no. 11, pp. 1540–1549, Nov. 2008.

[46] E. Brier and M. Joye, “Weierstaβ elliptic curves and side-channel attack,” in *PKC’02*, vol. 2274, Paris, France, 2002, pp. 335–345.

[47] J. López and R. Dahab, “Fast multiplication on elliptic curve over $GF(2^n)$ without precomputation,” in *CHES’99*, vol. 1717, Worcester, MA, USA, 1999, pp. 316–327.

[48] P. Fouque and F. Valette, “The doubling attack-why upwards is better than downwards,” in *CHES’03*, vol. 2779, Cologne, Germany, 2003, pp. 269–280.

[49] K. Itoh, T. Izu, and M. Takenaka, “Address-bit differential power analysis of cryptographic schemes OK-ECDH and OK-ECDSA,” in *CHES’02*, Redwood Shores, CA, USA, 2003, pp. 399–412.

[50] L. Goubin, “A refined power-analysis attack on elliptic curve cryptosystems,” in *PKC 2003*, ser. Lecture Notes in Computer Science, vol. 2567, Miami, Florida, USA, 2003, pp. 199–210.

[51] T. Akishita and T. Takagi, "Zero-value point attacks on elliptic curve cryptosystem," in *ISC 2003*, ser. Lecture Notes in Computer Science, vol. 2851, Bristol, UK, 2003, pp. 199–210.

[52] J. López and R. Dahab, "Improved algorithms for elliptic curve arithmetic in $GF(2^m)$," in *Sel. Areas Cryptography: 5th Annu. Int. Workshop(SAC)*, vol. 1556, Santa Fe, New Mexico, Aug. 1998, pp. 201–212.

[53] H. Cohen, A. Miyaji, and T. Ono, "Efficient elliptic curve exponentiation using mixed coordinates," in *in Proc. Adv. Cryptolog. (Asiacrypt'98)*.

[54] P. L. Montgomery, "Modular multiplication without trial division," *Mathematics of Computation*, vol. 44, no. 170, pp. 519–521, April 1985.

[55] G. Chen, G. Bai, and H. Chen, "A dual-field elliptic curve cryptographic processor based on a systolic arithmetic unit," in *IEEE Int. Symp. Circuit Sust. (ISCAS)*, Seattle, Washington, USA, May 2008, pp. 3298–3301.

[56] A. F. Tenca and Ç. K. Koç, "A scalable architecture for modular multiplication based on Montgomery's algorithm," *IEEE Transactions on Computers*, vol. 52, no. 9, pp. 1215–1221, Sept. 2003.

[57] D. E. Knuth, *The Art of Computer Programming*, 3rd ed. Addison-Wesley, 1998, vol. 2, ch. Seminumerical Algorithms.

[58] G. V. S. Raju and R. Akbani, "Elliptic curve cryptosystem and its applications," in *IEEE International Conference on Systems, Man and Cybernetics*, vol. 2, Crystal City Hyatt Regency Washington, D. C., USA, Nov. 2003, pp. 1540–1543.

[59] *FIPS 186—Digital signature standard*, National Institute of Standards and Technology (NIST) Std., June 2009.

[60] J.-S. Coron, "Resistance against differential power analysis for elliptic curve cryptography," in *CHES'99*, ser. Lecture Notes in Computer Science, Ç. K. Koç and C. Paar, Eds., vol. 1717, Worcester, MA, USA, 1999, pp. 292–302.

[61] T. Addabbo, M. Alioto, A. Fort, S. Rocchi, and V. Vignoli, "Long period pseudo random bit generators derived from a discretized chaotic map," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 2, Kobe, Japan, May 2005, pp. 892–895.

[62] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, D. B. M. Vangel, A. Heckert, J. Dray, and S. Vo, *A statistical test suite for random and pseudorandom number generators for cryptographic applications*, NIST Special Publication 800-22 Std., Aug. 2008.

[63] F. Zhou, C. Chen, D. Jin, C. Huang, and H. Ming, "Evaluating and optimizing power consumption of anti-collision protocols for application in RFID systems," in *AUTO-ID Labs, white paper*, 2008.

[64] *Information Technology-Radio Frequency Identification for Item Management-Part 3: Parameters for Air Interface Communications at 13.56 MHz*, ISO/IEC Std. 18 000-3:2004, 2004.

作者簡介

1. 姓名：陳耀琳
2. 出生：台南市
3. 學歷：81.9 ~ 87.6 台南市立德高國小、台南市立新興國小
   87.9 ~ 90.6 台南市立崇明國中
   90.9 ~ 93.6 國立台南第一高級中學
   93.9 ~ 97.6 國立交通大學電子工程學系
   97.9 ~ 99.7 國立交通大學電子研究所系統組

得獎事蹟

1. 98 年度教育部 IC 設計競賽研究所標準元件數位電路設計：特優
2. 2009 ARM Code-O-Rama 設計大賽：佳作

投稿論文

1. J.-W. Lee, Y.-J. Liu, **Y.-L. Chen**, H.-C. Chang, and C.-Y. Lee, "*A Dual-Field Elliptic Curve Cryptographic Processor with Universal Division Algorithm*," submitted to IEEE Trans. on Compu..
2. J.-W. Lee, **Y.-L. Chen**, C.-Y. Tseng, H.-C. Chang, and C.-Y. Lee, "*A 521-Bit Dual-Field Elliptic Curve Cryptographic Processor with Side-Channel Attacks Resistance*," submitted to 2010 International Solid State Circuits Conference.
3. J.-W. Lee, **Y.-L. Chen**, H.-C. Chang, and C.-Y. Lee, "*A Dual-Field Elliptic Curve Cryptographic Processor Using Free-Recovery Montgomery Division*," submitted to 2010 IEEE International Symposium on Circuits and Systems.
4. J.-W. Lee, **Y.-L. Chen**, C.-Y. Tseng, Y.-J. Liu, H.-C. Chang, and C.-Y. Lee, "*A 521-bit Dual-Field Elliptic Curve Cryptographic Processor with Power Analysis Resistance*," submitted to 2010 Symposium on VLSI Circuits.
5. J.-W. Lee, **Y.-L. Chen**, C.-Y. Tseng, H.-C. Chang, and C.-Y. Lee, "*A 521-bit Dual-Field Elliptic Curve Cryptographic Processor with Power Analysis Resistance*," in European Conference Solid-State Circuits (ESSCIRC), Sept., 2010.
6. **Y.-L. Chen**, J.-W. Lee, H.-C. Chang, and C.-Y. Lee, "*A 521-bit Dual-Field Elliptic Curve Cryptographic Processor with Power Analysis Resistance*," submitted to 2010 IEEE Asian Conference on Solid-State Circuits (ASSCC).