國立交通大學

電子工程學系 電子研究所

博 士 論 文

具側漏資訊攻擊防禦之高硬體效能橢圓曲線密碼處理器

High-Performance Elliptic Curve Cryptographic Processor with Side-Channel Attack Resistance

研 究 生：李人偉

指導教授：李鎮宜 教授

中 華 民 國 一〇二 年 六 月

具側漏資訊攻擊防禦之高硬體效能橢圓曲線密碼處理器

High-Performance Elliptic Curve Cryptographic Processor with Side-Channel Attack Resistance

研 究 生：李人偉　　Student：Jen-Wei Lee

指導教授：李鎮宜　　Advisor：Chen-Yi Lee

國 立 交 通 大 學

電子工程學系 電子研究所

博 士 論 文

A Dissertation
Submitted to Department of Electronics Engineering and
Institute of Electronics
College of Electrical and Computer Engineering
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy
in
Electronics Engineering

June 2013
Hsinchu, Taiwan

中華民國一〇二年六月

具側漏資訊攻擊防禦之高硬體效能橢圓曲線密碼處理器

學生：李人偉　　指導教授：李鎮宜 博士

國立交通大學

電子工程學系暨電子研究所

摘要

現今，電子通訊帶給人類社會極大便利的資訊交流快速發展，相對應的保護個人訊息安全需求也日趨漸增。在資訊安全領域裡面，傳統的對稱式密碼系統能在使用者端妥善的加密保護資料隱密性，但這都還不足以解決金鑰配置、明文完整性以及不合法授權使用的問題。非對稱式密碼系統，又稱公開金鑰密碼系統，其被開發用來滿足前述應用的需求。在過去的幾年中，橢圓曲線密碼學是一個被提出相對傳統 RSA 演算法安全度較高的可實現方法，但是目前還尚未有合適橢圓曲線密碼處理器的設計對應方法。

在本論文，我們從系統的角度探索密碼處理器的設計，包含從最上層的演算法、次之硬體運算單元架構以及底層的電子電路設計。為了追求高硬體效能，我們著手採用了一些改善硬體速度、硬體複雜度以及能量消耗的設計技巧，除此之外，一個合宜的密碼處理器，也必須包含側漏資訊攻擊的防禦。如何能在硬體運算時不洩漏和金鑰有關的訊息，也不因為防禦設計上造成硬體複雜度增加過度的代價，這些都將是設計上的挑戰也是我們實現電路的目標。

如上所述，我們提出了一些新的設計方法，包含隨機式運算與金鑰不相依的硬體排程方法，此設計的特色除了適合系統實現的整合，也因為不需額外的參數與離線計算，所以硬體計算可以符合標準化的規範，另外一個優點是和過去的文獻相比，我們的側漏資訊攻擊防禦硬體代價也相較為低。為了提供更穩健的保護能力，我們也提出一個新的單一晶片真實亂數產生器設計方法，其能提供足夠的亂度給硬體作隨機式運算。針對這些提出的設計方法，我們的橢圓曲線密碼處理器架構在硬體效能與側漏資訊攻擊防禦都有相較過去文獻的優異表現。

更進一步呈現我們的研究貢獻，透過聯電 90 奈米製程，我們針對各種應用製作開發晶片。

第一顆為 0.41 mm^2 160 位元長的橢圓曲線密碼處理器，其能各別在 GF(p_{160})與 GF(2^{160})有限域的 0.34 ms 11.7 μJ 與 0.29 ms 9.3 μJ 下完成一次橢圓曲線點乘法計算，此優異的硬體效能顯示其將適合在手機通訊產品上的開發使用。第二顆是 521 位元長的橢圓曲線密碼處理器，其能各別在 GF(p_{521})與 GF(2^{521})有限域的 3.40 ms 與 2.77 ms 時間內完成一次橢圓曲線點乘法計算，其中透過橢圓曲線點產生法，能減少一半的公開金鑰傳遞訊息量，此設計是達到至今運算最快的橢圓曲線密碼處理器，其將適合高速的雲端伺服器應用。另外一顆是操作在低電壓 0.5 V 與低時脈頻率 25 MHz 的 192 位元長的橢圓曲線密碼處理器，其能各別在 GF(p_{192})與 GF(2^{192})有限域的 10.8 ms 438 μJ 與 9.2 ms 437 μJ 下完成一次橢圓曲線點乘法計算，此優異的低能量消耗顯示其將適合在未來的物聯網產品上的開發使用。最後，這些晶片也都經過收集上百萬條能量軌跡的側漏資訊攻擊防禦量測驗證其安全性。

High-Performance Elliptic Curve Cryptographic Processor with Side-Channel Attack Resistance

Student：Jen-Wei Lee　　Advisors：Dr. Chen-Yi Lee

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

ABSTRACT

Nowadays, the fast development of network communication in electronics industry brings the people to a quick and convenient life, while the demand in safety for protecting the personal private data from revealing significantly increases as well. In security, the conventional symmetric-key scheme can locally achieve the encryption, but the decryption key and ciphertext are still needed to be sent without disclosure, modification, duplication, forgery, and even unauthorized access. The asymmetric-key scheme or so called public-key cryptosystems (PKC) is developed to satisfy these requirements. In recent years, a new coming approach, elliptic curve cryptography (ECC), has been adopted in several applications for ensuring the security of information exchange. However, the suitable solution of ECC processor has not appeared so far.

In this dissertation, we investigate the design of crypto engine through a system view, from top to down, including the algorithm, operation scheduling, processing-element architecture, and also circuit-level implementation. For pursuing the achievement of high-performance accelerator, several improvement techniques for the hardware speed, hardware complexity, and power consumption are promoted. Besides, to deliver a decent design of crypto engine, the device security such as the counter-measure of side-channel attacks (SCAs) is also included in our implementation

target. And then, both of these design issues lead to a big challenge, where it requires the device to be implemented without both of the key-dependent processed data and much overhead of SCA resistance.

As above, we proposed a new design method, which is based on the randomized computation and key-independent scheduling manner, to protect the private date stored in device from the side-channel information leakage. The feature is that it is suitable for the system integration and the usage of the standard without any pre-computation. Another advantage is that the overhead of protected design is lower than that of related previous works. The robustness of SCA resistance is examined by exploiting an on-chip true-random number generator (TRNG) with sufficient randomness. Moreover, the corresponding design architecture of hardware implementation is introduced, and our ECC processor outperforms both in the hardware efficiency and protection against SCAs as compared with the other approaches.

To show more our contributions, we further conduct our research for several standard applications. Fabricated by UMC 90-nm CMOS technology, a 0.41 $mm^2$160-bit ECC chip can achieve 0.34/0.29 ms 11.7/9.3 μJ for one GF(p)/GF(2^m) elliptic curve scalar multiplication (ECSM), which is effective at the hardware cost and suitable for the mobile device; a 521-bit ECC chip performs each GF(p_{521}) ECSM in 3.40 ms and GF(2^{521}) ECSM in 2.77 ms, where it saves 50% data transmission of public key by on-chip elliptic curve point generation (ECPG). This is the fastest design and also applicable for the cloud computing; a 192-bit ECC chip achieves 10.8/9.2 ms 438/437 μW GF(p_{192})/GF(2^{192}) ECSM at scaled 0.5 V and 25 MHz, where it is efficient at the power consumption and suitable for the applications of Internet of Things (IoT). In addition, the SCA resistance for each design is demonstrated by millions of measurements.

誌　謝

在博士班的求學生涯中，承蒙交大培育、多位師長提攜、朋友的協助以及家人的支持，讓我能一路度過各個挑戰，最後順利完成我的博士學位。我要非常感謝一起指導我研究的李鎮宜教授和張錫嘉教授，兩位教授給予我廣大的研究發揮空間以及豐富的研究軟硬體資源，除此之外，老師們也積極的領導我參與研究相關的國際學術活動，還有強調自我運動健康以及社交活動的重要性，同時藉由這幾年的研究過程，更讓我從老師們身上學習到待人處世和視野遠見的培養，讓我博士班生涯有了更全面性的成長。我也要感謝陳榮傑教授在我研究論文的過程，耐心地教導我建立起數學理論的基礎。另外，要感謝我的口試委員：吳安宇教授、周世傑教授、謝明得教授、蔡宗漢教授、黃元豪教授與賴伯承教授在百忙之中參加我的口試，給我許多寶貴的建議，讓我看到研究上許多不同的面向，並啟發我未來的研究方向。

接著，我要感謝 SI2 實驗室以及 Ocean 研究團隊的全體同仁，讓我得以在此學習成長，吸收許多寶貴經驗。在學業上，透過與大家的討論，使得研究更加完善充實；在生活上，也因為有了各位，在面對研究挑戰的路途上，有更多的歡笑和難以忘懷的甜美回憶。

在這段期間，謝謝交大電子系所提供給我博士班的獎助學金，讓我能心無旁騖、全心全力盡快完成博士班研究與不斷擴充自己的專業能力。也要感謝行政院國家科學委員會的出席國際學術會議補助，讓我能除了吸收世界各地優秀學者所提供的研究資訊之外，也提高台灣在國際學術研究上的能見度。

我要謝謝鍾菁哲學長、許騰仁學長、林建青學長、游瑞元學長、陳志龍學長、林義閔學長、余建螢學長與涂博銘先生，幫助我了解製作晶片過程與驗證系統行為。感謝范銘隆學長，從我第一次出國參加會議研討一直到我畢業，這一路上都是我最棒的心靈導師。我也很感謝天公伯和交大土地公爺爺，總是能在很多關鍵時刻給我強運。

然後，我要由衷的向我台北與宜蘭的大家庭所有人，獻上我最誠摯的感謝，謝謝你們這一路照顧、陪伴我從小到大。我也特別感謝我的爸爸、媽媽不辭辛勞對我的栽培，謝謝我的哥哥，能夠分享我的喜怒哀樂並給予經濟上的幫助，也謝謝我的二舅和屘姨，我知道你們一直都很關心我，待我有如自己的孩子一樣。這些日子多虧有全家人的體諒，有你們無悔的付出與關心，讓我無後顧之憂地完成博士學業。在這邊以此論文獻給你們，作為我們全家共同分享的成果。

最後，虔誠祈禱上天保佑台灣這塊美麗的寶島，我願和熱愛這片土地的任何人一起努力。大家加油！為台灣加油！

Contents

List of Figures

List of Tables

Glossary

ADD – Addition. 53

AES – Advanced Encryption Standard. 1

ALU – Arithmetic Logic Unit. 6

ASIC – Application-Specific Integrated Circuit. 8

CBC-MAC – Cipher Block Chaining Message Authentication Code. 19

CCM – CTR with CBC-MAC. 19

CE – Crypto Engine. 115

CIHDC – Configurable Interlaced Hysteresis Delay Cell. 60

CMAC – Cipher-based Message Authentication Code. 19

CPA – Collision Power-Analysis. 24

CTR – Counter. 19

DES – Data Encryption Standard. 1

DF – Dual Field or Dual-Field. 5

DF-ECC – Dual-Field Elliptic Curve Cryptography (or Cryptographic). 115

DHK – Diffie-Hellman Key. 115

DLP – Discrete Logarithm Problem. 12

DPA – Differential Power-Analysis. 24

DSA – Digital Signature Algorithm. 13

EC – Elliptic Curve. 16

ECC – Elliptic Curve Cryptography (or Cryptographic). 3, 4

ECDLP – Elliptic Curve Discrete Logarithm Problem. 3

ECIES – Elliptic Curve Integrated Encryption Scheme. 19

ECPA – Elliptic Curve Point Addition. 16

ECPC – Elliptic Curve Point Calculation. 17

ECPD – Elliptic Curve Point Doubling. 16

ECPG – Elliptic Curve Point Generation. 34

ECPS – Elliptic Curve Point Subtraction. 16

ECSM – Elliptic Curve Scalar Multiplication. 27

ECSP-DSA – Elliptic Curve Signature Primitive Digital Signature Algorithm. 19

ECSP-NR – Elliptic Curve Signature Primitive Nyberg-Rueppel. 19

ECSVDP-DH – Elliptic Curve Secret Value Derivation Primitive Diffie-Hellman. 19

ECSVDP-MQV – Elliptic Curve Secret Value Derivation Primitive Menezes-Qu-Vanstone. 19

FHE – Fully Homomorphism Encryption. 119

FPGA – Field-Programmable Gate Array. 8

$GF(2^m)$ – Notation of "Galois field with characteristic 2 and degree m" or "extension binary field". 5–7

GFAU – Galois Field Arithmetic Unit. 71

$GF(p)$ – Notation of "Galois field with characteristic p" or "prime field". 5–7

HT – Half Trace. 67

IBE – ID-based Encryption. 119

IC – Integrated Circuit. 54

IEEE – Institute of Electrical and Electronics Engineers. 4

IHDC – Interlaced Hysteresis Delay Cell. 59

IoT – Internet of Things. 119

JS-GFAU – Jacobi Symbol and Galois Field Arithmetic Unit. vii, 68

KO – Karatsuba-Ofman. 5

LR-DA – Left-to-Right Double-and-Add. 28

LR-DAA – Left-to-Right Double-and-Add-Always. 28

LR-DAS – Left-to-Right Double-and-Add/Substract. 112

LS – Lucas Sequence. 67

m – Notation of "bit size of the operating field length". 5, 6

MAS – Multiplier-Adder/Subtractor. 53

MD – Modular Division. 53

MK – Master Key. 20

MM – Modular Multiplication. 53

MS – Modular Squaring. 17

n – Notation of "maximum bit size of the operating field length". 41

ONB – Optimal Normal Basis. 5

p – Notation of "prime". 13

PKC – Public-Key Cryptosystems. 2, 3, 12

PRNG – Pseudo-Random Number Generator. 54

RADD – Randomized Addition. 42

RFID – Radio-Frequency Identification. 7

RL-DAA – Right-to-Left Double-and-Add-Always. 39

RMD – Randomized Montgomery Division. 42

RMM – Randomized Montgomery Multiplication. 42

RNG – Random Number Generator. 10

RNS – Residue Number System. 5

RO – Ring Oscillator. x, 55

RO-RNG – Ring-Oscillator-based Random Number Generator. 54

RSUB – Randomized Subtraction. 42

SCA – Side-Channel Attack. 4, 8

SPA – Simple Power-Analysis. 24

SUB – Subtraction. 53

T_{ADD} – Notation of "computation time of ADD". 53

T_{MD} – Notation of "computation time of MD". 53

T_{MM} – Notation of "computation time of MM". 53

TRNG – True-Random Number Generator. vii, 54

T_{SUB} – Notation of "computation time of SUB". 53

ZPA – Zero-value Power-Analysis. 24

Chapter 1

Introduction

A general model for the network security is shown in Figure 1.1, where the message is to be transferred from one party to another across some sort of wireless communications or Internet service. The two parties, who are the *principals* in this transaction, must cooperate the exchange to take place. A logical information channel is established by defining a communication protocol such as GSM, Wi-Fi, and TCP/IP. The trusted third party may be needed to achieve secure transmission. For example, a third party may be responsible for distributing the secret information to the two principals while keeping it from any opponent. Or a third party may be needed to arbitrate disputes between the two principals concerning the authenticity of a message transmission.

Symmetric-key encryption is a form of cryptosystem where the encryption and decryption are performed by using the same key. It is also known as *conventional encryption*. Symmetric-key encryption transforms security-related message, plaintext, into ciphertext using a secret key and an encryption algorithm, such as stream cipher RC4 [1], block cipher DES (Data Encryption Standard) [2], and block cipher AES (Advanced Encryption Standard) [3]. By using the same key and decryption algorithm, the plaintext is recovered from the ciphertext. The traditional symmetric-key ciphers use the *substitution* and/or *transposition* techniques. Substitution techniques map plaintext elements into ciphertext elements (each letter retains its position but changes its identity). Transposition techniques systematically transpose the positions of plaintext elements (each letter retains its identity but changes its position). An example model of conventional encryption is shown in Figure 1.2.

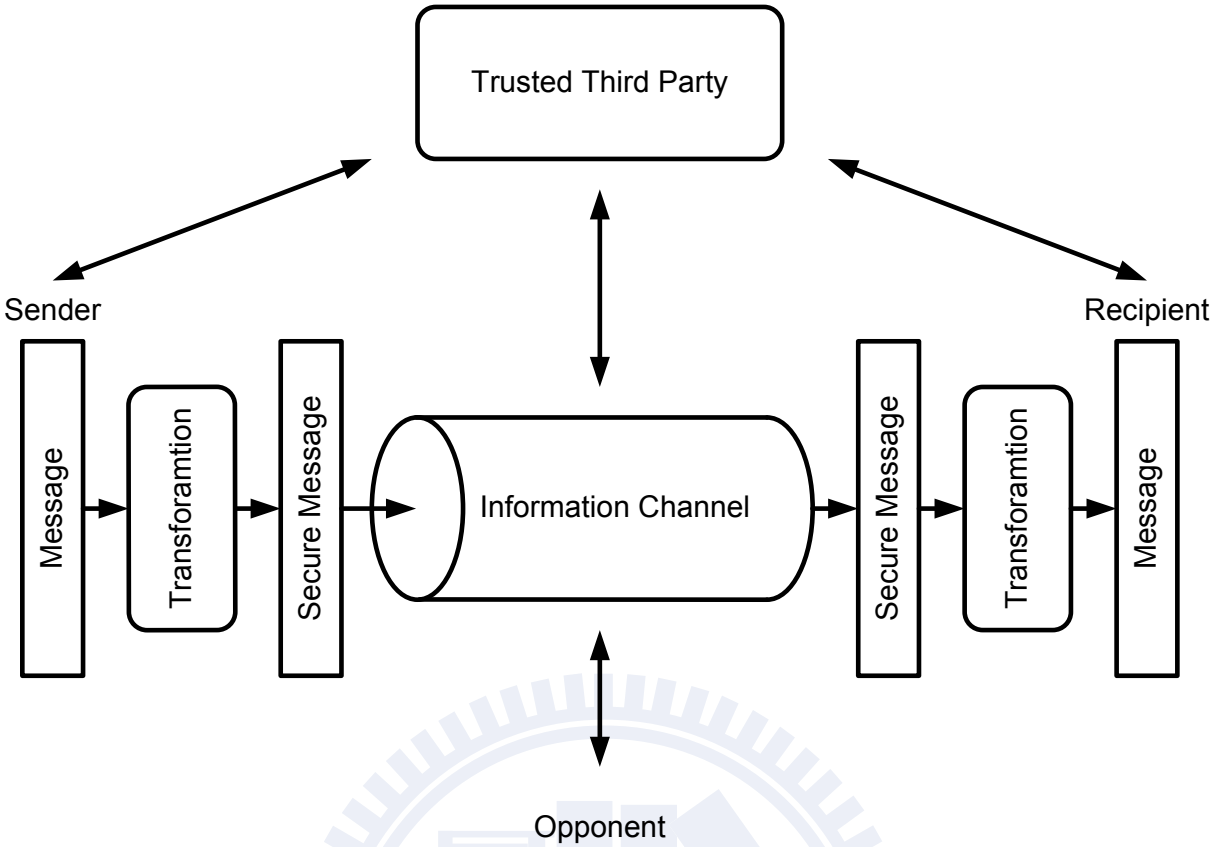


Figure 1.1: A model of network security.

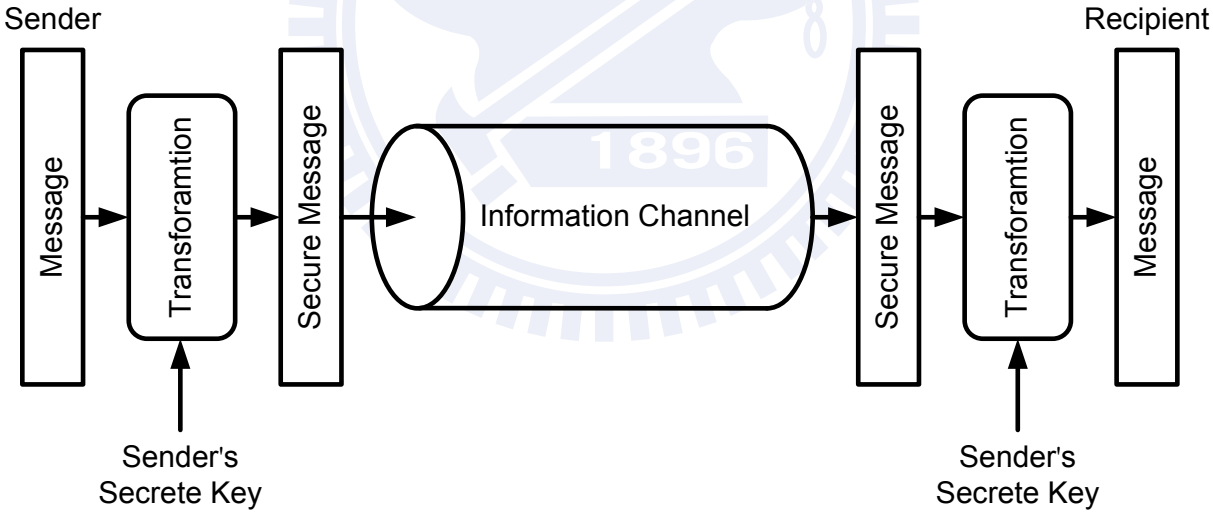


Figure 1.2: A model of symmetric-key encryption.

On the other hand, *asymmetric-key encryption* is developed to achieve the encryption and decryption with two different keys in which one is a public key and another one is a private key. It is also known as *public-key cryptosystems* (PKC). In contrast to

the symmetric-key encryption, asymmetric-key algorithms are based on the mathematical functions rather than on the substitution and transposition. For one thing, although the asymmetric-key ciphers can achieve the same function of encryption, the symmetric-key ciphers will not be abandoned because the computational overhead of current asymmetric-key encryption schemes. Usually, the both of symmetric-key and asymmetric-key encryption schemes are used together in a security system. For example, the short message such as encryption key is securely generated from the combination of recipient's public key and sender's private key by asymmetric-key ciphers in an open channel, and then the long message such as plaintext is scrambled by symmetric-key ciphers. In this case, only the corresponding recipient who has the correct private key can unscramble the ciphertext, where the decryption key can be obtained from the combination of sender's public key and recipient's private key in the similar way. In addition to the message encryption, due to the flexibility of using the public/private key pair, the PKC have profound consequences in the area of confidentiality and authentication. For more practical examples, they can be referred to the standardized applications, such as WPAN, NFC, SSL, and PGP. An example model of PKC is shown in Figure 1.3. Note that the PKC are an efficient approach to solve the problem of key distribution for the symmetric-key encryption, where the encryption key is usually assumed to be unknown for the recipient before establishing the communication session.

The traditional achievable method for the PKC is RSA [4], which was publicly described in 1977 by Ron Rivest, Adi Shamir, and Leonard Adleman. The difficulty of attacking RSA is based on the hard problem of finding the big prime factors of a composite number. To provide a sufficient security, the key size is usually selected to be several thousands of bits. This big key size results in a high complexity in computation, and it is inconvenient for user in practical implementation. According to these, in 1985, *elliptic curve cryptography* (ECC) is independently discovered by Victor Miller [5] and Neal Koblitz [6] to be an alternative scheme for PKC. Its security is based on the hardness of a different problem, namely the *elliptic curve discrete logarithm problem* (ECDLP). Currently, the best algorithms known to solve ECDLP have fully exponential running time, in contrast to the subexponential-time algorithms known for the *integer factorization*. This means that a desired security level can be attained with significantly smaller keys

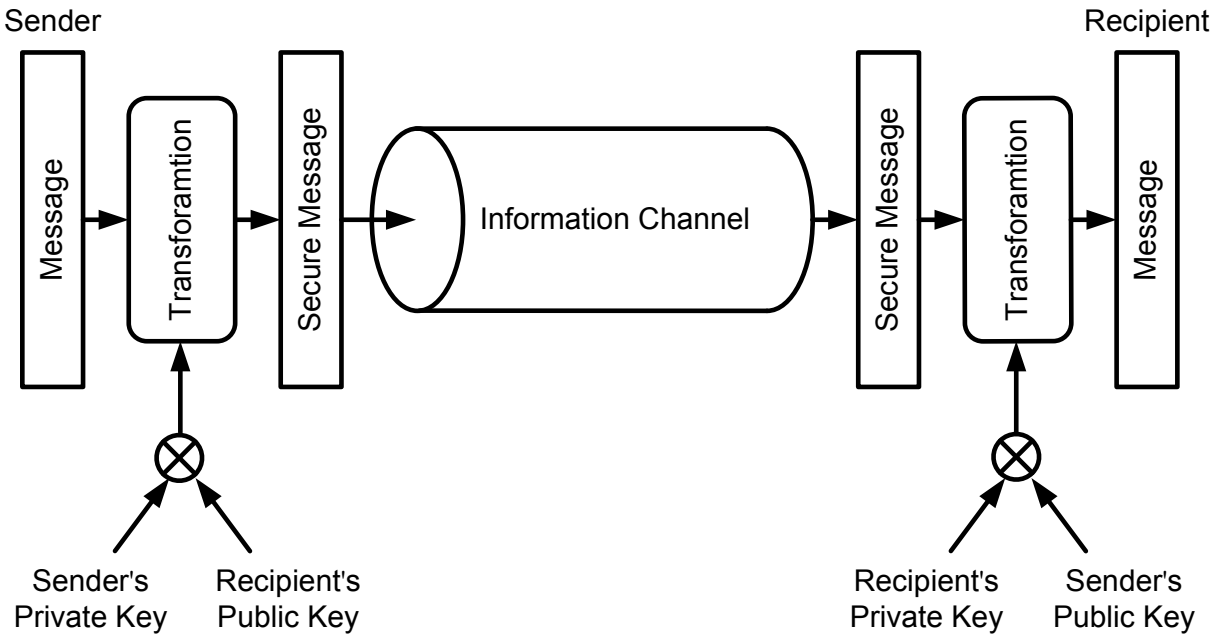


Figure 1.3: A simplified model of PKC.

in elliptic curve systems than those of RSA. For instance, it is generally accepted that a 160-bit elliptic curve key provides the same level of security as a 1024-bit RSA key. The advantages that can be gained from smaller key size include speed and efficient use of power consumption, transmission bandwidth, and memory storage.

The security protocol based on ECC schemes has been specified in IEEE standards, IEEE P1363 [7], in 2000 with an extended version [8] appeared in 2004. The ECC is also applied in practical commercial electronic products using IEEE 802.15.4 [9] and IEEE 802.15.6 [10]. They are the standards in physical layer for currently existing low power and low cost solution of ZigBee [11], Bluetooth low energy [12], and wireless body area networks [13], which are extensively used in industry, business, and medical treatment, respectively. Moreover, high-speed crypto engine is indispensable for the ubiquitous applications of computing server. The hardware accelerator of ECC, so called ECC processor, is dedicated to reducing the system retard from high computation complexity of ECC functions. Instead of conventional performance-oriented design methods, a current issue for delivering a decent crypto engine is the device security such as the protection of *side-channel attacks* (SCAs). In [14], a demonstration shows that the key in circuit device can be easily broken by power measurement, while the suitable solution of SCA resistance for ECC processor has not been discovered much yet.

1.1 Previous Works

1.1.1 Elliptic Curve Cryptographic (ECC) Processor

To date, several works of the ECC hardware implementation have been published in [15–28]. To save hardware complexity, single finite field architecture either for prime field $GF(p)$ [17, 19, 21, 26, 29] or extension binary field $GF(2^m)$ [15, 25, 27], and fixed modulus approach on specific elliptic curves (ECs) [20–22] can be used. However, the applications of IEEE P1363 including digital signature are approved for supporting dual-field (DF) functions on arbitrary ECs. Exploiting carry-save adder trees in word-based multipliers is a common technique to integrate DF data path [16, 24, 28], but the limit of integration for distinct arithmetic units still results in large hardware cost.

In general, the $GF(2^m)$ design is faster than the $GF(p)$ design because of carry-free addition over $GF(2^m)$. Besides, there are some well-known techniques to pursue high-speed $GF(2^m)$ ECC design. A divide-and-conquer algorithm, *Karatsuba-Ofman (KO) multiplication* [30], is applied to reduce the computation complexity of number of bit operations. Classical methods to multiply two m-bit polynomials require $\mathcal{O}(m^2)$ bit operations. The KO algorithm reduces this to $\mathcal{O}(m^{\log_2 3})$. As the polynomial modulus is fixed, the reduction over $GF(2^m)$ is simple [31], and then the throughput of KO multiplier can be elevated by adopting fully pipelining architecture [32, 33]. Another design technique over $GF(2^m)$ using fixed polynomial modulus is the *fast squaring* [34]. The binary representation of a polynomial $a(z)^2$ is obtained by inserting a zero-bit between consecutive bits of the binary representation of $a(z)$. Thus the computation complexity is most dominated by the reduction over $GF(2^m)$, which is easily achieved by combinational circuit using exclusive-OR gates only. In contrast to standard (polynomial) representation of elements over $GF(2^m)$, *optimal normal basis* (ONB) representation [7, 34] has benefits in squaring because it can be achieved by simple shifting operations. But it is inevitable for the computing overhead of conversion between the standard and ONB representation.

For arithmetic over $GF(p)$, based on Chinese remainder theorem, *residue number system* (RNS) [35, 36] represents a large integer using a set of smaller integers, so that computation may be performed more efficiently. This briefs the long delay within the data path of carry-propagation adder, and the multiple multipliers can be implemented

with parallelism. RNS implementations bear the extra cost of an input converter (binary-to-RNS) to translate numbers from a standard binary format into residues and an output converter (RNS-to-binary) to implement the translation from RNS to a binary representation. An RNS implementation applied to $GF(p)$ ECC processor is presented in [23], where the technique of data flow graph for the optimization of ECC function is utilized as well.

For the implementation of scalable architecture performing flexible field length and arbitrary modulus, *Montgomery algorithm* [37] is commonly adopted. It is an efficient approach to achieve the modular multiplication over DFs, where the long-precision integer division is not required during the calculation of Montgomery multiplication (or called Montgomery modular multiplication). The key idea is that the reduction after integer multiplication can be achieved by shifting bit position as the domain constant is selected to be two to the power of m or x with degree m (i.e., 2^m over $GF(p)$ and x^m over $GF(2^m)$), where the constant 2^m and x^m is so called Montgomery constant. Another benefit for the hardware implementation of Montgomery algorithm is that the $GF(p)$ and $GF(2^m)$ arithmetic logic unit (ALU) is suitable for integration in VLSI circuit because the sum of carry-save adder is equal to two bitwise exclusive-OR operators [15, 27, 38]. The overhead is the multiplexer to select the data path between operating fields. In [39, 40], a word-based Montgomery multiplier is presented to avoid the high fanout of AND operators in conventional serial-parallel architecture [15]. In [16, 24, 41], a $w \times w$ multiplier is exploited to tradeoff between the hardware speed and area cost with flexible size w. As w equals field length m, one modular multiplication can be performed within several cycle periods [17,42]. Note that, although the Montgomery algorithm still requires the overhead of conversion between integer and Montgomery domain, it can be immediately achieved by Montgomery division described in [38].

For high speed target, a usually adopted technique is the parallel computation with multiple processing elements (PEs) of *homogeneous* architecture [18, 24, 43]. However, in practice, this approach by directly duplicating the PEs has less hardware utilization for various operations. Another approach of improving computation speed of ECC processor is the *window methods* [34]. The key idea is to store some pre-computed data in device, and then the on-line running time can be reduced.

On the contrary, the parallel computation and window methods requiring the overhead of device memory would not be suitable for the low power and low cost applications such as radio-frequency identification (RFID). ISO/IEC 18000-3 [44] is an international standard for the item level identification of the passive RFID, and it also describes the parameters for air interface communications at 13, 56 MHz. Several previous works [22,29, 45, 46] are targeted at the implementation of low hardware complexity. In [45], a 192-bit $GF(p)/GF(2^m)$ ECC processor supporting hash function [47] and consuming less than 30 μW is reported, while the execution time is over 1 second per operation due to low operating frequency 175 kHz. In [46], the $GF(2^m)$ fast squaring approach is exploited to efficiently computed inversion in *affine coordinates*. In [29], a 192-bit $GF(p)$ ECC processor is presented, where a radix-4 Montgomery multiplication approach is used and the inversion is achieved by extended Euclidean algorithm [34]. In [22], a 163-bit $GF(2^m)$ ECC design with micro-controller and bus manager is implemented to connect to the front-end module in RFID device. A dedicated register file management is used to save the high complexity of multiplexers. To further save the number of temporary register, a *common Z projective coordinate system* modified from [48] is exploited.

To pursue the embedded system market, in [49], a hardware/software co-design of ECC processor is implemented and performed at 12 MHz on an 8051 micro-controller. Communication overhead due to operand transfers is reduced by integration of a direct memory access unit and through the inclusion of an additional I/O register into the hardware accelerator. In [50], a cryptographic core compliant with the IEEE 802.15.4 standard [9] and based on FPGA is described. It consists of three components including an AES-CCM module, a content-addressable memory achieving an access control list, and an RSA module based on Montgomery arithmetic.

1.1.2 Side-Channel Attacks (SCAs)

Traditional cryptanalysis assumes that an adversary only has access to input and output pairs without the knowledge about internal states of the device. However, the advent of side-channel analysis showed that a cryptographic device can leak critical information. By monitoring the timing, power consumption, electromagnetic emission of the device or by injecting faults, adversaries can obtain the information about internal processed data

or operations, and then the key is extracted out of the cryptographic device without mathematically breaking the primitives. This kind of attacks using side-channel information is so called side-channel attacks (SCAs).

In 1999, Kocher [51] has presented a real threat on the hardware device by power measurement. The detailed description for the attacks on symmetric-key crypto engine is given in [14], and the *power-analysis attacks* are successfully conducted on the microprocessor, ASIC, and even FPGA. The common techniques against power-analysis attacks for symmetric-key crypto engine are the dual-rail logic cell equalizing the power consumption and the masking in substitution which depends on the key value. The previous one needs to change the design flow including the back-end physical layout to ensure interconnect capacitances of the true and false output nodes of logic gates are equal; the last one requires the overhead of hardware speed and cost from combinational circuit. Several published papers [52–55] show other kinds of logic cells to "balance" the power consumption. On the other hand, a systematic overview for most of currently existing SCAs and countermeasure on asymmetric-key design is reported in [56]. However, most of the previous approaches illustrate the theoretical analysis rather than real implementation together with measurement results. In Chapter 3, we will give more description about the principle and show the evaluation of power-analysis attacks on ECC device from power measurement.

1.1.3 Summary of Paper Survey

The research age of ECC hardware implementation is briefly shown in Figure 1.4. The ECC processor with small key size and single field has less hardware complexity [22, 25, 29, 49], but it sacrifices the security. The DF design [24, 28, 45, 57, 58] and large key size approach [38, 59] have higher security level. However, there is still relatively little design targeted at the applications such as cloud computing and portable device, where the both of flexibility and device security are necessary.

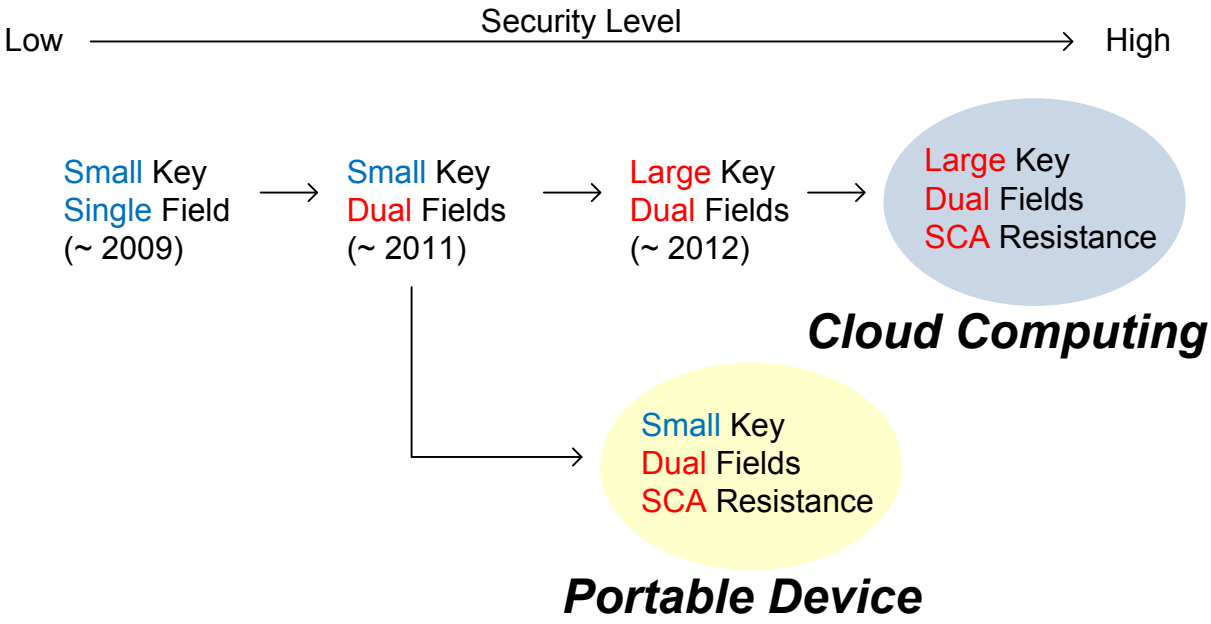


Figure 1.4: Research review of ECC hardware implementation.

1.2 Motivation and Design Challenge

As described in sub-section 1.1, the suitable solution of ECC processor to provide hardware efficiency against SCAs has not so far appeared. In our work, not only the performance but also the practical applications are taken into consideration. For instance, the speed is a key factor for server computing. But the RFID device and portable applications are targeted at the requirements of low power and low cost. These would bring a big difficulty to the hardware designer due to the trade-off between speed and cost for current design approaches.

The following are to list the items about our design target:

1. Low SCA-resistant overhead of speed, cost, power and no modification of circuit design flow
2. Performance improvement from delivering a new hardware architecture
3. Compliance with current standards, such as IEEE P1363 and IEEE 802.15.4/6
4. A high-speed ECC design for the cloud computing
5. An energy-efficient and cost-effectiveness ECC design for the portable applications

1.3 Our Solution

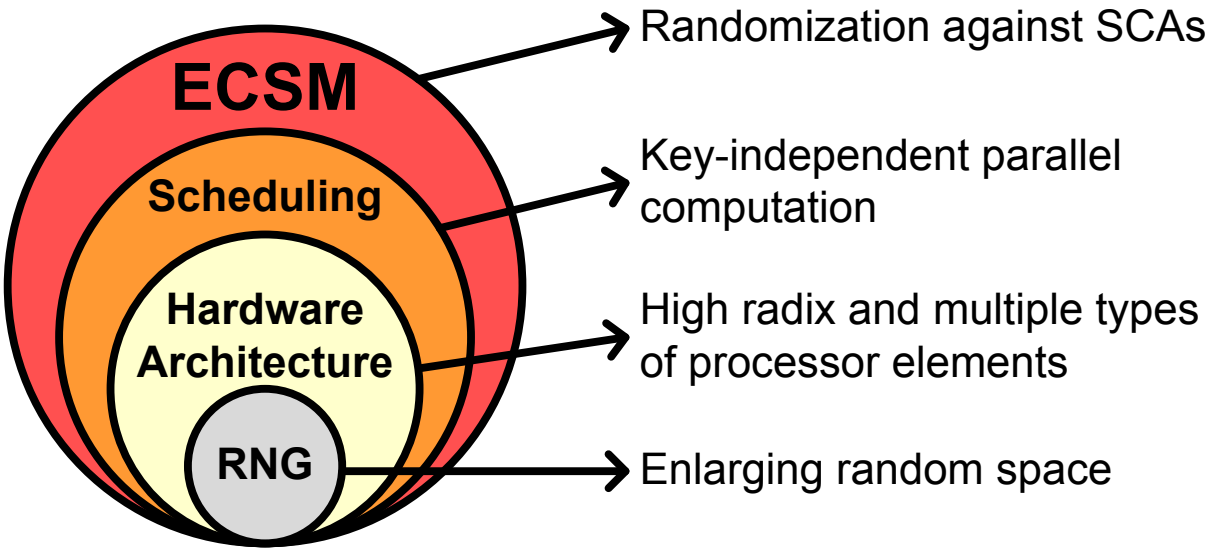


Figure 1.5: Our top-to-down design methods of SCA-resistant ECC processor.

Figure 1.5 briefly illustrates our proposed solution for the design objective. In the upper-level view, we try to randomize the processed data and schedule the operation tasks in the key-independent manner for breaking the dependence on attacking model. The noticeable things are that these methods would not bring much modification for both of the hardware architecture and circuit design flow, and little overhead is added to the protected design. For hardware components, the high-radix and *heterogeneous* processing element architecture is used to accelerate the modular operations with utilization improvement as compared to the conventional approaches. The reconfigurable computing is exploited by arithmetic unit integration for the reduction of hardware complexity. Besides, for the multiple processing element design, memory hierarchy is adopted to address the data bandwidth with benefits in power saving. Finally, we use circuit-level design techniques to improve the randomization ability of random number generator (RNG) in which the robustness against SCAs is achieved.

1.4 Dissertation Organization

The remainder of this dissertation is outlined as follows. Chapter 2 reviews the basics of PKC and arithmetic of ECC over finite field. Chapter 3 presents the principle and evaluation of various SCAs on ECC processor. Our proposed countermeasure of SCAs and hardware design of random source are introduced in Chapter 4 and Chapter 5, respectively. For the proposed processing elements, operation scheduling, parallel computation, and memory architecture of ECC processor, they are given in Chapter 6. Chapter 7 shows the implementation and experiment results of our ECC processor with performance analysis and power measurement. Finally, Chapter 8 concludes our work and gives several new research targets for the future as well.

Chapter 2

An Overview of Cryptographic Algorithms

2.1 Public-Key Cryptosystems (PKC)

Public-key cryptosystems (PKC) refer to a cryptographic system requiring two separate keys, one of which is secret and one of which is public. Although different, the two parts of the key pair are mathematically linked. One key locks or encrypts the plaintext, and the other unlocks or decrypts the ciphertext. Neither key can perform both functions by itself. The public key may be published without compromising security, while the private key must not be revealed to anyone not authorized to read the messages.

PKC use asymmetric-key algorithms and can also be referred to by the more generic term "asymmetric-key encryption." The algorithms used for PKC are based on mathematical relationships that presumably have no efficient solution. The most notable ones being the integer factorization and *discrete logarithm problem* (DLP). Although it is computationally easy for the intended recipient to generate the public and private keys, to decrypt the message using the private key, and easy for the sender to encrypt the message using the public key, it is extremely difficult or effectively impossible for anyone to derive the private key, based only on their knowledge of the public key. This is why, unlike symmetric-key algorithms, a public-key algorithm does not require a secure initial exchange of one or more secret keys between the sender and receiver. The use of these algorithms also allows the authenticity of a message to be checked by creating a digital

signature of the message using the private key, which can then be verified by using the public key. In practice, only a hash of the message is typically encrypted for signature verification purposes.

There are three primary kinds of PKC: public-key distribution systems, digital signature systems, and public-key cryptosystems, which can perform both public key distribution and digital signature services. *Diffie-Hellman key* (DHK) exchange is the most widely used public-key distribution system, while the *digital signature algorithm* (DSA) is the most widely used digital signature system.

For the history of PKC, the pioneering paper by Diffie and Hellman [60] presented an approach to cryptography and challenged cryptologists to come up with a cryptographic algorithm that met the requirements for public-key systems. The first achievable method is the RSA [4]. It is a block cipher in which the plaintext and ciphertext are integers between 0 and $n-1$ for some n. Plaintext is encrypted in blocks, with each block having a binary value less than some number n. A typical size for n is 1024 bits or 309 decimal digits. The following are the brief description of the RSA algorithm.

For some plaintext block M and ciphertext block C, encryption and decryption of RSA are of the following form.

$$\begin{cases} C = M^e \pmod{n} \\ M = C^d \pmod{n} = (M^e)^d \pmod{n} = M^{ed} \pmod{n}. \end{cases}$$

Both sender and receiver must know the value of n. The sender knows the value of e, and only the receiver knows the value of d. Thus, this is a public-key encryption algorithm with a public key of $PU = \{e, n\}$ and a private key of $PR = \{d, n\}$. For this algorithm to be satisfactory for public-key encryption, the following requirements must be met.

1. It is possible to find values of e, d, n such that $M^{ed} \pmod{n} = M$ for all $M < n$.
2. It is relatively easy to calculate $M^e \pmod{n}$ and $C^d \pmod{n}$ for all values of $M < n$.
3. It is infeasible to determine d given e and n.

The preceding relationship holds if e and d are multiplicative inverses modulo $\phi(n)$, where $\phi(n)$ is the Euler's totient function. For p, q prime, $\phi(pq) = (p-1) \times (q-1)$. The

relationship between e and d can be expressed as $ed \pmod{\phi(n)} = 1$. This is equivalent to saying $ed \equiv 1 \pmod{\phi(n)}$ and $d \equiv e^{-1} \pmod{\phi(n)}$. That is, e and d are multiplicative inverses $\pmod{\phi(n)}$. Note that, according to the rules of modular arithmetic, this is true only if d (and e) is relatively prime to $\phi(n)$ (i.e., $\gcd(\phi(n), d) = 1$).

We are now ready to state the RSA scheme. The ingredients are the following.

- p, q two prime numbers (private, chosen)
- $n = pq$ (public, calculated)
- e, with $\gcd(\phi(n), e) = 1$ and $1 < e < \phi(n)$ (public, chosen)
- $d \equiv e^{-1} \pmod{\phi(n)}$ (private, calculated)

The private key consists of $\{d, n\}$ and the public key consists of $\{e, n\}$. Suppose that user Alice has published her public key and that user Bob wishes to send the message M to Alice. Then Bob calculates $C = M^e \pmod{n}$ and transmits C. On receipt of this ciphertext, user Alice decrypts by calculating $M = C^d \pmod{n}$.

For the security of RSA, there are three approaches to attacking RSA mathematically.

1. Factor n into its two prime factors. This enables calculation of $\phi(n) = (p-q) \times (q-1)$, which, in turn, enables determination of $d \equiv e^{-1} \pmod{\phi(n)}$.
2. Determine $\phi(n)$ directly, without first determining p and q. Again, this enables determination of $d \equiv e^{-1} \pmod{\phi(n)}$.
3. Determine d directly, without first determining $\phi(n)$.

Most discussions of the cryptanalysis of RSA have focused on the task of factoring n into its two prime factors. Determining $\phi(n)$ given n is equivalent to factoring n. With presently known algorithms, determining d given e and n appears to be at least as time-consuming as the factoring problem [61]. Thus, we can use factoring performance as a benchmark against which to evaluate the security of RSA.

For the size of n, a number of other constraints have been suggested by researchers. To avoid values of n that may be factored more easily, the algorithm's inventors suggest the following constraints on p and q.

1. p and q should differ in length by only a few digits. Thus, for a 1024-bit key, both p and q should be on the order of magnitude of 10^{75} to 10^{100}.

2. Both $(p-1)$ and $(q-1)$ should contain a large prime factor.

3. $\gcd(p-1, q-1)$ should be small.

The key size of 1024 bits was generally considered the minimum necessary for the RSA encryption algorithm. However, it would result in high complexity of hardware cost and time execution. Figure 2.1 shows the comparison of security strengths for ECC versus RSA. It is shown that the key size of ECC can be several tens of times shorter than that of RSA with equivalent security. This also means that the user has convenience in using the shorter key by ECC approach.

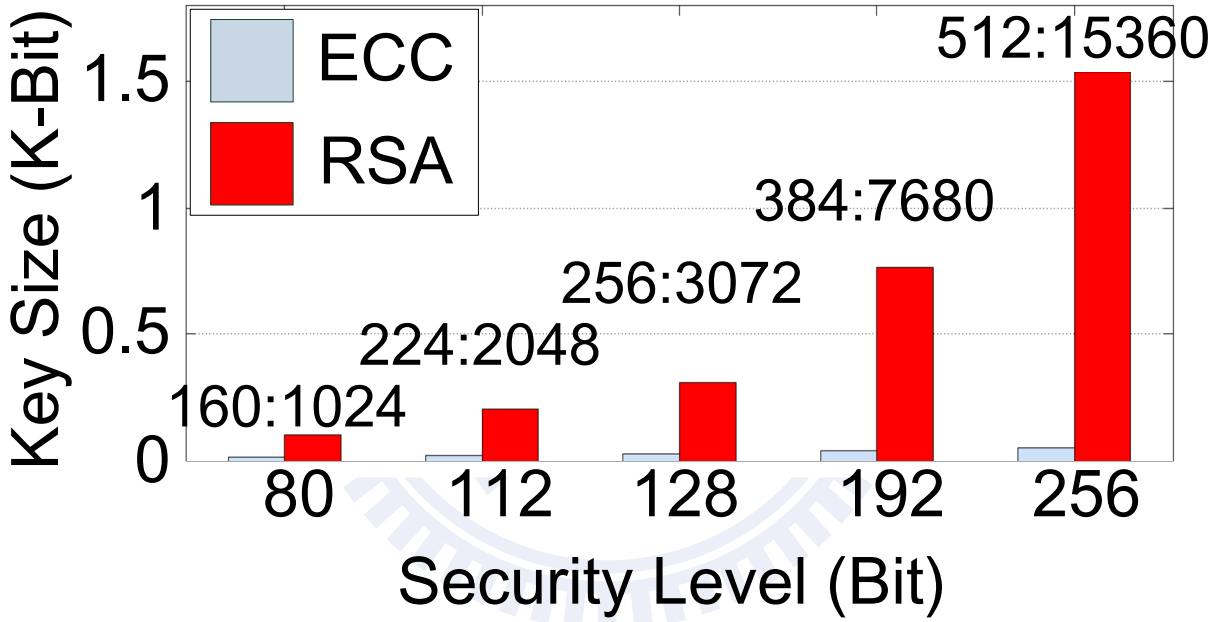


Figure 2.1: Security comparison of ECC versus RSA.

2.2 Arithmetic of Elliptic Curve Cryptography (ECC) over $GF(p)$ and $GF(2^m)$

As described in IEEE P1363 [7], the standardized elliptic curve (EC) over $GF(p)$ is $y^2 = x^3 + a_p x + b_p$, where $x, y \in GF(p)$ and $4a_p^3 + 27b_p^2 \neq 0 \pmod p$, and the other one over $GF(2^m)$ is $y^2 + xy = x^3 + a_b x^2 + b_b$ with $x, y \in GF(2^m)$ and $b_b \neq 0$. For the ECC schemes, the discrete logarithm problem (DLP) is based on the *elliptic curve scalar multiplication* (ECSM) such that $KP = P + P + \cdots + P$ with an integer private key K and a point $P(x, y)$ on EC. The ECSM is applied in ECC as a means of producing a trapdoor function. Thus the security of ECC depends on the intractability of determining K from $Q = KP$ given known values of Q and P. The fundamental theorem of arithmetic about ECC is described in the guide books [62, 63].

For implementation, the ECSM is the most time-critical operation, and it can be achieved by the serial EC point addition and doubling (ECPA and ECPD) with binary method. Note that the ECPA is to perform $P_3(P_{3_x}, P_{3_y}) \leftarrow P_1(P_{1_x}, P_{1_y}) + P_2(P_{2_x}, P_{2_y})$ with $P_1 \neq \pm P_2$, and the ECPD calculates $P_3(P_{3_x}, P_{3_y}) \leftarrow 2P_1(P_{1_x}, P_{1_y})$ with $P_1 \neq -P_1$. The dual-field (DF) arithmetic of ECPA and ECPD in affine coordinates is summarized in Table 2.1, where the EC point subtraction (ECPS) can be achieved by performing the ECPA with modification of coordinate values such as $P(x, y) \rightarrow -P(x, -y)$ over $GF(p)$ and $P(x, y) \rightarrow -P(x, x + y)$ over $GF(2^m)$.

Table 2.1: Formulas of EC Point Calculation (ECPC) in Affine Coordinates

Field	ECPA	ECPD
$GF(p)$	$\lambda = \frac{P_{1_y} - P_{2_y}}{P_{1_x} - P_{2_x}}$ $P_{3_x} = \lambda^2 - P_{1_x} - P_{2_x}$ $P_{3_y} = \lambda(P_{2_x} - P_{3_x}) - P_{2_y}$	$\lambda = \frac{3P_{1_x}^2 + a_p}{2P_{1_y}}$ $P_{3_x} = \lambda^2 - 2P_{1_x}$ $P_{3_y} = \lambda(P_{1_x} - P_{3_x}) - P_{1_y}$
$GF(2^m)$	$\lambda = \frac{P_{1_y} + P_{2_y}}{P_{1_x} + P_{2_x}}$ $P_{3_x} = \lambda^2 + \lambda + P_{1_x} + P_{2_x} + a_b$ $P_{3_y} = \lambda(P_{2_x} + P_{3_x}) + P_{3_x} + P_{2_y}$	$\lambda = P_{1_x} + \frac{P_{1_y}}{P_{1_x}}$ $P_{3_x} = \lambda^2 + \lambda + a_b$ $P_{3_y} = \lambda(P_{1_x} + P_{3_x}) + P_{3_x} + P_{1_y}$

The ECPC can be implemented in several coordinate systems, where the computational complexity analysis can be referred to [64] and [22]. The major operations of ECPC over both $GF(p)$ and $GF(2^m)$ are summarized in Table 2.2, where the notation of MD, MM, and MS represents the modular division, multiplication, and squaring, respectively. Note that the $GF(2^m)$ MS with a fixed irreducible polynomial [34] can be performed within relative fewer cycles than those of the MD and MM, but the fixed irreducible polynomial method restricts the flexibility and results in the low security. Since our work is targeted at supporting the arbitrary irreducible polynomial, the MS is regarded as the MM with the same multiplier and multiplicand. From comparison Table 2.2, it can be found that as the time ratio $\frac{\text{MD}}{\text{MM}}$ is smaller than 3, the ECSM performance is the fastest in the affine coordinates over DFs. Otherwise, the computation time is less in the *projective coordinates*.

Table 2.2: Operations for ECPC over DFs in Various Coordinates

Field	ECPC		Modular Operations
$GF(p)$	ECPD	$A \leftarrow 2A$	1MD + 1MM + 2MS
		$SP \leftarrow 2SP$	7MM + 5MS
		$J \leftarrow 2J$	4MM + 6MS
		$J_{\mathrm{m}} \leftarrow 2J_{\mathrm{m}}$	4MM + 4MS
		$J_{\mathrm{C}} \leftarrow 2J_{\mathrm{C}}$	5MM + 6MS
	ECPA	$A \leftarrow A + A$	1MD + 1MM + 1MS
		$SP \leftarrow SP + SP$	12MM + 2MS
		$J \leftarrow J + J$	12MM + 4MS
		$J_{\mathrm{m}} \leftarrow J_{\mathrm{m}} + J_{\mathrm{m}}$	13MM + 6MS
		$J_{\mathrm{C}} \leftarrow J_{\mathrm{C}} + J_{\mathrm{C}}$	11MM + 3MS
		$J \leftarrow J + A$	8MM + 3MS
		$J_{\mathrm{m}} \leftarrow J_{\mathrm{m}} + A$	9MM + 5MS
		$J_{\mathrm{C}} \leftarrow J_{\mathrm{C}} + A$	8MM + 3MS
$GF(2^m)$	ECPD	$A \leftarrow 2A$	1MD + 1MM + 1MS
		$LD \leftarrow 2LD$	4MM + 1MS
		$LD_{\mathrm{m}} \leftarrow 2LD_{\mathrm{m}}$	5MM + 1MS *
	ECPA	$A \leftarrow A + A$	1MD + 1MM + 1MS
		$LD \leftarrow LD + LD$	2MM + 4MS
		$LD_{\mathrm{m}} \leftarrow LD_{\mathrm{m}} + LD_{\mathrm{m}}$	2MM + 3MS

A:affine, SP:standard projective, J:Jacobian, J_{m}:modified Jacobian,

J_{C}: Chudnovsky Jacobian, LD:López-Dahab, LD_{m}:modified López-Dahab.

* The respective coordinates z are unequal values.

2.3 Specifications for Applications

2.3.1 IEEE P1363

The standard IEEE P1363 [7] with its extension version [8] specify several primitives based on ECC to achieve the cryptographic schemes. For the key agreement schemes, the primitives include *elliptic curve secret value derivation primitive Diffie-Hellman* (ECSVDP-DH) [5, 6, 65] and *elliptic curve secret value derivation primitive Menezes-Qu-Vanstone* (ECSVDP-MQV) [66]. For the signature schemes with appendix, the primitives include *elliptic curve signature primitive Nyberg-Rueppel* (ECSP-NR) [5,6, 67] and *elliptic curve signature primitive digital signature algorithm* (ECSP-DSA) [5,6,68]. In addition, *elliptic curve integrated encryption scheme* (ECIES) [69] is adopted to implement encryption and decryption.

2.3.2 IEEE 802.15.4/6

As specified in the IEEE 802.15.4 [9], the symmetric-key cryptographic algorithm uses block cipher AES [3] with three operation modes; that is, *counter* (CTR), *cipher block chaining message authentication code* (CBC-MAC), and *CTR with CBC-MAC* (CCM) [70]. Also, it is applied to conduct the security schemes involving with encryption, authentication, and message integrity, respectively. In addition to two AES operation modes, *cipher-based message authentication code* (CMAC) [71] and CCM exerted in IEEE 802.15.6 [10], an asymmetric-key cryptographic algorithm based on ECC [7] is adopted to achieve the message exchange with *Diffie-Hellman key* (DHK) agreement [65] on an open channel.

AES algorithm

As described in [3], the AES cipher processes a 128-bit plaintext block with either 128, 192, or 256-bit secret key to generate a 128-bit ciphertext block. The design with larger key size provides higher security level but it has more processed cycles. A *round* is the basic transformation function in AES algorithm, and the number of rounds for one AES encryption depends on the key size. Key sizes 128, 192, and 256-bit refer to 10, 12, and 14 rounds respectively for single 128-bit input message. The round function consists of four

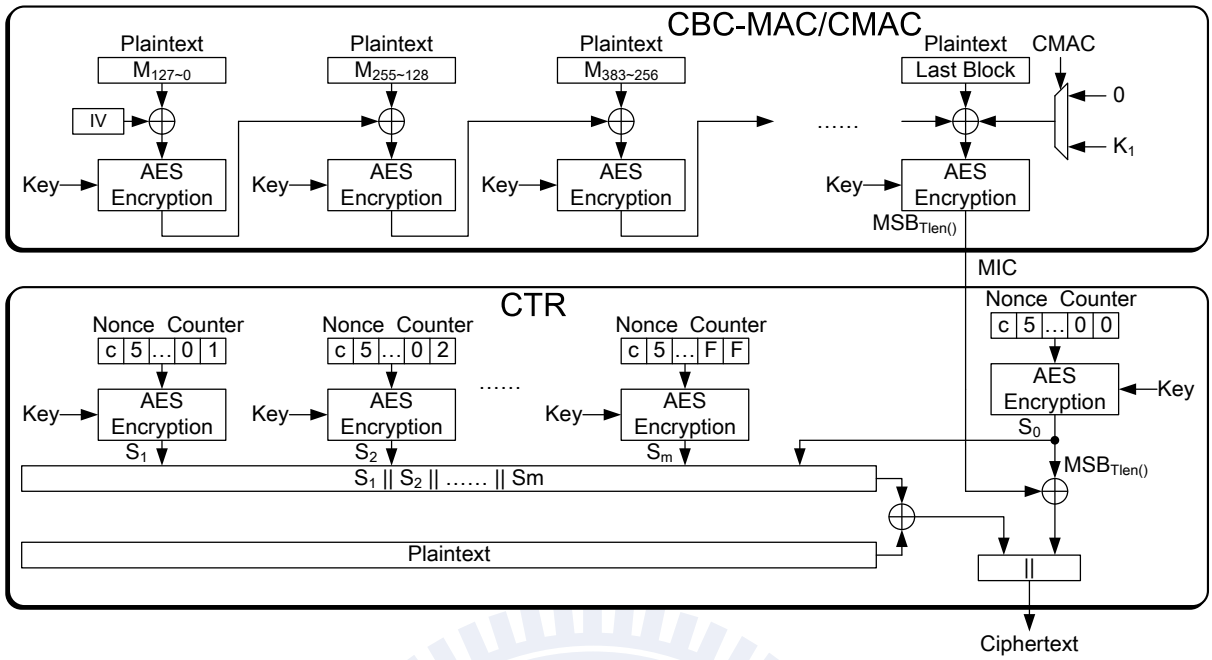
basic transformations: `SubByte`, `ShiftRow`, `MixColumn`, and `AddRoundKey`, except for the last round, which is without `MixColumn`. The `KeySchedule` algorithm expends the secret key in a word-oriented fashion, and it generates a 128-bit round key every round to add the *state* value by a simple bit-wise exclusive-OR operation in `AddRoundKey`, where the state value is a 16 8-bit temporary data for AES round calculation.

Encryption, authentication, and message integrity

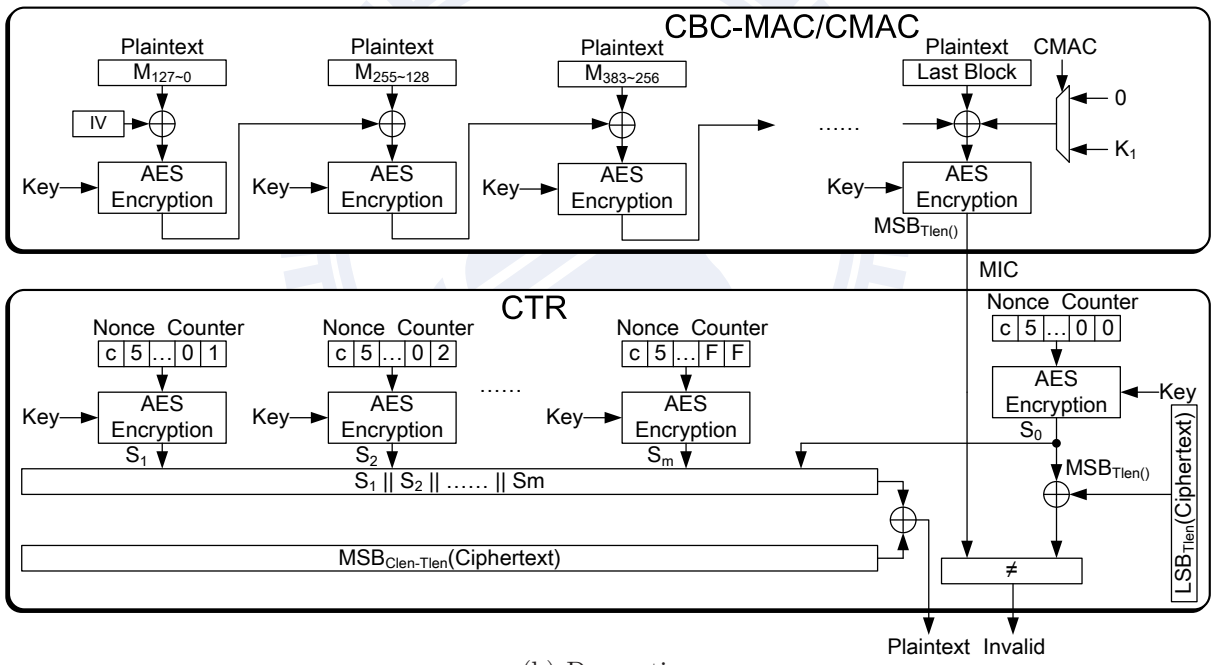
Figure 2.2(a) and Figure 2.2(b) show the AES schemes including encryption (or decryption), authentication, and message integrity by using CTR, CBC-MAC/CMAC, and CCM modes, respectively. In the CTR mode, the plaintext is encrypted by performing bit-wise exclusive-OR logic operator with a block-stream ciphertext, which is produced from the AES output by feeding in a block message consists of nonce and counter. Note that the data flow of decryption in CTR mode is the similar with that of encryption. For the CBC-MAC and CMAC modes, a *message integrity code* (MIC) is produced by a chain reaction of AES encryption for detecting any tampering in the plaintext. For achieving the message integrity scheme (i.e., authenticated encryption), the CCM mode is efficiently implemented by a combined operation of CTR and CBC-MAC modes.

Message exchange with DHK agreement

Figure 2.3 shows the procedure before message exchange between two parties communicating over an insecure channel based on well-known DHK agreement [65]. Address A and Address B represent the media access control address of Alice and Bob, respectively, and security suite indicates the security level of cipher function. Note that both of the public-key generation and DHK agreement can be achieved by performing the ECSM from a selected private key. As communicating in an open channel, the delivered message is encrypted and decrypted by using the AES CCM mode based on a *master key* (MK), which is refreshed and activated when a new party is joining in the network.



(a) Encryption



(b) Decryption

Figure 2.2: The data flow of each AES mode, where the nonce and initial vector (IV) are an arbitrary number and secrete value, respectively. The functional notation $\mathrm{MSB}_{\mathrm{Tlen}}/\mathrm{LSB}_{\mathrm{Tlen}}$ denotes the most/least significant Tlen bits of the data, and Tlen/Clen is the bit length of the MIC/ciphertext.

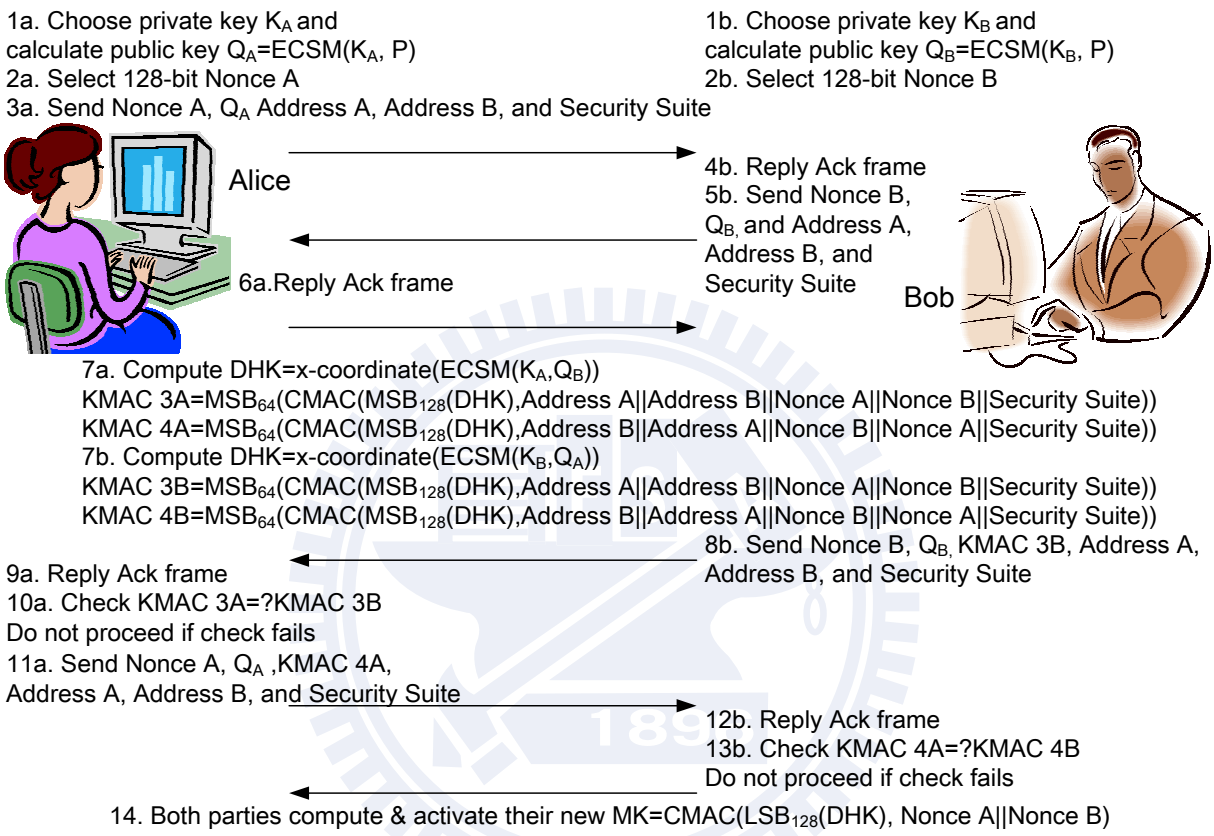


Figure 2.3: Message can be securely sent based on MK to a specific party by using both of asymmetric and symmetric-key algorithm without pre-knowledge encryption and decryption keys.

Chapter 3

Side-Channel Attacks (SCAs)

Modern security systems apply the cryptographic algorithms to provide confidentiality, integrity, and authenticity of data, where the cryptographic algorithms are mathematical functions that usually take two input parameters, including message (also called plaintext) and a cryptographic key. The cryptographic algorithms map these parameters to an output, called ciphertext, and this process is regarded as the encryption. In current cryptography, the cryptographic algorithms are assumed to be known, which means that all details about the cryptographic algorithms are publicly available and only the cryptographic key is kept secret. This notion can be traced back to Auguste Kerckhoffs [72], who was a Dutch cryptographer of the 19th century, and the concept is famous as "Kerckhoffs' principle".

Breaking a cryptographic algorithm typically means that finding the secret key is based on some public information, such as instance pairs of plaintexts and ciphertexts. A cryptographic algorithm is considered to be *secure* in practice if there are no attacks known that can break it within a reasonable amount of time and with a reasonable amount of computing power. Many algorithms are designed such that the effort of breaking them grows significantly or exponentially with the number of bits of the key. Consequently, the length of the key is an important factor in the security of a cryptographic algorithm.

Crypto engines are the electronic devices, such as an application-specified integrated circuit (ASIC), field-programmable gate array (FPGA), or microprocessor, that implement cryptographic algorithms using the keys stored on them. The fact that crypto engines are used to accelerate the cryptographic algorithms, while this leads to a new

issue for the practical security of the algorithms. In practice, not only the security of the cryptographic algorithm should be taken into concern. The security of the whole system, i.e. the crypto engine that implements the cryptographic algorithms, needs to be considered. Breaking a crypto engine means extracting the key of the device. A person who tries to extract the key of a crypto engine in an unauthorized way is the *attacker*, and then any attempt to extract the key in an unauthorized way is viewed as an *attack*. In order to evaluate the security of a crypto engine, it is necessary to make assumptions about the knowledge that an attacker has about it. The strongest assumption is that the attacker is assumed to know the details of the crypto engines.

In recent years, several kinds of attacks on crypto engines have become public. *Side-channel attacks* (SCAs) are the attacks based on information leakage obtained from the physical implementation of cryptosystems, rather than brute force or theoretical weaknesses in the algorithms. In Figure 3.1, for example, the timing information, power consumption, electromagnetic leaks or even sound can provide an extra source of information, which can be exploited to break the system. Among of them, the *power-analysis attacks*, initially presented by Kocher [51], have received such a large amount of attention because they are very powerful and because they can be conducted relatively easily. The basic idea of this kind of attacks is to reveal the key of a crypto engine by analyzing its power consumption. The variation of power consumption is directly to reflect the difference of key-dependent processed data, where the total power consumption P_{total} of a cell is the sum of static power P_{stat} and dynamic power P_{dyn} as shown in Figure 3.2. Consequently, the power-analysis attacks pose a serious threat to the security of crypto engines in practice.

In this dissertation, we have tried our best to investigate the state-of-the-art approaches of power-analysis attacks. They include the *simple power-analysis* (SPA) attacks [51], *differential power-analysis* (DPA) attacks [73], *zero-value power-analysis* (ZPA) attacks [74], and *collision power-analysis* (CPA) attacks [75]. The concepts of them are described in the following sub-sections, and we also show the successful attacks of the power measurement conducted on the devices. Figure 3.3(a) and Figure 3.3(b) show our power-analysis verification environment of the chip, where it is powered by an ECC crypto engine fabricated by UMC 90-nm CMOS technology.

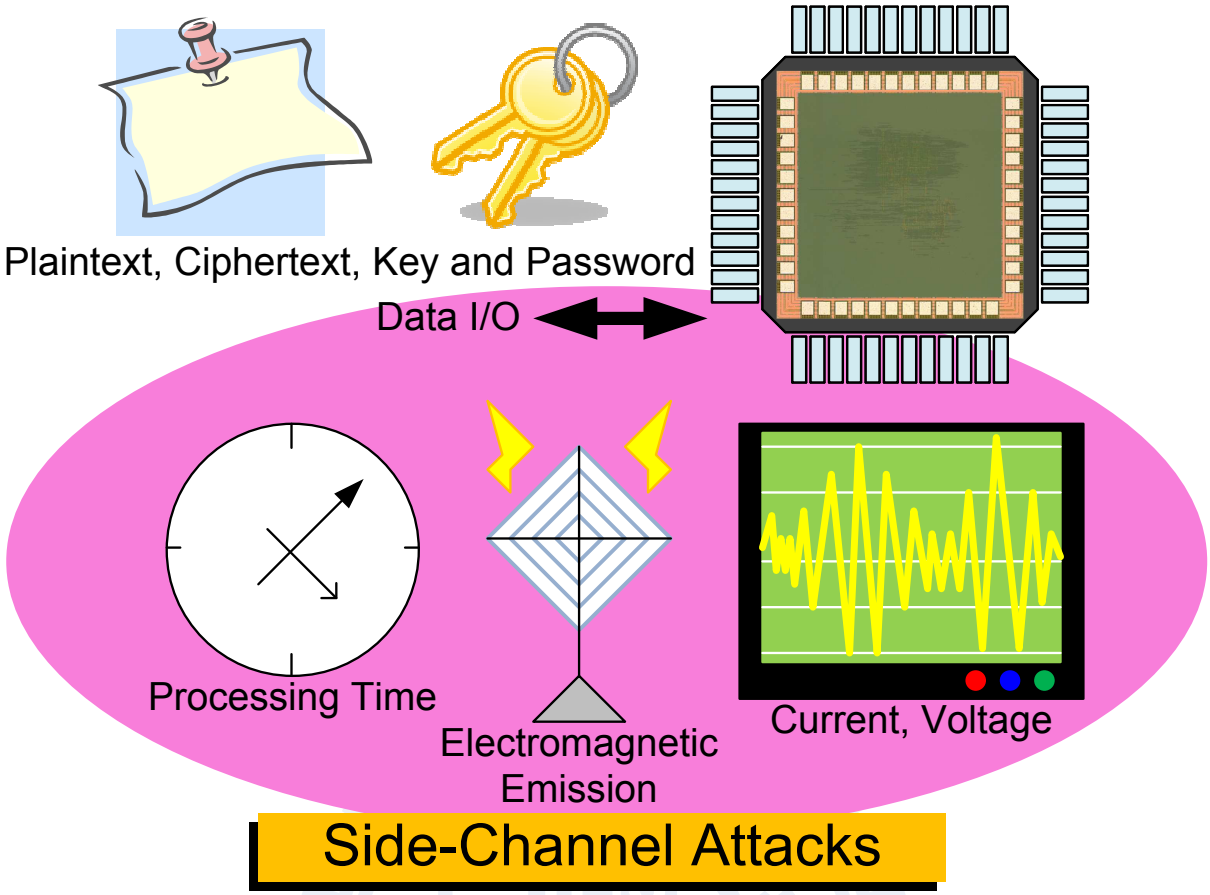


Figure 3.1: Scenario of side-channel attacks on hardware device.

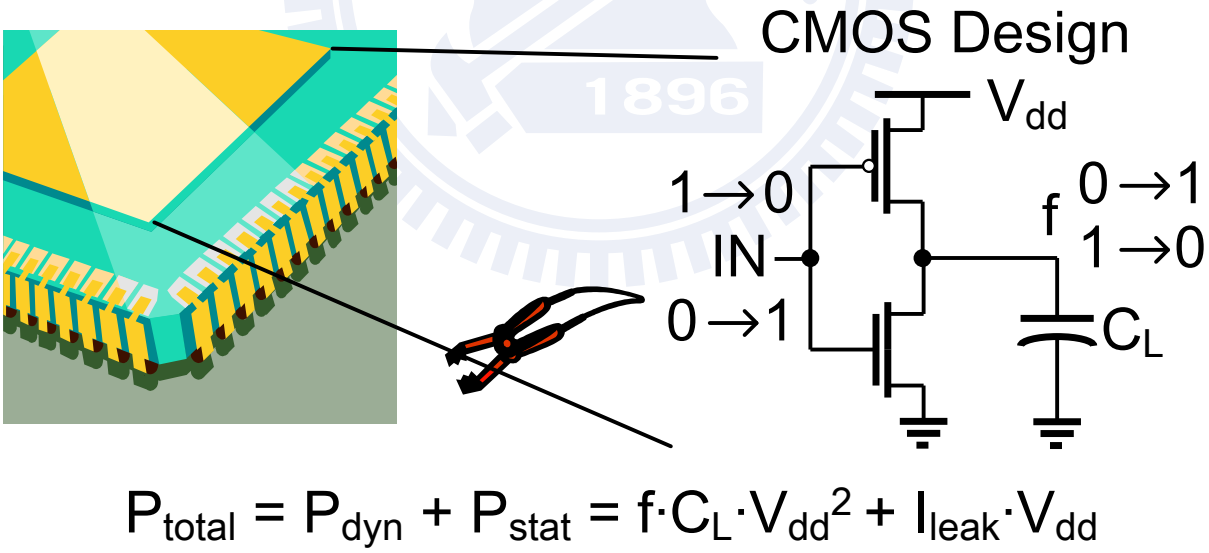
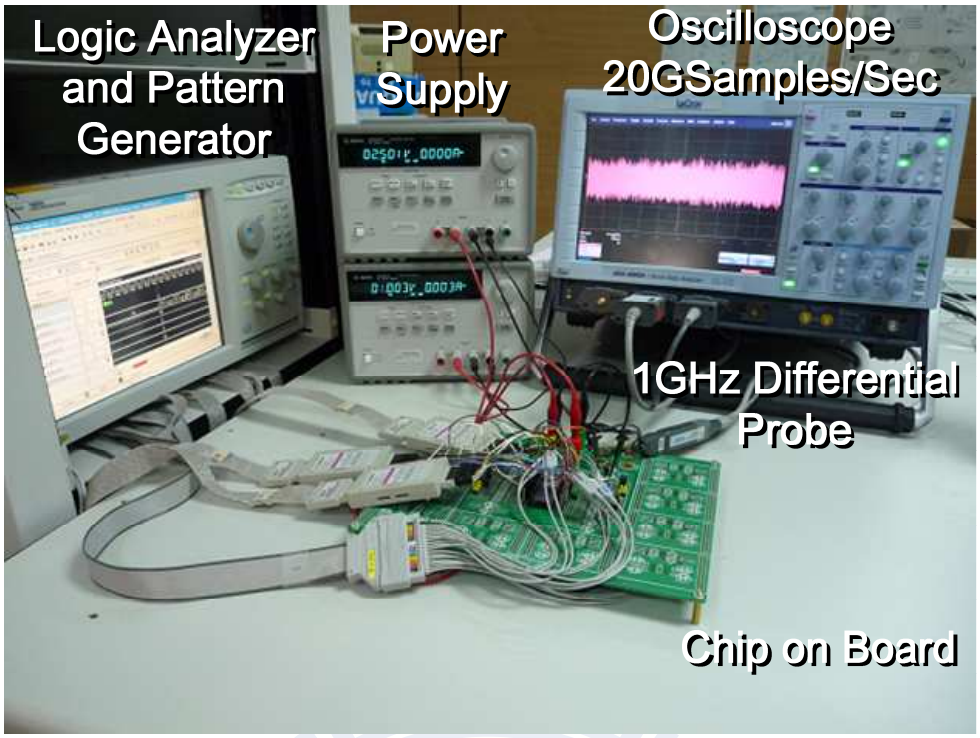
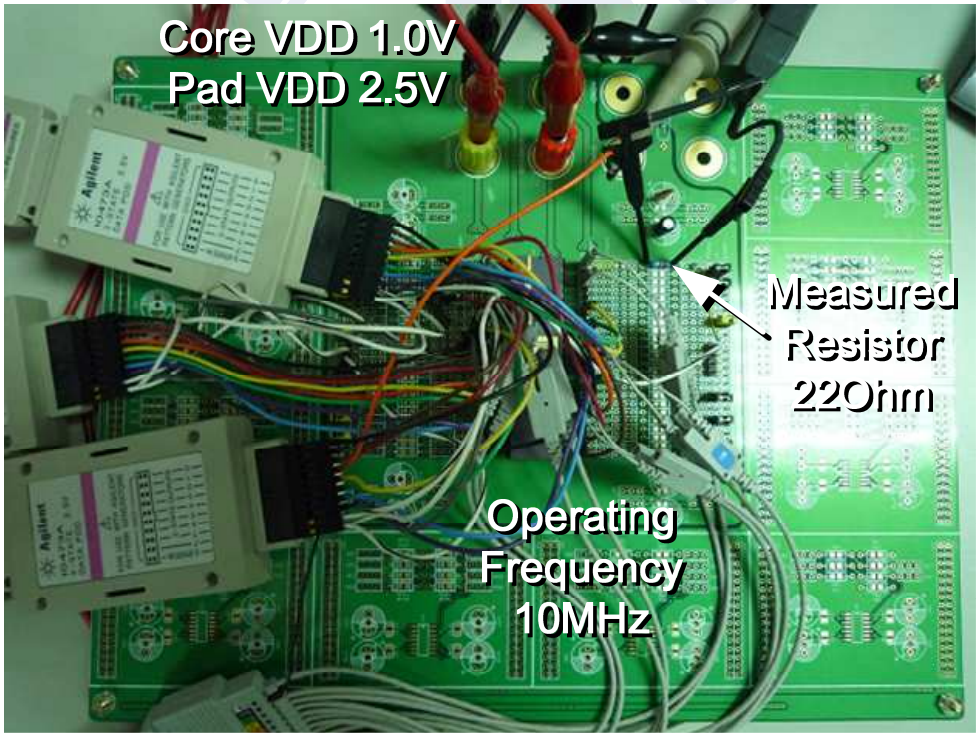


Figure 3.2: Power consumption of CMOS circuits with supply voltage V_{dd} and leakage current I_{leak}.



(a)



(b)

Figure 3.3: (a) Environment of power measurement. (b) Current running through the chip is recorded by measuring the voltage drop via a resistor in series with the core power and supply power.

For a quick preview, the ECC processor is targeted at accelerating the elliptic curve scalar multiplication (ECSM) KP, where K is the private key and P is the point on elliptic curve. Thus the object of power-analysis attacks on ECC processor is to extract the private key K by the measured power traces of ECSM calculation. Since P is usually public, it is reasonable to assume that the attacker has the information about P, and the attacker can control or inject any input values of P as possible.

3.1 Simple Power-Analysis (SPA) Attacks

Simple power-analysis (SPA) is the technique that involves directly interpreting power measurement collected during cryptographic operations. In other words, the attacker tries to derive the key more or less directly from a given power trace. The SPA attacks are useful in practice if only one or very few power traces are available for a given set of inputs. In the attacked device, the key must have a significant impact on the power consumption, otherwise the effectiveness of SPA attacks is reduced by the noise.

Algorithm 1 shows the conventional ECSM by the left-to-right double-and-add (LR-DA) binary method [17]. With this approach, there is a branch in Step 4, where the ECPA depends on the value of i-th bit position of the key K. It means that the execution time of ECSM is correlated to the *hamming weight* of the key, and then the SPA attacks become a threat to reveal the key value through recording power traces over time.

Algorithm 1 LR-DA ECSM

Input: K and P

Output: KP

1: Let $Q_0 \leftarrow 0$

2: **For** i from $m-1$ to 0 **do**

3: $\quad Q_0 \leftarrow 2Q_0$

4: $\quad$ **If** $K_i = 1$ **then** $Q_0 \leftarrow Q_0 + P$

5: **Return** Q_0

Figure 3.4 shows the power traces for different hamming weight of the key over time obtained from an unprotected ECC chip performing LR-DA ECSM in Algorithm 1, where the hamming weight of the key is denoted by H(K). As the chip is processing, it consumes 1.79 mW at 10 MHz, which results in a voltage drop above 50 mV across the measured resistor. From these waveforms, the key value in the chip using LR-DA ECSM can be distinguished by visual inspections because the processing time is dependent on the hamming weight of the key.

As shown in Algorithm 2, the left-to-right double-and-add-always (LR-DAA) ECSM performing the uniformed ECPC in each iteration can resist the SPA attacks [22], but it averagely requires 50% ECPA operation overhead. In sub-section 4.3, we present our

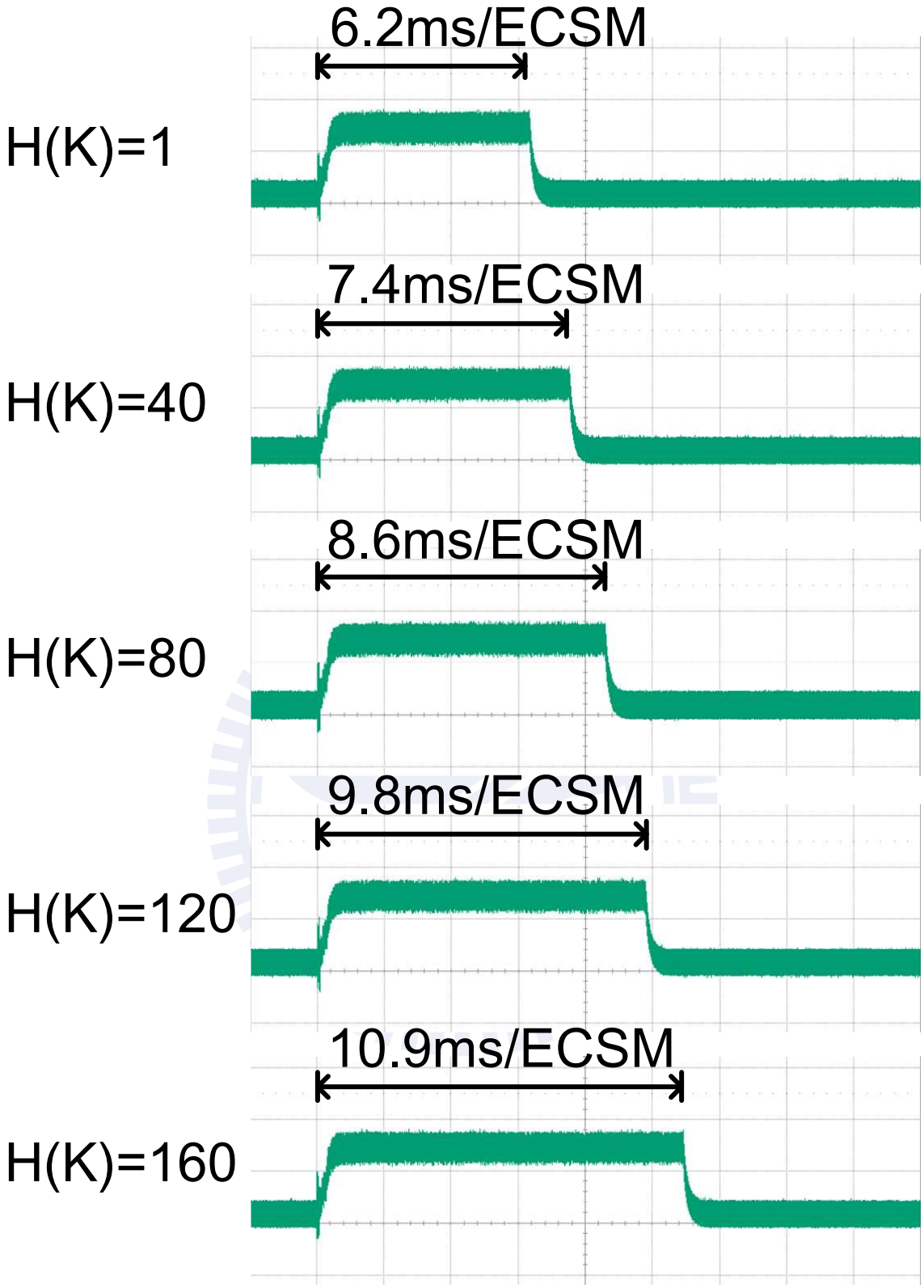


Figure 3.4: SPA attacks on the unprotected ECC chip using LR-DA binary method of ECSM, where the power traces are recorded by 50.0 mV/div voltage resolution and 2.0 ms/div time base.

new method which not only resists the SPA attacks but also has more efficiency in the overhead of computing ECSM.

Algorithm 2 LR-DAA ECSM

Input: K and P

Output: KP

1: Let $Q_0 \leftarrow 0, Q_1 \leftarrow P$

2: **For** i from $m-1$ to 0 **do**

3: $\quad Q_0 \leftarrow 2Q_0$

4: $\quad Q_1 \leftarrow Q_0 + P$

5: $\quad Q_0 \leftarrow Q_{K_i}$

6: **Return** Q_0

3.2 Differential Power-Analysis (DPA) Attacks

In contrast to SPA attacks, *differential power-analysis* (DPA) attacks exploit a large number of power traces to analyze the power consumption at a fixed moment of time as a function of the processed data. In general, even though the execution time is independent on the key value or the noise dominates the device power, the DPA attacks can be conducted on the SPA-resistant crypto engine to extract private information as any key-dependent power still exists. In SPA attacks, the power consumption of a device is mainly analyzed along the time axis. The attacker tries to find patterns or tries to match templates in a single trace. In case of DPA attacks, the shape of the traces along the time axis is not so important. The DPA attacks evaluate how the power consumption at a specific sampling time depends on the processed data. Hence, the DPA attacks focus exclusively on the data dependency of the power traces.

The strategy to apply DPA attacks on crypto engine has four steps.

1. Choosing a key-dependent intermediate result of the executed algorithm.
2. Measuring the reference power consumption to build the power model.
3. Measuring the target power consumption.
4. Comparing the correlation between the target and reference power traces.

For example, to attack the SPA-resistant ECC processor using Algorithm 2, the scenario is shown in Figure 3.5. The intermediate values of ECPD in Steps 3 depend on the zero and non-zero bits of the key value in Step 5. Hence, with a chosen point P, the key value can be distinguished by matching the power trace segment of ECPD calculation.

In Figure 3.6, the correlation coefficients between the target traces and power model for all possible hamming distances of the point coordinate Q_0 in Algorithm 2 are plotted over power traces, and those of the correct and incorrect key hypothesis are plotted in black and gray, respectively. In this case, as more than ten thousand power traces are used, the correlation value of the correct key is the highest one among that of all the other key hypotheses, and then the bit value of key can be found easily. As a result, by this approach, an overall 160 to 521-bit key of ECC device can be extracted within hundreds experiments.

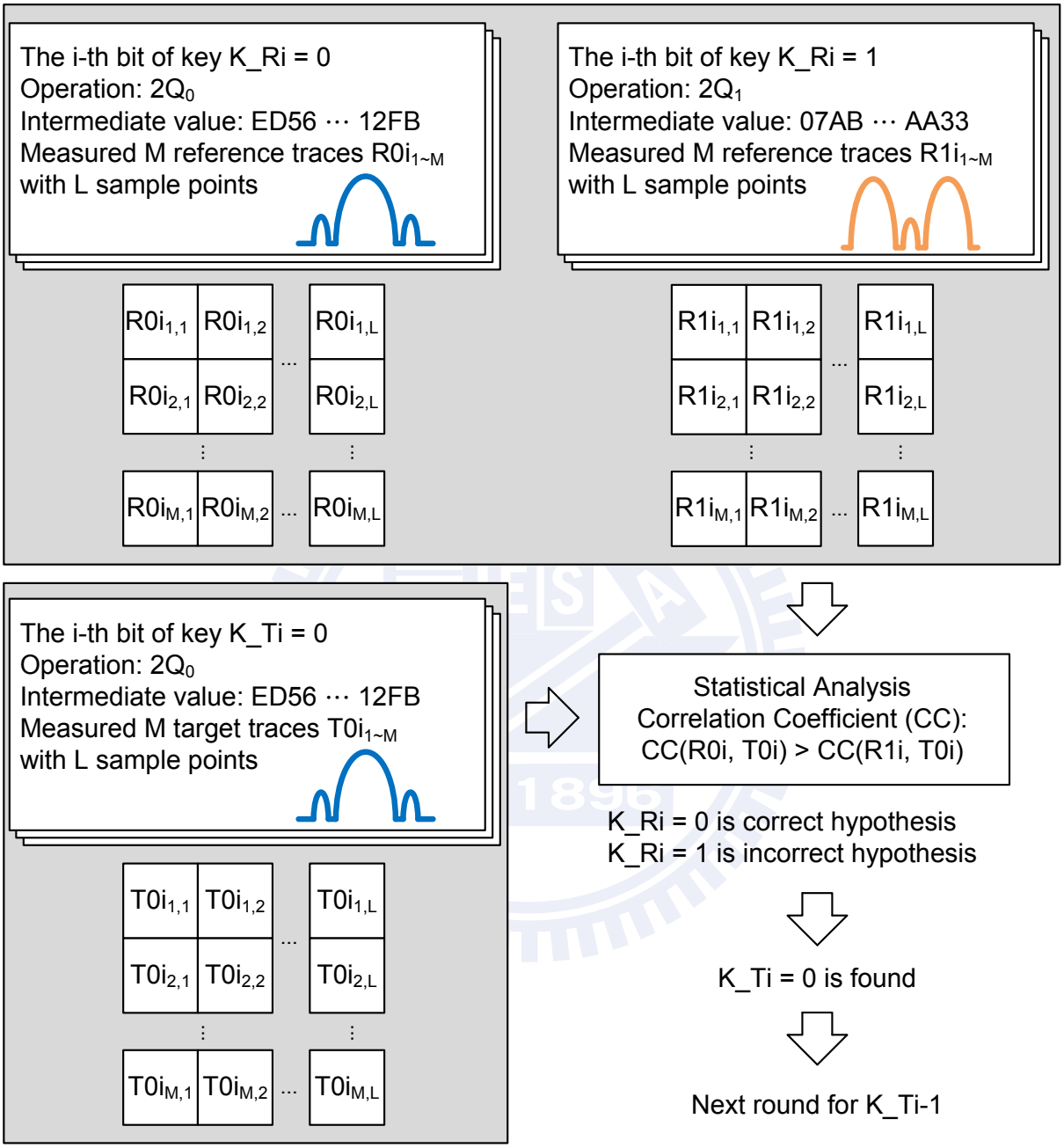


Figure 3.5: DPA attacks on an ECC device.

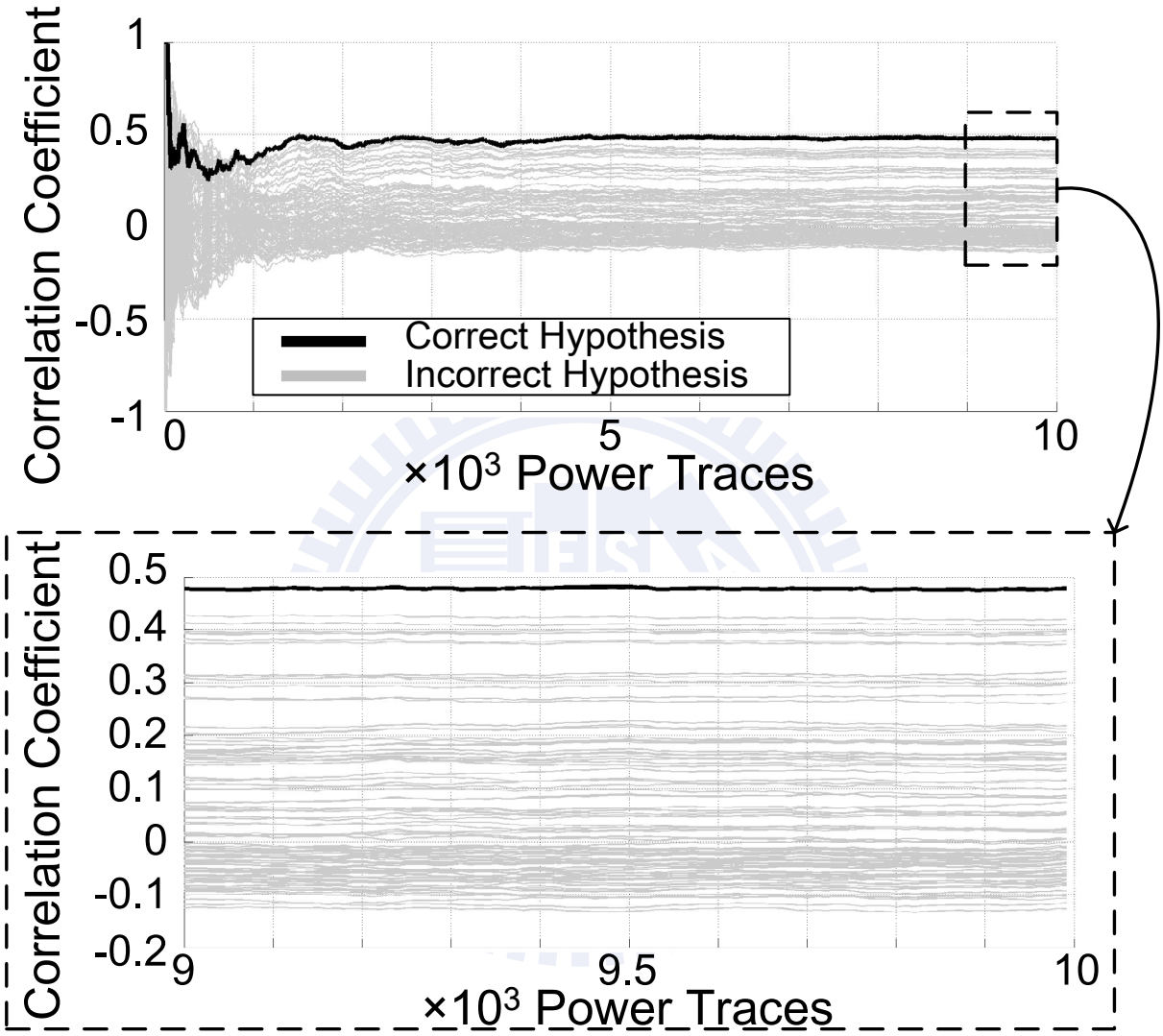


Figure 3.6: Correlation analysis obtained from an unprotected ECC chip by conducting the DPA attacks.

Hiding technique with algorithm-independent dedicated circuit is a common approach to protect crypto engines from attackers collecting the key-dependent characteristics of power traces. In [76], wave dynamic differential logic (WDDL) circuit with regular routing algorithm is exploited to equalize the current between rising and falling transitions. However, more than 200% overhead in area, performance, and power consumption is added to the unprotected crypto engines due to precharging for half cycle, and generating complementary logic outputs from divided single ended modules with equivalent power consumption. Switched capacitor [77] is able to isolate the encryption core from the external power supplies, but this approach results in 50% speed loss for replenishing charge every cycle. In order to avoid the throughput degradation, a countermeasure circuit using digital controlled ring oscillators [78] is designed outside of the critical path. The concept is to generate random noise power to dominate the power consumption of arithmetic unit, and then, the correlation peak would not be found even matching the correct key value. However, this demands extra 100% power overhead for the key-dependent processing element.

At the algorithm level, *masking* the processed data independent of power consumption is another approach to avoid the DPA attacks. For the ECC schemes, since the scalar K of ECPC is periodic with the point order $\#E$, a key-blinded technique proposed by Coron [79] can be adopted to change the key value by adding $\alpha \cdot \#E$ for every ECSM calculation such as $KP = (K + \alpha \cdot \#E)P$, where α is a random integer. However, with this method, the throughput overhead is inevitable due to extending the key length. In [38], the ECSM of 521-bit key extended with a 32-bit random value needs 10% more execution time to be carried out than that of the unprotected approach. Another DPA countermeasure also presented in [79] is to mask the primary base point with the pre-computed random points M and $N = KM$. Then the ECSM is achieved by computing $K(P + M) = KP'$ and subtracting N before returning such that $KP' - N = KP$. For every next ECSM calculation, the random points M and N are refreshed by performing $M \leftarrow (-1)^{\beta}2M$ and $N \leftarrow (-1)^{\beta}2N$ with a single random bit β. But the time-cost random *elliptic curve point generation* (ECPG) is not suitable for real-time applications as the EC parameters are various with different users. In [80], the EC isomorphism method can randomize the primary base point by simple finite field operations without pre-computing

random points. However, it is limited to be applied in single finite field $GF(p)$.

In sub-section 4.1, we present our method against the DPA attacks, and it has benefits in the overhead of the hardware cost, speed, and even the power consumption. Also, our DPA-resistant approach is suitable for DF implementation and standard applications without any pre-computation.

3.3 Zero-Value Power-Analysis (ZPA) Attacks

Zero-value power-analysis (ZPA) attacks generalize the *refined power-analysis* (RPA) attacks presented by Goubin [81], where the RPA attacks focus on the analysis of the existence of zero-value for the coordinates of point P in ECSM calculation, such as $(x, 0)$ and $(0, y)$. In contrast, the ZPA attacks exploit any key-dependent information of zero-value in device calculation. Even if a point has no zero-value point coordinates, the temporary registers or processing elements might take zero-value. A large amount of transition to zero-value in transistor results in a large variation of power consumption due to charging or discharging current. These attacks are available because the point in ECC scheme is usually public. Thus the attacker can control the primary base point P, and then let the chip perform ECSM as usual.

Both of the ZPA and RPA attacks can still extract the key value of the SPA and DPA-resistant device. The unified or double-and-add-always ECSM such as Algorithm 2 and the randomized techniques including random projective coordinates [79], randomized EC isomorphism [80], randomized field isomorphism [80] cannot avoid the key-dependent zero-value. Thus a unified countermeasure of SCAs is necessary.

After applying the correlation analysis, Figure 3.7 shows a successful ZPA attack on an unprotected ECC device, where the correlation value of correct and incorrect hypothesis is plotted in black and gray color, respectively. The correlation value of the correct key is the highest one among that of all the other key hypotheses as the zero-value happens, and then the bit value of key can be revealed. As the attacker intends to collect more information of the key value, then the primary base point P can be changed and apply the ZPA attacks during ECSM calculation until disclosing the overall bit value of the key.

To protect the ZPA attacks, the randomized base point technique [79] can be applied for eliminating the correlation between point coordinates and key value, and it also defeats the fault attacks by injecting low order point [74]. In sub-section 4.2, we present an efficient method to generate the random EC points. Besides, our corresponding implementation architecture is described in sub-section 6.1 and sub-section 6.3.

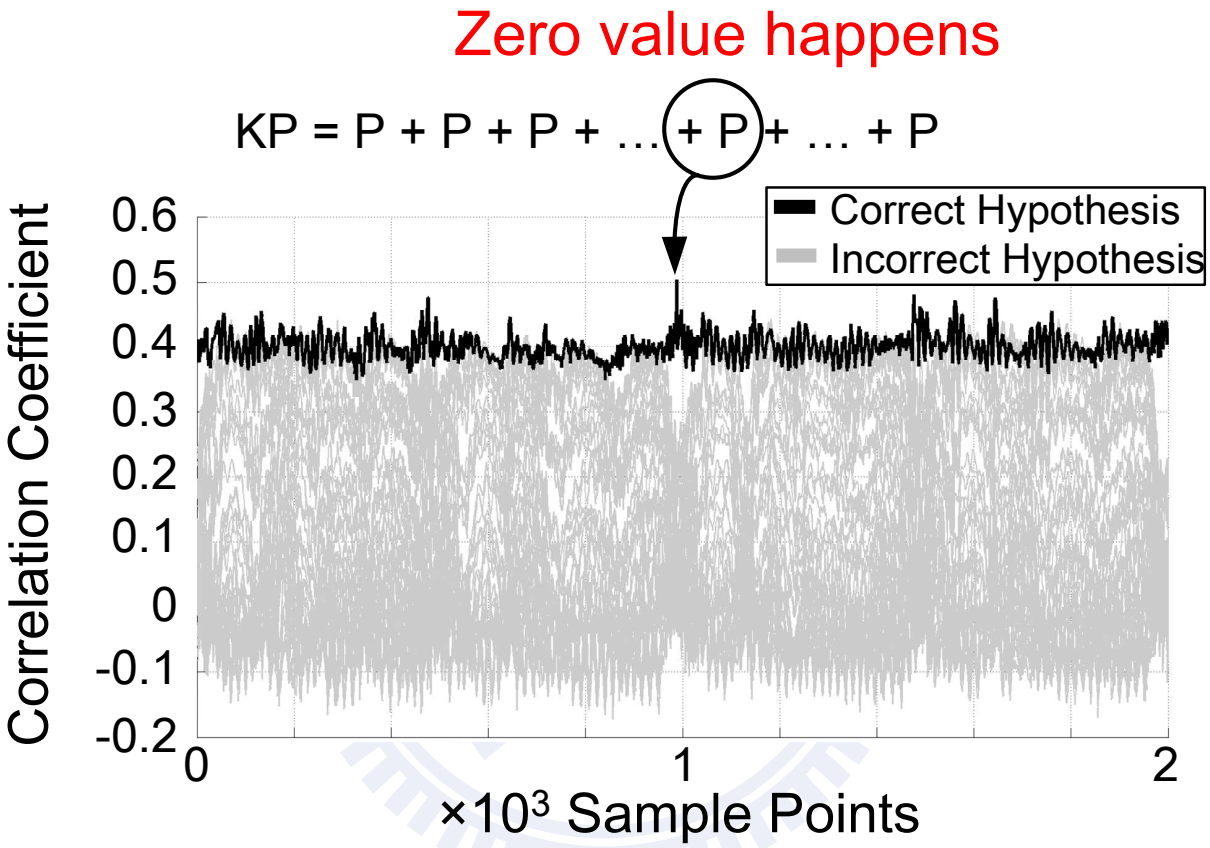


Figure 3.7: Correlation analysis obtained from an unprotected ECC chip by conducting the ZPA attacks.

3.4 Collision Power-Analysis (CPA) Attacks

Collision power-analysis (CPA) attacks use a pre-decided pair of messages (or primary inputs) to generate the key-dependent collisions between their power traces at arbitrary time frames. In ECC scheme, the successful CPA attacks can be conducted to reveal the key value as even the conventional SPA-resistant LR-DAA ECSM in Algorithm 2 is adopted. By injecting a pair of primary input points P and $2P$, the attacker is able to classify the bit value of private key K from matching the power segment waveforms of ECPD operations.

To formally illustrate the CPA attacks on the design using LR-DAA ECSM, the j-th ECPD operations for input points P and $2P$ are given as follows:

$$2(2(\cdots(2(2(2P + K_{m-2}P) + K_{m-3}P) + K_{m-4}P) + \cdots) + K_{m-(j-1)}P)$$

and

$$2(2(\cdots(2(2(2(2P) + K_{m-2}(2P)) + K_{m-3}(2P)) + K_{m-4}(2P)) + \cdots) + K_{m-(j-1)}(2P)),$$

respectively. According to these formulations, if the bit $K_{m-(j-1)}$ is zero value, then the $(j-1)$-th ECPD for the case of input point $2P$ is the same as the j-th ECPD with input point P. On the other hand, if the value of $K_{m-(j-1)}$ is non-zero, the ECPD operations are different due to the ECPA calculation. An example of the CPA attacks for Algorithm 2 is shown in Figure 3.8. As a result, the zero bits and non-zero bits of key value can be distinguished from collisions and non-collisions by comparing the correlation of ECPD power traces.

Input Point	ECPC	K =	1	0	0	1	0	1	1
P	ECPD	Q_0 =	0	2P	4P	8P	18P	36P	74P
	ECPA	Q_1 =	P	3P	5P	9P	19P	37P	75P
2P	ECPD	Q_0 =	0	4P	8P	16P	36P	72P	148P
	ECPA	Q_1 =	2P	6P	10P	18P	38P	74P	150P

Figure 3.8: Example of the CPA attacks for the LR-DAA ECSM.

From the experiment results for unprotected ECC device shown in Figure 3.9, the correlation analysis shows that there are high dependence and independence between the zero

and non-zero bit value of key due to collision and non-collision operations, respectively. The correlation coefficients of the power traces related to the zero and non-zero bits of the key are drawn in circle and star, respectively. The mean of correlation coefficients for zero and non-zero bits is over 0.8 and near 0 due to the key-dependent collisions and non-collisions, respectively. Note that the bit value of the key can be distinguished from a difference of at least 0.4 in correlation coefficients.

To protect the CPA attacks, the right-to-left double-and-add-always (RL-DAA) binary method of ECSM in Algorithm 3 can be applied for eliminating the key-dependent ECPD. However, Algorithm 3 prevents the private key from being revealed by detecting the difference among ECPD operations with specific primary input points, the *read-after-write* scheduling hazard inherently exists in ECPC. The ECPA $Q_{1_i} \leftarrow Q_{0_{i-1}} + Q_{2_{i-1}}$ for i-th iteration in Step 3 can only be processed after finishing the ECPD $Q_{2_{i-1}} \leftarrow 2Q_{2_{i-2}}$ for previous iteration in Step 4. This operand dependency results in a long latency for idling through parallel computations. In sub-section 4.3, we present a new method to explore the parallelism of Algorithm 3. Besides, our corresponding operation scheduling to improve the hardware utilization is described in sub-section 6.2.

Algorithm 3 RL-DAA ECSM

Input: K and P

Output: KP

1: Let $Q_0 \leftarrow 0, Q_1 \leftarrow 0, Q_2 \leftarrow P$
2: **For** i from 0 to $m-1$ **do**
3: $\quad Q_1 \leftarrow Q_0 + Q_2$
4: $\quad Q_2 \leftarrow 2Q_2$
5: $\quad Q_0 \leftarrow Q_{K_i}$
6: **Return** Q_0

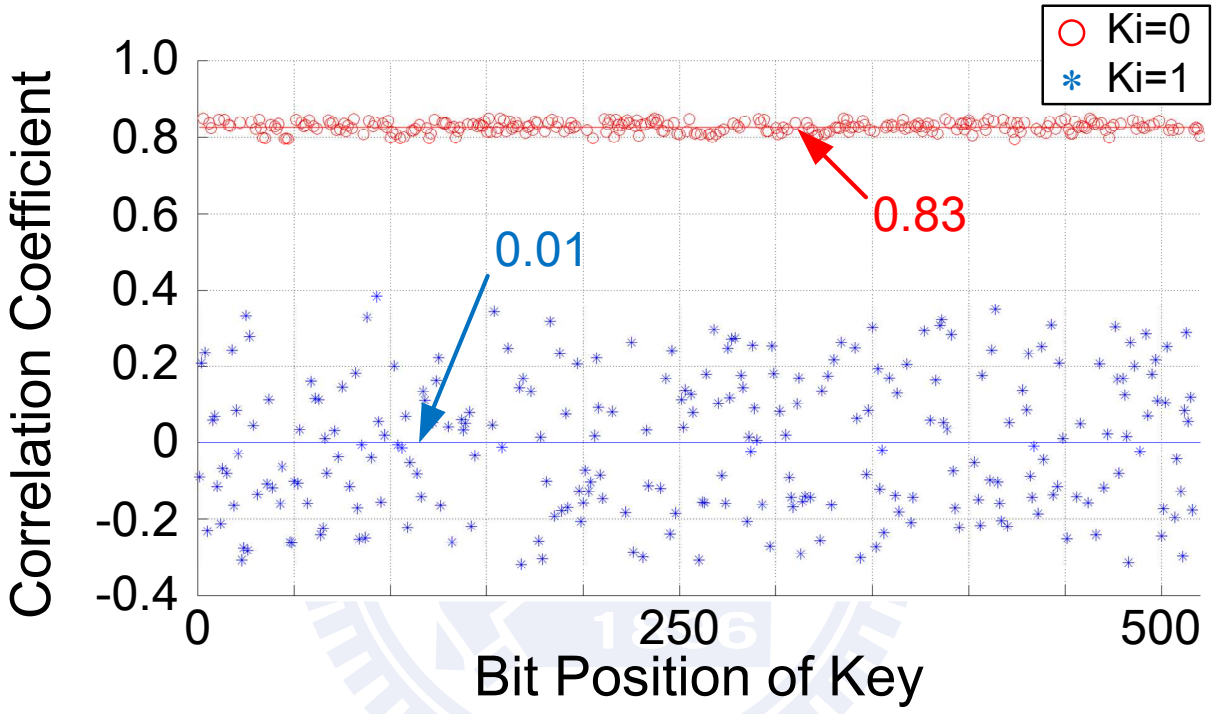


Figure 3.9: Correlation analysis obtained from an unprotected ECC chip by conducting the CPA attacks.

Chapter 4

Proposed Countermeasure of SCAs

4.1 Randomized Montgomery Operations

The fundamental concept of DPA countermeasure is to break the dependency between intermediate values and power traces. For achieving the ECPC, the well-known Montgomery algorithm [37] is usually adopted to perform the field arithmetic in a specific domain such that $A \equiv a \cdot r \pmod{p}$, where a is in the integer domain and $r = 2^m$ is the Montgomery constant with m-bit field length. In this work, we introduce an approach to resist the DPA attacks at modular algorithm by calculating the operands in a *randomized Montgomery domain* $A \equiv a \cdot 2^{\lambda} \pmod{p}$, where it is also a kind of random field automorphism. The domain value λ equals the Hamming weight of an n-bit random value α, and it is represented by $\lambda = \mathrm{H}(\alpha)$. Note that n is the maximum field length and the bit values of $(\alpha_{n-1}, \alpha_{n-2}, \ldots, \alpha_m)$ are set to zero for preventing λ from exceeding m. By exploiting this approach, the intermediate values can be masked because they are various with different domain values such as $2^g \pmod{p} \neq 2^h \pmod{p}$ when $0 \leq g \neq h < m$. The definition of overall *randomized Montgomery operations* for input operands $X \equiv x \cdot 2^{\lambda} \pmod{p}$ and $Y \equiv y \cdot 2^{\lambda} \pmod{p}$ is summarized in Table 4.1.

An example of the randomized Montgomery operations with modulus 7 is shown in Figure 4.1. As $\lambda = 0$, the elements 1 to 6 is aligned as the same as those in the integer domain. As $\lambda = 1$, the elements $(1, 2, 3, 4, 5, 6)$ with $\lambda = 0$ are mapped to the elements $(2, 4, 6, 1, 3, 5)$. Similarly, as $\lambda = 2$, the elements $(1, 2, 3, 4, 5, 6)$ with $\lambda = 0$ are mapped to the elements $(4, 1, 5, 2, 6, 3)$. These mean that the processed data can be randomized by

the proposed approach.

Table 4.1: Operations in Randomized Montgomery Domain

Operation	Arithmetic
Randomized Montgomery multiplication (RMM)	$\text{RMM}(X, Y) \equiv x \cdot y \cdot 2^{\lambda} \pmod{p}$
Randomized Montgomery division (RMD)	$\text{RMD}(X, Y) \equiv x \cdot y^{-1} \cdot 2^{\lambda} \pmod{p}$
Randomized addition (RADD)	$\text{RADD}(X, Y) \equiv (x + y) \cdot 2^{\lambda} \pmod{p}$
Randomized subtraction (RSUB)	$\text{RSUB}(X, Y) \equiv (x - y) \cdot 2^{\lambda} \pmod{p}$

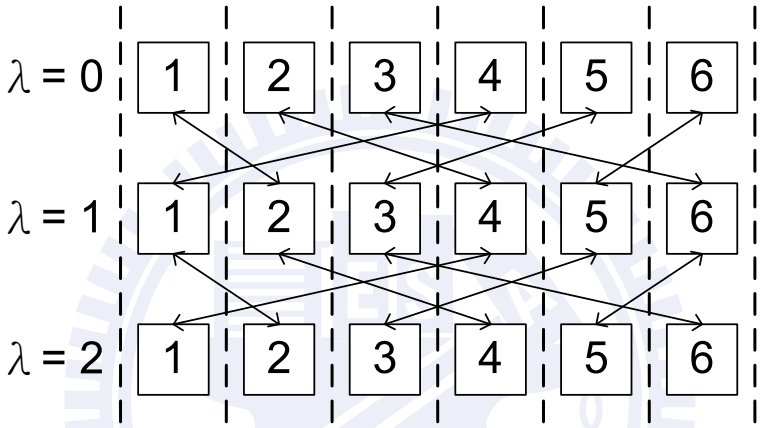


Figure 4.1: Example of randomized Montgomery operations.

4.1.1 Randomized Montgomery Multiplication (RMM)

Algorithm 4 shows our proposed randomized Montgomery multiplication (RMM) which contains two operating steps in every iteration to change the intermediate domain value λ', and these steps are determined by the i-th bit of random value α. If $\alpha_i = 1$, the domain value of output operand R decreases by one in Step 4 such as $R = \frac{R+V_0 \cdot S}{2} \pmod{p}$; the domain value remains the same as $\alpha_i = 0$ in Step 5 such as $R = R + V_0 \cdot S \pmod{p}$. The initial values of operands (V, R, S) are set to be $(X, 0, Y)$. In further iterative calculation, the bit value V_0 is equal to the i-th bit value of X since $V = \frac{V}{2}$, and the operand S doubles its value as $\alpha_i = 0$. Based on these, the functionality can be derived as follows:

- For 1-st iteration, the intermediate result of R is $(X_0 \cdot Y) \cdot 2^{-\alpha_0} \pmod{p}$.

- For 2-nd iteration, R becomes $((X_0 \cdot Y) \cdot 2^{-\alpha_0} \pmod p + X_1 \cdot (2^{1-\mathrm{H}(\alpha_0)} \cdot Y)) \cdot 2^{-\alpha_1} \pmod p$.

- Until m-th iteration, the final result of R is $(\cdots(((X_0 \cdot Y) \cdot 2^{-\alpha_0} \pmod p + X_1 \cdot (2^{1-\mathrm{H}(\alpha_0)} \cdot Y)) \cdot 2^{-\alpha_1} \pmod p + X_2 \cdot (2^{2-\mathrm{H}(\alpha_1,\alpha_0)} \cdot Y)) \cdot 2^{-\alpha_2} \pmod p + \cdots + X_{m-1} \cdot (2^{m-1-\mathrm{H}(\alpha_{m-2},\cdots,\alpha_1,\alpha_0)} \cdot Y)) \cdot 2^{-\alpha_{m-1}} \pmod p$
  $\equiv (X_0 \cdot Y \cdot 2^{-\mathrm{H}(\alpha_{m-1},\ldots,\alpha_0)}) \pmod p + (X_1 \cdot Y \cdot 2^{-\mathrm{H}(\alpha_{m-1},\ldots,\alpha_0)+1}) \pmod p + \cdots + (X_{m-1} \cdot Y \cdot 2^{-\mathrm{H}(\alpha_{m-1},\ldots,\alpha_0)+m-1}) \pmod p$
  $\equiv X \cdot Y \cdot 2^{-\mathrm{H}(\alpha_{m-1},\ldots,\alpha_0)} \pmod p$
  $\equiv X \cdot Y \cdot 2^{-\lambda} \pmod p$.

Hence, the RMM in Algorithm 4 can be performed in m iterations, the same as those in conventional radix-2 Montgomery multiplication [38].

Algorithm 4 Proposed radix-2 RMM

Input: X, Y, p, and α

Output: $R = \mathrm{RMM}(X, Y)$

1: Let $V = X$, $R = 0$, $S = Y$
2: **For** i from 0 to $m - 1$ **do**
3: $\quad R \equiv R + V_0 \cdot S \pmod p, V = \frac{V}{2}$
4: $\quad$ **If** $\alpha_i = 1$ **then** $R \equiv \frac{R}{2} \pmod p$
5: $\quad$ **else** $S \equiv 2S \pmod p$
6: **Return** R

Algorithm 5 shows a radix-4 approach to Algorithm 4 for almost 50% iteration reduction. The domain value of R is determined by the Hamming weight of two continuous bits of random value α in Steps 5, 6, and 7. For the case of $\mathrm{H}(\alpha_{2\mathrm{i}+1}, \alpha_{2\mathrm{i}}) = 2$, it is reduced by two through performing quartering operation such as $R \equiv \frac{R}{4} \pmod p$. While halving R and doubling S operations are performed as $\mathrm{H}(\alpha_{2\mathrm{i}+1}, \alpha_{2\mathrm{i}}) = 1$, these are deduced by computing one iteration of radix-2 Montgomery reduction and one iteration of radix-2 modular reduction in single period. For the rest case of $\mathrm{H}(\alpha_{2\mathrm{i}+1}, \alpha_{2\mathrm{i}}) = 0$, the operand $S \equiv 4S \pmod p$ is performed due to the unchanged domain value of R.

Algorithm 5 Proposed radix-4 RMM

Input: X, Y, p, and α

Output: $R = \text{RMM}(X, Y)$

1: Let $V = X$, $R = 0$, $S = Y$

2: **For** i from 0 to $\lceil \frac{m}{2} \rceil - 1$ **do**

3: **If** $m \pmod 2 \equiv 1$ and $i = \lceil \frac{m}{2} \rceil - 1$ **then** $R \equiv R + V_0 \cdot S \pmod p$, $V = \frac{V}{2}$

4: **else** $R \equiv R + V_0 \cdot S + V_1 \cdot 2S \pmod p$, $V = \frac{V}{4}$

5: **If** $(\alpha_{2i+1}, \alpha_{2i}) = (1, 1)$ **then** $R \equiv \frac{R}{4} \pmod p$

6: **else if** $(\alpha_{2i+1}, \alpha_{2i}) = (1, 0)$ or $(0, 1)$ **then** $R \equiv \frac{R}{2} \pmod p$, $S \equiv 2S \pmod p$

7: **else** $S \equiv 4S \pmod p$

8: **Return** R

4.1.2 Randomized Montgomery Division (RMD)

To achieve the division in Montgomery domain, Kaliski [82] first proposed an iterative algorithm which needs $m \sim 2m$ iterations of successive reduction, $0 \sim m$ iterations for degree recovery (reduce intermediate domain value λ' to be m as $\lambda' > m$), and two additional Montgomery multiplications with a final modular reduction $p - R$. The algorithm presented in [82] is formulated from the identical equations as follows:

$$\begin{cases} Y \cdot R \equiv -U \cdot 2^{\lambda'} \pmod p \\ Y \cdot S \equiv V \cdot 2^{\lambda'} \pmod p. \end{cases}$$

Based on Kaliski's method, we derive a new randomized Montgomery division (RMD) which is described in Algorithm 6. To directly achieve the division operation without additional multiplication and final modular reduction, our method is to modify the initial values of (U, V, R, S) to be $(p, Y, 0, X)$ in Step 1 and the RS data path with modular subtraction in Steps 10, 11, 13, 14. Then the identities become

$$\begin{cases} X^{-1} \cdot Y \cdot R \equiv U \cdot 2^{\lambda'} \pmod p \\ X^{-1} \cdot Y \cdot S \equiv V \cdot 2^{\lambda'} \pmod p. \end{cases}$$

Similar to RMM, the RS data path between the Montgomery domain and integer domain is determined by the i-th bit value of α. The domain value of operands R and S increases by one as $\alpha_i = 1$ and remains the same as $\alpha_i = 0$.

Algorithm 6 Proposed radix-2 RMD

Input: X, Y, p, and α

Output: $R = \text{RMD}(X, Y)$

Let $U = p, V = Y, R = 0, S = X$
While $(V > 0)$ **do**
  If U is even **then** $U = \frac{U}{2}$
    If $\alpha_i = 1$ **then** $S \equiv 2S \pmod{p}$
    else $R \equiv \frac{R}{2} \pmod{p}$
  else if V is even **then** $V = \frac{V}{2}$
    If $\alpha_i = 1$ **then** $R \equiv 2R \pmod{p}$
    else $S \equiv \frac{S}{2} \pmod{p}$
  else if $U > V$ **then** $U = \frac{U-V}{2}$
    If $\alpha_i = 1$ **then** $R \equiv R - S \pmod{p}$, $S \equiv 2S \pmod{p}$
    else $R \equiv \frac{R-S}{2} \pmod{p}$
  else $V = \frac{V-U}{2}$
    If $\alpha_i = 1$ **then** $S \equiv S - R \pmod{p}$, $R \equiv 2R \pmod{p}$
    else $S \equiv \frac{S-R}{2} \pmod{p}$
  If $i < m$ **then** $i = i + 1$
Return R

For further reducing the degree recovery phase, the RS data path turns into dividing values by two in Steps 5, 8, 11, 14 to keep the intermediate domain value in $\lambda = \mathrm{H}(\alpha)$ as $i = m$. Thus the identities in Algorithm 6 are given as follows:

$$\text{If } i < m, \text{then} \begin{cases} X^{-1} \cdot Y \cdot R \equiv U \cdot 2^{\lambda'} \pmod{p} \\ X^{-1} \cdot Y \cdot S \equiv V \cdot 2^{\lambda'} \pmod{p} \end{cases}$$
$$\text{else} \begin{cases} X^{-1} \cdot Y \cdot R \equiv U \cdot 2^{\lambda} \pmod{p} \\ X^{-1} \cdot Y \cdot S \equiv V \cdot 2^{\lambda} \pmod{p}. \end{cases}$$

Before the last iteration, both U and V are 1 because the initial values of U and V are relatively prime. Then after finishing the iterative operations in Step 2, the values of (U, V, R, S) become $(1, 0, X \cdot Y^{-1} \cdot 2^{\lambda} \pmod{p}, 0)$. As a result, the proposed randomized division algorithm requires at most $2m$ iterations of successive reduction. Table 4.2 shows the expected operation time and the comparison with related works on modifying radix-2 Montgomery division algorithm. With randomization capability, Algorithm 6 will also benefit the hardware design owing to the low latency.

Table 4.2: Analysis of Various Division Algorithms

	Algorithm 6	TCAS-I'06 [17]	ESSCIRC'10 [38]
Iteration Time	$m \sim 2m$	$m \sim 2m$	$m \sim 3m$
Multiplication	0	$2 \sim 3$	0
Domain	Random 2^{λ}, $0 \leq \lambda \leq m$	Fixed 2^m	Fixed 2^m

Algorithm 7 shows the radix-4 RMD derived from Algorithm 6, and there are more branches in the algorithm as the radix becomes lager. To remain the domain value of R unpredictable in the flexible range of $[0, m-1)$, it is determined by the Hamming weight of random value α_i or (α_{i+1}, α_i). The values of UV are reduced to at least $\frac{UV}{4}$ except $U \equiv 1 \pmod 4$, $V \equiv 3 \pmod 4$ or $U \equiv 3 \pmod 4$, $V \equiv 1 \pmod 4$ in Steps 17 and 18. With this approach and the radix-4 RMM given in Algorithm 5, the ECPC can be carried out faster in affine coordinates than that in projective coordinates, where the iteration time ratio $\frac{\mathrm{RMD}}{\mathrm{RMM}} \cong 1.32$ over $GF(p)$ and 1.44 over $GF(2^m)$.

Algorithm 7 Proposed radix-4 RMD

Input: X, Y, p, and α

Output: $R = \text{RMD}(X, Y)$

1: Let $U = p, V = Y, R = 0, S = X, i = 0$
2: **While** $(V > 0)$ **do**
3: $c \equiv U \pmod 4, d \equiv V \pmod 4, t = 2$
4: **If** $i = m - 1$ **then** $R \equiv 2R \pmod p, S \equiv 2S \pmod p, t = 1$
5: **else if** $c = 0$ **then** $U = \frac{U}{4}, S \equiv 4S \pmod p$
6: **else if** $d = 0$ **then** $V = \frac{V}{4}, R \equiv 4R \pmod p$
7: **else if** $c = d$ **then**
8: **If** $U > V$ **then** $U = \frac{U-V}{4}$, $R \equiv R - S \pmod p, S \equiv 4S \pmod p$
9: **else** $V = \frac{V-U}{4}$, $S \equiv S - R \pmod p, R \equiv 4R \pmod p$
10: **else if** $c = 2$ **then**
11: **If** $\frac{U}{2} > V$ **then** $U = \frac{\frac{U}{2}-V}{2}$, $R \equiv R - 2S \pmod p, S \equiv 4S \pmod p$
12: **else** $V = \frac{V-\frac{U}{2}}{2}, U = \frac{U}{2}$, $S \equiv 2S - R \pmod p, R \equiv 2R \pmod p$
13: **else if** $d = 2$ **then**
14: **If** $U > \frac{V}{2}$ **then** $U = \frac{U-\frac{V}{2}}{2}, V = \frac{V}{2}$, $R \equiv 2R - S \pmod p, S \equiv 2S \pmod p$
15: **else** $V = \frac{\frac{V}{2}-U}{2}$, $S \equiv S - 2R \pmod p, R \equiv 4R \pmod p$
16: **else**
17: **If** $U > V$ **then** $U = \frac{U-V}{2}$, $R \equiv R - S \pmod p, S \equiv 2S \pmod p, t = 1$
18: **else** $V = \frac{V-U}{2}$, $S \equiv S - R \pmod p, R \equiv 2R \pmod p, t = 1$
19: **If** $i < m$ **then**
20: **If** $i = m - 1$ or $t = 1$ **then**
21: **If** $\alpha_i = 1$ **then** $R \equiv R \pmod p, S \equiv S \pmod p$
22: **else** $R \equiv \frac{R}{2} \pmod p, S \equiv \frac{S}{2} \pmod p$
23: **else**
24: **If** $(\alpha_{i+1}, \alpha_i) = (1, 1)$ **then** $R \equiv R \pmod p, S \equiv S \pmod p$
25: **else if** $(\alpha_{i+1}, \alpha_i) = (1, 0)$ or $(0, 1)$ **then** $R \equiv \frac{R}{2} \pmod p, S \equiv \frac{S}{2} \pmod p$
26: **else** $R \equiv \frac{R}{4} \pmod p, S \equiv \frac{S}{4} \pmod p$
27: $i = i + t$
28: **else** $R \equiv \frac{R}{2^t} \pmod p, S \equiv \frac{S}{2^t} \pmod p$
29: **Return** R

4.1.3 Domain Conversion

Since the primary inputs of EC coefficient and points are in integer domain, the domain conversion of field automorphic function ψ can be performed by the proposed RMD operation such that $\mathrm{RMD}(a, 1) = a2^{\lambda} \pmod{p}$. On the other hand, to return the point coordinates in integer domain, the RMM can be exploited to perform ψ^{-1} such as $\mathrm{RMM}(a2^{\lambda}, 1) = a \pmod{p}$. For calculating one ECSM in affine coordinates, the overhead of domain conversion is three RMD and two RMM operations as shown in Figure 4.2, where both of them can be performed by the divisor and multiplier to avoid any pre-computation from host system.

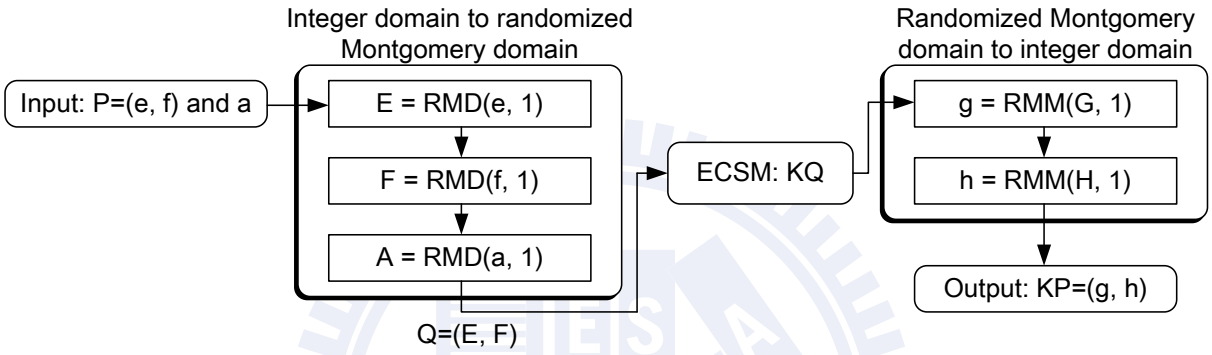


Figure 4.2: The domain conversion can be achieved in pre/post-process stage, where this overhead of several modular operations can be neglected for overall ECSM.

4.2 Elliptic Curve Point Generation (ECPG)

Figure 4.3 shows the approach of generating a random EC point over DFs. For $GF(p)$, the first step is to initialize the non-zero coordinate value of x. The Jacobi symbol (JS) can be applied to check whether the square root of $q \equiv x^3 + ax + b \pmod{p}$ exists or not, where $\mathrm{JS}(\frac{a}{p})$ represents the JS, defined for all integers a and all odd primes p by

$$\mathrm{JS}(\frac{a}{p}) = \begin{cases} 0 \text{ if } a \equiv 0 \pmod{p} \\ +1 \text{ if } a \neq 0 \pmod{p} \text{ and for some integer } x, a \equiv x^2 (\mathrm{mod} p) \\ -1 \text{ if there is no such } x. \end{cases}$$

The JS can be performed by an iterative algorithm described in IEEE P1363. Algorithm 8 is our proposed high radix approach of JS, which has 34% iteration reduction as compared with the radix-2 JS in IEEE P1363. As the square root of q exists, the coordinate value of y is the square root of q with a positive or negative value. For $GF(2^m)$, after the initialization of the non-zero coordinate value of x, the next step is to compute the square root of z, where $z^2 + z \equiv \beta \pmod{p(x)}$ with $\beta \equiv \frac{q}{x^2} \pmod{p(x)}$. And then, the coordinate value of y is equal to $(z + \mu)x \pmod{p(x)}$ with random 1-bit μ.

Figure 4.4 shows the methods of computing a square root over DFs. The computation complexity is dominated by the exponentiation in Step 2.2 (or Step 4.2) over $GF(p)$ and Step 2.2 (or Step 3) over $GF(2^m)$. The computation time is proportional to the field length of m. To improve the performance of hardware implementation, we adopt a parallel computing approach, which is described in sub-section 6.3 with more details.

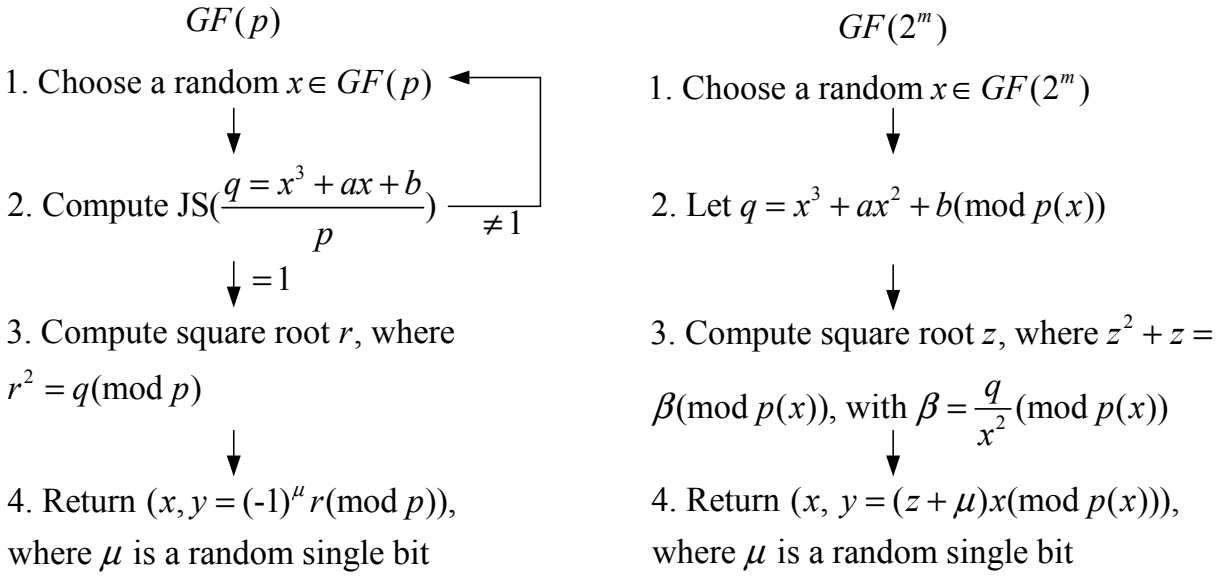


Figure 4.3: Generating a random EC point over DFs.

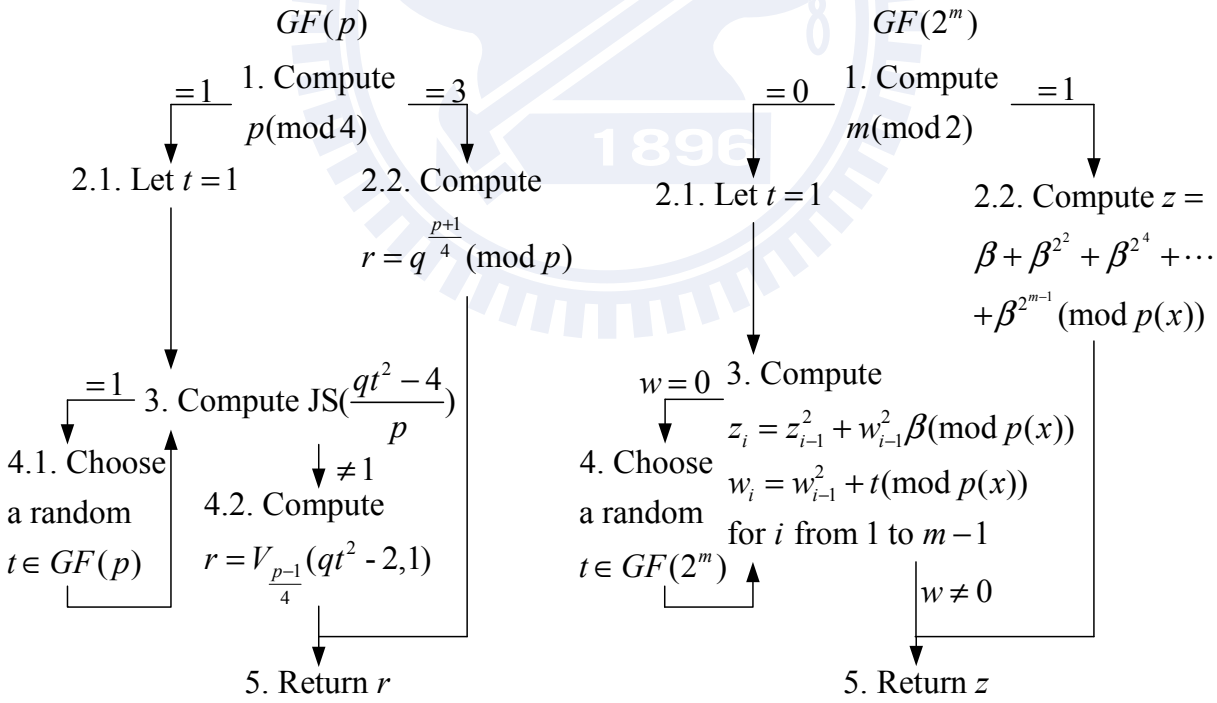


Figure 4.4: Computing a square root over DFs.

Algorithm 8 Proposed radix-4 JS algorithm

Input: $a \in GF(p)$ and p

Output: $\text{JS}(\frac{a}{p}) \in \{\pm 1, 0\}$

Let $U = p, V = a, T = 1$
While $(V > 0)$ **do**
  If $V \pmod 4 \equiv 0$ **then** $V = \frac{V}{4}$
  else if $U \pmod 4 \equiv V \pmod 4$ **then**
    If $U > V$ **then**
      If $U \equiv V \pmod 4 \equiv 3$ **then** $T = -T$
      $\text{SWAP}(U, V), V = \frac{V-U}{4}$
    else $V = \frac{V-U}{4}$
  else if $V \pmod 2 \equiv 0$ **then**
      If $U \pmod 8 \equiv 3$ or 5 **then** $T = -T$
      $V = \frac{V}{2}$
  else
    If $U > V$ **then**
      If $U \equiv V \pmod 4 \equiv 3$ **then** $T = -T$
      $\text{SWAP}(U, V)$
      If $U \pmod 8 \equiv 3$ or 5 **then** $T = -T$
      $V = \frac{V-U}{2}$
    else
      If $U \pmod 8 \equiv 3$ or 5 **then** $T = -T$
      $V = \frac{V-U}{2}$
If $U = 1$ **then Return** T
else Return 0

4.3 Right-to-Left Binary Method of Double-and-Add-Always Elliptic Curve Scalar Multiplication (RL-DAA ECSM)

Although Algorithm 3 in sub-section 3.4 prevents the private key from being revealed by detecting the difference among ECPD operations with specific primary input points, the *read-after-write* scheduling hazard inherently exists in ECPC. The ECPA $Q_{1_i} \leftarrow Q_{0_{i-1}} + Q_{2_{i-1}}$ for i-th iteration in Step 3 can only be processed after finishing the ECPD $Q_{2_{i-1}} \leftarrow 2Q_{2_{i-2}}$ for previous iteration in Step 4. This operand dependency results in a long latency for idling through parallel computations. For exploring parallelism in ECSM calculation, Algorithm 9 shows the reformulation of Algorithm 3. By using a temporary point Q_T to store the values of point $Q_{2_{i-1}}$ before starting the i-th ECPD, the iterative ECPC $Q_{2_i} \leftarrow 2Q_{2_{i-1}}$ in Step 4 and $Q_{1_i} \leftarrow Q_{0_{i-1}} + Q_{T_i} = Q_{0_{i-1}} + Q_{2_{i-1}}$ in Step 5 can be computed into two parallel threads, where the field operations of ECPC are regarded as the tasks.

Algorithm 9 Modified RL-DAA ECSM

Input: K and P

Output: KP

1: Let $Q_T \leftarrow 0, Q_0 \leftarrow 0, Q_1 \leftarrow 0, Q_2 \leftarrow P$
2: **For** i from 0 to $m-1$ **do**
3: $\quad Q_T \leftarrow Q_2$
4: $\quad Q_2 \leftarrow 2Q_2$
5: $\quad Q_1 \leftarrow Q_0 + Q_T$
6: $\quad Q_0 \leftarrow Q_{K_i}$
7: **Return** Q_0

A design method for accelerating Algorithm 9 by parallel computations is to exploit two duplicated PEs of homogeneous architecture, and each PE specifically performs the ECPD in Step 4 or ECPA in Step 5. With this approach, the overall execution time in each iteration of processing $GF(p)$ and $GF(2^m)$ ECSM is dominated by the ECPD operations. The homogeneous architecture using two identical PEs can outperform the

single PE design nearly two times in speed, but the hardware complexity increases double as well.

The computation time of distinct field operations is different such as $T_{\mathrm{MD}} > T_{\mathrm{MM}} >> T_{\mathrm{ADD}}$, T_{SUB}, where T_{MD}, T_{MM}, T_{ADD}, T_{SUB} represent the computation time of modular division (MD), multiplication (MM), addition (ADD), and subtraction (SUB). The PE can be simplified since the MD is not necessary to be processed all the time. In this work, we introduce a heterogeneous architecture including a powerful *Galois field arithmetic unit* (GFAU) and a synergistic multiplier-adder/subtractor (MAS) to speed up the ECSM with lower hardware complexity than that of two-GFAU design using two duplicated GFAU accelerators. The GFAU supports the overall field operations, and its detailed circuit unit design is described in sub-section 6.1.

Chapter 5

Proposed Design of True-Random Number Generator (TRNG)

As described in sub-section 4.1 and sub-section 4.2, to achieve the randomized computation, the random number sequence is necessarily required. For hardware implementation, the *pseudo-random number generator* (PRNG) is the most frequently implemented in circuit because the designer can adopt several given deterministic functions [83]. However, there are design issues for pseudo-random sequence used in protection against power-analysis attacks. It is that the attacker can still predict the side-channel information by applying system reset [84]. Thus, to avoid using predictable sequence for randomization, a *true-random number generator* (TRNG) without initially deterministic state is necessary for crypto-ICs.

To produce the true-random sequence, we implement the ring-oscillator-based random number generator (RO-RNG) [85–87]. It exploits the metastability and cycle-to-cycle time jitter in free running ring oscillators and sampling clock, respectively. An elementary RO-RNG is shown in Figure 5.1. Another benefit of RO-RNG is that it is suitable for integration on single chip. But the design problem is that, in actual implementation, the jitter of sampling clock is not sufficiently large for wide range of bit sequence. Then the sampled output sequence would not have much randomness or fail some patterns in the NIST random test, NIST P800-22 [88]. To give a more robust resistance against side-channel attacks, we proposed a new design method of TRNG, where the sampled output sequence passes all patterns in NIST P800-22, and the experiment results in sub-

section 7.2.2 show that the key value of ECC chip cannot be extracted with millions of measurements.

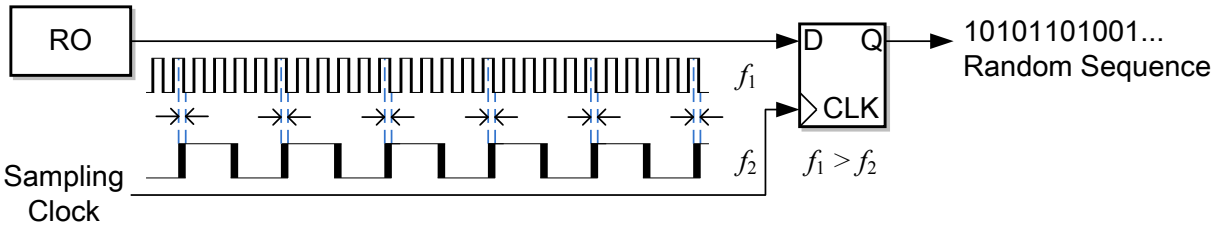


Figure 5.1: RO-RNG circuit, where the frequency of ring oscillator (RO), f_1 is faster than that of sampling clock, f_2.

5.1 Delay Chain of Jitter Amplifier

Our proposed method for improving the randomness is to amplify the jitter of sampling clock [89], where the *jitter amplifier* is designed between the primary sampling clock and flip-flop. Figure 5.2 shows the RO-RNG with jitter amplifier. The concept is shown as Figure 5.3, where the gray and white color regions are the 1 (high-logic) and 0 (low-logic) sequence over time, respectively. The regions hatched with slash lines are the probability of sampled sequence. If there is a bias for the random normal distribution of clock jitter, the area under probability density function curve between 1 and 0 samples is unequal. However, without changing the characteristic function of the distribution for clock jitter, the probability for sampling 1 and 0 values can be balanced by sampling more bit sequence during the cycle period.

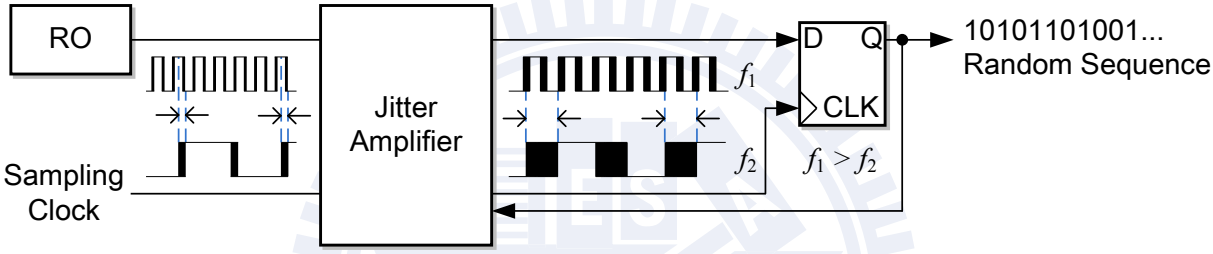


Figure 5.2: RO-RNG with jitter amplifier.

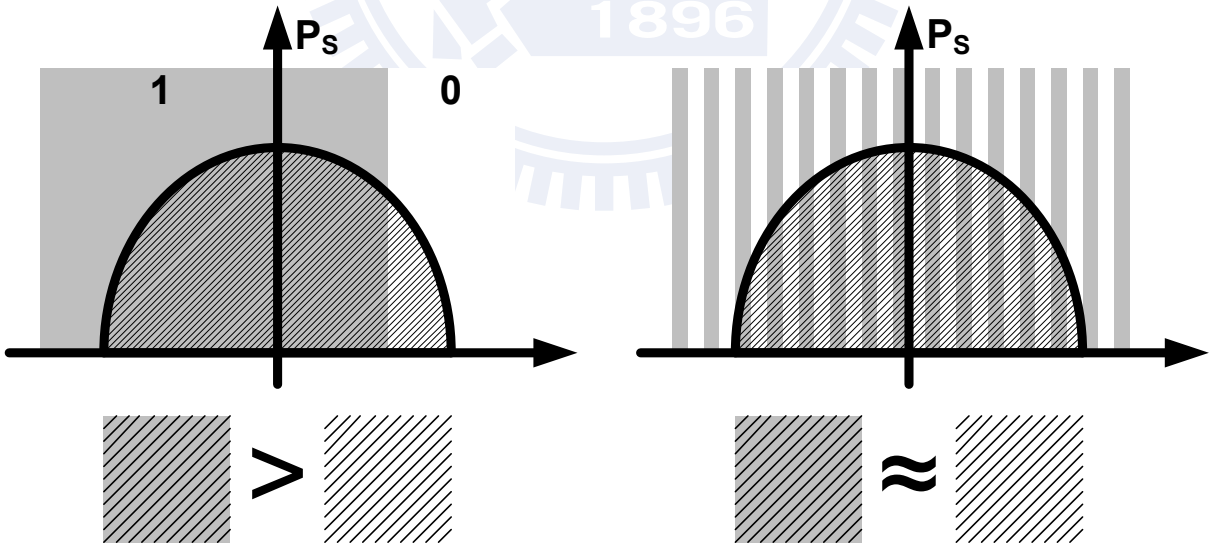


Figure 5.3: Random normal distribution of clock jitter for the sampled sequence.

An analog frequency divider which amplifies the sampling clock jitter is conventionally

used [90]. However, this approach results in degraded throughput from decreasing the sampling rate. Thus, it is not a decent design of real-time and high-throughput encryption which is in high demand for security applications. For addressing this, we proposed a new all-digital design of jitter amplifier by utilizing a delay chain with a random configuration scheme of delay time to increase the jitter uncertainty scale. Because the sampling source of flip-flop is just buffered from the system clock, the output sequence can be generated without the penalty of operating frequency.

Figure 5.4 illustrates the design for amplifying the jitter of input oscillating signal SC IN by using a delay chain with configurable delay time, where the control signal RAN is to select the rise/fall time of the delay cells. The unit jitter of output oscillating signal SC OUT can be enlarged by changing an operation mode from fast mode to slow mode through immediately varying the RAN control signals every clock period. To randomly scale the jitter further, a linear feedback shift register (LFSR) is exploited, and the RAN control signals are fed from the output of distinct registers. Figure 5.5 depicts the implemented control signal generator, where the bit value (*seed*) stored in temporary register is real-time randomly refreshed by the sampling output bitstream with a Galois-type LFSR. The delay chain is designed so that each delay t_d can be changed by RAN from fastest delay t_{df} to slowest delay t_{ds}, where $t_{df} < t_{ds}$. Besides, to change the scaling of t_d, every delay cell can have different delay time in fast mode and slow mode.

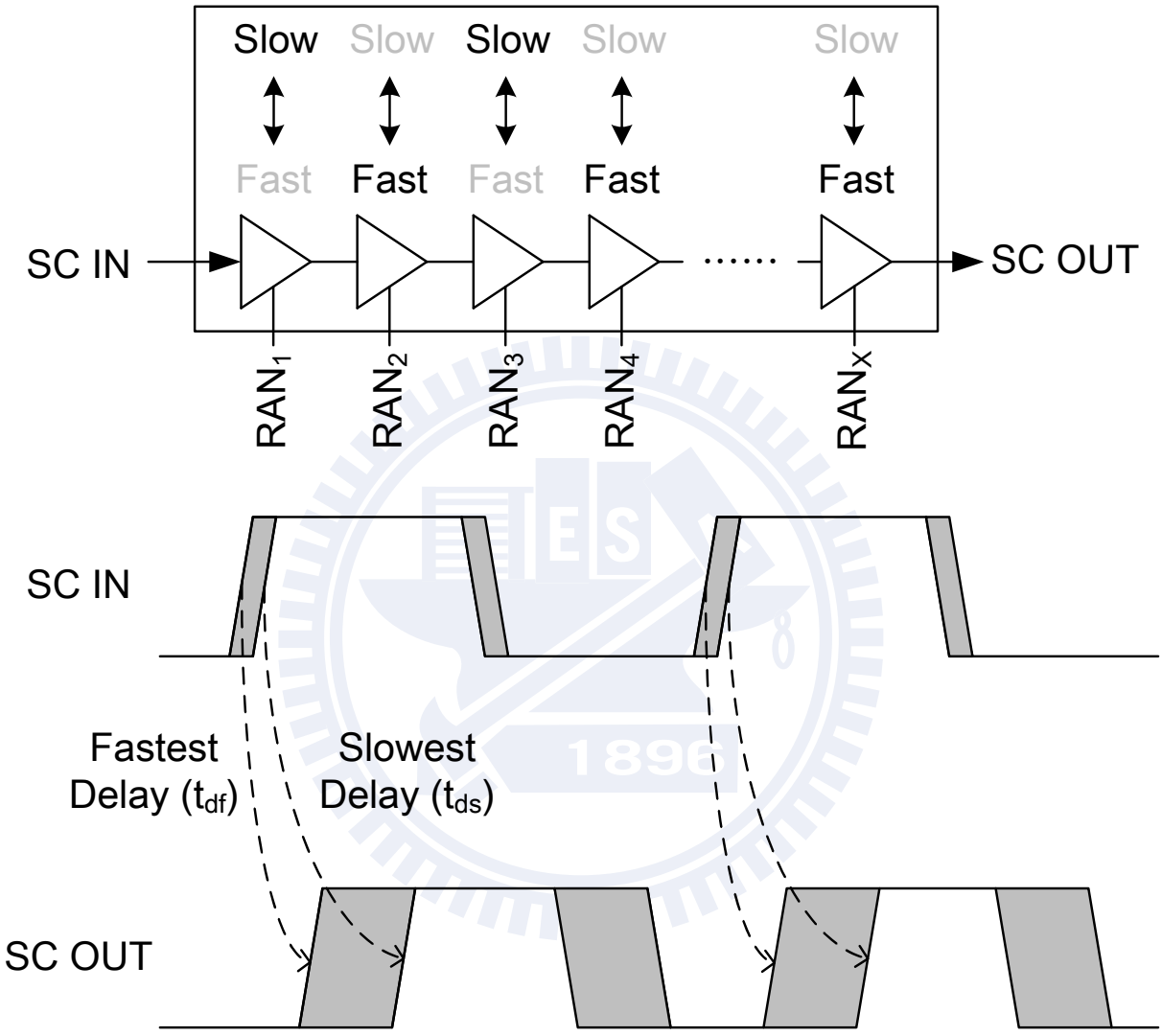


Figure 5.4: Proposed method to amplify jitter with configurable delay cell.

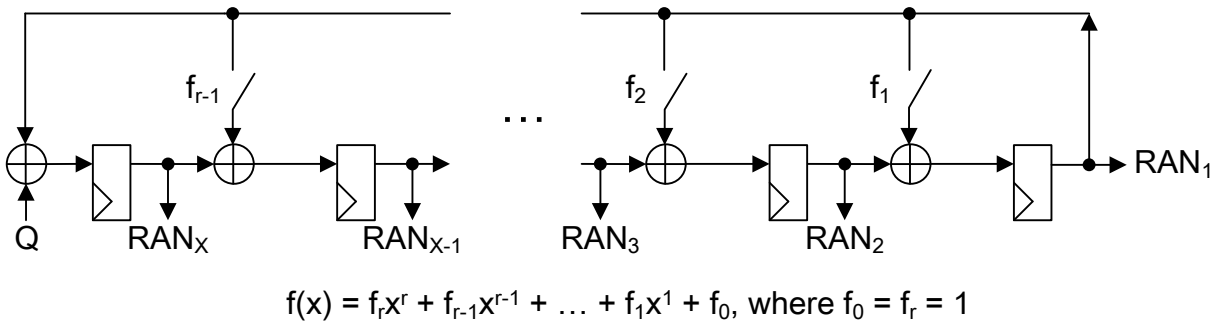


Figure 5.5: On-the-fly generation of control signals based on the LFSR.

5.2 Configurable Interlaced Hysteresis Delay Cell (CI-HDC)

To achieve a long delay time on unit cell without large power dissipation, an *interlaced hysteresis delay cell* (IHDC) [91] which functions as a buffer is exploited, and its schematic is shown in Figure 5.6. One IHDC consists of two serial inverters realized by eight MOS transistors (P-MOS transistors P1 $\sim$ P4 and N-MOS transistors N1 $\sim$ N4) for its basic functionality with four bypass transistors (A, B, C, D) for charge redistribution. It generates propagation delay signal from IN to OUT by charging/discharging cascaded transistors through cross wiring of each gate node to the drain node. Functionality is as follows: As IN charged to be logic high, transistor N1 switches on and then discharge its drain to be logic low. This turns transistor P3 on, charging its drain to be logic high with the gate of transistor N2. Then transistor N2 discharges the node INTERM and turns transistor P4 on, which charges the node OUT to be logic high. Hence the IHDC achieves a time-delay mechanism. Another case for that IN is charged to be power ground is deduced as the similar way, where the transistor delay sequence is P1 $\Rightarrow$ N3 $\Rightarrow$ P2 $\Rightarrow$ N4. These chain reactions throughout the cell contain the charging/discharging of four transistor gates; thus further on there will be referred to it as four transistor delays. Besides, for changing the delay path, the node INTERM can be pre-charged by a switch implemented with pass-transistor logic circuit. The switch circuit, shown in Figure 5.7, is controlled by the RAN signal. The fact that the switch creates three transistor delays can be seen as follows: As the RAN signal is logic low, the P-MOS pass-transistor S4 switches on. At the same time as IN is in logic high, S4 propagates the high value to

open transistor S6. After that, node INTERM is discharged to be power ground and then the gate of transistor P4 turns to be opening. Because the drain node of transistor P3 is charged to be logic high only after discharging transistor N1, a logic high value is immediately propagated to node OUT. The switch has two transistor delays and turns one transistor on/off in the IHDC. Therefore, this pre-charging technique for configuring delay time makes it one transistor delay faster than that in regular operation. The overall functionality of proposed delay cell is summarized in Table 5.1, and the propagated delay waveform shown in Figure 5.8 is simulated by HSPICE using the UMC 90-nm CMOS process. Since the majestic peak power caused from short-circuit current is prevented before turning on the last transistor during delay period, the proposed configurable IHDC (CIHDC) saves 93% average power as compared to the standard unit delay cell with same transistor sizing. Another benefit is that the hardware cost can be improved by achieving various delay time on single circuit unit without using the multiplexer to select different delay path.

Table 5.1: Functionality of CIHDC

IN	RAN	OUT	Delay Sequence	Transistor Delay
$0 \rightarrow 1$	0	$0 \rightarrow 1$	S4 $\Rightarrow$ S6 $\Rightarrow$ P4	3
$0 \rightarrow 1$	1	$0 \rightarrow 1$	N1 $\Rightarrow$ P3 $\Rightarrow$ N2 $\Rightarrow$ P4	4
$1 \rightarrow 0$	1	$1 \rightarrow 0$	S1 $\Rightarrow$ S3 $\Rightarrow$ N4	3
$1 \rightarrow 0$	0	$1 \rightarrow 0$	P1 $\Rightarrow$ N3 $\Rightarrow$ P2 $\Rightarrow$ N4	4

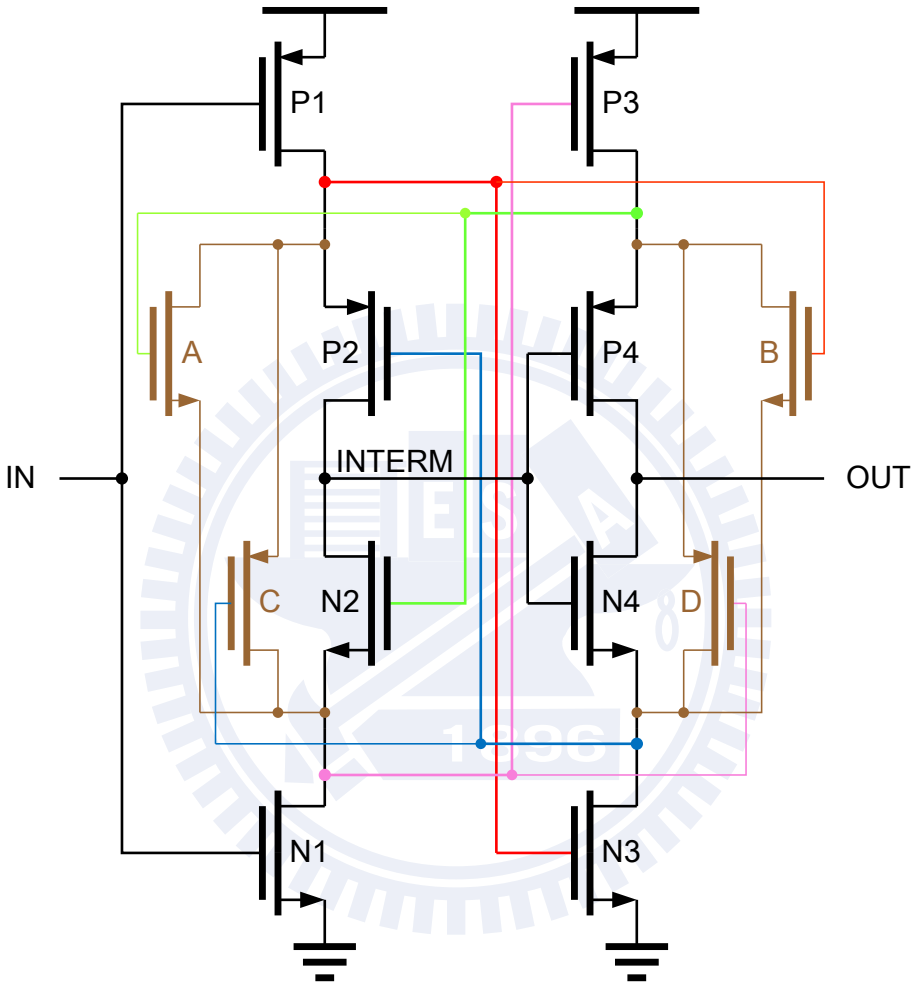


Figure 5.6: Elementary IHDC.

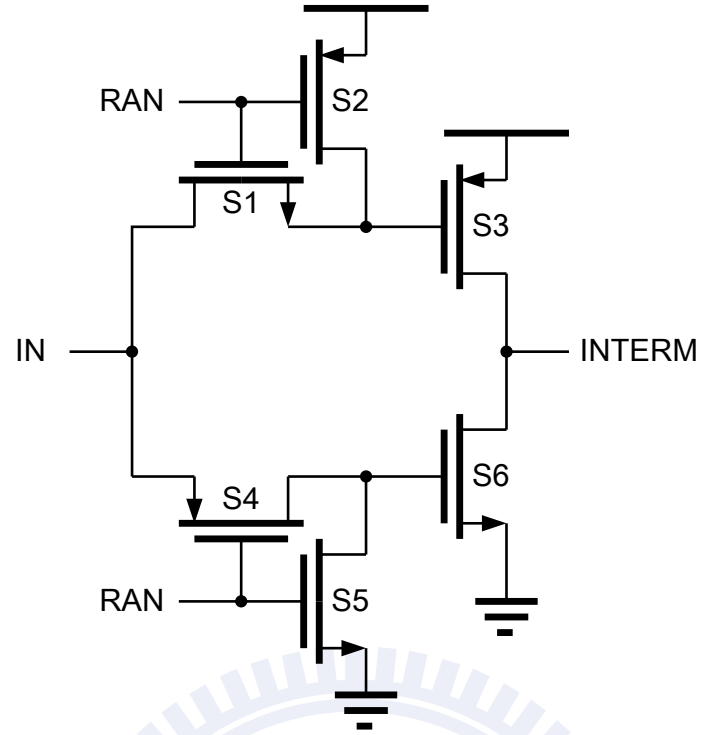


Figure 5.7: On-off switch circuit.

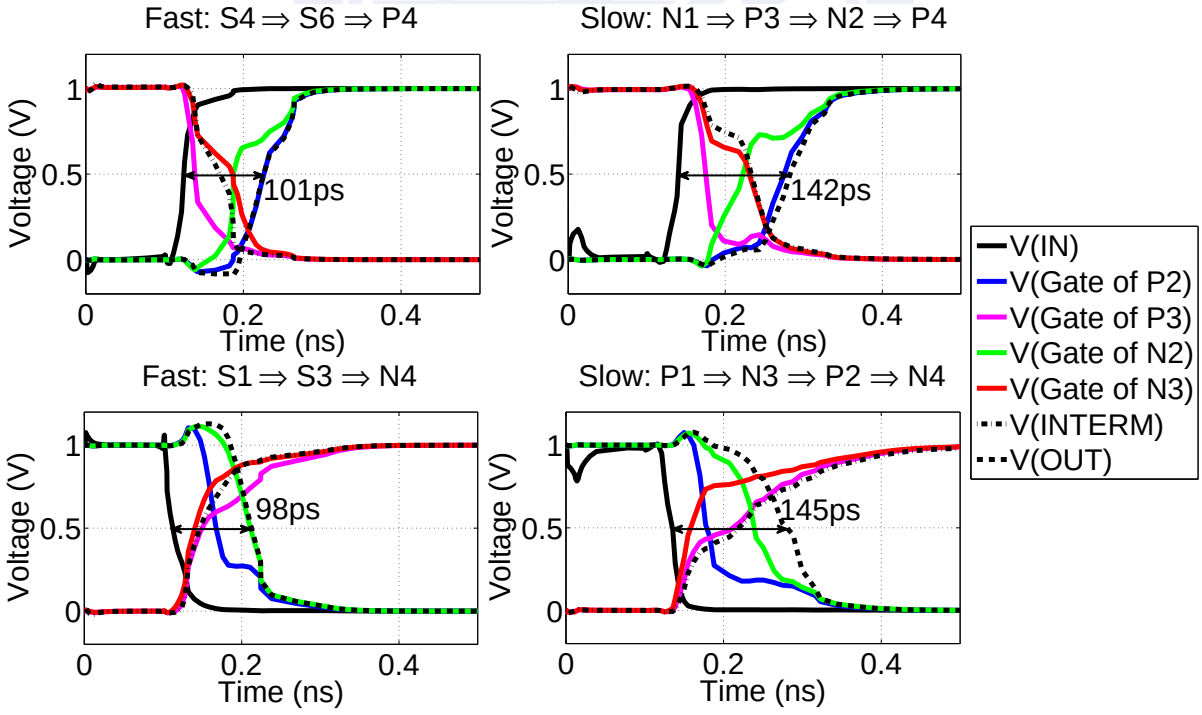


Figure 5.8: The transition waveform of CIHDC, where one CIHDC can increase jitter by several tens of picosecond at rising and falling edge.

5.3 Ring-Oscillator-Based TRNG with Jitter Amplifier

The proposed RO-RNG was fabricated by 90-nm UMC 1P9M CMOS process, and its chip photo is shown in Figure 5.9, where the feedback polynomials of Fibonacci RO, Galois RO, and LFSR are $x^{20}+x^{18}+x^{16}+x^{15}+x^{13}+x^{12}+x^5+x^4+x^2+1$, $x^{21}+x^{19}+x^{17}+x^{16}+x^7+x^3+x^2+1$, and $x^{32}+x^7+x^6+x^2+1$, respectively. To efficiently avoid the bias of individual bits, 0 and 1, induced by metastability in ROs, the final random output sequence is post-processed by reusing the LFSR with simple logic operator $Q \oplus (\mathrm{RAN}_2 \vee \mathrm{RAN}_7)$. The delay chain for amplifying jitter consists of 32 three-level CIHDCs and occupies 903 μm^2, and the total area of the RO-RNG is 1,935 μm^2. The schematic of three-level IHDC is shown in Figure 5.10 with its switch circuit shown in Figure 5.11. The physical layout of one three-level CIHDC with $11.2 \times 2.25 = 28.224\mu$m^2 area is shown in Figure 5.12. Note that another RO-RNG without jitter amplifier is integrated into this test chip, and the recorded random bitstream of ten one-million-bit sequences is verified by the NIST randomness tests, NIST P800-22 [88], where the significance level α is chosen to be the default of 0.01 (99% confidence). From measurement results, 115 Mbits/sec 0.76 mW throughput without degradation is achieved. The RNG without jitter amplification failed 4 tests of NIST test suite including non-overlapping template matching, approximate entropy, random excursions, and random excursions variant. Contrarily, all tests are passed in the NIST test suite due to the proposed jitter amplifier.

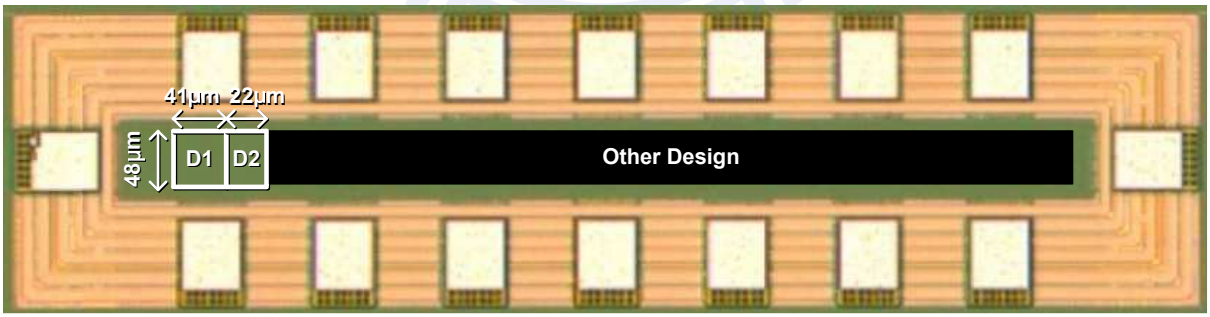


Figure 5.9: The die photo, where D1 and D2 are the RO-RNG with and without jitter amplifier, respectively.

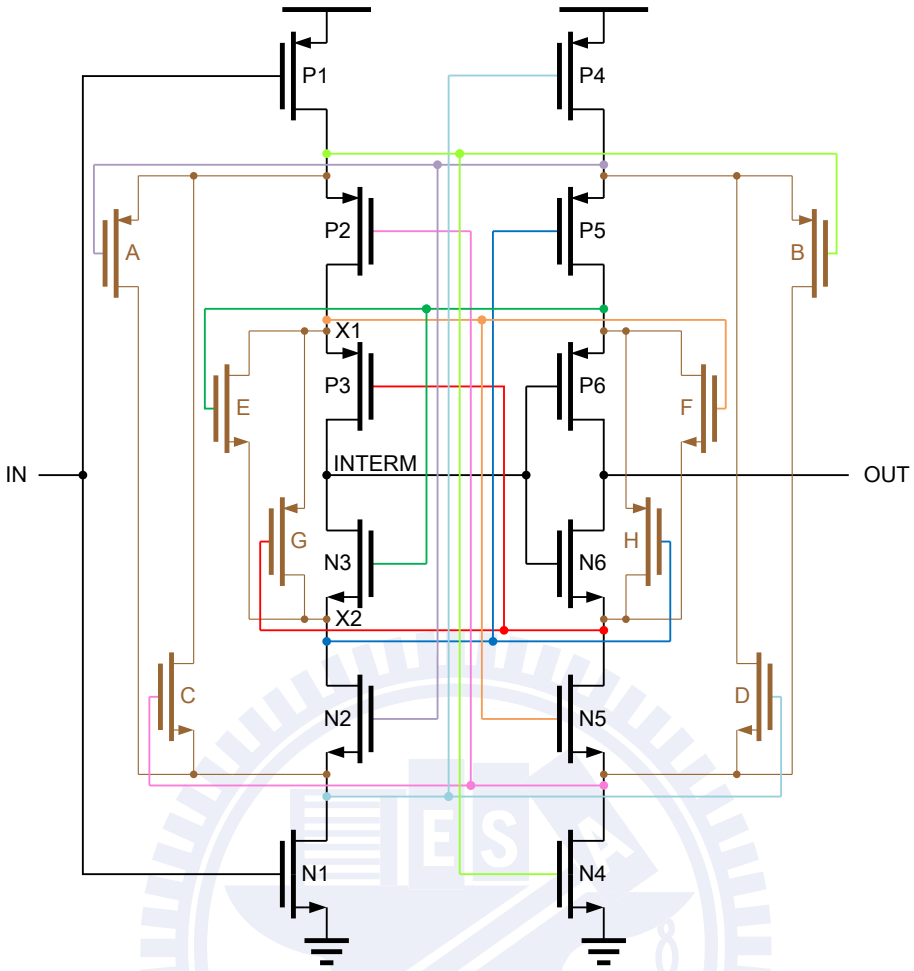


Figure 5.10: 3-level IHDC.

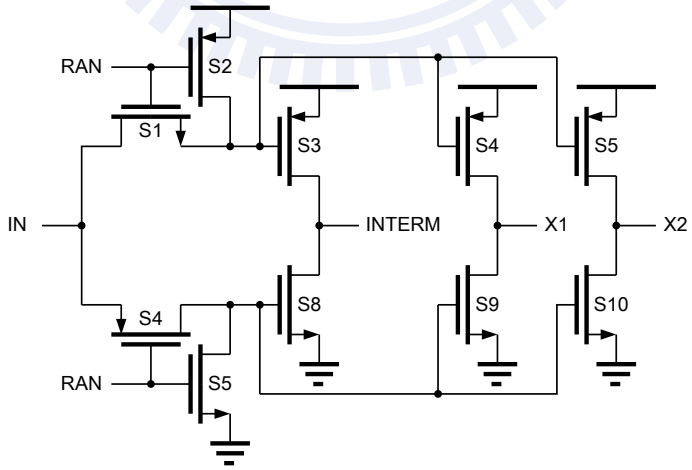


Figure 5.11: On-off switch circuit of 3-level IHDC.

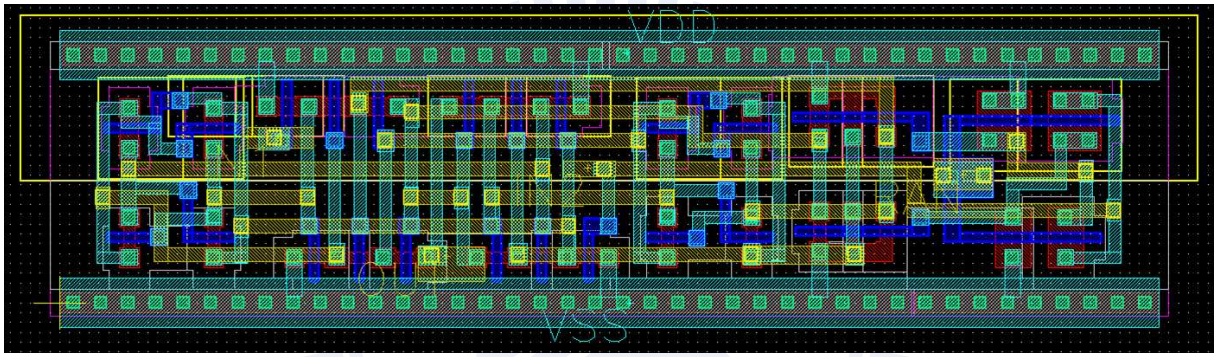


Figure 5.12: Layout of 3-level CIHDC.

Chapter 6

Proposed Architecture of Dual-Field ECC (DF-ECC) Processor

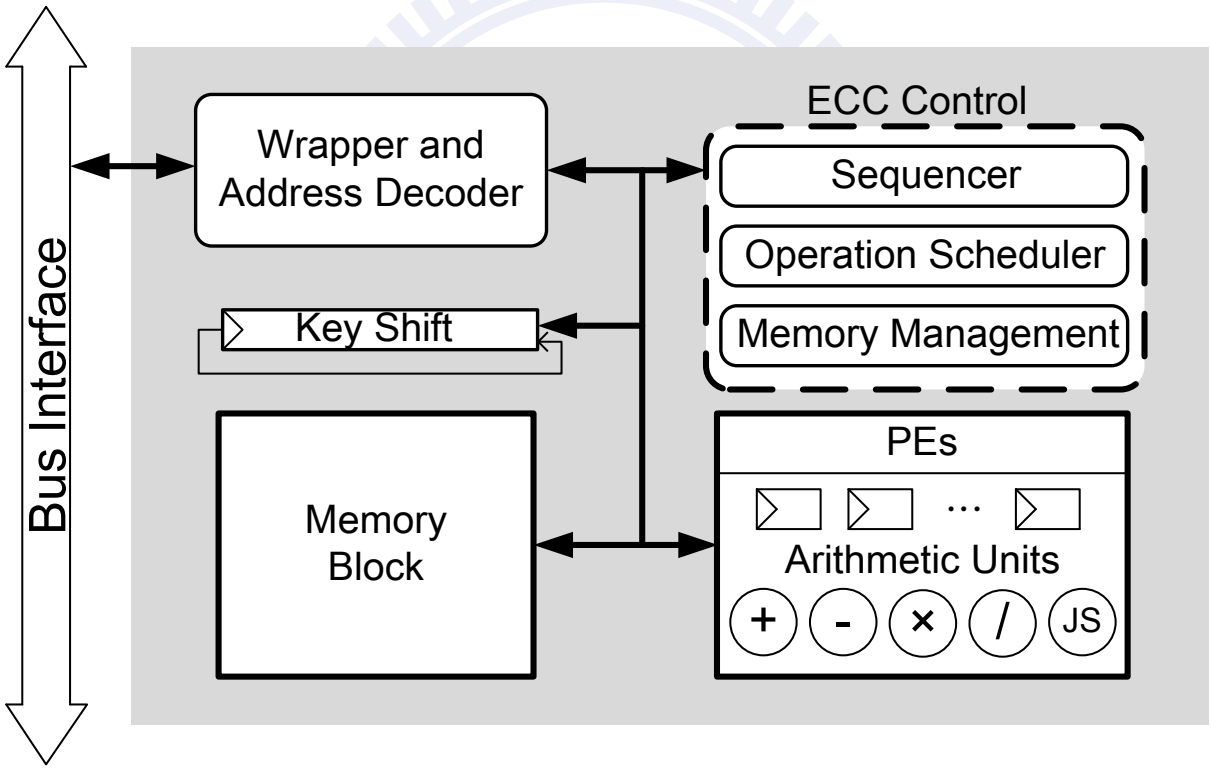


Figure 6.1: Block diagram of our DF-ECC processor.

Figure 6.1 shows the overall block diagram of our proposed dual-field ECC (DF-ECC) processor. The user-specific commands are decoded by the address decoder from the predetermined address. The DF-ECC processor supports a variety of functions such as

modular operations, Jacobi symbol, ECPC (ECPA, ECPS, ECPD, ECSM), and ECPG required in the ECC schemes over $GF(p)$ and $GF(2^m)$. The ECC control combines the sequencer, operation scheduler, and memory management. Due to the binary method of ECSM calculation, the key is scanned and shifted by 1-bit precision. To efficiently handle the transmission of intermediate values, a memory block separated from the register in PEs is used. For the secure implementation, users are prohibited from read access to the intermediate values and the private key.

The sequencer shown in Figure 6.2 has a four-level hierarchical structure. The highest level, Level 4, executes the ECC schemes. Level 3 supports the calculation of ECSM and ECPG. The sequencer at Level 2 provides the basic ECPC of ECPA, ECPS, and ECPD, and the function of half trace (HT) over $GF(2^m)$ and Lucas sequence (LS) over $GF(p)$ for ECPG. Level 1 supports the low-level functions such as the basic (Montgomery) modular operations and Jacobi symbol. Our architecture has a clearly separated control structure, and it is easy to design and modify the logic and it has high flexibility for functional extensions.

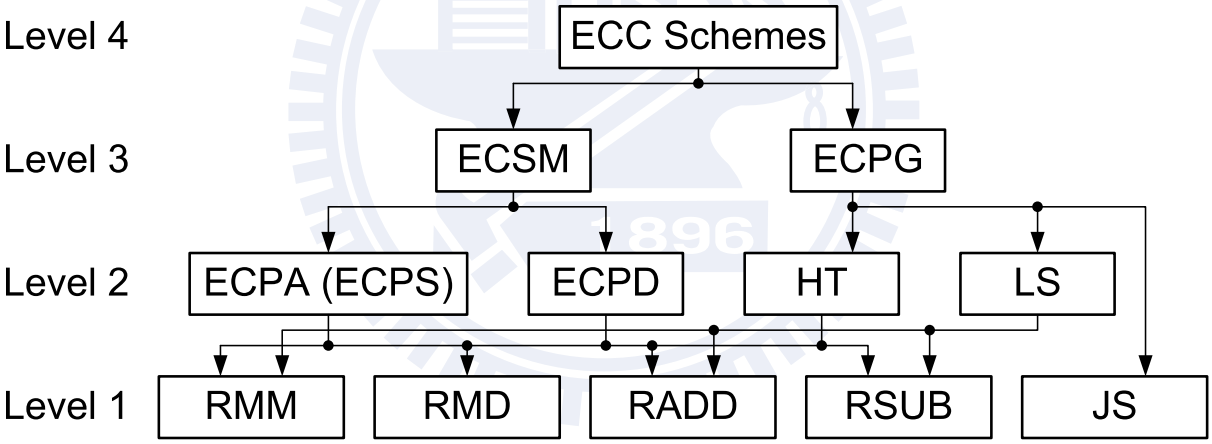


Figure 6.2: Hierarchy implementation of ECC schemes.

6.1 Jacobi Symbol and Galois Field Arithmetic Unit (JS-GFAU)

6.1.1 Fully-Pipelining Scheme

As the iterative operations shown in Algorithm 7 are performed within one cycle, the critical path is to calculate the results of operands R or S, which consists of the UV comparison with modular operations. The time-critical comparison operations such as $U > V$, $\frac{U}{2} > V$, $U > \frac{V}{2}$ in Steps 8, 11, 14 achieved by subtraction are nearly equal to an addition delay. Since the results of operands R, S are irrelevant to the results of operands U or V, a fully-pipeline stage is inserted between the UV and RS data path to reduce the critical path delay. Figure 6.3(a) and Figure 6.3(b) illustrate the hardware behavior of the pipelining scheme. After initialization, the UV data path is determined at the first cycle. Then the next cycle is to set the values of operands R, S and simultaneously determine the second case of UV comparison. The following cycles can be deduced from this approach until $V = 0$. Although an additional cycle is needed after pipelining, this is negligible as the division takes hundreds of cycles.

6.1.2 Programmable Data Path of Modular Reduction with Ladder Selection

To calculate the operands within finite filed set over $GF(p)$ in Algorithm 5, Algorithm 7, and Algorithm 8, a low-level parallel architecture with 2's complement number system is exploited. The values of all operands are bounded by the interval [0, p). For instance, as processing the modular reduction of $S \equiv 4S \pmod{p}$, the $4S$ can be achieved by bitwise shifting operand S left two bits, and the result is needed to be bounded in the interval [0, p). To achieve this, the arithmetic functions $f_{S_{p1}} = 4S - 3p$, $f_{S_{p2}} = 4S - 2p$, $f_{S_{p3}} = 4S - p$, and $f_{S_{p4}} = 4S$ are carried out simultaneously, while the correct value is sequentially determined with a ladder selection by checking the signed bit. The arithmetic functions substrated by different multiple modulus are carried out simultaneously, while the correct value is sequentially determined with a ladder selection [58] by checking the signed bit: if $f_{S_{p1}}$ is positive, then $S = f_{S_{p1}}$; else if $f_{S_{p2}}$ is positive, then $S = f_{S_{p2}}$; else if

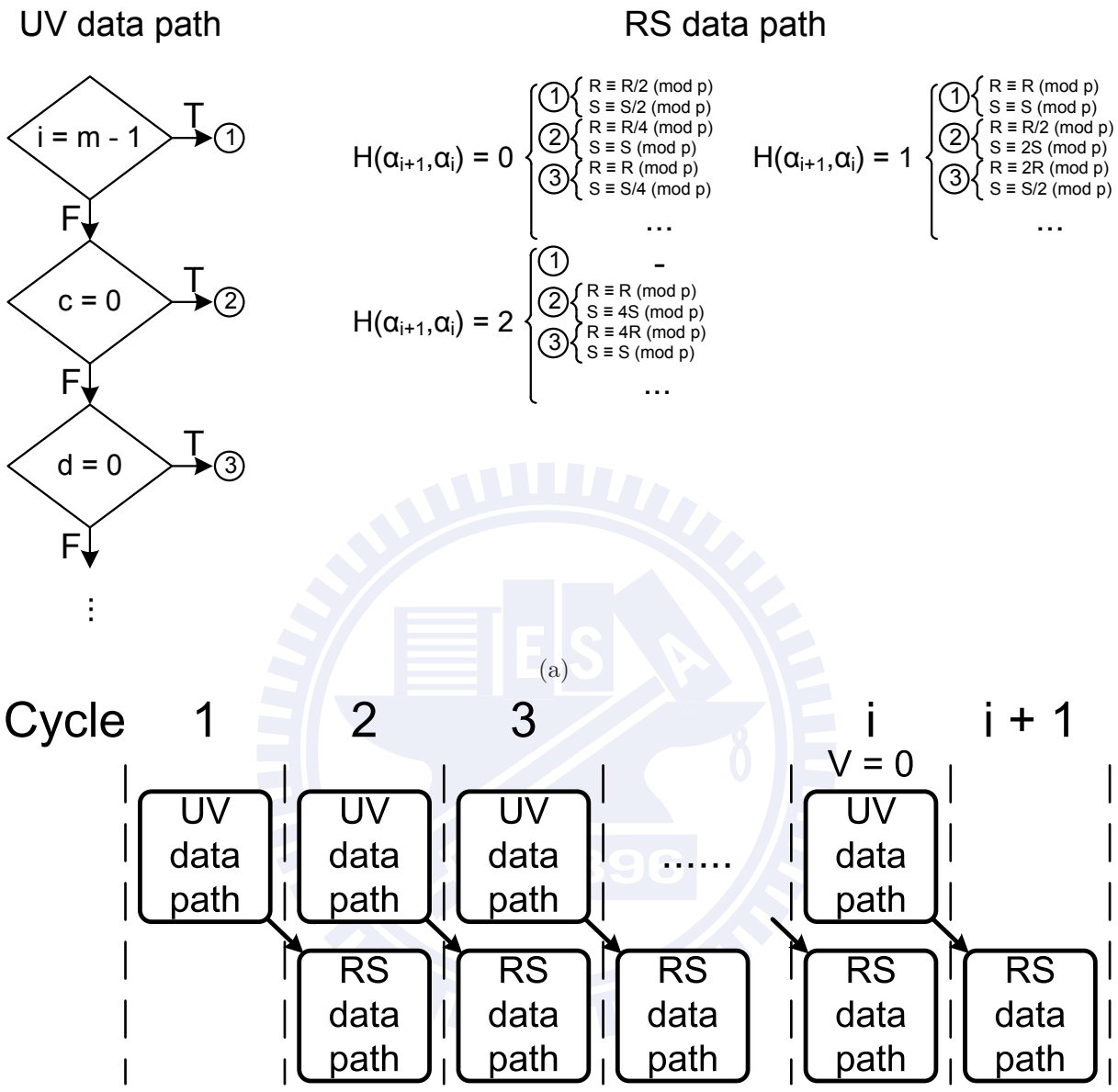


Figure 6.3: (a) Data path separation of UV comparison and RS calculation. (b) The fully-pipelining scheme of hardware implementation for the proposed radix-4 RMD in Algorithm 7.

$f_{S_{p3}}$ is positive, then $S = f_{S_{p3}}$; else $S = f_{S_{p4}}$. These multiple modular operations in the iterative calculation can be effectively implemented by using a programmable data path of bit-level architecture, which consists of the carry-save adders with a carry-lookahead adder at the last stage [38].

6.1.3 Modular Halving, Quartering by Bitwise Shifting

In Algorithm 7, the halving and quartering of the UV data path can be easily achieved by shifting right one and two bit positions because the least significant one and two bits of intermediate values are definitely zero. However, the least significant bit values of operands R and S are undetermined in the iterative calculation. Here, we use the modulus p to on-the-fly fix the least significant one and two bits of R, S to be zero. To simplify the illustration, the intermediate value of R, S is denoted as X, where the subscribed means the bit position in binary representation. For calculating the modular halving operation $\frac{X}{2} \pmod p$, it is achieved by performing $(X + X_0 \cdot p) >> 1$ since the prime p must be an odd value. For the modular quartering operation $\frac{X}{4} \pmod p$, it is conducted by performing the following calculation: if $(X_1, X_0) = (0, 0)$, X is shifted right two bit positions; if $(X_1, X_0) = (1, 0)$, $(X - 2p) >> 2$ is performed; if $(X_1, X_0) = (1, 1)$ or $(0, 1)$, and there are two sub-cases. As the least significant two bits of $X - p$ are $(1, 0)$, $(X - 3p) >> 2$ is performed because $-p_1$ is the complement value of $-3p_1$. On the other hand, it is achieved by $(X - p) >> 2$. As a result, the overall modular halving and quartering operations in Algorithm 5, Algorithm 7, and Algorithm 8 can be implemented by bitwise shifting with simple logic gates without time-cost modular division.

6.1.4 Arithmetic Unit Integration

To map the multiple modular operations in Algorithm 7 into hardware unit without using distinct circuit components and without a quite complex multiplexer of operand selection, symmetric operations such as $\frac{R-S}{4} \pmod p$ and $\frac{S-R}{4} \pmod p$ can be executed by using the same computational unit with a swap logic circuit. In Algorithm 7, the RS data path within Step 4 to Step 18 is classified into two groups: the first group includes Steps 6, 9, 12, 15, and 18; the second one consists of the remainder. The two operands R and S are switched to each other as the processing group is different from the group in

previous cycle. Furthermore, since the ECPC and ECPG are the computation of serial field operations, both of the temporary registers and modular operations in Algorithm 5, Algorithm 7, and Algorithm 8 can be reused.

Figure 6.4 shows the detailed architecture of *Galois field arithmetic unit* (GFAU), where it supports the radix-4 RMM in Algorithm 5, radix-4 RMD in Algorithm 7, and modular addition, subtraction over DFs. Without pipelining, the delay path is equal to (1) + (2) + (3) + (5) over $GF(p)$ and (1) + (2) + (4) + (5) over $GF(2^m)$. The delay path (1) can be eliminated due to the fully-pipeline stage of data path separation, so that the RS select signal is delayed one cycle from the UV select signal. Besides, the swap logic circuit can be implemented by an exclusive-OR logic operator to change the input operands of RS data path as the previous and current swap signals have inverse values. After arithmetic processing, the ladder selection is to pick out the value belonging to the finite field set. Note that the MAS is implemented by similar design approach with less hardware complexity than that of GFAU, and the circuit components of MAS are depicted in gray color in Figure 6.4.

In comparison with the previous works on $GF(p_{256})$ field arithmetic unit in [92] and [93], we also implement our processing elements (PEs) using the identical field length by the same FPGA family. Table 6.1 gives the performance results. Due to pipelined and highly integrated architecture, our design has benefits in the area-time (AT) product and outperforms others at least two times in the hardware speed.

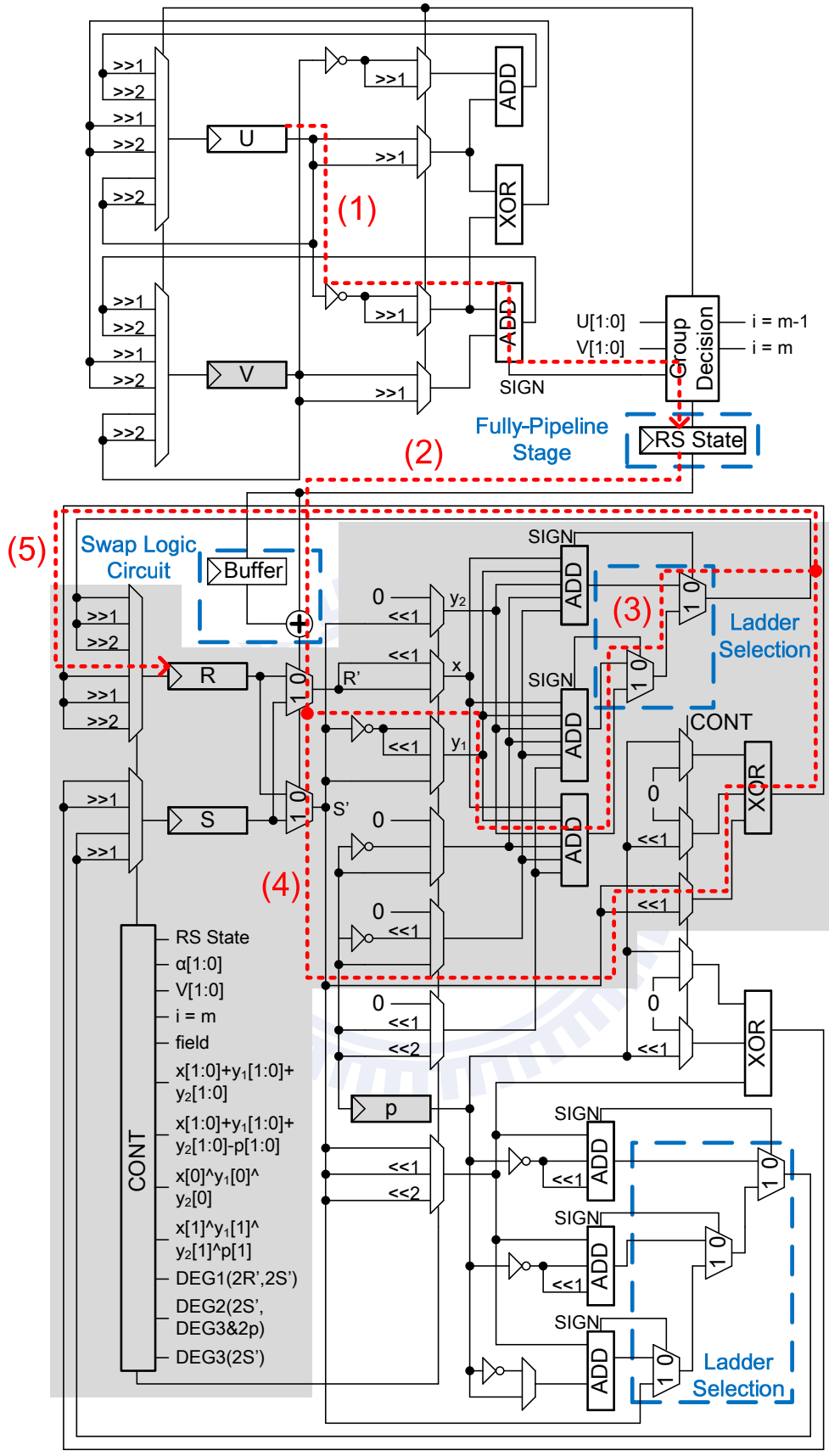


Figure 6.4: The overall DF modular operations are integrated into a fully-pipelined GFAU.

Table 6.1: Implementation Results of $GF(p_{256})$ GFAU and MAS on Xilinx Virtex-II FPGA Device with Comparison

	Area (Slices)	f (MHz)	Multiplication		Division	
			Time (μs/Op.)	AT	Time (μs/Op.)	AT
[93]	5,477	14	18.28	1	43.89	1
[92]	5,379	34	7.53	0.40	13.55	0.30
Our GFAU	9,213	37	3.46	0.29	4.98	0.18
Our MAS	4,843	37	3.46	0.13	-	-

AT product = area $\times$ time.

6.2 Heterogeneous Processing Elements (PEs) and Priority-Oriented Scheduling

To further ensure that the PEs are utilized as much as possible, the *priority-oriented scheduling* which queues higher priority task before lower priority task is exploited [94]. Algorithm 10 is our proposed operation scheduling for the modified RL-DAA ECSM in Algorithm 9, and it has two stages. The first stage in Step 1 is to configure the tasks with higher priority based on larger computation time. At the second stage in a loop of Step 4, the current task is processed as the capable PEs are available. Otherwise, when the current task is pushed into the instruction FIFO (first-in-first-out), it will be issued as the GFAU is available in Step 9. The task and thread counter are refreshed in Step 10 to Step 13 after checking thread dependence. By this interleaved processing approach, the PEs can cooperate with each other to carry out the ECSM for utilization improvement.

Figure 6.5(a) and Figure 6.5(b) illustrate the major operations of ECPC by Algorithm 3 and Algorithm 9 with priority-oriented scheduling, respectively. In these figures, the horizontal direction is the hardware behavior and the vertical direction is the timing. Also, the block in gray color signifies the idle execution. As adopting Algorithm 3, even though the last two multiplications of $(i-1)$-th ECPD can be performed by the MAS, the tasks of i-th ECPA still have to wait to be issued until generating the coordinates of $2Q_2$

Algorithm 10 Proposed priority-oriented scheduling

1: Prioritize tasks:

MD is *high priority*

MM is *medium priority*

ADD and SUB are *low priority*

2: Create ECPD and ECPA to be a thread individually

3: Initialize task and thread counter:

$u = 1, L = 1$

4: **While** $(L \leq m)$ **do**

5: Get u^{th} task in L^{th} thread

6: **If** (task priority $<$ *high*) **then**

Assign task on PE

7: **else**

8: **If** (PE ID is GFAU) **then**

Assign task on PE

9: **else** / $*$ Interleaved Processing $*$ /

Push task into FIFO, exchange PE ID,

and then wait until GFAU is available

10: **If** (u^{th} task is the last task) **then**

11: **If** (L^{th} thread is independent of all $L + 1^{\text{th}}$ threads) **then**

$u = 1, L = L + 1$

12: **else**

Wait until all parallel L^{th} threads are done,

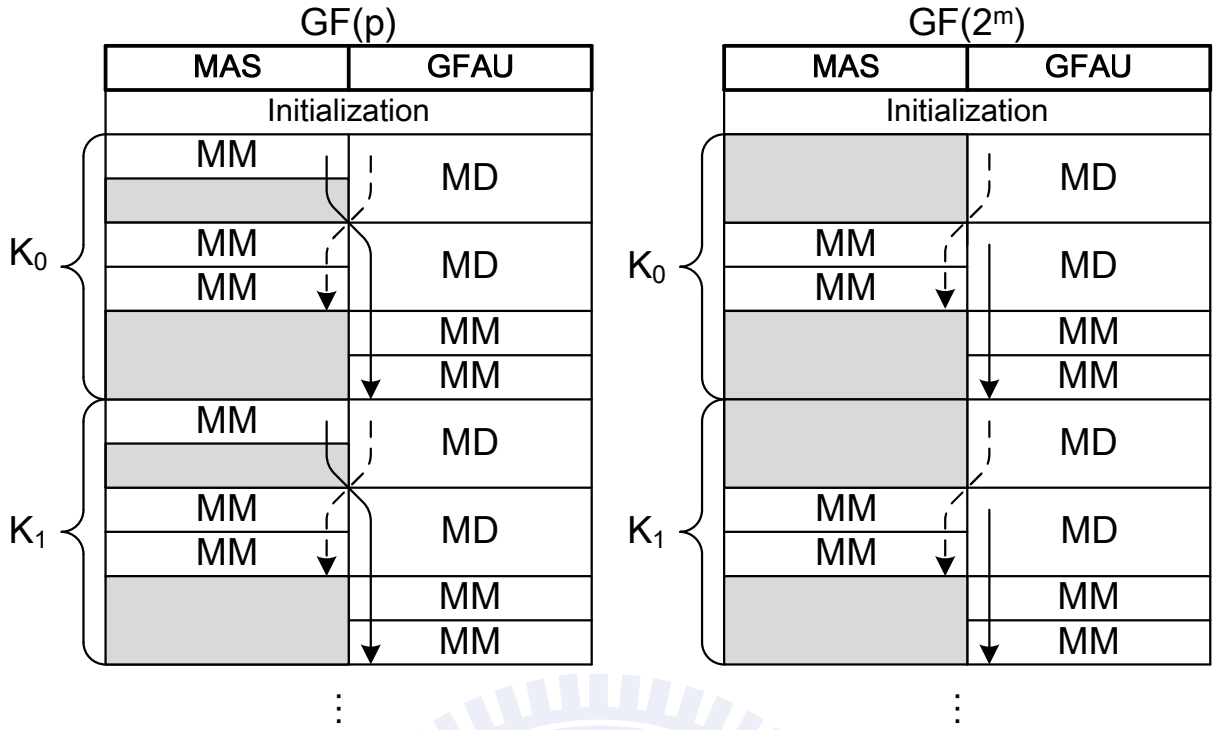
$u = 1, L = L + 1$

13: **else**

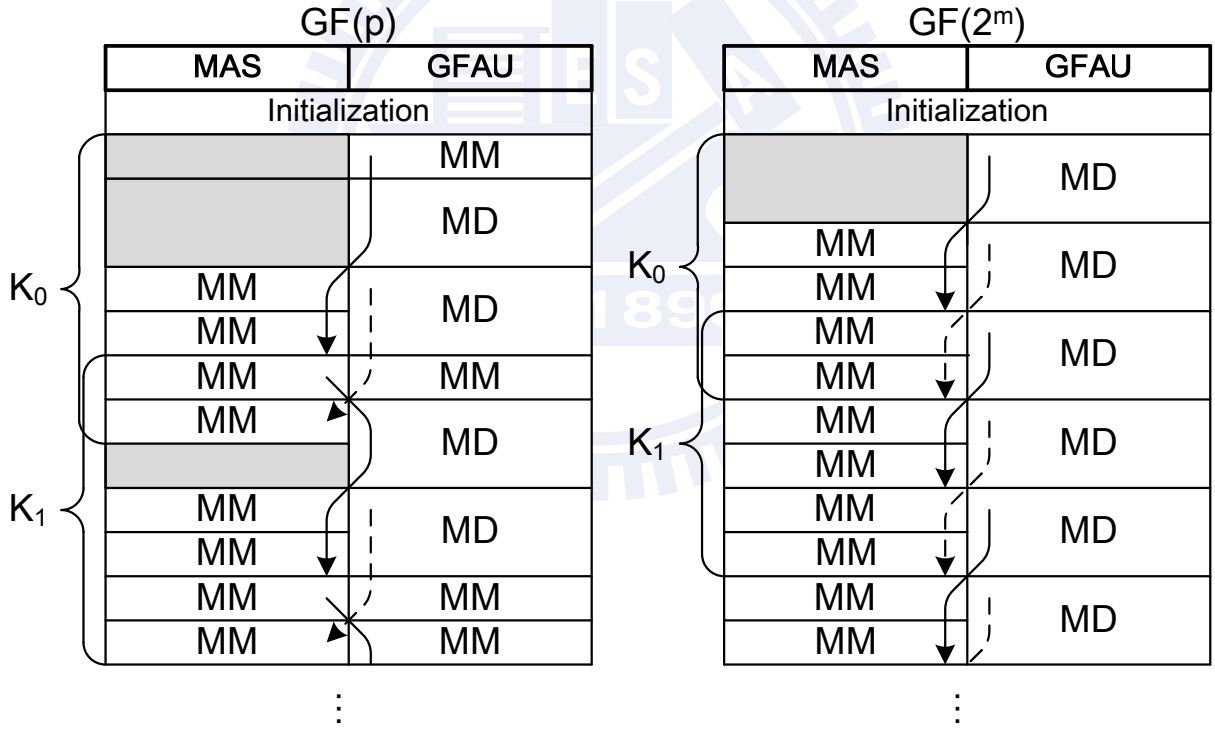
$u = u + 1$

14: ECSM is done

in previous iteration. On the other hand, with Algorithm 9, the GFAU can immediately start calculation as the value of $2Q_{2_{i-1}}$ is stored before i-th ECPA. For the average execution time at one bit of key value, the interleaved processing shown in Figure 6.5(a) needs $1T_{\mathrm{MD}}+4T_{\mathrm{MM}}$ over $GF(p)$ and $1T_{\mathrm{MD}}+4T_{\mathrm{MM}}$ over $GF(2^m)$. In respect of the case shown in Figure 6.5(b), it takes $1T_{\mathrm{MD}}+3T_{\mathrm{MM}}$ over $GF(p)$ and $4T_{\mathrm{MM}}$ over $GF(2^m)$. Therefore, the modified RL-DAA binary method of ECSM calculation with our proposed architecture and operation scheduling has fewer idle operations and more advantages in the hardware utilization than those of conventional RL-DAA approach, where the detailed operation flow of Figure 6.5(b) is described in sub-section 7.1.



(a)



(b)

Figure 6.5: The priority-oriented scheduling for (a) conventional RL-DAA ECSM and (b) modified RL-DAA ECSM, where the solid line is the ECPD operation flow and the dash line is the ECPA operation flow.

6.3 Parallel Computation of Elliptic Curve Point Generation (ECPG)

Table 6.2 shows our parallel computation of Step 2.2 and Step 4.2 in Figure 4.4 over $GF(p)$. There are two cases determined by value of $p \pmod 4$. For a field element $\alpha \in GF(p)$ and an integer e with valid length l, as $p \pmod 4 = 3$, it simply computes the modular exponentiation $\alpha^e \pmod p$ by iterative multiplications. The computation time is l MM operations. On the other hand, as $p \pmod 4 = 1$, it performs the Lucas sequence (LS) $V_e(\alpha, 1)$ which is modified by Müller [95]. The LS can be similarly achieved by iterative modular operations of multiplication followed by subtraction. The computation time is l MM + l SUB operations. These iterative operations can be implemented by the heterogeneous two-PE architecture described in sub-section 6.2. For $GF(2^m)$, similarly, the method of parallel computation of Step 2.2 ($z = \mathrm{HT}(\beta)$) and Step 3 in Figure 4.4 is shown in Table 6.3. The computation time requires $\frac{m}{2}$(2 MM + ADD) and m(2 MM + ADD) operations for Step 2.2 and Step 3 in Figure 4.4, respectively.

For $GF(p)$, assume that the rate of JS value in Step 2 of Figure 4.3 and Step 3 of Figure 4.4 is denoted by f_{JS}. The executed operations of computing ECPG over $GF(p)$ in average is

$$\underbrace{\underbrace{\frac{1}{f_{\mathrm{JS}=1}}(3\mathrm{MM}+2\mathrm{ADD}+\mathrm{JS})}_{\text{Step 2 in Figure 4.3}}+ \{\frac{1}{2}\underbrace{m\mathrm{MM}}_{\text{Step 2.2 in Figure 4.4}}+\frac{1}{2}[\underbrace{\frac{1}{f_{\mathrm{JS}\neq 1}}(2\mathrm{MM}+\mathrm{SUB}+\mathrm{JS})}_{\text{Step 3 in Figure 4.4}}+\underbrace{m\mathrm{MM}+m\mathrm{SUB}}_{\text{Step 4.2 in Figure 4.4}}]\}}_{\text{Step 3 in Figure 4.3}}.$$

On the other hand, for $GF(2^m)$, assume that the rate of w value in Step 3 of Figure 4.4 is denoted by f_w. The executed operations of computing ECPG over $GF(2^m)$ in average is

$$\underbrace{\underbrace{3\mathrm{MM}+2\mathrm{ADD}}_{\text{Step 2 in Figure 4.3}}+ \{\frac{1}{2}[\underbrace{\frac{m}{2}(2\mathrm{MM}+\mathrm{ADD})}_{\text{Step 2.2 in Figure 4.4}}]+\frac{1}{2}[\underbrace{\frac{1}{f_{w\neq 0}}m(2\mathrm{MM}+\mathrm{ADD})}_{\text{Step 3 in Figure 4.4}}]\}}_{\text{Step 3 in Figure 4.3}}.$$

Table 6.2: Architecture for Parallel Computing $GF(p)$ Square Roots

Input: $e = \sum_{j=0}^{l} e_j 2^j, \alpha \in GF(p)$, and p

Output: $\begin{cases} \alpha^e \pmod p \\ V_e(\alpha, 1) \in GF(p) \end{cases}$

1. $R_0 = \alpha, R_1 = \begin{cases} \alpha^2 \pmod p \\ \alpha^2 - 2 \pmod p \end{cases}, t_0 = \begin{cases} - \\ \alpha \end{cases}, t_1 = \begin{cases} - \\ 2 \end{cases}$
2. **For** i **from** $l-1$ to 0 **do**

   /* JS-GFAU */ /* MAS */

   2.1.a. $\text{MM}(R_0, R_1, R_{\overline{e_j}})$ 2.1.b. $\text{MM}(R_{e_j}, R_{e_j}, R_{e_j})$

   2.2.a. $\begin{cases} - \\ \text{SUB}(R_{\overline{e_j}}, t_0, R_{\overline{e_j}}) \end{cases}$ 2.2.b. $\begin{cases} - \\ \text{SUB}(R_{e_j}, t_1, R_{e_j}) \end{cases}$
3. **Return** $\begin{cases} R_1 \\ R_0 \end{cases}$

Table 6.3: Architecture for Parallel Computing $GF(2^m)$ Square Roots

Input: $\beta \in GF(2^m)$, $\begin{cases} - \\ t \end{cases}$, and m

Output: z and $\begin{cases} - \\ w \end{cases}$

1. $R_0 = \begin{cases} \beta \\ 0 \end{cases}, R_1 = \begin{cases} - \\ t \end{cases}, t_0 = \beta, t_1 = \begin{cases} - \\ t \end{cases}$
2. **For** i **from** 1 to $\begin{cases} \frac{m-1}{2} \\ m-1 \end{cases}$ **do**

   /* JS-GFAU */ /* MAS */

   2.1.a. $\text{MM}(R_0, R_0, R_0)$ 2.1.b. $\begin{cases} - \\ \text{MM}(R_1, R_1, R_1) \end{cases}$

   2.2.a. $\begin{cases} \text{MM}(R_0, R_0, R_0) \\ \text{MM}(R_1, t_0, R_2) \end{cases}$ 2.2.b. $-$

   2.3.a. $\begin{cases} \text{ADD}(R_0, t_0, R_0) \\ \text{ADD}(R_0, R_2, R_0) \end{cases}$ 2.3.b. $\begin{cases} - \\ \text{ADD}(R_1, t_1, R_1) \end{cases}$
3. **Return** R_0 and $\begin{cases} R_1 \\ R_0 \end{cases}$

6.4 Memory Hierarchy with Local Memory Coherence

The memory bandwidth is also a critical factor of system performance for the interleaved processing within various PEs; thus, we design a hierarchical memory architecture shown in Figure 6.6 with a *local memory synchronization* scheme to reduce the memory access time. Note that a w-bit register buffer is used to avoid the intrinsic latency of reading data from SRAM, where w is the data width of shared memory. For arbitrary field length m, one data transition between the PEs and MEM needs $T_{\text{MEM}} = \lceil \frac{m}{w} \rceil + 1$ cycles. The *on-demand registers*, implemented by using the D-type flip-flops, are the local memory for PEs to perform arithmetic without fetching instantly used data from the shared memory every time. To ensure the data consistency, the memory management strategy is as follows:

- *Write Back:* As the data are predicted to be used in the same PE only for next calculation such as the intermediate values for iterative calculation of MD, MM and ADD, SUB, they are saved into the on-demand registers.
- *Write Through:* The data are written into both of the on-demand registers and shared memory when they are predicted to be used for further calculation, such as the values of EC slope λ and point coordinates (x, y).
- *Local Memory Synchronization:* As the task for interleaved processing in Algorithm 10 is issued, the data in on-demand registers are exchanged between PEs.

Figure 6.7(a) and Figure 6.7(b) give an example to show that the data bandwidth is improved by applying the local memory synchronization scheme. The data in the local memory of GFAU and MAS have to be exchanged as the sequences `MOV GFAU(R reg) to MAS(S reg)` and `MOV MAS(R reg) to GFAU(S reg)` are performed. Without local memory synchronization as shown in Figure 6.7(a), the data in R reg of MAS and GFAU are written through to MEM such as GFAU (R reg) $\rightarrow$ MEMx and MAS (R reg) $\rightarrow$ MEMy; then the data are serially fetched from MEM to GFAU and MAS such as MEMx $\rightarrow$ MAS (S reg) and MEMy $\rightarrow$ GFAU (S reg). Contrarily, by exploiting the scheme of local memory synchronization, the exchanged data between processing elements can be achieved in one

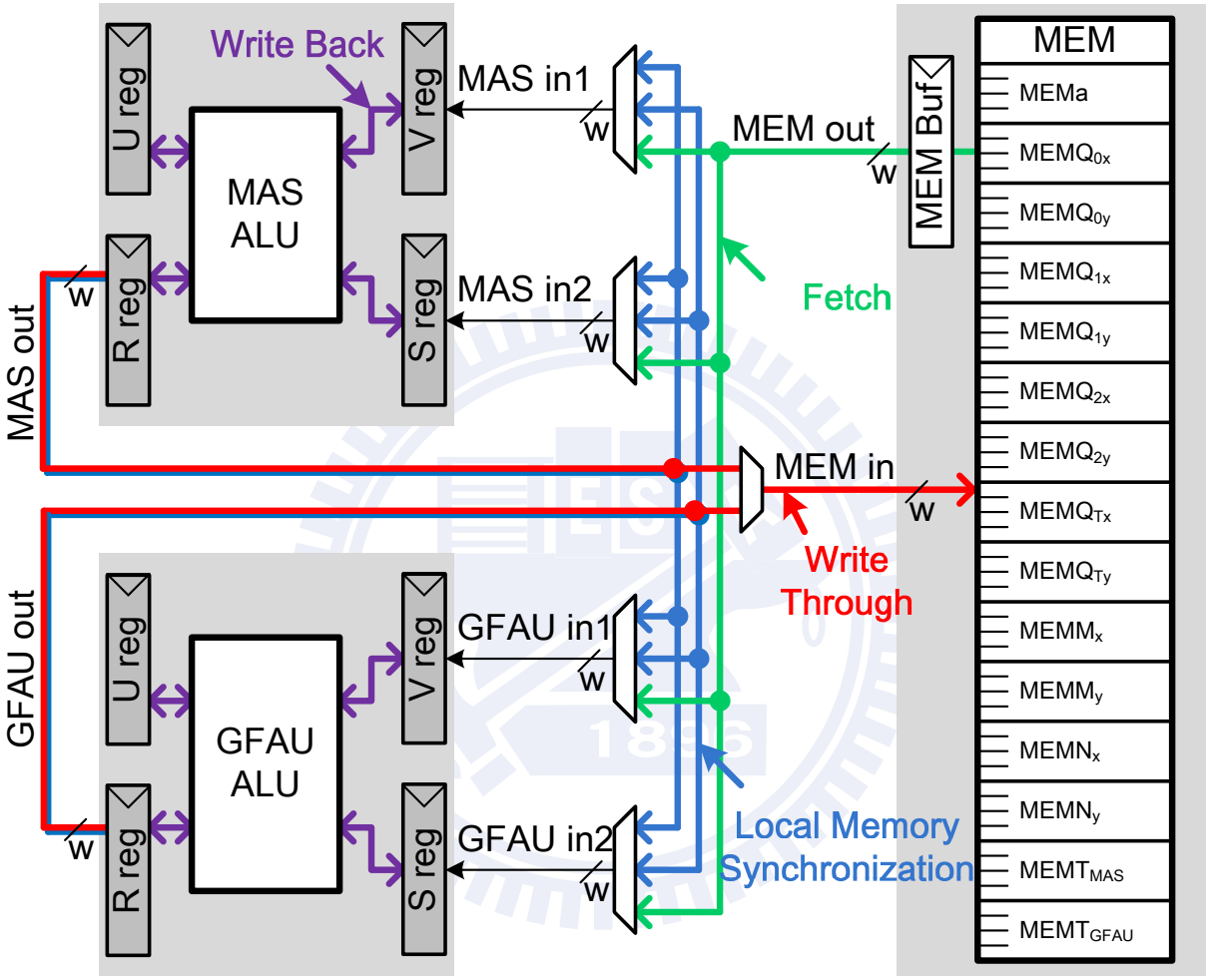


Figure 6.6: Two-level memory hierarchy for heterogeneous two-PE architecture.

data transition such as GFAU (R reg) → MAS (S reg) and MAS (R reg) → GFAU (S reg) as shown in Figure 6.7(b). Compared with a shift-register based memory architecture [38] leading to a large amount of active circuit, our proposed hierarchical memory architecture with local memory synchronization scheme gains an average of 14.2% of power reduction.

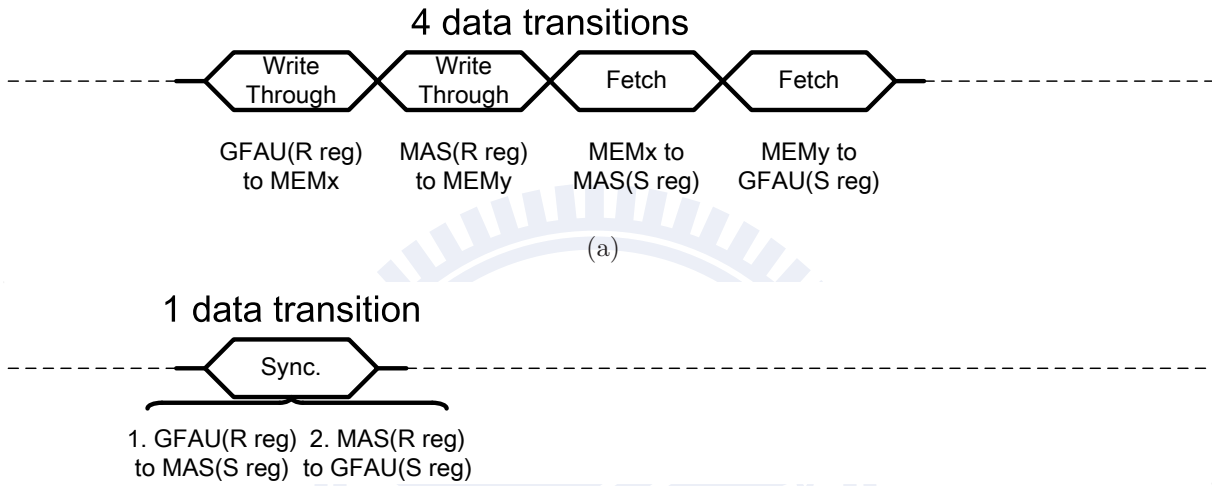


Figure 6.7: Example of data access sequences `MOV GFAU(R reg) to MAS(S reg)` and `MOV MAS(R reg) to GFAU(S reg)` (a) without (b) with local memory synchronization scheme. The data transitions through MEM for interleaved processing in (a) can be eliminated in (b).

Chapter 7

Implementation and Experiment Results

7.1 Performance Analysis

Figure 7.1 shows the explicit scheduling of our proposed parallel computation scheme. To effectively align the data transitions during processing ECSM, the atomic block is split into several stages over $GF(p)$ and $GF(2^m)$. In Algorithm 9, the coordinates of Q_0 are zero until finishing the first iteration including the initial step. Thus the ECPA operation $Q_1 = Q_0 + Q_T$ can be simply achieved by moving the value of Q_T to that of Q_1. Stages IS1, IS2, IS3 over $GF(p)$ and Stages IS1, IS2 over $GF(2^m)$ are the initial stages to process the operations as $Q_0 = 0$. Stages I, II, III over $GF(p)$ and Stages I, II over $GF(2^m)$ are the operating stages between interleaved processing for the iterative ECSM calculation as $Q_0 \neq 0$. In Figure 7.1, the computation in Stages IS1, IS2, IS3 over $GF(p)$ and Stages IS1, IS2 over $GF(2^m)$ are similar to that in Stages II, III, I over $GF(p)$ and Stages II, I over $GF(2^m)$ except disabling the ECPA operations, respectively.

On the basis of the cycle analysis results of MD, MM, ADD, SUB operations, and data transitions, the execution time for the proposed heterogeneous architecture using priority-oriented scheduling can be computed. Table 7.1 gives the operation time among distinct operating stages, and the execution time of one ECSM over DFs for a valid key

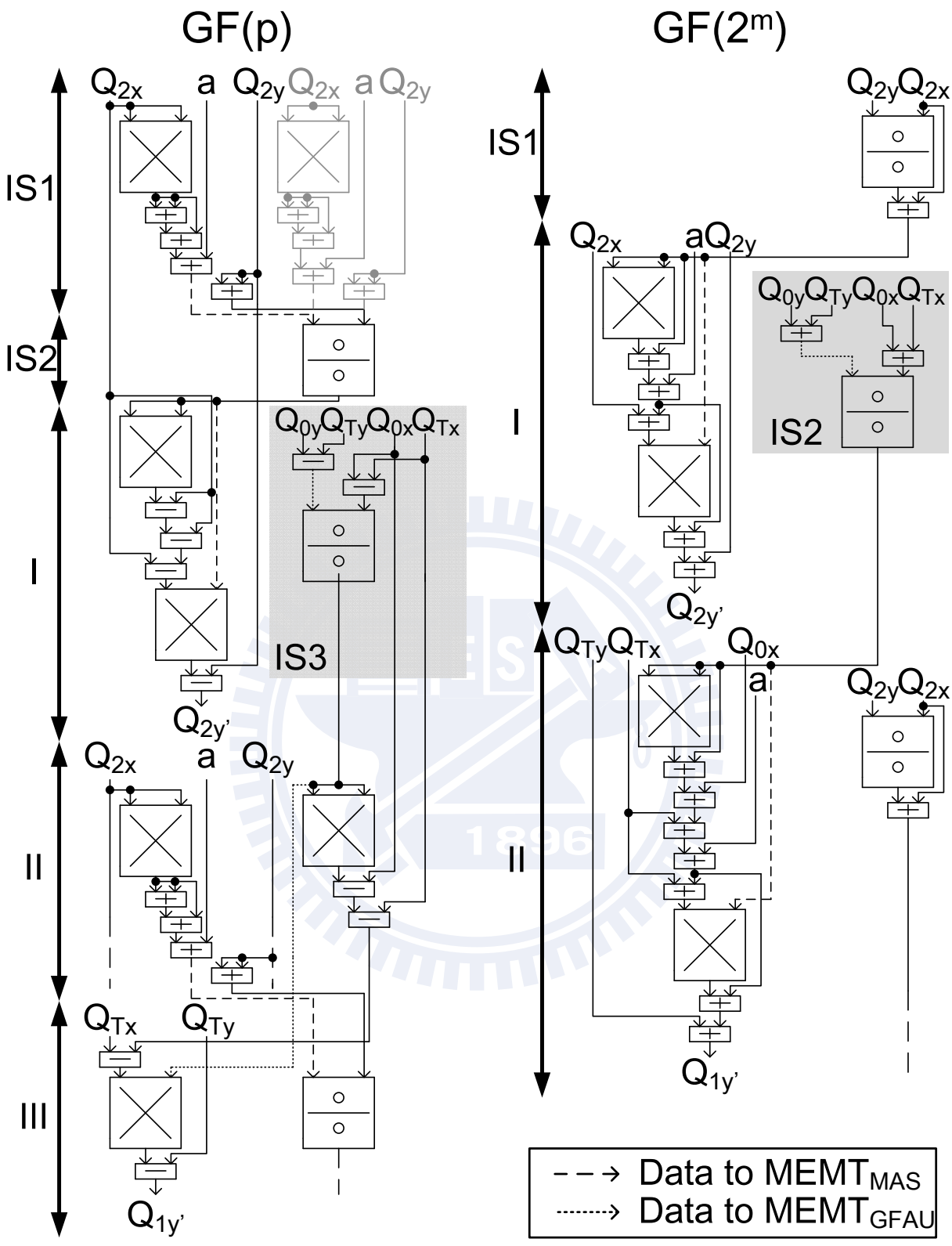


Figure 7.1: Detailed data flow for the proposed priority-oriented scheduling of ECSM calculation over DFs.

length L_K is summarized as follows:

$$\begin{cases} GF(p) : T_{p,\mathrm{PRE}} + T_{p,\mathrm{MK}} + 2(T_{p,\mathrm{IS1}} + T_{p,\mathrm{IS2}}) + T_{p,\mathrm{IS3}} + (L_K - 1)T_{p,\mathrm{S1}} + \\ (L_K - 2)(T_{p,\mathrm{S2}} + T_{p,\mathrm{S3}}) + T_{p,\mathrm{UK}} + T_{p,\mathrm{POST}} \\ GF(2^m) : T_{b,\mathrm{PRE}} + T_{b,\mathrm{MK}} + 2T_{b,\mathrm{IS1}} + T_{b,\mathrm{IS2}} + (L_K - 1)T_{b,\mathrm{S1}} + \\ (L_K - 2)T_{b,\mathrm{S2}} + T_{b,\mathrm{UK}} + T_{b,\mathrm{POST}}. \end{cases}$$

Note that, with radix-4 approaches, $T_{\mathrm{MM}} = 0.5m$, $T_{\mathrm{MD}} = 0.66m$, $T_{\mathrm{ADD}} = T_{\mathrm{SUB}} = 1$, $T_{\mathrm{MEM}} = \lceil \frac{m}{w} \rceil + 1$ with w-bit data width of shared memory. For one 160-bit ECSM, the overhead of the masking and unmasking primary point is 0.80%, and the overhead of the pre-processing and post-processing is 0.72%.

To compare the different design methods under the consideration of power-analysis resistance, the post-layout simulations of ECC hardware implementation are given in Table 7.2. Single-GFAU [58] and two-GFAU designs are the tradeoff between hardware complexity and speed due to the difference between serial and parallel computations. By using a cooperative MAS which has lower hardware complexity than that of GFAU, the heterogeneous architecture moderates the cost from duplicating GFAU. But the parallelism ability is still required to be improved further. Algorithm 9 reducing the data hazard in Algorithm 3 has fewer idle operations as exploiting the proposed scheduling in Algorithm 10. As a result, the design using the heterogeneous architecture and a newly introduced priority-oriented scheduling with the independent parallel threads for ECPC has advantages in the hardware efficiency.

Table 7.1: Time Analysis of Proposed Priority-Oriented Scheduling

(a) $GF(p)$

Operating Stage	Computation Time
Pre-process	$T_{p,\mathrm{PRE}} = 3T_{\mathrm{MD}} + 6T_{\mathrm{MEM}}$
Mask	$T_{p,\mathrm{MK}} = T_{\mathrm{MD}} + 2T_{\mathrm{MM}} + 6T_{\mathrm{SUB}} + 13T_{\mathrm{MEM}}$
IS1	$T_{p,\mathrm{IS1}} = T_{\mathrm{MM}} + 4T_{\mathrm{ADD}} + 6T_{\mathrm{MEM}}$
IS2	$T_{p,\mathrm{IS2}} = T_{\mathrm{MD}} + T_{\mathrm{MEM}}$
IS3	$T_{p,\mathrm{IS3}} = 2T_{\mathrm{MM}} + 4T_{\mathrm{SUB}} + 9T_{\mathrm{MEM}}$
I	$T_{p,\mathrm{S1}} = T_{\mathrm{MEM}} + 2T_{\mathrm{MM}} + 4T_{\mathrm{SUB}} + 8T_{\mathrm{MEM}}$
II	$T_{p,\mathrm{S2}} = T_{\mathrm{MM}} + 4T_{\mathrm{ADD}} + 7T_{\mathrm{MEM}}$
III	$T_{p,\mathrm{S3}} = T_{\mathrm{MEM}} + T_{\mathrm{MD}}$
Unmask	$T_{p,\mathrm{UK}} = T_{\mathrm{MD}} + 2T_{\mathrm{MM}} + 7T_{\mathrm{SUB}} + 15T_{\mathrm{MEM}}$
Post-process	$T_{p,\mathrm{POST}} = 2T_{\mathrm{MM}} + 4T_{\mathrm{MEM}}$

(b) $GF(2^m)$

Operating Stage	Computation Time
Pre-process	$T_{b,\mathrm{PRE}} = 3T_{\mathrm{MD}} + 6T_{\mathrm{MEM}}$
Mask	$T_{b,\mathrm{MK}} = T_{\mathrm{MD}} + 2T_{\mathrm{MM}} + 9T_{\mathrm{ADD}} + 16T_{\mathrm{MEM}}$
IS1	$T_{b,\mathrm{IS1}} = T_{\mathrm{MD}} + T_{\mathrm{ADD}} + 2T_{\mathrm{MEM}}$
IS2	$T_{b,\mathrm{IS2}} = 2T_{\mathrm{MM}} + 5T_{\mathrm{ADD}} + 9T_{\mathrm{MEM}}$
I	$T_{b,\mathrm{S1}} = T_{\mathrm{MEM}} + 2T_{\mathrm{MM}} + 5T_{\mathrm{ADD}} + 8T_{\mathrm{MEM}}$
II	$T_{b,\mathrm{S2}} = 2T_{\mathrm{MM}} + 7T_{\mathrm{ADD}} + 10T_{\mathrm{MEM}}$
Unmask	$T_{b,\mathrm{UK}} = T_{\mathrm{MD}} + 2T_{\mathrm{MM}} + 10T_{\mathrm{ADD}} + 18T_{\mathrm{MEM}}$
Post-process	$T_{b,\mathrm{POST}} = 2T_{\mathrm{MM}} + 4T_{\mathrm{MEM}}$

Table 7.2: Implementation Analysis for Different DF-ECC Designs

Design Method	Area (mm^2/KGates)	Operating Field	Time (ms/ECSM) @f (MHz)	AT
Single-GFAU DF-ECC with Algorithm 3	0.29/70	$GF(p_{160})$	0.44@256	1
		$GF(2^{160})$	0.38@260	1
Two-GFAU DF-ECC with Algorithm 9	0.54/129	$GF(p_{160})$	0.25@256	1.05
		$GF(2^{160})$	0.19@260	0.92
Heterogeneous DF-ECC with Algorithm 3	0.39/95	$GF(p_{160})$	0.39@256	1.20
		$GF(2^{160})$	0.30@260	1.07
Heterogeneous DF-ECC with Algorithm 9	0.40/96	$GF(p_{160})$	0.25@256	0.77
		$GF(2^{160})$	0.22@260	0.78

AT product = gate count $\times$ time.

7.2 Power Measurement

In this sub-section, the power analysis for the ECC chip using our SCA countermeasure in Chapter 4 is presented.

7.2.1 SPA

Figure 7.2 shows the power traces for different hamming weight of the key over time obtained from a protected ECC chip performing LR-DAA ECSM in Algorithm 2, where the hamming weight of the key is denoted by H(K). As the chip is processing, it consumes 1.79 mW at 10 MHz, which results in a voltage drop above 50 mV across the measured resistor. From these waveforms, the key value in the chip using LR-DAA ECSM cannot be distinguished by visual inspections because the processing time is independent on the hamming weight of the key.

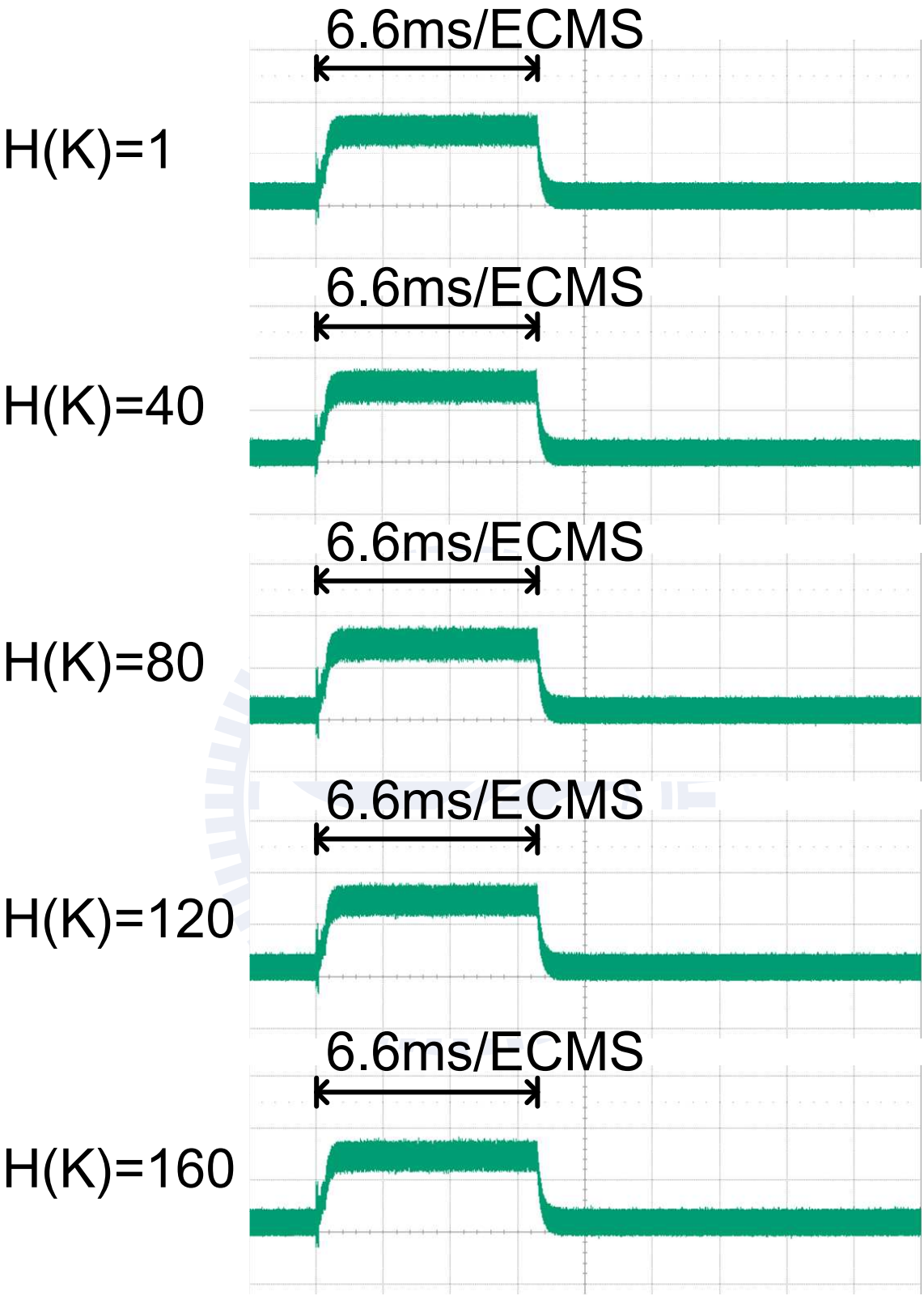


Figure 7.2: SPA attacks on the protected ECC chip using LR-DAA binary method of ECSM, where the power traces are recorded by 50.0 mV/div voltage resolution and 2.0 ms/div time base.

7.2.2 DPA

In Figure 7.3, the correlation coefficients between the target traces and power model for all possible hamming distances of the point coordinate Q_0 in Algorithm 2 with randomized computation are plotted over power traces, and those of the correct and incorrect key hypothesis are plotted in black and gray, respectively. By using a basic RO-RNG without the jitter amplifier, the reset problem [84] is solved. However, the chip is still susceptible to DPA attacks owing to low randomness as shown in Figure 7.3, where the jitter of the sampling clock is not sufficiently large in practical implementations and the random sequence fails several patterns of the NIST randomness test [88]. In this case, the key of ECC chip using randomized computation with the primary random sequence can still be revealed after three million power traces.

In contrast, Figure 7.4 shows the correlation analysis of DPA attacks for the ECC chip using random sequence generated from the RO-RNG with jitter amplifier. The random sequence is examined to meet the 15 patterns of NIST randomness test with $\alpha > 0.01$. For this, due to randomness improvement, the key value of the protected ECC chip using randomized computation cannot be revealed even after 12 million power traces.

7.2.3 ZPA

After applying the correlation analysis, Figure 7.5 shows the ZPA attacks on a protected ECC device using masked base point approaches. The correlation value of correct and incorrect hypothesis is plotted in black and gray color, respectively. The correlation value of the correct key is not the highest one among that of all the other key hypotheses because the zero-value does not happen. Thus the bit value of key cannot be revealed.

7.2.4 CPA

From the experiment results for protected ECC device using RL-DAA ECSM, as shown in Figure 7.6, the correlation analysis shows that the correlation value of correct and incorrect hypothesis cannot be scattered in high dependence because of collision operations. The correlation coefficients of the power traces related to the zero and non-zero bits of the key are drawn in circle and star, respectively. In Figure 7.6, the bit value

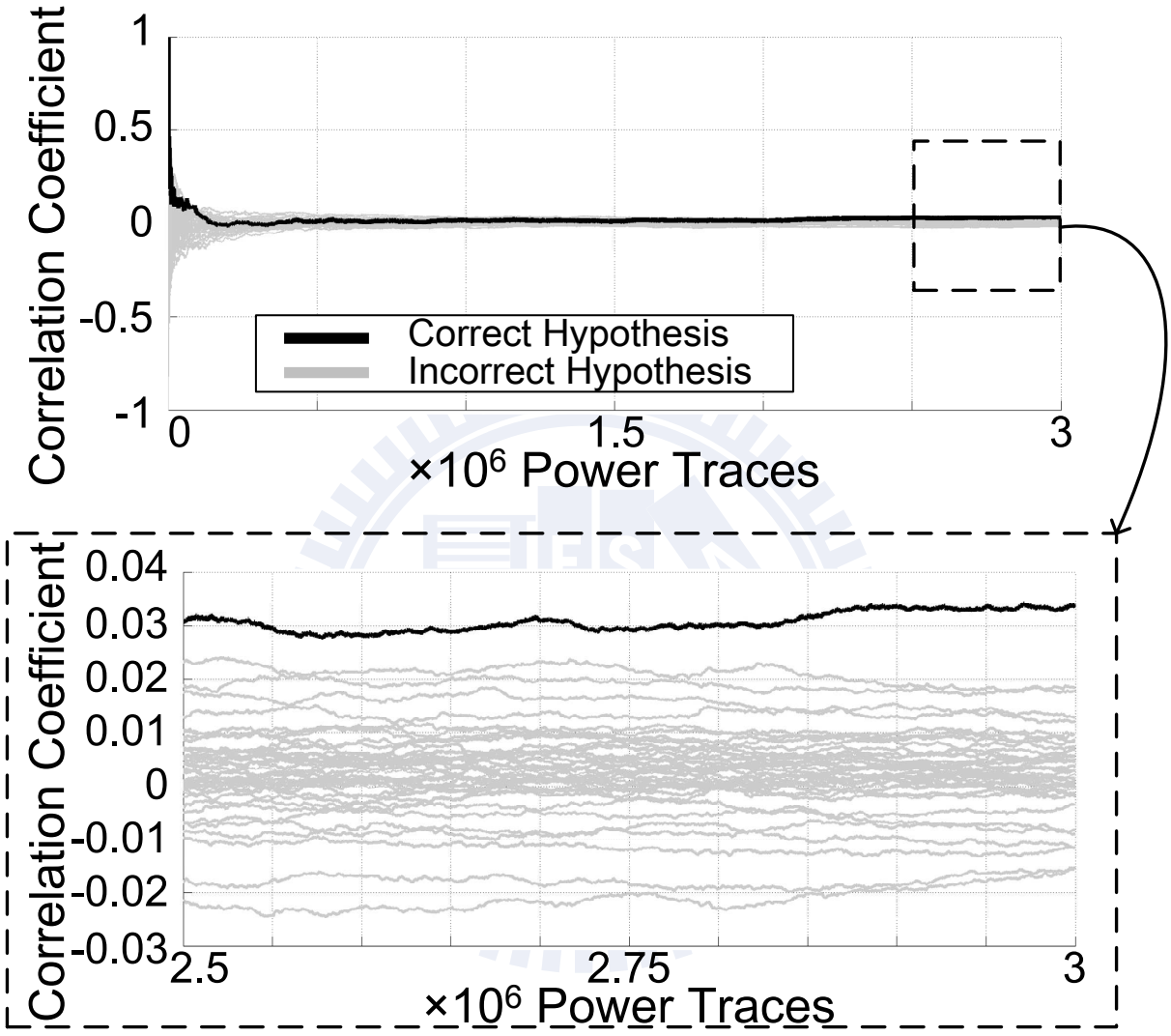


Figure 7.3: DPA attacks on protected ECC device processing ECSM with randomized computation, where the random sequence fails NIST P800-22 test suite.

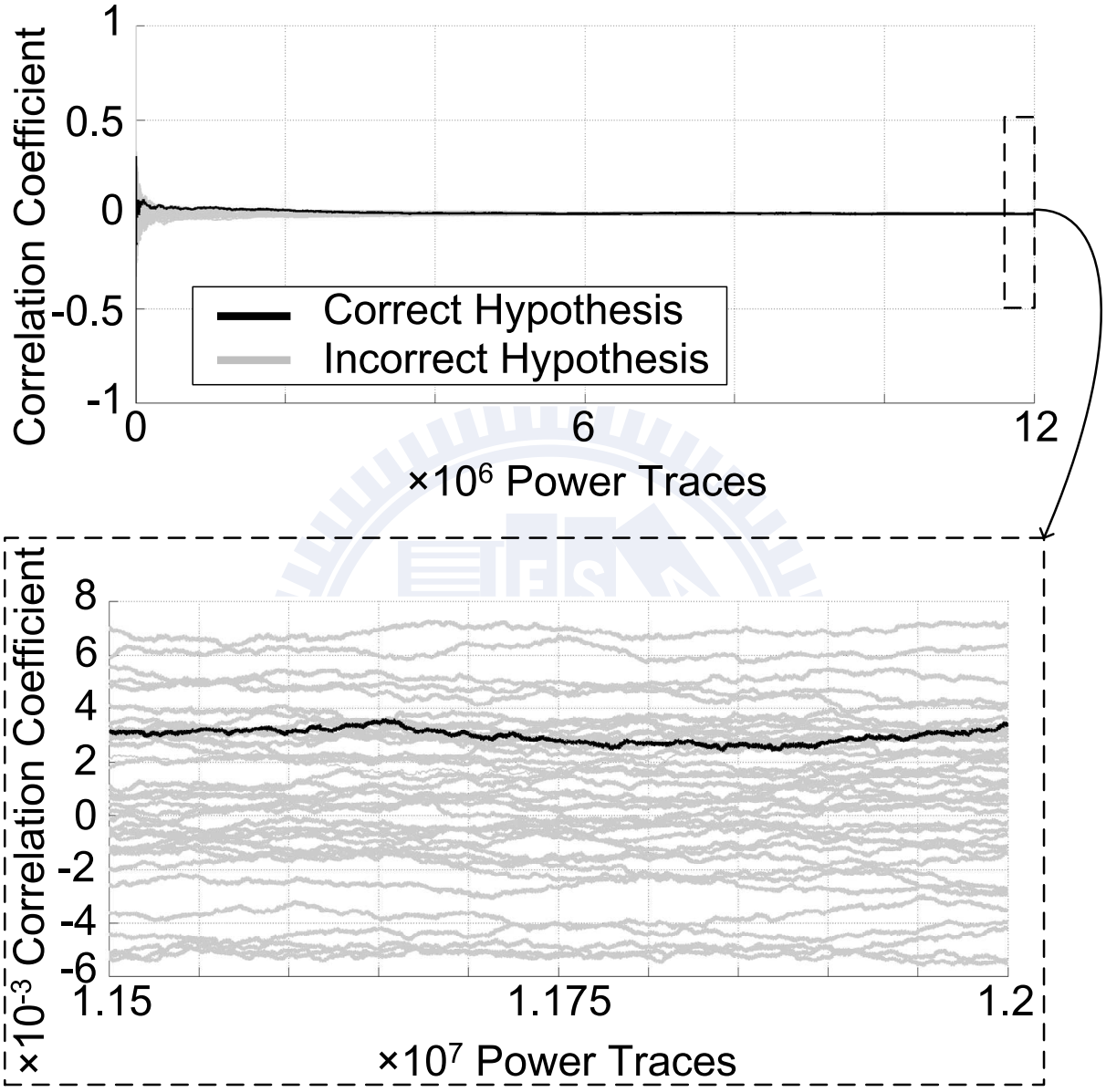


Figure 7.4: DPA attacks on protected ECC device processing ECSM with randomized computation, where the random sequence passes NIST P800-22 test suite.

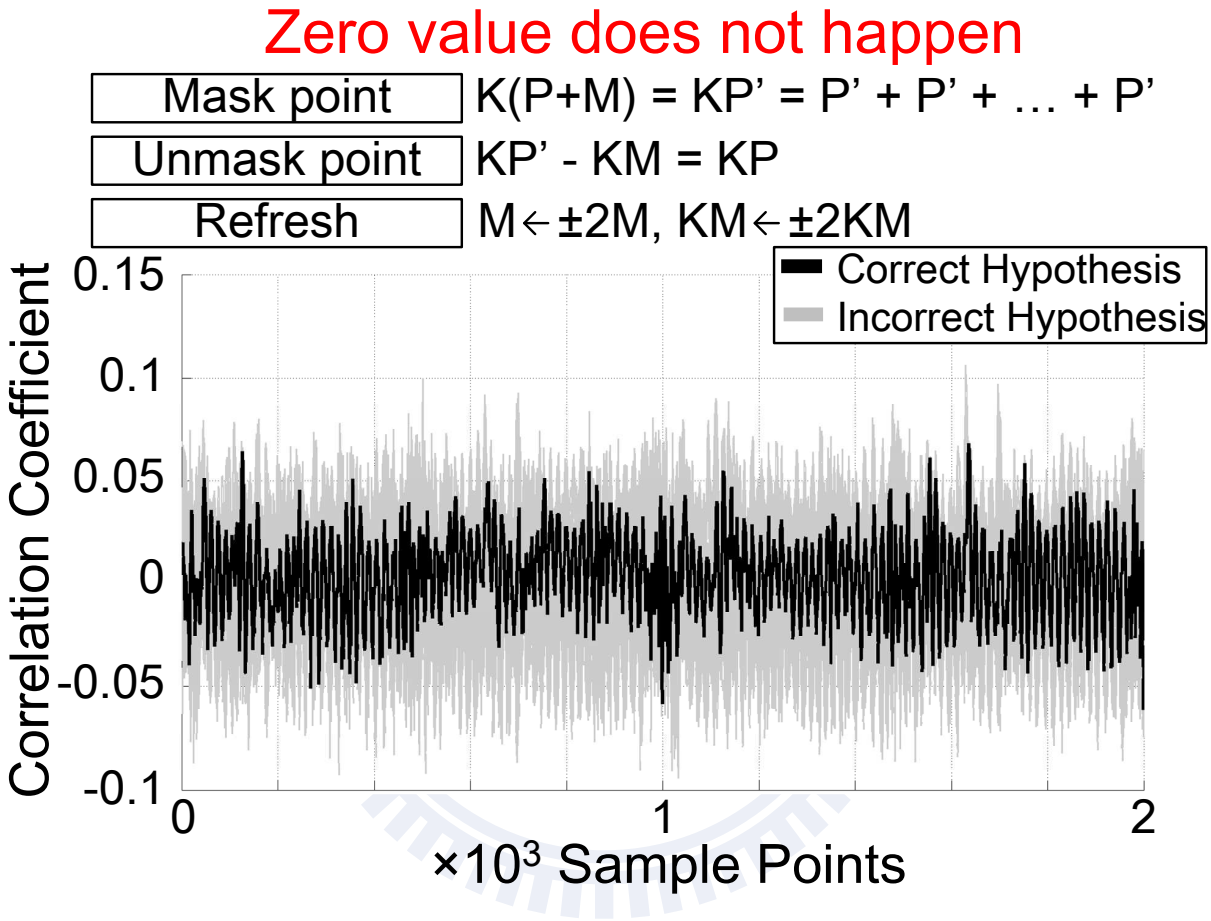


Figure 7.5: Correlation analysis obtained from a protected ECC chip by conducting the ZPA attacks.

of the key cannot be distinguished, where the mean of correlation coefficients is nearly equal because the collision operations are generated for all possible key values.

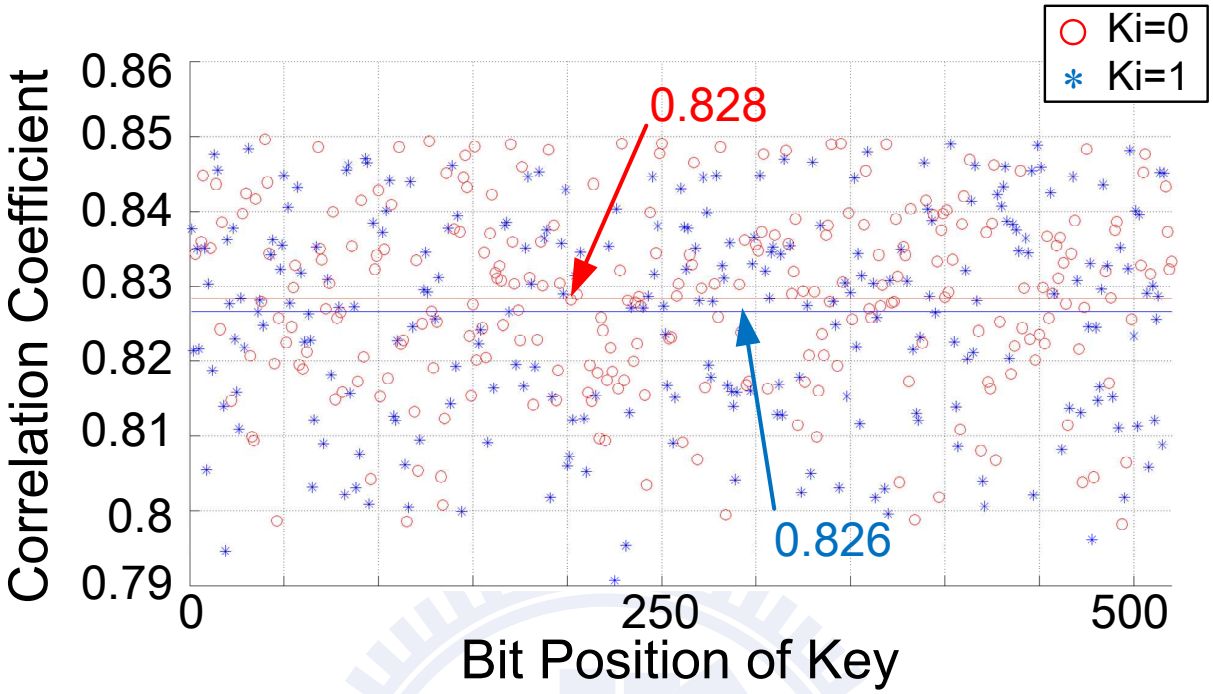


Figure 7.6: Correlation analysis obtained from a protected ECC chip by conducting the CPA attacks.

7.3 Overhead of SCA Resistance

For the DPA resistance, our approach is to mask the processed data uncorrelated with power traces without changing the logic family and without dominating the power consumption of key-dependent operations. The overhead is the simple control logic gate for determining the domain value. Compared with the unprotected design, the area and time overhead of protected design is 3.6% and 0%, respectively.

For the ZPA resistance, our countermeasure is to hide the information of primary input point by ECPA, and the results can be correctly returned by ECPS. The overhead is the combinational logic circuit for the operations required in the ECPG. Compared with the unprotected design, the area and time overhead of protected design is 4.2% and 1.52%, respectively.

To resist the SPA and CPA attacks, we exploit the RL-DAA ECSM. The overhead is the control logic gate and temporary register for saving the coordinates of EC points. Compared with the unprotected design, the area and time overhead of protected design is 1.04% and 0%, respectively.

As above, the low ($< 10\%$) overhead of SCA resistance has benefits in both of high-speed and resource-constrained applications. Although the random sequence is assumed to be an input for the design using randomized Montgomery operations and ECPG. The RO-RNG with jitter amplifier with several thousands of gates does not dominate the area complexity of ECC processor which needs hundreds thousands of gates.

7.4 Chip Achievement

By using a UMC 90-nm CMOS technology, our four IEEE P1363-compliant ECC chips with different specifications and design techniques are fabricated for various applications, including the mobile device, computing server, and Internet of Things (IoT). The standard AMBA AHB bus interface [96] is integrated, and the inputs of ECC processor are the user public/private-key, EC coordinates, EC parameters, and protocol instructions. To real-time perform these contents, the instruction decoder and pre/post-processing of data domain conversion are also combined in the ECC processor. The four ECC chips are named for our research groups, where the combination of "silicon" and "ocean", *so*, is used. To give a brief classification, the *so*ECC-B chip is for the acceleration of basic ECC function over DFs; the *so*ECC-P chip is targeted at the performance in terms of hardware speed and area cost; the *so*ECC-S chip achieves the fastest speed; the *so*ECC-G chip is designed for the green-energy requirements by several low power techniques of VLSI circuit.

7.4.1 *so*ECC-B: A 0.55 mm^2 19.2/8.2 ms $GF(p_{521})/GF(2^{409})$ 521-bit SCA-Resistant DF-ECC Processor Using Single-GFAU Architecture

Figure 7.7 shows the block diagram of the system of *so*ECC-B. For performing division operations, a radix-2 unified division algorithm [38] is used to save 62% execution cycles for the implementation using both of Kaliski's Montgomery invertion [82] and conventional radix-2 Montgomery multiplication. A single processing element, GFAU, is exploited to accelerate the ECPC and modular operations over DFs. The multiplexer complexity for the long bit length of registers is reduced by using the separated 32-bit circular shift registers. With LR-DAA ECSM and key-blinded approaches, the SPA, DPA, ZPA, and CPA attacks can be counteracted as the random value α is selected to be 32 bits. Note that, for this design using PRNG, we assume that the attacker does not apply the system reset before collecting power trances. Table 7.3 shows the summary of chip performance for our 521-bit SCA-resistant DF-ECC processor. Figure 7.8(a) and Figure 7.8(b) show the die photos, where an unprotected ECC chip and a protected ECC chip are implemented

to show the evaluation of SCA countermeasure.

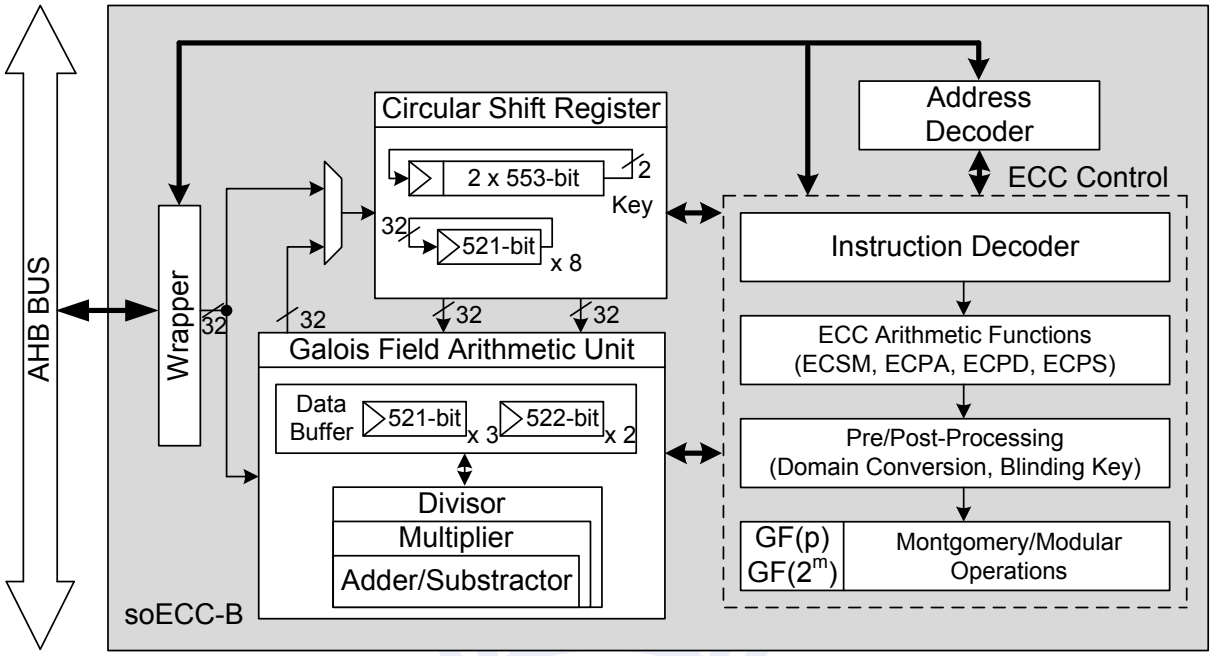
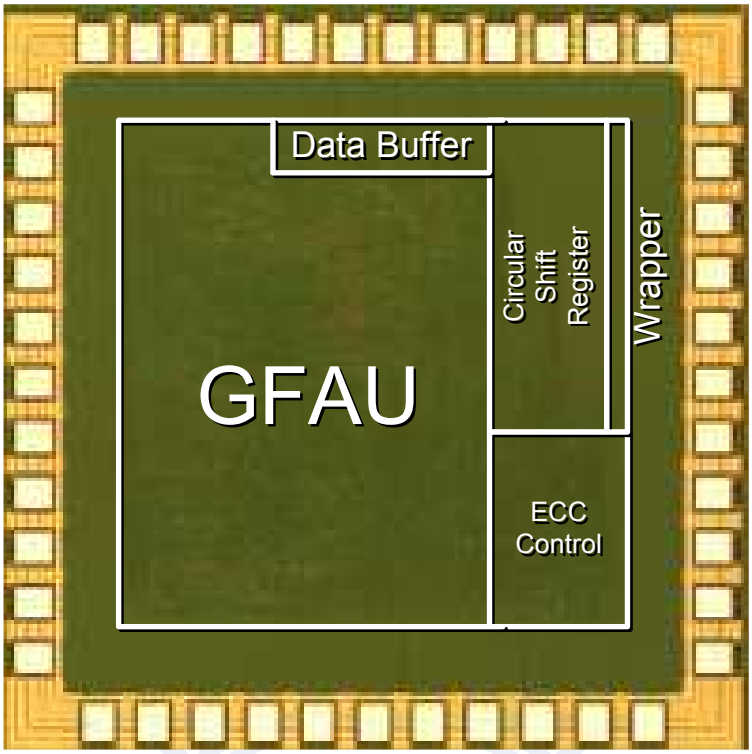


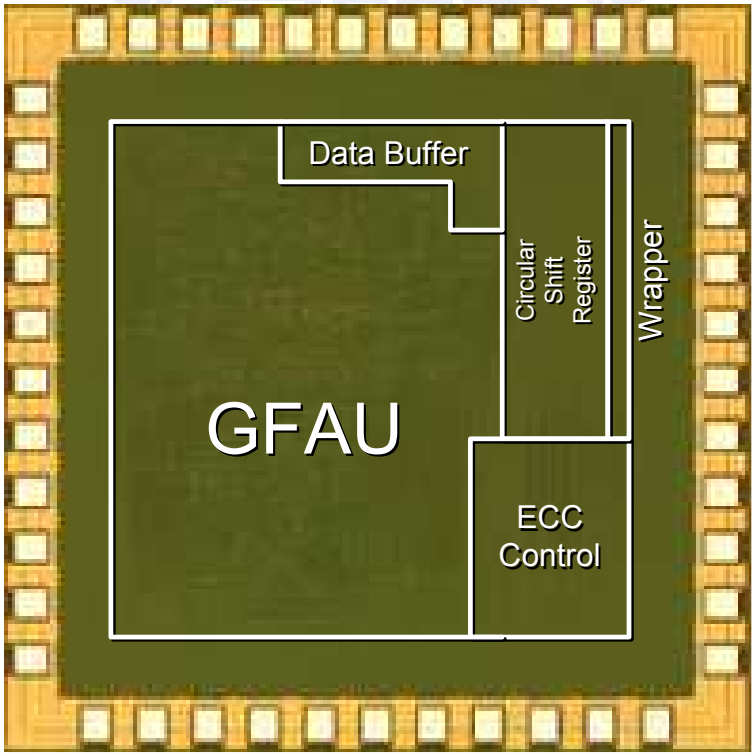
Figure 7.7: System architecture of *so*ECC-B.

Table 7.3: Chip Summary of *so*ECC-B

Technology	90-nm					
Core Area	0.55 mm^2					
Gate Count	170 K					
Key Size	521					
Field	Dual					
Field Length	$GF(p)$			$GF(2^m)$		
	160	256	521	163	283	409
Time (ms/ECSM)	1.62	4.4	19.2	1.15	3.33	8.2
f (MHz)	154	147	132	188	182	166
Power (mW)	66.3	67.6	58.5	72.5	87.0	86.4



(a) Without SCA resistance



(b) With SCA resistance

Figure 7.8: Chip micrograph of our 521-bit DF-ECC processor, where *so*ECC-B is shown in (b).

7.4.2 *so*ECC-P: A 0.41 mm^2 0.34/0.29 ms $GF(p_{160})/GF(2^{160})$ 160-bit SCA-Resistant DF-ECC Processor Using Heterogeneous Two-PE Architecture

Figure 7.9 shows the block diagram of the system of *so*ECC-P. For high speed, the radix-4 modular operations [58] are exploited to save the execution cycles for the implementation using radix-2 approach. To improve hardware utilization, the heterogeneous two-PE architecture, composed of one GFAU and one MAS, with priority-oriented scheduling is adopted. The memory hierarchy with local memory coherency is used to save 14.2% power consumption as compared with the circular shift registers [38]. With RL-DAA ECSM in Algorithm 9 and masked base point approaches, the SPA, DPA, ZPA, and CPA attacks can be counteracted. For this design, a basic RO-RNG without jitter amplifier is implemented in device. Table 7.4 lists the summary of chip performance for our 160-bit SCA-resistant DF-ECC processor. The measurement results of operating frequency and power consumption over supply voltage are shown in Figure 7.10. The maximum frequency is higher as the field length is lower because the critical path depends on the field length. Figure 7.11 shows the die photo of the ECC chip.

Table 7.4: Chip Summary of *so*ECC-P

Technology	90-nm	
Core Area	0.41 mm^2	
Gate Count	98 K	
Key Size	160	
Field	Dual	
Field Length	$GF(p)$	$GF(2^m)$
	160	160
Time (ms/ECSM)	0.34	0.29
f (MHz)	194	204
Energy (μJ/ECSM)	11.7	9.3

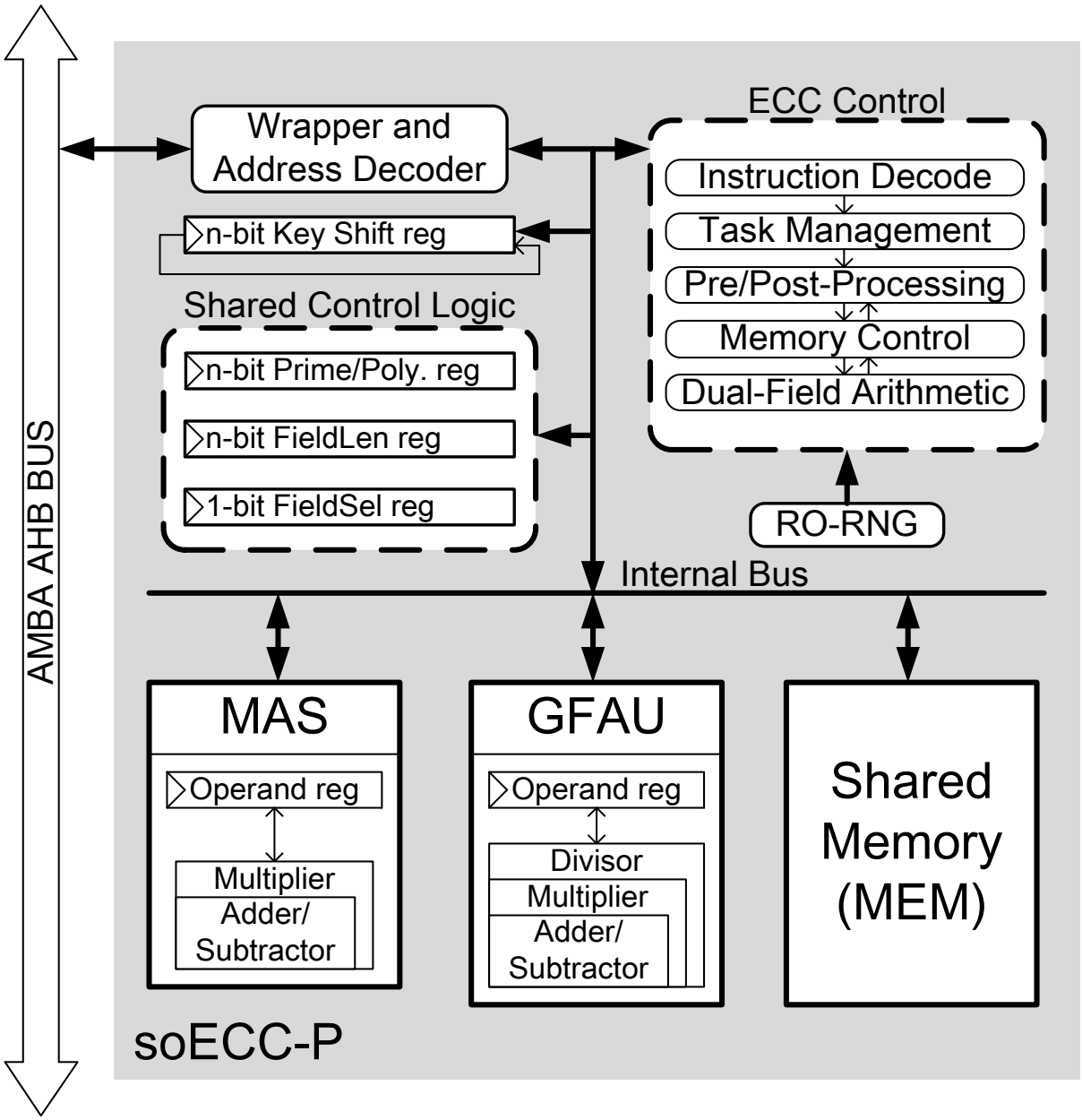


Figure 7.9: System architecture of *so*ECC-P.

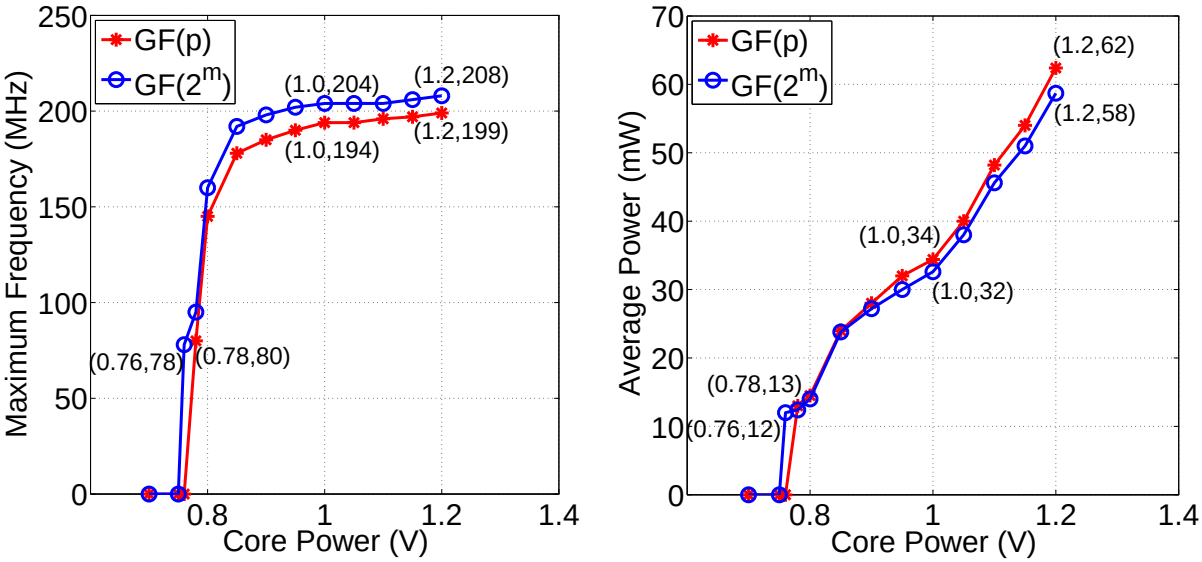


Figure 7.10: Shmoo plot for the measurement results of chip *so*ECC-P.

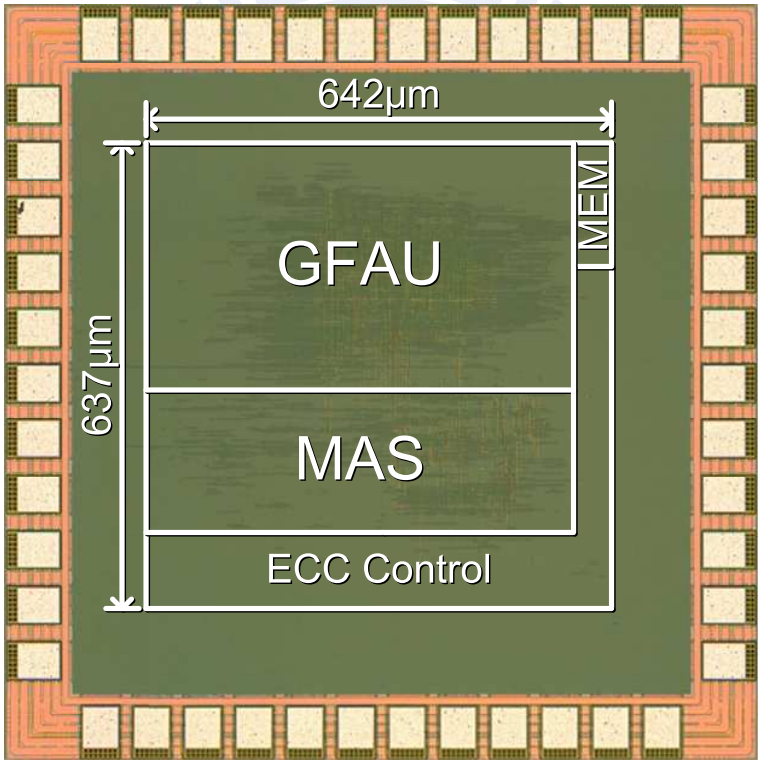


Figure 7.11: Chip micrograph of our 160-bit DF-ECC processor, *so*ECC-P.

7.4.3 *so*ECC-S: A 1.38 mm^2 3.40/2.77 ms $GF(p_{521})/GF(2^{521})$ 521-bit SCA-Resistant DF-ECC Processor Using Heterogeneous Two-PE Architecture

Figure 7.12 shows the block diagram of the system of *so*ECC-S. To save the computation overhead against SCAs from the key-blinded approach with extended key size, the radix-4 randomized Montgomery operations [97] described in sub-section 4.1 are exploited. The heterogeneous two-PE architecture is adopted to accelerate the ECPC, ECPG, and modular operations over DFs, where it consists of one JS-GFAU and one MAS. The memory hierarchy with local memory coherency is used to transfer data efficiently. With RL-DAA ECSM in Algorithm 9 and masked base point approaches, the SPA, DPA, ZPA, and CPA attacks can be defeated. To give the robustness against SCAs, a RO-RNG with jitter amplifier is implemented in device. For accelerating the calculation of ECPG, the EC points can be randomly generated by parallel computation from the components of JS-GFAU and MAS. Since the ECPG can be achieved in chip device, another advantage is the transmission reduction of public key [7]. The compressed form is that the y coordinate, denoted $\tilde{y}$, is a single bit, where $\tilde{y} = y \pmod 2$. The decompression of y is to compute a square root z of $g \equiv x^3 + a_p x + b_p \pmod p$ over $GF(p)$. Let $\tilde{z}$ be the rightmost bit of z. If $\tilde{z} = \tilde{y}$, then $y \leftarrow z$, else $y \leftarrow p - z$. In the case of field $GF(2^m)$, the decompression of y is first to compute $\beta \equiv \alpha(x^2)^{-1}$, where $\alpha \equiv x^3 + a_b x^2 + b_b \pmod{p(x)}$. And then find a field element z such that $z^2 + z = \beta$. Let $\tilde{z}$ be the rightmost bit of z. Finally, compute $y \equiv (z + \tilde{z} + \tilde{y})x$ and return coordinate value y.

Table 7.5 lists the summary of chip performance for our 521-bit SCA-resistant DF-ECC processor. The measurement results of operating frequency and energy dissipation over supply voltage are shown in Figure 7.13(a) and Figure 7.13(b), respectively. The range of supply voltage is from 0.6 V to 1.2 V. The maximum frequency is higher as the field length is lower because the critical path depends on the field length. In contrast, the energy consumption per ECSM operation is proportional to the field length because of the binary method of scalar multiplication. Figure 7.14 shows the die photo of the ECC chip.

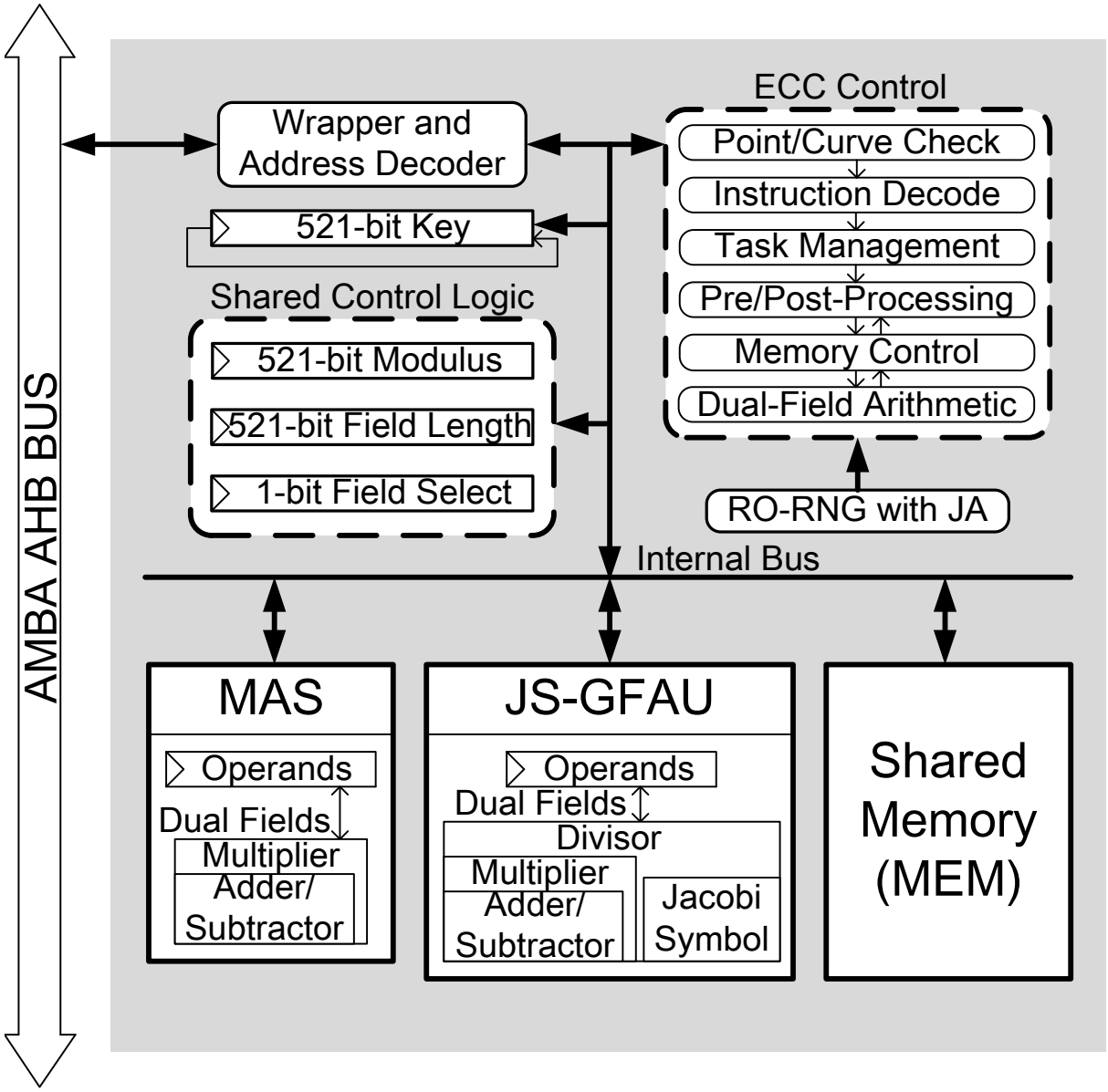
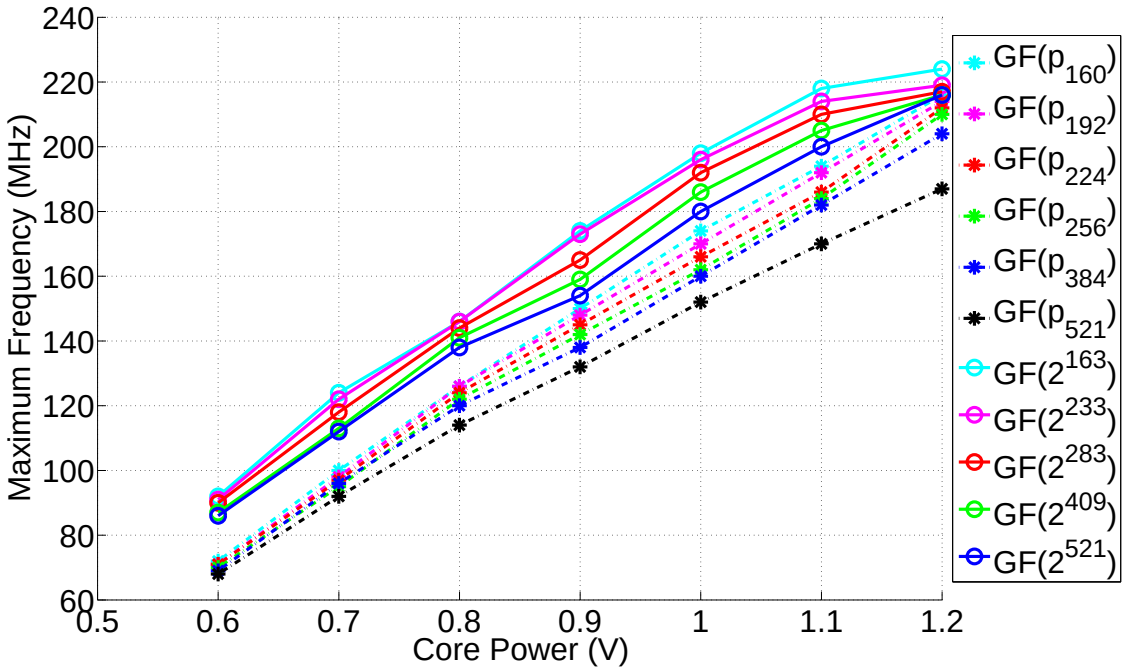


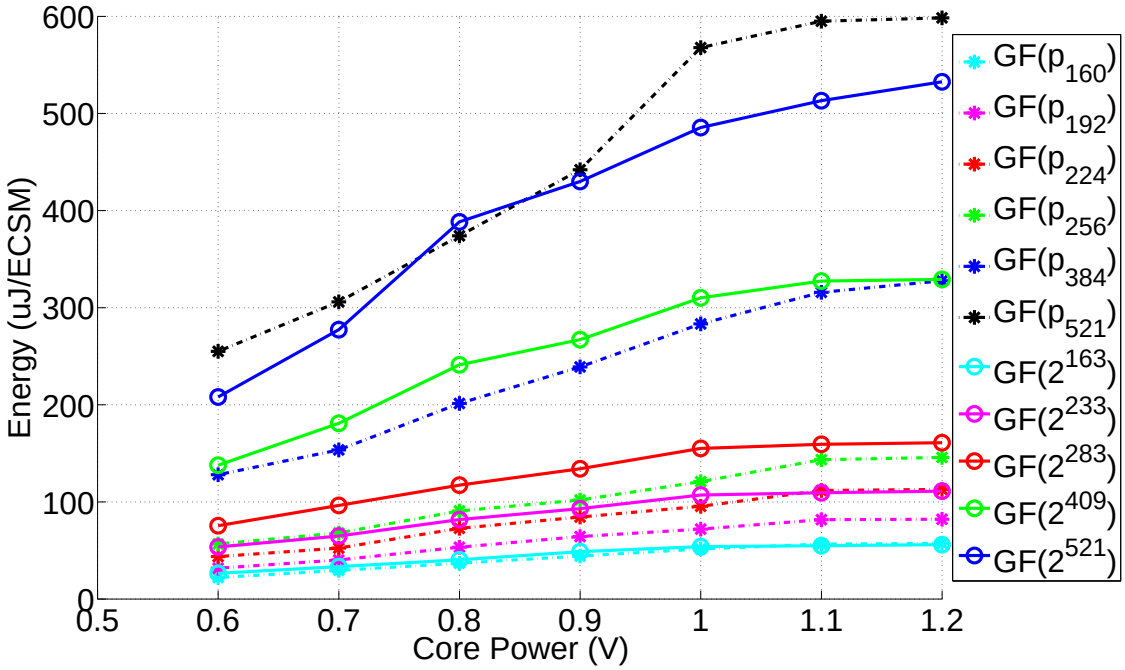
Figure 7.12: System architecture of *so*ECC-S.

Table 7.5: Chip Summary of *so*ECC-S

Technology	90-nm				
Core Area	1.38 mm^2				
Gate Count	342 K				
Key Size	521				
Field	Dual				
Field Length	$GF(p)$		$GF(2^m)$		
	160	521	163	409	521
Time (ms/ECSM)	0.29	3.40	0.25	1.72	2.77
f (MHz)	214	187	224	217	216
Energy (μJ/ECSM)	57	598	56	329	532



(a) Operating frequency over supply voltage



(b) Energy dissipation over supply voltage

Figure 7.13: Shmoo plot for the measurement results of chip *so*ECC-S.

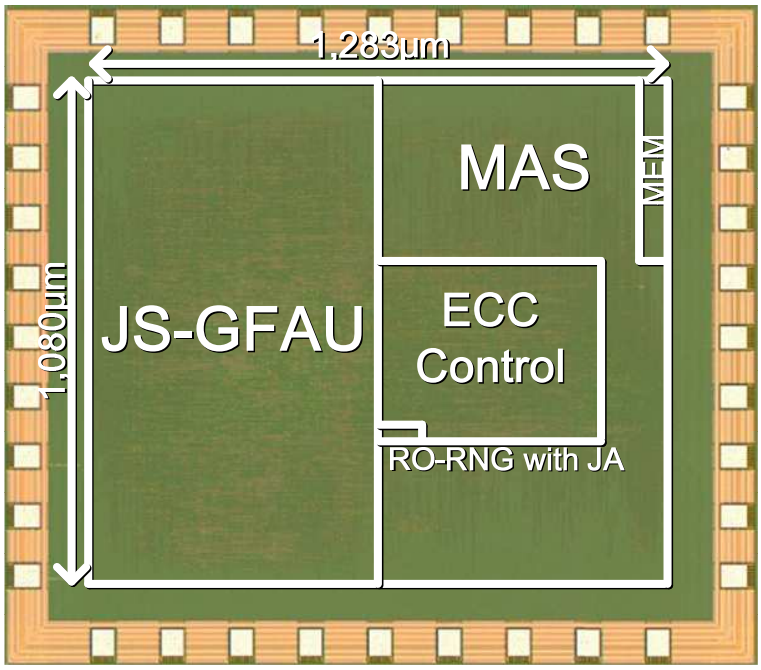


Figure 7.14: Chip micrograph of our 521-bit DF-ECC processor, *so*ECC-S.

7.4.4 *so*ECC-G: A 10.8/9.2 ms 438/437 μW $GF(p_{192})/GF(2^{192})$ 192-bit SCA-Resistant DF-ECC Processor Using Single-GFAU Architecture

Figure 7.15 shows the block diagram of the system of our proposed crypto engine (CE), where the following are the supported security schemes manipulated by CE control.

- AES schemes: CTR, CBC-MAC, CMAC, and CCM modes, where the encryption and decryption key sizes are 128 bits.
- ECC schemes: ECPA, ECPD, ECPS, ECSM operations, and DHK agreement over $GF(p)$ and $GF(2^m)$, where the public and private key sizes are 192 bits.
- Modular operations: addition, subtraction, multiplication, inversion, and division over $GF(p)$ and $GF(2^m)$.
- Random number sequence: 8-bit true random bitstream per cycle.

To conveniently integrate our proposed CE into an embedded system, a standard AMBA AHB bus interface [96] is used. Also, for real-time transmitting the encrypted and decrypted message frame in system applications, a direct data path from AES core to

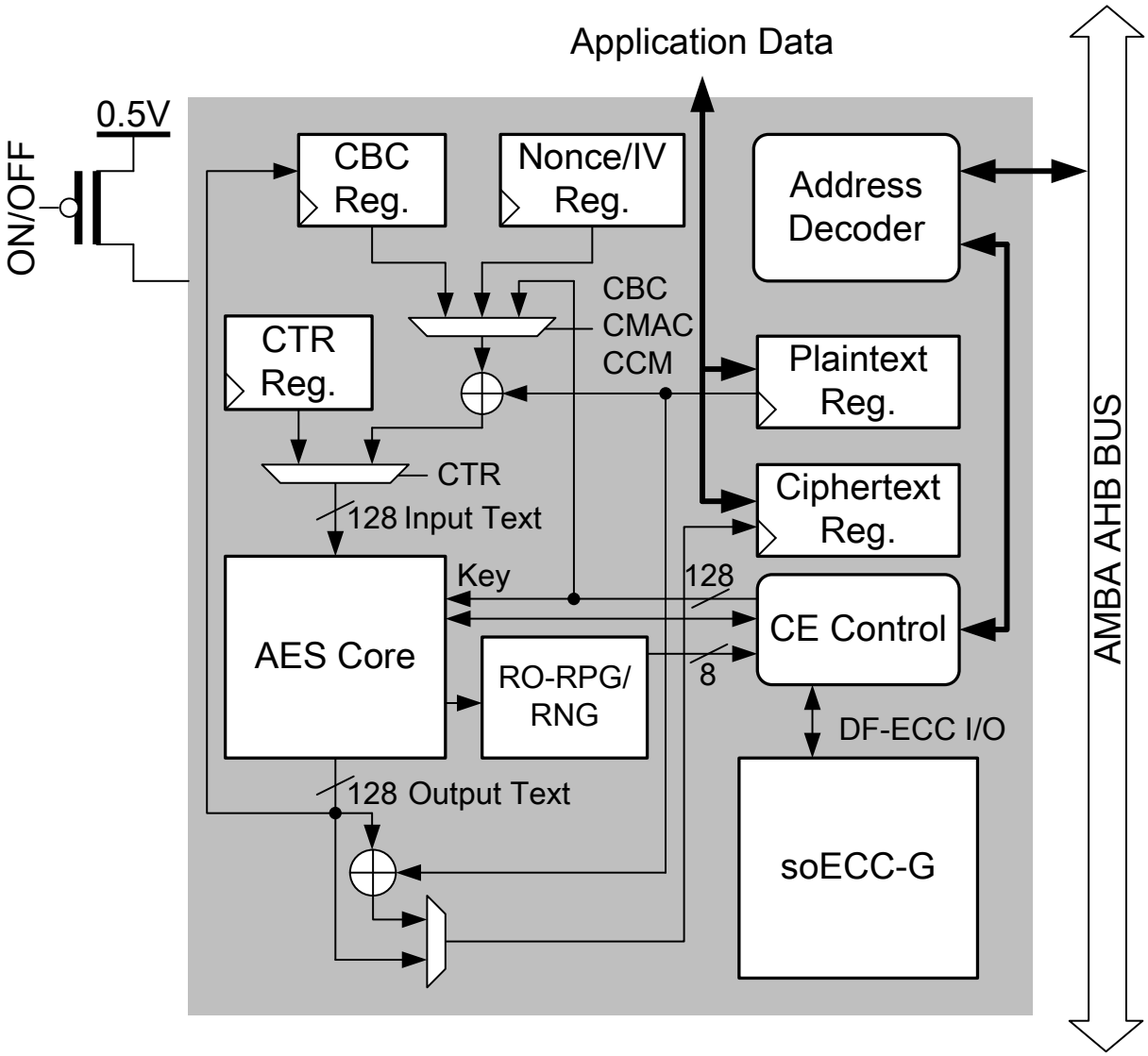


Figure 7.15: System architecture of our CE.

data memory is exploited to access outside memory without dominating system bus. Since the electronic metastability inherently exists in free running ring oscillators (ROs), the ROs can be efficiently reused to implement both random power generator (RO-RPG) [84] and random number generator (RO-RNG), where they are required to protect the key in AES core from revealing by power-analysis attacks. To save the SCA-resistant overhead of ECC processor, *so*ECC-G, from the key-blinded approach with extended key size, the radix-2 randomized Montgomery operations described in sub-section 4.1 are exploited. A single processing element, GFAU, is efficiently exploited to accelerate the ECPC in ECC schemes.

To support the requirements of security functions in the applications of IoT, a 128-bit AES core, a 192-bit DF-ECC processor, and an 8-bit RO-RNG are integrated with bio-signal processing system [98]. The digital processing module and sensing interface are included, and a 32-bit RISC CPU core, Andes N903-C05 [99], is utilized to enhance the instruction scheduling. To reduce the system power, the processor sleeps in the data collection stage and activates in the data processing stage by a wakeup/power control logic. To improve the battery life of portable device, the chip working at 0.5 V low supply voltage is achieved by a reconstructed logic cell [100]. Moreover, to apply the voltage scaling scheme, the cell behavior and timing information in the range of 1 V to 0.5 V supply voltage are simulated and re-calibrated, then the cell library after picking out the cells which work normally is reconstructed. With the reconstructed cell library, the proposed CE chip can be implemented by using standard-cell based design procedure. By scaling the supply voltage from 1.0 V to 0.5 V, the power is reduced by 80-84%, where Figure 7.16 plots the power consumption versus the voltage and frequency. Since the power consumption is dominated by leakage power at low frequency, the CE operating frequency is raised to work at 25 MHz for the sake of energy efficiency. Additionally, as the operations in security schemes are finished, the CE can be turned off by the power gating for leakage power saving. The hardware performance of our AES core achieves 60 Mb/s 99 μW, where the throughput is 6 times higher than the 10 Mb/s required in IEEE 802.15.6 standard and 30 times higher than the 2 Mb/s specified in IEEE 802.15.4 standard. The RO-RNG generates 25 Mb/s random sequence and consumes 47 μW. For the DF-ECC processor it can perform one $GF(p_{192})$ ECSM in 10.8 ms with 438 μW and

one $GF(2^{192})$ in 9.2 ms with 437 μW, sufficiently passing the 250 ms at 13.56 MHz reaction requirement in ISO 18000-3 of RFID tag applications [44] using Schnorr's identification protocol [22]. For the hardware complexity, the equivalent gate counts of AES core, RO-RNG, and DF-ECC processor are 7.24 K, 0.43 K, and 61.68 K, respectively. Figure 7.17 shows the die photo and Table 7.6 shows the summary of chip performance for our 192-bit SCA-resistant DF-ECC processor.

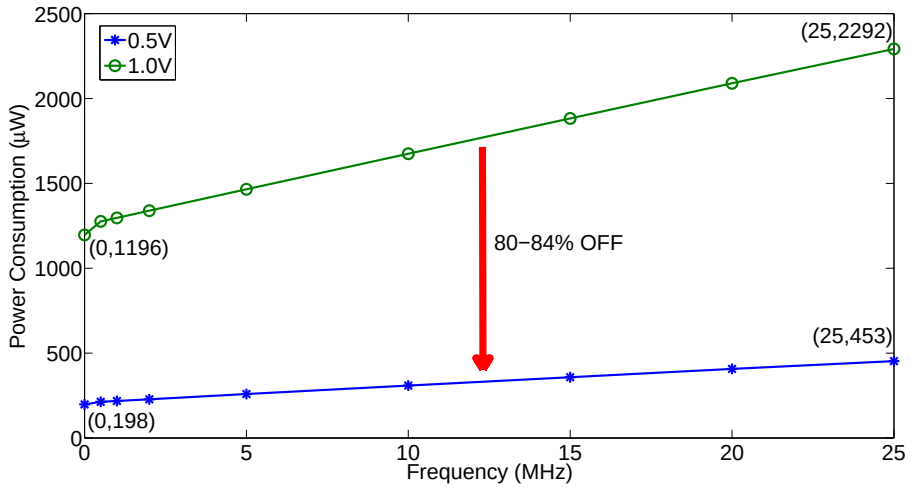


Figure 7.16: The power consumption of CE chip working at different supply voltage and operation frequency.

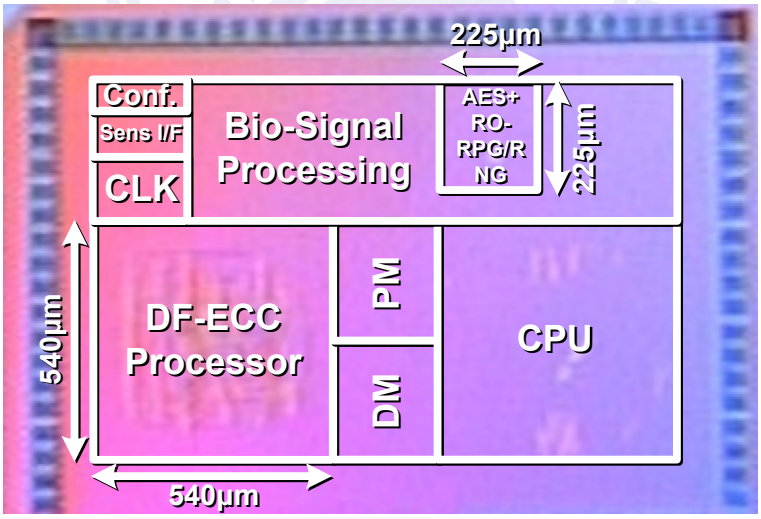


Figure 7.17: Chip micrograph of our CE cooperating with embedded processor and other components, such as data memory (DM), program memory (PM), sensing interface, and bio-signal processing module.

Table 7.6: Chip Summary of *so*ECC-G

Technology	90-nm	
Core Area	0.34 mm^2	
Key Size	192	
Field	Dual	
Field Length	$GF(p)$	$GF(2^m)$
	192	192
Time (ms/ECSM)	10.8	9.2
f (MHz)	25	25
Energy (μJ/ECSM)	4.73	4.02

7.5 Comparison

7.5.1 High-Performance and SCA-Resistant ECC Processor for IEEE P1363 Applications

Based on our proposed programmable design architecture described in sub-section 7.4.2, six additional ECC designs, including the 192-bit DF (ECC-DF192), 521-bit DF (ECC-DF521), 192-bit $GF(p)$ (ECC-P192), 256-bit $GF(p)$ (ECC-P256), 163-bit $GF(2^m)$ (ECC-B163), 192-bit $GF(2^m)$ (ECC-B192) ECC processors, are also implemented to compare with the previous works. The layout view of ECC-DF521 is shown in Figure 7.18. The chip performance and implementation results with comparison to other related ECC hardware implementations over $GF(p)$ and $GF(2^m)$ are summarized in Table 7.7 and Table 7.8, respectively. Note that, in consideration for the scaling effect of fabrication technology and supply voltage, the normalization factor of area-time product and energy can be referred to [101] and [102], respectively. The normalization factor of area-time product is proportional to the ratio of minimum gate length for transistor; the normalization factor of energy is proportional to the square ratio of minimum gate length for transistor multiplied by the square ratio of supply voltage. By interleaved processing the

ECSM operations without duplicating PEs, our heterogeneous two-PE ECC processor with arithmetic unit integration outperforms previous works using four identical multipliers architecture [24,28], separated arithmetic units [17,25,29,46,103], and single integrated arithmetic unit [22, 38, 45, 49] in terms of cost effectiveness. Moreover, since an operation scheduling in a key-independent manner with randomized intermediate values is used to protect the chip from power-analysis attacks including SPA, DPA, ZPA, and CPA attacks, our design supports higher security level. These benefits demonstrate that the proposed solution is well suited for the portable applications such as mobile device.

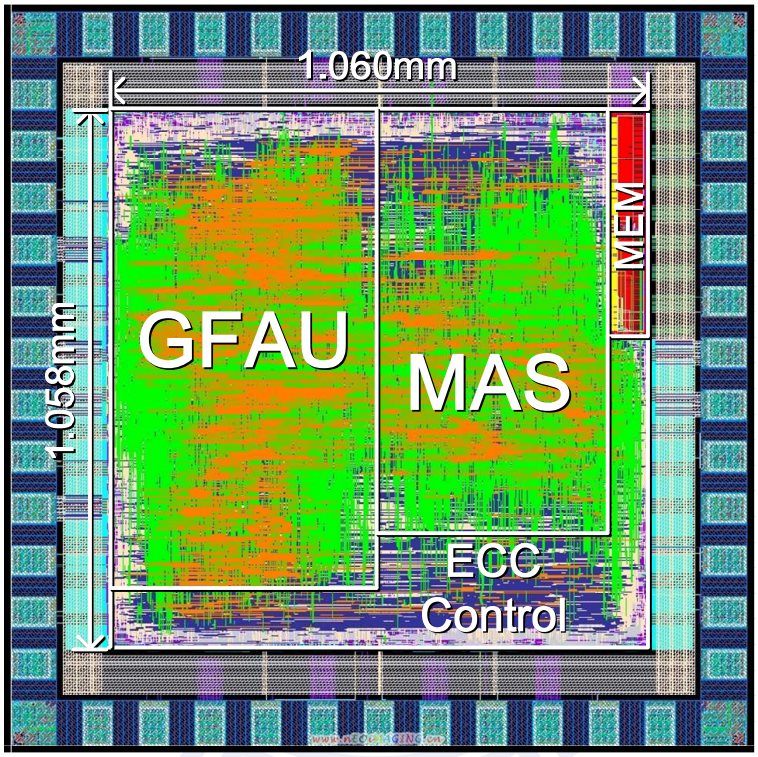


Figure 7.18: Layout view of our 521-bit DF-ECC processor, ECC-DF521.

Table 7.9 shows the comparison of our another ECC design described in sub-section 7.4.3 with related works. Compared with single-PE 521-bit designs [38,59] and a four-multiplier-based 160-bit design [28], our *so*ECC-S chip outperforms both in hardware speed and in protection against SCAs. These advantages demonstrate the proposed design solution is suitable for high-end applications, such as cloud computing.

Table 7.7: Comparison Among Previous Approaches for $GF(p)$

	Technology	Area (mm^2/KGates)	Field	Field Length	Time(ms/ ECSM)@f(MHz)	KCycles	AT	Energy (μJ/ECSM)	ECSM Method	Power-Analysis Resistance
Our *so*ECC-P (Measurement@1.0V)	90-nm	0.41/98	Dual	160	0.34@194	66.2	1	11.7	RL-DAA	SPA, DPA, ZPA, and CPA attacks
TCAS-II'09 [24] (Measurement@1.2V)	130-nm	1.44/169	Dual	160	0.61@121	74.0	3.09 (2.14†)	42.6 (14.2♮)	LR-DAS	-
TVLSI'11 [28] (Measurement@1.2V)	130-nm	1.35/179	Dual	160	0.39@141	54.4	2.09 (1.45†)	31.0 (10.3♮)	LR-DAS	-
Our ECC-DF192 (Post-layout@1.0V)	90-nm	0.46/122	Dual	192	0.36@263	94.2	1	24.4	RL-DAA	SPA, DPA, ZPA, and CPA attacks
RFIDSec'05 [45]* (Post-layout)	90-nm	0.09/23.8	Dual	192	1,300@0.545	677	704.5	39	LR-DAA	SPA and DPA attacks
Our ECC-DF521 (Post-layout@1.0V)	90-nm	1.12/313	Dual	160	0.30@220	66.2	1	12	RL-DAA	SPA, DPA, ZPA, and CPA attacks
				192	0.43@220	94.2	-	26		
				224	0.59@217	127.2	-	39		
				256	0.76@217	165.1	1	54		
				384	1.69@217	366.1	-	143		
				521	3.15@212	668.6	1	292		
ESSCIRC'10 [38] (Measurement@1.0V)	90-nm	0.55/170	Dual	160	1.62@154	249.5	2.93	107	LR-DAA	SPA, DPA, ZPA, and CPA attacks ¶
				256	4.40@147	646.8	3.14	297		
				521	19.2@132	2,534	3.31	1,123		
Our ECC-P192 (Post-layout@1.0V)	90-nm	0.41/108	$GF(p)$	192	0.36@263	94.2	1	23.9	RL-DAA	SPA, DPA, ZPA, and CPA attacks
ISCAS'07 [29] (Post-layout)	130-nm	0.15/23.6	$GF(p)$	192	2.5@200	502	1.52 (1.05†)	-	LR-DAA	SPA and DPA attacks
Our ECC-P256 (Post-layout)	Virtex-II Pro	8,272 CLB Slices	$GF(p)$	256	4.41@37	165.1	1	-	RL-DAA	SPA, DPA, ZPA, and CPA attacks
TCAS-I'06 [17] (Post-layout)	Virtex-II Pro	15,755 CLB Slices	$GF(p)$	256	3.86@39	151.4	1.67	-	LR-DA	-

AT product = gate count (or CLB slices) × time.

Energy = average power × time.

† Normalization factor is 0.69 (90-nm/130-nm).

♮ Normalization factor is 0.33 ((90-nm/130-nm)^2×(1.0V/1.2V)^2).

¶ Resistance failed by activating reset signal before ECSM calculation.

* Support hash function.

LR-DAS: left-to-right double-and-add/subtract.

Table 7.8: Comparison Among Previous Approaches for $GF(2^m)$

	Technology	Area (mm^2/KGates)	Field	Field Length	Time(ms/ ECSM)@f(MHz)	KCycles	AT	Energy (μJ/ECSM)	ECSM Method	Power-Analysis Resistance
Our *so*ECC-P (Measurement@1.0V)	90-nm	0.41/98	Dual	160	0.29@204	62.5	1	9.3	RL-DAA	SPA, DPA, ZPA, and CPA attacks
TCAS-II'09 [24] (Measurement@1.2V)	130-nm	1.44/169	Dual	160	0.37@146	54.3	2.20 (1.52†)	30.5 (10.1$^{\natural}$)	LR-DAS	-
TVLSI'11 [28] (Measurement@1.2V)	130-nm	1.35/179	Dual	160	0.27@158	43.0	1.70 (1.18†)	21.6 (7.1$^{\natural}$)	LR-DAS	-
Our ECC-DF192 (Post-layout@1.0V)	90-nm	0.46/122	Dual	192	0.32@263	84.7	1	18.2	RL-DAA	SPA, DPA, ZPA, and CPA attacks
RFIDSec'05 [45]* (Post-layout)	90-nm	0.09/23.8	Dual	192	800@0.545	426	487.7	24	LR-DAA	SPA and DPA attacks
Our ECC-DF521 (Post-layout@1.0V)	90-nm	1.12/313	Dual	163	0.26@238	62.5	1	14	RL-DAA	SPA, DPA, ZPA, and CPA attacks
				233	0.52@238	124.3	-	34		
				283	0.76@238	181.3	1	55		
				409	1.58@235	372.5	1	141		
ESSCIRC'10 [38] (Measurement@1.0V)	90-nm	0.55/170	Dual	163	1.15@188	216.2	2.40	76	LR-DAA	SPA, DPA, ZPA, and CPA attacks ¶
				283	3.33@182	606.1	2.36	225		
				409	8.20@166	1,361	2.82	480		
Our ECC-B163 (Post-layout@1.0V)	90-nm	0.24/65	$GF(2^m)$	163	0.22@277	62.5	1	8.2	RL-DAA	SPA, DPA, ZPA, and CPA attacks
TC'08 [22] (Synthesis@1.2V)	130-nm	- /12.5	$GF(2^m)$	163	244@0.001	275.8	213.2 (147.6†)	8.94 (3.0$^{\natural}$)	LR-DAA	SPA attacks
MWSCAS'09 [25] (Post-layout@1.8V)	180-nm	2.10/69	$GF(2^m)$	163	1.89@181	228.1	9.12 (4.56‡)	257 (15.4§)	LR-DA	-
ICITA'05 [103] (Synthesis@3.3V)	350-nm	- /46	$GF(2^m)$	163	3.05@44	134	9.81 (2.52$^{\flat}$)	-	LR-DAS	-
RFIDSec'06 [46] (Synthesis@3.3V)	350-nm	- /16	$GF(2^m)$	163	27.9@13.56	376.8	31.22 (8.03$^{\flat}$)	-	LR-DAA	SPA and DPA attacks
Our ECC-B192 (Post-layout@1.0V)	90-nm	0.32/84.6	$GF(2^m)$	192	0.32@263	84.7	1	17.1	RL-DAA	SPA, DPA, ZPA, and CPA attacks
CHES'06 [49] (Synthesis@3.3V)	350-nm	- /29.4	$GF(2^m)$	192	118@12	1,416	128.1 (32.95$^{\flat}$)	-	-	-

AT product = gate count × time.

Energy = average power × time.

† Normalization factor is 0.69 (90-nm/130-nm).

‡ Normalization factor is 0.50 (90-nm/180-nm).

♭ Normalization factor is 0.26 (90-nm/350-nm).

♮ Normalization factor is 0.33 ((90-nm/130-nm)^2×(1.0V/1.2V)^2).

§ Normalization factor is 0.08 ((90-nm/180-nm)^2×(1.0V/1.8V)^2).

¶ Resistance failed by activating reset signal before ECSM calculation.

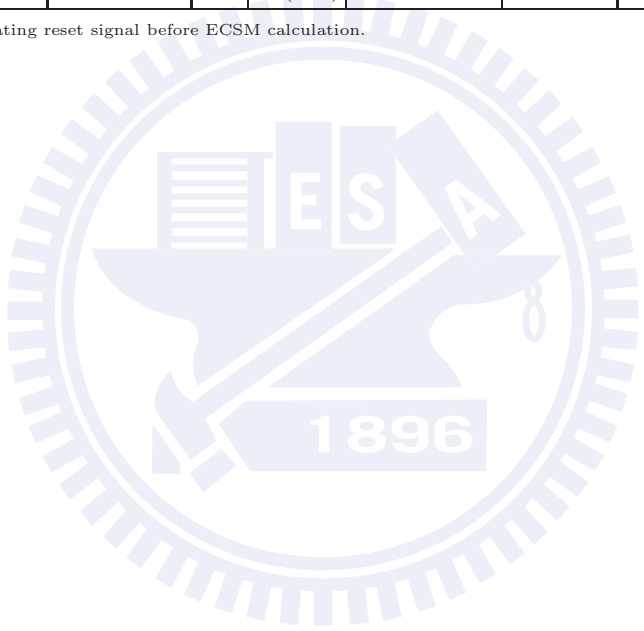
* Support hash function.

LR-DAS: left-to-right double-and-add/subtract.

Table 7.9: Comparison Among Previous Approaches

	Technology	Area (mm^2/KGates)	Field	Field Length	Time(ms/ ECSM)@f(MHz)	Energy (μJ/ECSM)	Compressed Public-Key	Power-Analysis Resistance
Our *so*ECC-S (Measurement@1.2V)	90-nm	1.38/342	Dual	$GF(p_{160})$	0.29@214	57	Yes	SPA, DPA, ZPA, and CPA attacks
				$GF(p_{521})$	3.40@187	598		
				$GF(2^{163})$	0.25@224	56		
				$GF(2^{409})$	1.72@217	329		
				$GF(2^{521})$	2.77@216	532		
ESSCIRC'10 [38] (Measurement@1.0V)	90-nm	0.55/170	Dual	$GF(p_{160})$	1.62@154	107	No	SPA, DPA, ZPA, and CPA attacks ¶
				$GF(p_{521})$	19.2@132	1,123		
				$GF(2^{163})$	1.15@188	76		
				$GF(2^{409})$	8.20@166	480		
TCAS-II'12 [59] (Post-Layout@1.0V)	90-nm	0.58/168	Dual	$GF(p_{160})$	0.74@256	20	No	SPA and DPA attacks
				$GF(p_{521})$	8.08@250	450		
				$GF(2^{163})$	0.63@270	20		
				$GF(2^{409})$	4.65@263	240		
TVLSI'11 [28] (Measurement@1.2V)	90-nm	1.35/179	Dual	$GF(p_{160})$	0.39@141	31.0	No	-
				$GF(2^{160})$	0.27@158	21.6		

¶ Resistance failed by activating reset signal before ECSM calculation.



7.5.2 Energy-Efficient and SCA-Resistant Crypto Engine for IEEE 802.15.4/6 Applications

The comparison of our CE chip with other related AES [76, 104–107] and ECC [22, 24, 25, 28, 57, 58] hardware implementations are summarized in Table 7.10. By using our highly-integrated architecture with low overhead of randomized techniques, it shows advantages in energy efficiency and power-analysis resistance. Besides, through system-level integration, security schemes specified in both of IEEE 802.15.4 and IEEE 802.15.6 standards are supported. Compared to a previous work of IEEE 802.15.4 security device with RSA-based Diffie-Hellman Key (DHK) agreement [50], we also implement another crypto engine (CE-II) by the same FPGA family. Note that the key size of DF-ECC processor in CE-II is set to 224 bits, where it achieves the same level of security as 2048-bit RSA used in [50]. The synthesized results are shown in Table 7.10. Our CE-II occupies 20,166 slice LUTs and 4,399 slice registers which are 13% and 65% less than those of [50], respectively. On the other hand, our DF-ECC processor needs at most 336K cycles to complete the DHK agreement, which is about 94% less than those of RSA-based DHK agreement design. In addition, our design supports higher security level against power-analysis attacks. These advantages indicate that our proposed solution is well suitable for the resource constrained applications such as IoT.

Table 7.10: ASIC and FPGA Comparison Among Previous Works

	Technology	Area (mm^2)	AES Throughput (Mb/s)@f(MHz)	AES Energy (μJ/Mb)	ECSM Time (ms)@f(MHz)		ECSM Energy (μJ)	Power-Analysis Resistance	Standards
Our CE (Measurement)	90-nm	0.34	60@25	1.65	$GF(p_{192})$	10.8@25	4.73	SPA and DPA attacks	IEEE 802.15.4 IEEE 802.15.6
					$GF(2^{192})$	9.2@25	4.02		
JSSC'11 [104] (Measurement)	45-nm	0.052	53,000@2,100	2.36	-	-	-	SPA attacks	-
ESSCIRC'11 [105] (Measurement)	90-nm	0.104	2,970@255	2.39	-	-	-	SPA and DPA attacks	-
JSSC'10 [106] (Measurement)	130-nm	0.44	1,280@100	34.64	-	-	-	SPA and DPA attacks	-
TVLSI'10 [107] (Measurement)	130-nm	0.02	4.3@12	23.02	-	-	-	SPA attacks	-
JSSC'06 [76] (Measurement)	180-nm	2.45	990@85.5	202	-	-	-	SPA and DPA attacks	-
TVLSI'11 [28] (Measurement)	130-nm	1.35	-	-	$GF(p_{160})$	0.39@141	31.0	-	-
					$GF(2^{160})$	0.27@158	21.6		
ISCAS'11 [58] (Post-layout)	90-nm	0.29	-	-	$GF(p_{160})$	0.31@256	6.98	-	-
					$GF(2^{160})$	0.19@290	4.92		
TCAS-II'09 [24] (Measurement)	130-nm	1.44	-	-	$GF(p_{160})$	0.61@121	42.6	-	-
					$GF(2^{160})$	0.37@146	30.5		
MWSCAS'09 [25] (Post-layout)	180-nm	2.10	-	-	-	-	-	-	-
					$GF(2^{163})$	1.89@181	257		
TC'08 [22] (Synthesis)	130-nm	-	-	-	-	-	-	SPA attacks	-
					$GF(2^{163})$	244@0.001	8.94		
CRASH'05 [57] (Post-layout)	90-nm	0.09	-	-	$GF(p_{192})$	1.13@600	37.29	-	-
					$GF(2^{191})$	0.71@600	23.46		
Our CE-II (Synthesis)	Spartan 6 xc6slx75-3	20,166/4,399† (LUTs/REGs)	133.295@56.234	-	$GF(p_{224})$/ $GF(2^{224})$	336/286K cycles	-	SPA and DPA attacks	IEEE 802.15.4 IEEE 802.15.6
SCVT'11 [50] (Synthesis)	Spartan 6 xc6slx75-3	23,079/12,679‡ (LUTs/REGs)	452.85@233.503	-	RSA_{2048}	6,291K* cycles	-	-	IEEE 802.15.4

Energy = average power × time.

† AES core + DF-ECC processor.

‡ AES core + RSA processor.

∗ Estimated by $m \times (m + 0.5 \times m)$ cycles [108], where m is the key size.

Chapter 8

Conclusion

8.1 Summary

In this dissertation, our research works about the design and implementation for PKC have been reported. We reviewed the state-of-the-art approaches of the ECC hardware implementation and SCA countermeasure. Several design techniques have been presented for hardware performance improvement, and some arithmetical methods of ECC have been used to avoid key-dependent processed data against SCAs. Unfortunately, there is relatively little total solution for the SCA-resistant ECC processor. Also, although the ECC has been adopted in some existing standards, there is a lack of design methods for distinct realistic applications. According to these considerations, our design objectives are not only the hardware efficiency against SCAs but also the standard compliance. In our work, we adopt a new top-to-down design approach including the basic modular operations over DFs, operation scheduling for both the ECSM and ECGP, and on-chip implementation of the random bitstream generation.

The randomized Montgomery operations with low overhead of hardware complexity are proposed for DPA resistance, where the iteration reduction of randomized Montgomery division is also achieved to improve the execution time as compared to Kaliski's Montgomery inversion. The domain conversion can be immediately performed by several operations of Montgomery multiplication and division, and it is time costless for the computation of ECSM. To prevent the attackers from observing some key-dependent specific processed data such as zero value, the masked base point technique of ZPA resistance is

exploited. By reusing the PEs, a parallel computation of ECPG is efficiently implemented to save time overhead. Besides, to avoid the potential threats in operation scheduling, the RL-DAA ECSM with SPA and CPA resistance is used, where a modification for that is to explore the parallelism. The execution time can be further improved by the priority-oriented scheduling method. Note that our SCA countermeasure does not need to modify the standard cell and ASIC/FPGA design flow, and it can be applied for the standard ECC function over DFs.

As above, the randomized computation requires the random sequence. The evidence is that the low randomness of bitstream results in weakness of DPA resistance. To defeat this problem, we introduce a new design of RO-RNG, which generates the random sequence without deterministic state. A jitter amplifier is exploited to enlarge the sample space, leading to higher capability against bias sampling. After applying this technique, ten one-million-bit sequences pass the 15 patterns in NIST random tests with 99% confidence. From measurement results, the DPA attacks cannot reveal the key value in ECC chip using these random sequences even with 12 million power traces.

For hardware architecture, the integrated and fully-pipelined PEs (i.e., JS-GFAU, GFAU, and MAS) are used to save cost area from multiple ALUs. To improve the utilization further, the heterogeneous two-PE architecture is exploited for the parallel computation of ECSM and ECPG without duplicating PEs. Besides, the two-level memory hierarchy with local memory coherence is applied to reduce the data transition, gaining benefits in the power dissipation as compared with conventional shift-register based approaches.

By using a UMC 90-nm CMOS technology, several SCA-resistant ECC chips with different specifications and design criteria were fabricated for various applications, including the mobile device, computing server, and IoT. A 0.41 mm^2 160-bit ECC chip, *so*ECC-P, performs one $GF(p)/GF(2^m)$ ECSM in 0.34/0.29 ms 11.7/9.3 μJ. It is effective at the hardware cost for the mobile device. A 521-bit ECC chip, *so*ECC-S, supporting compressed public-key form can achieve each $GF(p_{521})$ ECSM in 3.40 ms and $GF(2^{521})$ ECSM in 2.77 ms. This is the fastest design for the cloud computing. Furthermore, a 192-bit ECC chip, *so*ECC-G, operating at low supply voltage 0.5 V and cooperating with bio-signal module achieves 10.8/9.2 ms 438/437 μW $GF(p_{192})/GF(2^{192})$ ECSM. This is

targeted at the power efficiency and suitable for the applications of IoT.

8.2 Future Work

For pursuing broader and more flexible range of security requirements in various applications, such as cloud computing and big data system, our future work includes the *ID-based encryption* (IBE) and *fully homomorphism encryption* (FHE) [109]. IBE is a type of public-key encryption in which the public key of a user is some unique information about the identity of the user (e.g. a user's email address). This can use the text-value of the name or domain name as a key or the physical IP address. On the other hand, FHE is a form of encryption which allows specific types of computations to be carried out on ciphertext and obtain an encrypted result which decrypted matches the result of operations performed on the plaintext, where "fully" means that the operations of addition and multiplication are supported. We are constructing a system level and demonstrable prototype with our ECC processor. A variety of security applications can therefore be evaluated on this prototype considering the tradeoff of speed, cost, energy consumption, and also the SCA resistance. The evaluation results can then be applied to a realistic cryptosystem and be used to justify the feasibility and effectiveness of our security platform.

Appendix A

Summary of Research Status

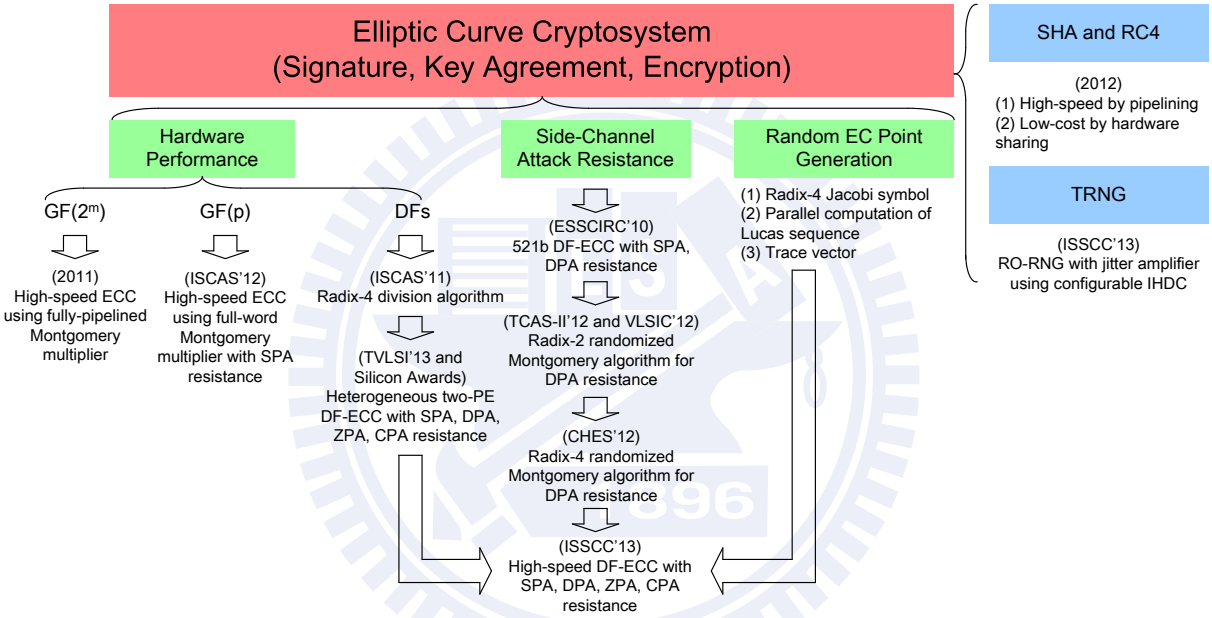


Figure A.1: Overview of our research status.

Appendix B

Montgomery Multiplication

Montgomery multiplication is an efficient approach for the calculation of finite filed multiplication without long-precision division in reduction. In fact, the long-precision integer multiplication of two m-bit operands x and y is still required, while the reduction can be simply achieved by bitwise shifting. The key concept is that the modular reduction by N is to be the division by constant $r = 2^m > N$, where $\gcd(r, N) = 1$; as the last m-bit of intermediate value is zero value, the division by r can be implemented by bitwise shifting. For $0 \leq T < Nr$, the *Montgomery reduction* of T modulo N is defined as the value $Tr^{-1} \pmod N$, and the constant r is so called Montgomery constant. To conduct the Montgomery algorithm, the primary input operands require the domain conversion such that $X \equiv x \cdot r \pmod N$ and $Y \equiv y \cdot r \pmod N$, and then the X and Y are in the *Montgomery domain*. Algorithm 11 shows the Montgomery multiplication, where $r \cdot r^{-1} - N \cdot N' = 1$ with $0 < r^{-1} \equiv 2^{-m} \pmod N < N$ and $0 < N' < r$.

To understand why Algorithm 11 gives the right answer, let's consider the following: $UN \pmod r \equiv TNN' \pmod r \equiv T(rr^{-1} - 1) \pmod r \equiv -T \pmod r$. Thus, $T + UN \pmod r \equiv 0 \pmod r$, and then $(T + UN)$ is exactly divisible by r which also means that the last m-bit of t is zero value. Then the division of t by r can be easily implemented by bitwise shifting. Furthermore, since $0 \leq T < Nr$, $t \leq 2N$ as $U < r$, the return value R is always less than N.

Note that $N' = \frac{r \cdot r^{-1} - 1}{N} \equiv -N^{-1} \pmod r$ and the computation of $-N^{-1} \pmod r$ is shown in Algorithm 12. In the *for* loop of Algorithm 12, $N \cdot q \equiv 1 \pmod{2^i}$. Algorithm 13 shows the $GF(2^m)$ version of Algorithm 12, where $q \equiv N(x)^{-1} \pmod r$ with $r = x^m$. The

Algorithm 11 Montgomery Multiplication

Input: X, Y, N, N', and m

Output: $R \equiv X \cdot Y \cdot r^{-1} \pmod{N}$

1: $T = X \cdot Y$ ($m \times m$-bit Multiplication) ($\log_2 T = 2m$)

2: $U = (T \pmod{r}) \cdot N' \pmod{r}$ (m-bit Truncation + $m \times m$-bit Multiplication + m-bit Truncation) ($\log_2 U = m$)

3: $t = \frac{T+U \cdot N}{r}$ ($m \times m$-bit Multiplication + $2m$-bit Addition + m-bit Bitwise Shift Right) ($\log_2 t = m + 1$)

4: **If** $t \geq N$ **then** $R = t - N$ (m-bit Subtraction)

5: **Return** R

operation of $N(x) \cdot q$ in Step 3 is implemented by the polynomial multiplication, while the operation of $2^{i-1} < N(x) \cdot q \pmod{2^i}$ in Step 3 is implemented by the integer comparison.

Algorithm 12 Computation of $-N^{-1} \pmod{r}$ for $r = 2^m$

Input: N with $0 < N < r$ and m

Output: $q \equiv -N^{-1} \pmod{r}$

1: $q = 1$

2: **For** i from 2 to m **do**

3: **If** $(2^{i-1} < N \cdot q \pmod{2^i})$

4: **then** $q = q + 2^{i-1}$

5: **Return** $q = 2^m - q$

Algorithm 13 Computation of $N^{-1} \pmod{r}$ for $r = x^m$

Input: $N(x) = x^m + a(x)$ and m

Output: $q \equiv N(x)^{-1} \pmod{r}$

1: $q = 1$

2: **For** i from 2 to m **do**

3: **If** $(2^{i-1} < N(x) \cdot q \pmod{2^i})$

4: **then** $q = q \oplus x^{i-1}$

5: **Return** q

Appendix C

Barrett Reduction

Barrett reduction shown in Algorithm 14 finds $T \pmod N$ with a given $(k+1)$-bit value $\mu = \lfloor \frac{b^{2k}}{N} \rfloor$ and a suitably-chosen base $b = 2^L$, where μ can be obtained from pre-computation. For example, as $b = 2$, $\mu = \lfloor \frac{2^{2k}}{N} \rfloor$; $q_1 = T >> (k-1)$; $q_2 = q_1 \cdot \mu$; $q_3 = q_2 >> (k+1)$; $R_1 = T[k:0]$; $R_2 = q_3 \cdot N$; $R_2 = R_2[k:0]$; $R = R_1 - R_2$; If $(R < 0)$ then $R = R + 2^{k+1}$; While $(R \geq N)$ $R = R - N$; Return R. To compute the value of μ required in Algorithm 14, it is shown in Algorithm 15.

Algorithm 14 Barrett Reduction

Input: $0 \leq T = (T_{2k-1}, T_{2k-2}, T_1, T_0)_b < b^{2k}$, N, $b \geq 3$, $k = \lfloor \log_b N \rfloor + 1$, and μ

Output: $R \equiv (R_{k-1}, \ldots, R_0)_b = T \pmod N$

1: $q_1 = \lfloor \frac{T}{b^{k-1}} \rfloor$ (Bitwise Shift Right)
2: $q_2 = q_1 \cdot \mu$ (Multiplication)
3: $q_3 = \lfloor \frac{q_2}{b^{k+1}} \rfloor$ (Bitwise Shift Right)
4: $R_1 = T \pmod{b^{k+1}}$ (Truncation)
5: $R_2 = q_3 \cdot N \pmod{b^{k+1}}$ (Multiplication + Truncation)
6: $R = R_1 - R_2$ (Subtraction)
7: **If** $(R < 0)$ **then** $R = R + b^{k+1}$ (Addition)
8: **While** $(R \geq N)$ **do** $R = R - N$ (Subtraction)
9: **Return** R

Both of Barrett reduction and Montgomery reduction are the approach to find the field element in $GF(p)$ for an integer of double bit length without trial division. They also require a pre-computed constant. For the differences, Barrett reduction does not

Algorithm 15 Computation of μ

Input: p, and $k = \lfloor \log_b N \rfloor + 1$

Output: $\mu = \lfloor \frac{b^{2k}}{N} \rfloor$

1: $\mu = b^k$
2: **Repeat**
3: $\quad S = \mu$
4: $\quad \mu = 2\mu - \lfloor \frac{\lfloor \frac{\mu^2}{b^k} \rfloor \cdot N}{b^k} \rfloor$
5: **Until** $\mu \leq S$
6: $t = b^{2k} - N \cdot \mu$
7: **While** $(t < 0)$ **do**
8: $\quad \mu = \mu - 1$
9: $\quad t = t + N$
10: **Return** R

need the domain conversion of primary inputs. In contrast, Montgomery reduction needs the domain conversion of primary inputs.

Bibliography

[1] R. Rivest, *Rivest cipher 4 (RC4)*, Std., 1987.

[2] Federal Information Processing Standard (FIPS), *Data encryption standard (DES)*, FIPS Std. 46-3, Oct. 1999. [Online]. Available: http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf

[3] ——, *Advanced encryption standard (AES)*, FIPS Std. 197, Nov. 2001. [Online]. Available: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

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

[5] V. Miller, "Use of elliptic curve in cryptography," in *Proc. Advances in Cryptology (Crypto)*, 1986, pp. 417–426.

[6] N. Koblitz, "Elliptic Curve Cryptosystems," *Math. Computing*, vol. 48, pp. 203–209, 1987.

[7] Institute of Electrical and Electronics Engineers (IEEE), *Standard specifications or public-key cryptography*, IEEE Std. 1363, Jan. 2000. [Online]. Available: http://grouper.ieee.org/groups/1363/

[8] ——, *Standard specifications or public-key cryptography – Amendment 1: Additional techniques*, IEEE Std. 1363a, Sep. 2004. [Online]. Available: http://grouper.ieee.org/groups/1363/P1363a/

[9] ——, *Wireless medium access control and physical layer specifications for low-rate wireless personal area networks*, IEEE Std. 802.15.4, May 2000.

[10] ——, *IEEE standard for local and metropolitan area networks - Part 15.6: Wireless body area networks*, IEEE Std. 802.15.6, Feb. 2012.

[11] Z. Alliance, *ZigBee specifications*, Std., 2006. [Online]. Available: http://www.zigbee.org

[12] B. SIG, *Bluetooth specification version 4.0 [vol 0]*, Std., Jun. 2010. [Online]. Available: http://www.bluetooth.org

[13] J.-Y. Yu, C.-C. Chung, W.-C. Liao, and C.-Y. Lee, "A sub-mW multi-tone CDMA baseband transceiver chipset for wireless body area network applications," in *ISSCC Dig. Tech. Papers*, Feb. 2007.

[14] S. Mangard, E. Oswald, and T. Popp, *Power analysis attacks - Revealing the secretes of smart cards.* Springer, 2007.

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

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

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

[18] K. Sakiyama, L. Batina, B. Preneel, and I. Verbauwhede, "Superscalar coprocessor for high-speed curve-based cryptography," in *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, vol. 4249, Oct. 2006, pp. 415–429.

[19] 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*, vol. 54, no. 5, pp. 412–416, May 2007.

[20] K. Sakiyama, L. Batina, B. Preneel, and I. Verbauwhede, "Multicore curve-based cryptoprocessor with reconfigurable modular arithmetic logic units over $GF(2^n)$," *IEEE Trans. Comput.*, vol. 56, no. 9, pp. 1269–1282, Sep. 2007.

[21] T. Güneysu and C. Paar, "Ultra high performance ECC over NIST primes on commercial FPGAs," in *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, vol. 5154, Aug. 2008, pp. 62–78.

[22] Y. K. Lee, K. Sakiyama, L. Batina, and I. Verbauwhede, "Elliptic-curve-based security processor for RFID," *IEEE Trans. Comput.*, vol. 57, no. 11, pp. 1514–1527, Nov. 2008.

[23] D. M. Schinianakis, A. P. Fournaris, H. E. Michail, A. P. Kakarountas, and T. Stouraitis, "An RNS implementation of an F_p elliptic curve point multiplier," *IEEE Trans. Circuits Syst. I*, vol. 56, no. 6, pp. 1202–1213, Jun. 2009.

[24] J.-Y. Lai and C.-T. Huang, "A highly efficient cipher processor for dual-field elliptic curve cryptography," *IEEE Trans. Circuits Syst. II*, vol. 56, no. 5, pp. 394–398, May 2009.

[25] J.-H. Hong and W.-C. Wu, "The design of high performance elliptic curve cryptographic," in *IEEE Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Aug. 2009, pp. 527–530.

[26] N. Guillermin, "A high speed coprocessor for elliptic curve scalar multiplications over F_p," in *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, vol. 6225, Aug. 2010, pp. 48–64.

[27] J.-H. Chen, M.-D. Shieh, and W.-C. Lin, "A high-performance unified-field reconfigurable cryptographic processor," *IEEE Trans. VLSI Syst.*, vol. 18, no. 8, pp. 1145 –1158, Aug. 2010.

[28] J.-Y. Lai and C.-T. Huang, "Energy-adaptive dual-field processor for high-performance elliptic curve cryptographic applications," *IEEE Trans. VLSI Syst.*, vol. 19, no. 8, pp. 1512–1517, Aug. 2011.

[29] F. Fürbass and J. Wolkerstorfer, "ECC processor with low die size for RFID applications," in *IEEE Int. Symp. on Circuits Syst. (ISCAS)*, May 2007, pp. 1835–1838.

[30] A. Karatsuba and Y. Ofman, "Multiplication of many-digital numbers by automatic computers," *Soviet Physics-Doklady (English Translation)*, pp. 595–596, 1962.

[31] S. M. Shohdy, A. B. El-Sisi, and N. Ismail, "Hardware implementation of efficient modified Karatsuba multiplier used in elliptic eurves," *International Journal of Network Security*, vol. 11, no. 3, pp. 155–162, Nov. 2010.

[32] G. Zhou, H. Michalik, and L. Hinsenkamp, "Improving throughput of AES-GCM with pipelined karatsuba multipliers on FPGAs," in *Springer*, vol. 5453, 2009, pp. 193–203.

[33] L. Henzen and W. Fichtner, "FPGA parallel-pipelined AES-GCM core for 100G Ethernet applications," in *European Solid-State Circuits Conference (ESSCIRC)*, Sep. 2010, pp. 202–205.

[34] D. Hankerson, A. Menezes, and S. Vanstone, *Guide to elliptic curve cryptography*. New York: Springer, 2004.

[35] H. Nozaki, M. Motoyama, A. Shimbo, and S. Kawamura, "Implementation of RSA algorithm based on RNS Montgomery multiplication," in *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, vol. 2162, May 2001, pp. 364–376.

[36] J.-C. Bajard and L. Imbert, "A full RNS implementation of RSA," *IEEE Trans. Comput.*, vol. 53, no. 6, pp. 769–774, Jun. 2004.

[37] P. L. Montgomery, "Modular multiplication without trial division," *Math. Computing*, vol. 44, no. 170, pp. 519–521, Apr. 1985.

[38] 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)*, Sep. 2010, pp. 206–209.

[39] A. F. Tenca and Çetin K. Koç, "A scalable architecture for Montgomery multiplication," in *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, vol. 1717, 1999, pp. 94–108.

[40] W.-C. Lin, J.-H. Ye, and M.-D. Shieh, "Scalable Montgomery modular multiplication architecture with low latency and low memory bandwidth requirement," *IEEE Trans. Comput.*

[41] A. Miyamoto, N. Homma, T. Aoki, and A. Satoh, "Systematic design of RSA processors based on high-radix Montgomery multipliers," *IEEE Trans. VLSI Syst.*, vol. 19, no. 7, pp. 1136–1146, Jun. 2011.

[42] S.-C. Chung, J.-W. Lee, H.-C. Chang, and C.-Y. Lee, "A high-performance elliptic curve cryptographic processor over $GF(p)$ with SPA resistance," in *IEEE Int. Symp. on Circuits Syst. (ISCAS)*, May 2012, pp. 1456–1459.

[43] S. M. H. Rodríguez and F. Rodríguez-Henríquez, "An FPGA arithmetic logic unit for computing scalar multiplication using the half-and-add method," in *International Conference on Reconfigurable Computing and FPGAs (ReConFig)*, Sep. 2005.

[44] ISO/IEC, *Information technology-radio frequency identification (RFID) for item management-part 3: parameters for air interface communications at 13.56 MHz*, ISO/IEC Std. 18 000-3, 2004.

[45] J. Wolkerstorfer, "Is elliptic-curve cryptography suitable to secure RFID tags?" in *Proc. Workshop RFID and Light-Weight Cryptography (RFIDSec)*, Aug. 2005.

[46] S. S. Kumar and C. Paar, "Are standards compliant elliptic curve cryptosystems feasible on RFID?" in *Proc. Workshop on RFID Security (RFIDSec)*, Jul. 2006.

[47] F. I. P. S. (FIPS), *Secure Hash Standard (SHS)*, FIPS Std. 180-3, Oct. 2008.

[48] J. López and R. Dahab, "Fast multiplication on elliptic curves over $GF(2^m)$ without precomputation," in *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, vol. 1717, 1999, pp. 316–327.

[49] A. W. M. Koschuch, J. Lechner and J. G. schädl, "Hardware/software co-design of elliptic curve cryptography on an 8051 microcontroller," in *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, vol. 4249, Oct. 2006, pp. 430–444.

[50] A. de la Piedra, A. Touhafi, and G. Cornetta, "Cryptographic accelerator for 802.15.4 transceivers with key agreement engine based on Montgomery arithmetic," in *IEEE Symposium on Communications and Vehicular Technology in the Benelux (SCVT)*, Nov. 2011, pp. 1–5.

[51] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *International Cryptology Conference on Advances in Cryptology.*, 1999, pp. 388–397.

[52] T. Popp and S. Mangard, "Masked dual-rail pre-charge logic: DPA resistance without routing constraints," in *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, vol. 3659, Aug. 2005, pp. 172–186.

[53] K. Tiri, M. Akmal, and I. Verbauwhede, "A dynamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on smart cards," in *European Solid-State Circuits Conference (ESSCIRC)*, Sep. 2002, pp. 403–406.

[54] M. Bucci, L. Giancane, R. Luzzi, and A. Trifiletti, "Three-phase dual-rail pre-charge logic," in *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, vol. 4249, Oct. 2006, pp. 232–241.

[55] M. Bucci, L. Giancane, R. Luzzi, G. Scotti, and A. Trifiletti, "Delay-based dual-rail precharge logic," *IEEE Trans. VLSI Syst.*, vol. 19, no. 7, pp. 1147–1153, Jul. 2011.

[56] J. Fan, X. Guo, E. D. Mulder, P. Schaumont, B. Preneel, and I. Verbauwhede, "State-of-the-art of secure ECC implementations: a survey on known side-channel attacks and countermeasures," in *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, 2010, pp. 76–87.

[57] J. Wolkerstorfer, "Scaling ECC hardware to a minimum," in *ECRYPT Workshop - Cryptographic Advances in Secure Hardware (CRASH)*, 2005.

[58] Y.-L. Chen, J.-W. Lee, P.-C. Liu, H.-C. Chang, and C.-Y. Lee, "A dual-field elliptic curve cryptographic processor with a radix-4 unified division unit," in *IEEE Int. Symp. on Circuits Syst. (ISCAS)*, May 2011, pp. 713–716.

[59] J.-W. Lee, J.-H. Hsiao, H.-C. Chang, and C.-Y. Lee, "An efficient DPA countermeasure with randomized Montgomery operations for DF-ECC processor," *IEEE Trans. Circuits Syst. II*, vol. 59, no. 5, pp. 287–291, May 2012.

[60] W. Diffie and M. E. Hellman, "Multiuser cryptographic techniques," in *Proceedings of the AFIPS National Computer Conference*, vol. 45, Jun. 1976, pp. 109–112.

[61] B. Kaliski and M. Robshaw, "The secure use of RSA," *CryptoBytes*, 1995.

[62] L. C. Washington, *Elliptic curves: number theory and cryptography.* Chapman and Hall/CRC, 2003.

[63] H. Cohen, G. Frey, R. Avanzi, C. Doche, T. Lange, K. Nguyen, and F. Vercauteren, *Handbook of elliptic and hyperelliptic curve cryptography.* Chapman and Hall/CRC, 2005.

[64] H. Cohen, A. Miyaji, and T. Ono, "Efficient elliptic curve exponentiation using mixed coordinates," in *Proc. Adv. Cryptolog. (Asiacrypt)*, vol. 1514, 1998, pp. 51–65.

[65] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Trans. Info. Theory*, vol. IT-22, pp. 644–654, Nov. 1976.

[66] L. Law, A. Menezes, M. Qu, J. Solinas, and S. Vanstone, "An efficient protocol for authenticated key agreement," CORR 98-05, Dept. of C & O, University of Waterloo, Canada, Tech. Rep., Mar. 1998. [Online]. Available: http://www.cacr.math.uwaterloo.ca/

[67] K. Nyberg and R. Rueppel, "A new signature scheme based on the DSA giving message recovery," in *Proceedings of First ACM Conference on Computer and Communications Security, ACM Press*, 1993, pp. 58–61.

[68] D. W. Kravitz, "Digital signature algorithm," Patent 5,231,668, Jul., 1993.

[69] American National Standards Institute (ANSI), *Public Key Cryptography for the Financial Services Industry: Key Agreement and Transport Using Elliptic Curve Cryptography*, ANSI Std. X9.63-2002, 2002.

[70] National Institute of Standards and Technology (NIST), *Recommendation for block cipher modes of operation - methods and techniques*, NIST Std. 800-38A, Dec. 2001. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf

[71] ——, *Recommendation for block cipher modes of operation - The CMAC mode for authentication*, NIST Std. 800-38B, May 2005. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf

[72] A. Kerckhoffs, "La cryptographie militaire," *Journal des sciences militaires*, vol. 9, pp. 5–38, Jan. 1883.

[73] C. Tokunaga and D. Blaauw, "Secure AES engine with a local switched-capacitor current equalizer," in *ISSCC Dig. Tech. Papers*, Feb. 2009, pp. 64–65.

[74] J. Fan, B. Gierlichs, and F. Vercauteren, "To infinity and beyond: combined attack on ECC using points of low order," in *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, vol. 6917, Sep. 2011, pp. 143–159.

[75] N. Homma, A. Miyamoto, T. Aoki, A. Satoh, and A. Shamir, "Collision-based power analysis of modular exponentiation using chosen-message pairs," in *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, vol. 5154, Aug. 2008, pp. 15–29.

[76] D. Hwang, K. Tiri, A. Hodjat, B.-C. Lai, S. Yang, P. Schaumont, and I. Verbauwhede, "AES-based security coprocessor IC in 0.18-μm CMOS with resistance to differential power analysis side-channel attacks," *IEEE J. Solid-State Circuits*, vol. 41, no. 4, pp. 781–792, Apr. 2006.

[77] C. Tokunaga and D. Blaauw, "Securing encryption systems with a switched capacitor current equalizer," *IEEE J. Solid-State Circuits*, vol. 45, no. 1, pp. 23–31, Jan. 2010.

[78] P.-C. Liu, H.-C. Chang, and C.-Y. Lee, "A low overhead DPA countermeasure circuit based on ring oscillators," *IEEE Trans. Circuits Syst. II*, vol. 57, no. 7, pp. 546–550, Jul. 2010.

[79] J. S. Coron, "Resistance against differential power analysis for elliptic curve cryptosystems," in *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, vol. 1717, Aug. 1999, pp. 725–725.

[80] M. Joye and C. Tymen, "Protections against differential analysis for elliptic curve cryptography – an algebraic approach," in *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, vol. 2162, May 2001, pp. 377–390.

[81] L. Goubin, "A refined power-analysis attack on elliptic curve cryptosystems," in *Workshop on Theory and Practice in Public Key Cryptography*, vol. 2567, 2003.

[82] B. S. Kaliski, "The Montgomery inverse and its applications," *IEEE Trans. Comput.*, vol. 44, no. 8, pp. 1064–1065, Aug. 1995.

[83] H. Wang and H. Zhang, "A fast pseudorandom number generator with BLAKE hash function," *Wuhan University Journal of Natural Sciences*, vol. 15, no. 5, pp. 393–397, 2010.

[84] P.-C. Liu, H.-C. Chang, and C.-Y. Lee, "A true random-based differential power analysis countermeasure circuit for an AES engine," *IEEE Trans. Circuits Syst. II*, vol. 59, no. 2, pp. 103–107, Feb. 2012.

[85] C. S. Petrie and J. A. Connelly, "A noise-based IC random number generator for applications in cryptography," *IEEE Trans. Circuits Syst. I*, vol. 47, pp. 615–621, May 2000.

[86] J. Golic, "New methods for digital generation and postprocessing of random data," *IEEE Trans. Comput.*, vol. 55, no. 10, pp. 1217–1229, Oct. 2006.

[87] M. Bucci and R. Luzzi, "Fully digital random bit generators for cryptographic applications," *IEEE Trans. Circuits Syst. I*, vol. 55, no. 3, pp. 861–875, Apr. 2008.

[88] National Institute of Standards and Technology (NIST), *A statistical test suite for the validation of random number generators and pseudorandom number generators for cryptographic applications*, NIST Std. 800-22, May 2001. [Online]. Available: http://csrc.nist.gov/groups/ST/toolkit/rng/documents/SP800-22rev1a.pdf

[89] J.-W. Lee, S.-C. Chung, H.-C. Chang, and C.-Y. Lee, "A 3.40ms/$GF(p_{521})$ and 2.77ms/$GF(2^{521})$ DF-ECC processor with side-channel attack resistance," in *ISSCC Dig. Tech. Papers*, Feb. 2013, pp. 50–52.

[90] H. Bock, M. Bucci, and R. Luzzi, "An offset-compensated oscillator-based random bit source for security applications," in *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, vol. 3156, Aug. 2004, pp. 268–281.

[91] C.-Y. Yu, C.-C. Chung, C.-J. Yu, and C.-Y. Lee, "A low-power DCO using interlaced hysteresis delay cells," *IEEE Trans. Circuits Syst. II*, vol. 59, no. 10, pp. 673–677, Oct. 2012.

[92] S. Ghosh, D. Mukhopadhyay, and D. Roychowdhury, "Petrel: power and timing attack resistant elliptic curve scalar multiplier based on programmable $GF(p)$ arithmetic unit," *IEEE Trans. Circuits Syst. I*, vol. 58, no. 8, pp. 1798–1812, Aug. 2011.

[93] A. Daly, W. Marnane, T. Kerins, and E. Popovici, "An FPGA implementation of a $GF(p)$ ALU for encryption processors," *Microprocess. Microsyst.*, vol. 28, pp. 253–260, 2004.

[94] J.-W. Lee, S.-C. Chung, H.-C. Chang, and C.-Y. Lee, "Efficient power-analysis-resistant dual-field elliptic curve cryptographic processor using heterogeneous dual-processing-element architecture," *IEEE Trans. VLSI Syst.*, 2013.

[95] S. Müller, "On the computation of square roots in finite fields," *Designs, Codes and Cryptography*, vol. 31, no. 3, pp. 301–312, 2004.

[96] ARM, "AMBA Design Kit," 2007. [Online]. Available: http://infocenter.arm.com/help/topic/com.arm.doc.ddi0243c/DDI0243C_adk_r3p0_trm.pdf

[97] J.-W. Lee, S.-C. Chung, H.-C. Chang, and C.-Y. Lee, "An efficient countermeasure against correlation power-analysis attacks with randomized Montgomery operations for DF-ECC processor," in *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, vol. 7428, Sep. 2012, pp. 548–564.

[98] S.-Y. Hsu, Y.-C. Ho, Y.-W. Tseng, T.-Y. Lin, P.-Y. Chang, J.-W. Lee, J.-H. Hsiao, S.-M. Chuang, T.-Z. Yang, P.-C. Liu, T.-F. Yang, R.-J. Chen, C.-C. Su, and C.-Y. Lee, "A sub-100μW multi-functional cardiac signal processor for mobile healthcare applications," in *Symposium on VLSI Circuits (VLSIC)*, Jun. 2012, pp. 156–157.

[99] Andes, Andes, Tech. Rep. [Online]. Available: http://www.andestech.com/p2-3.htm

[100] T.-W. Chen, J.-Y. Yu, C.-Y. Yu, and C.-Y. Lee, "A 0.5V 4.85 Mbps dual-mode baseband transceiver with extended frequency calibration for biotelemetry applications," *IEEE J. Solid-State Circuits*, vol. 44, no. 11, pp. 2966–2976, Nov. 2009.

[101] H.-Y. Hsu, A.-Y. Wu, and J.-C. Yeo, "Area-efficient VLSI design of Reed-Solomon decoder for 10GBase-LX4 optical communication systems," *IEEE Trans. Circuits Syst. II*, vol. 43, no. 4, pp. 1019–1027, Nov. 2006.

[102] C.-C. Wong and H.-C. Chang, "High-efficiency processing schedule for parallel turbo decoders using QPP interleaver," *IEEE Trans. Circuits Syst. I*, vol. 58, no. 6, pp. 1412–1420, Jun. 2011.

[103] J. Park, J.-T. Hwang, and Y.-C. Kim, "FPGA and ASIC implementation of ECC processor for security on medical embedded system," in *Proc. IEEE Int. Conf. Inf. Technol. Appl.*, vol. 2, 2005, pp. 547–551.

[104] S. K. Mathew, F. Sheikh, M. Kounavis, S. Gueron, A. Agarwal, S. K. Hsu, H. Kaul, M. A. Anders, and R. K. Krishnamurthy, "53 Gbps native $GF(2^4)^2$ composite-field AES-encrypt/decrypt accelerator for content-protection in 45 nm high-performance microprocessors," *IEEE J. Solid-State Circuits*, vol. 46, no. 4, pp. 767–776, Apr. 2011.

[105] P.-C. Liu, J.-H. Hsiao, H.-C. Chang, and C.-Y. Lee, "A 2.97 Gb/s DPA-resistant AES engine with self-Generated random sequence," in *European Solid-State Circuits Conference (ESSCIRC)*, Sep. 2011, pp. 71–74.

[106] C. Tokunaga and D. Blaauw, "Securing encryption systems with a switched capacitor current equalizer," vol. 45, no. 1, pp. 23–31, Jan. 2010.

[107] T. Good and M. Benaissa, "692-nW advanced encryption standard (AES) on a 0.13-μm CMOS," *IEEE Trans. VLSI Syst.*, vol. 18, no. 12, pp. 1753–1757, Dec. 2010.

[108] F. Rodriguez-Henriquez, N. A. Saqib, A. Diaz-Pérez, and Çetin Kaya Koç, "Cryptographic algorithms on reconfigurable hardware," in *Springer Series on Signals and Communication Technology*, 2006.

[109] C. Gentry, "A fully homomorphic encryption scheme," Ph.D. dissertation, Stanford University, 2009. [Online]. Available: http://crypto.stanford.edu/craig/