KATHOLIEKE UNIVERSITEIT LEUVEN
FACULTEIT INGENIEURSWETENSCHAPPEN
DEPARTEMENT ELEKTROTECHNIEK–ESAT
Kasteelpark Arenberg 10, 3001 Leuven-Heverlee

Arithmetic and Architectures for Secure Hardware Implementations of Public-Key Cryptography

Promotors:
Prof. Dr. ir. Bart Preneel
Prof. Dr. ir. Ingrid Verbauwhede

Proefschrift voorgedragen tot
het behalen van het doctoraat
in de ingenieurswetenschappen

door

Lejla BATINA

December 2005

KATHOLIEKE UNIVERSITEIT LEUVEN
FACULTEIT INGENIEURSWETENSCHAPPEN
DEPARTEMENT ELEKTROTECHNIEK–ESAT
Kasteelpark Arenberg 10, 3001 Leuven-Heverlee

Arithmetic and Architectures for Secure Hardware Implementations of Public-Key Cryptography

Jury:
Prof. Dr. Yves Willems, voorzitter
Prof. Dr. ir. Bart Preneel, promotor
Prof. Dr. ir. Ingrid Verbauwhede, promotor
Prof. Dr. ir. Francky Catthoor
Prof. Dr. ir. Georges Gielen
Prof. Dr. ir. David Naccache, ENS, Paris
Prof. Dr. ir. Henk C.A. van Tilborg, T.U. Eindhoven
Dr. Pim Tuyls, Philips Research, Eindhoven, The Netherlands
Prof. Dr. ir. Joos Vandewalle

Proefschrift voorgedragen tot het behalen van het doctoraat in de ingenieurswetenschappen

door

Lejla BATINA

U.D.C. 681.3*D46 December 2005

D/2005/7515/98

ISBN 90-5682-670-0

Acknowledgments

First of all I would like to thank my promotors Bart Preneel and Ingrid Verbauwhede for their constant guidance and support that resulted in this thesis. Bart is especially thanked for giving me the chance to pursue a Ph.D. at COSIC and for his help and advice around the clock. Ingrid is thanked for giving me directions and freedom. They both provided a number of helpful comments on this text and I thank them for that as well.

I would also like to thank Prof. Francky Catthoor, Prof. Georges Gielen and Prof. Joos Vandewalle for careful reading of the manuscript, many valuable suggestions and comments.

I thank Prof. David Naccache, Prof. Henk van Tilborg and Dr. Pim Tuyls for accepting to be members of the jury, and Prof. Yves Willems for chairing it.

I am very grateful to Henk van Tilborg, who was my advisor at T.U. Eindhoven for always being friendly and supportive. I would also like to thank Berry Schoenmakers for his encouragement and help with my MTD thesis.

Furthermore, I would like to express my gratitude to Dr. Cees Jansen, who made it possible for me to start with my Ph.D. studies while I was still working at Pijnenburg, Securealink. I would like to thank Geeke Bruin-Muurling for being a wonderful colleague back there and my first co-author. We put together our first RSA-conference paper and we had a lot of fun working together.

Many thanks to Nele Mentens for being a great colleague, travel mate, co-author and my best friend at COSIC. Of my other COSIC colleagues, special thanks go to Berna Örs for her friendship and kindness and many, many other things in which she helped me ever since I joined COSIC. Kazuo Sakiyama and Elke De Mulder are a great addition to our hardware group, which we hope to see growing. I very much appreciated a pleasant and fruitful collaboration with Alireza Hodjat and David Hwang from UCLA.

I thank Fré Vercauteren for his insightful comments and criticism and for reading a large part of the text. I add also Jasper Scholten who was always willing to help me with all kinds of questions, from mathematical to technical. A big thanks for Péla Noë, Elvira Wouters and Marleen Somers for taking care of all those administrative things and always having interest and compassion for all other things

as well. Special thanks go to Karel Wouters for making my long rides to work and back much shorter. I thank all COSIC fellows for their friendliness and help, in particular Claudia Diaz, Svetla Nikova, Stefaan Seys and Christopher Wolf.

Chapter 6 in this thesis is joint work with Dr. Pim Tuyls and our cooperation resulted in more broadness in my work, which was truly rewarding. Gregory Neven is also acknowledged for his generous help and advice relating to this part of my work.

I thank my long-time friends for putting up with me through all these years and for having interest in all I do and cope with. Finally, I would like to thank my family for their unconditional support and love, especially to my mum and to my son Davor I thank for everything, everything, everything.

The last fifteen months of my doctoral studies were funded by a research grant of the Katholieke Universiteit Leuven, Belgium.

Lejla Batina
December 2005

Abstract

This thesis studies implementations of cryptographic algorithms and protocols for embedded systems, which contribute towards the development of the future secure pervasive computing environment; this context puts tight constraints on performance, memory, power, area and bandwidth. Effective information protection against eavesdropping and modifications in open systems can only be achieved using Public Key Cryptography (PKC). Implementing PKC requires detailed insights in the algorithms and the underlying arithmetic for all application platforms. While software platforms typically provide low-cost and flexible solutions, only hardware implementations offer a suitable level of tamper resistance and protection against side-channel attacks. In these attacks, information that is based on physical characteristics of the implementation (such as timing, power consumption or electromagnetic radiation) is exploited to deduce secret information (*e.g.* keys). This thesis investigates algorithms that are suitable for efficient and secure hardware implementations of PKC. The emphasis is on constrained environments and increased resistance to side-channel attacks.

In the first part of our research, we focus on a hardware architecture for the modular multiplication operation, which is efficient for bit-lengths suitable for both commonly used types of PKC *i.e.* RSA and Elliptic Curve Cryptography (ECC). We reconsider Montgomery's Modular Multiplication (MMM) algorithm and we make some adjustments for both RSA and ECC implementations on an FPGA platform. As a follow-up we develop a one common architecture for RSA and ECC. Fields of characteristic two *i.e.* binary fields for ECC are also addressed. A complete ECC implementation is developed for which the point operations are designed to preclude simple side-channel attacks. However, it should be pointed out that the hardware implementations presented in the thesis serve mainly as "proof of concept" for the proposed algorithms, hence further architectural optimizations are certainly possible.

Furthermore, we investigate implementations of Hyperelliptic Curves Cryptography (HECC). HECC was proposed in 1988 as a generalization of ECC. We describe the first HECC implementation using a hardware/software co-design on the 8051 micro-processor, which uses a small hardware co-processor to optimize

the performance.

RFID-tags are becoming very popular tools for the identification of products. As they have a small microchip on board, they offer functionality that can be used for security purposes. In particular, this functionality makes it possible to verify the authenticity of a product and hence to detect and prevent counterfeiting. In order to be suitable for these security purposes RFID-tags have to be resistant against many attacks, including cloning of the tag. It is shown that our solution for authentication of RFID-tags prevents counterfeiting and is feasible for both on-line and off-line scenarios.

We also propose an information theoretic model for side-channel leakage. In particular, we investigate the situation in which the Hamming weight of the secret data is disclosed. Our conclusion is that this particular leakage does not have a substantial impact on the security of a cryptosystem.

Among many proposed PKC alternatives, one of the most recent ones is torus based cryptography. In this thesis we also investigate efficient implementations of the CEILIDH cryptosystem. A concept of a hardware architecture and the first estimates for performance are presented. Furthermore, we compare this implementation with implementations of ECC and RSA.

Samenvatting

Dit proefschrift beschrijft implementaties van cryptografische algoritmen en protocollen voor ingebedde systemen. Het draagt bij tot de ontwikkeling van toepassingen voor beveiligde gegevensverwerking. Ingebedde systemen leggen strenge eisen op aan de performantie, het geheugengebruik, het vermogenverbruik, de oppervlakte en de bandbreedte.

Effectieve bescherming van gevoelige informatie tegen luistervinken en wijzigingen van open systemen kan enkel bereikt worden met behulp van publieke sleutel cryptografie (PKC). Het implementeren van PKC op diverse platformen vergt een grondig inzicht in de algoritmen en de onderliggende wiskunde. Software implementaties bieden meestal lage-kost en flexibele oplossingen, maar enkel hardware platformen kunnen voldoende beschermen tegen tampering en aanvallen via nevenkanalen. In dit soort aanvallen worden de fysische eigenschappen van de implementatie (zoals de tijdsduur van bewerkingen, het vermogenverbruik of de elektromagnetische straling) uitgebuit om zo geheime informatie te achterhalen (*bv.* sleutels). In dit project wordt er gezocht naar passende algoritmen voor efficiënte en veilige hardware implementaties van PKC. De nadruk ligt op toepassingen met beperkte implementatie-mogelijkheden en verhoogde weerstand tegen nevenkanaal aanvallen.

In het eerste deel van ons onderzoek, richten we ons op een hardware architectuur voor modulaire vermenigvuldiging, die efficiënt is voor bit-lengtes die gebruikt worden door de twee meest gebruikte types van PKC *d.i.* RSA en elliptische kromme cryptografie (ECC). We bekijken Montgomery's algoritme voor modulaire vermenigvuldiging (MMM) en maken enkele aanpassingen voor zowel RSA als ECC implementaties op FPGA platformen. Vervolgens ontwikkelen we een enige architectuur die kan gebruikt worden voor zowel RSA als ECC. Velden van karakteristiek twee *d.i.* binaire velden voor ECC worden eveneens behandeld. Een volledige ECC implementatie werd ontwikkeld waarbij de punt bewerkingen ontworpen zijn om te weerstaan aan enkelvoudige neven-kanaal aanvallen. Hierbij wensen we te benadrukken dat de hardware implementaties die in dit project worden voorgesteld vooral een concept willen bewijzen, waardoor verdere architecturale verbeteringen zeker nog mogelijk zijn.

Verder worden in dit proefschrift implementaties van hyperelliptische kromme cryptografie (HECC) bestudeerd. HECC werd in 1988 voorgesteld als een veralgemening van ECC. We beschrijven de eerste HECC implementatie met behulp van een hardware/software co-ontwerp op een 8051 micro-processor die een kleine hardware co-processor gebruikt om de performantie te optimaliseren.

Radiofrequentie-identificatie (RFID) wint aan populariteit voor de identificatie van producten. Omdat deze "tags" een kleine microchip aan boord hebben, bieden zij de mogelijkheid om bewerkingen uit te voeren voor beveiligingsdoeleinden. Deze functionaliteit maakt het mogelijk om de authenticiteit van een product te verifiëren en zo vervalsingen tegen te gaan. Om geschikt te kunnen zijn voor deze beveilingsdoeleinden moeten RFID tags bestand zijn tegen diverse aanvallen, waaronder het klonen van de tag. Zoals aangetoond, voorkomt onze oplossing voor authenticatie vervalsingen en kan ze aangewend worden in zowel on-line als off-line omstandigheden.

We introduceren eveneens een informatie theoretisch model dat het lekken van neven-kanaal informatie becshrijft. We onderzoeken de situatie waar het Hamming gewicht van de geheime gegevens onthuld wordt. Ons besluit is dat dit soort lekken geen substantiële impact hebben op de veiligheid van een cryptosysteem.

Het meest recente van alle nieuwe PKC alternatieven is torus-gebaseerde cryptografie. In dit project zoeken we daarom naar efficiënte implementaties van het CEILIDH cryptosysteem. Een eerste versie van een hardware architectuur en de eerste schattingen voor de performantie worden beschreven. Verder vergelijken we deze implementatie met ECC en RSA implementaties.

Contents

List of Abbreviations

AES	Advanced Encryption Standard
ALU	Arithmetic Logic Unit
AOP	All One Polynomial
ASC	Addition/Subtraction Circuit
ASIC	Application Specific Integrated Circuit
ASIP	Application Specific Instruction Set Processor
ASM	Algorithmic State Machine
AtoP	Affine to Projective coordinates converter
AU	Arithmetic Unit
AUC	Arithmetic Unit Controller
BCH	Bose, Ray-Chaudhuri, Hocquenghem
C	Challenge
CA	Certification Authority
CEILIDH	**C**ompact, **E**fficient, **I**mproves on **L**UC, Improves on **D**iffie-**H**ellman
CLB	Configurable Logic Block
CMOS	Complementary Metal Oxide Semiconductor
CRC	Cyclic Redundancy Check
CRT	Chinese Remainder Theorem
CRP	Challenge-Response Pair
CSA	Carry Save Adder
CU	Control Unit
DB	Dual Basis
DEMA	Differential ElectroMagnetic Analysis
DES	Data Encryption Standard
DFA	Differential Fault Attacks
DH	Diffie-Hellman
DHP	Diffie-Hellman Problem
DL	Discrete Logarithm
DLP	Discrete Logarithm Problem
DPA	Differential Power Analysis
DSA	Digital Signature Algorithm

DSRCP	Domain-Specific Reconfigurable Cryptographic Processor
EC	Elliptic Curve
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Helman
ECDLP	Elliptic Curve Discrete Logarithm Problem
ECDSA	Elliptic Curve Digital Signature Algorithm
ECIES	Elliptic Curve Integrated Encryption Scheme
ECM	Elliptic Curve factoring Method
ECP	Elliptic Curve Processor
ECPDC	Elliptic Curve Point Doubling Circuit
ECPM	Elliptic Curve Point Multiplier
EEA	Extended Euclidean Algorithm
EEPROM	Electrically Erasable Programmable Read Only Memory
EM	ElectroMagnetic
EMA	Electromagnetic Analysis
FA	Full Adder
FAR	False Acceptance Rate
FF	Flip-Flops
FPGA	Field Programmable Gate Array
FPSLIC	Field Programmable System Level Integration Circuit
FRR	False Rejection Rate
FSM	Finite State Machine
GNFS	General Number Field Sieve
HA	Half Adder
HDL	Hardware Description Language
HECC	Hyperelliptic Curve Cryptography
HECDLP	Hyperelliptic Curve Discrete Logarithm Problem
IBI	Identity-Based Identification scheme
IC	Integrated Circuit
I/O	Input/Output
IOB	Input/Output Block
IP	Internet Protocol
I-PUF	Integrated PUF
ISA	Instruction Set Architecture
LC	Logic Cell
LNCP	Large Number Co-Processor
LSB	Least Significant Bit
LSD	Least Significant Digit
LSR	Left-Shift Register
LSW	Least Significant Word
LUT	Look-Up Table
MA	Modular Addition

MAC	Message Authentication Code
MC	Main Controller
MMI	Modular Multiplicative Inverter
MMM	Montgomery Modular Multiplication
MMMC	Montgomery Modular Multiplication Circuit
MSB	Most Significant Bit
MtoN	Montgomery to Normal representation converter
NAF	Non-Adjacent Form
NB	Normal Basis
NFSNET	Number Field Sieve project
NMOS	Negative-channel Metal-Oxide Semiconductor
NOP	No-OPerations
NtoM	Normal to Montgomery representation converter
ONB	Optimal Normal Basis
P	Prover
PA	Power Analysis
PB	Polynomial Basis
PC	Processing Cell
PE	Processing Element
PKC	Public-Key Cryptography
PtoA	Projective to Affine coordinates converter
PUF	Physical Unclonable Function
RAM	Random Access Memory
RCA	Ripple Carry Adder
REM	Random Exponent Model
RFID	Radio Frequency Identification
RNG	Random Number Generator
ROM	Read Only Memory
RSA	Rivest-Shamir-Adelman
RSR	Right-Shift Register
RST	Reset Signal
SCA	Side-Channel Analysis
SEMA	Simple ElectroMagnetic Analysis
SD	Signed Digit
SI	Standard Identification
SPA	Simple Power Analysis
S-RAM	Static Random Access Memory
SS	Standard Signature
SSH	Secure Shell
SSL	Secure Sockets Layer
SSR	Serial Shift Register
TA	Timing Analysis

TTP	Trusted Third Party
V	Verifier
VHDL	Very High Speed Integrated Circuit Hardware Description Language
VLSI	Very Large Scale Integration
VPN	Virtual Private Networks
XRAM	External RAM

List of Notations

Basic Notation

$\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$	set of integers, rationals, reals and complex numbers
$\mathbb{Z}_p^*$	ring of p-adic integers
$\mathbb{F}$	field
$\overline{\mathbb{F}}$	algebraic closure of a field $\mathbb{F}$
$\mathbb{F}_p$	finite field of p elements
$\mathbb{F}_p^*$	cyclic group of a field $\mathbb{F}$
$\mathbb{A}^2$	affine plane
$\mathbb{P}^2$	projective plane
$char(\mathbb{F})$	characteristic of a field $\mathbb{F}$
$deg(f)$	degree of a polynomial f
$ord(g)$	order of an element g in a group
$Mont(x, y)$	Montgomery modular multiplication of x and y
$\lambda(n)$	$lcm(p-1, q-1)$
φ	Euler totient function
$\Phi_n(x)$	n-the cyclotomic polynomial
$N_{L/K}$	norm map
$Tr_{L/K}$	trace map

ECC/HECC Notation

xP	elliptic curve point multiplication of integer x and P
∞	point at infinity
E	elliptic curve
P	point on an elliptic curve E
(x, y)	affine point
(X, Y, Z)	projective point

$E(\mathbb{F}_p)$	elliptic curve over $\mathbb{F}_p$
$\#E(\mathbb{F}_p)$	number of rational points on a curve E
J	Jacobian of a hyperelliptic curve
D	Divisor of a hyperelliptic curve
$deg(D)$	degree of a divisor
$\sim$	equivalence of divisors

List of Figures

List of Tables

Chapter 1

Introduction and Motivation

1.1 Introduction

Our ever increasing dependence on information implies that the importance of information security is growing. Numerous examples of security applications are present in everyday life such as data encryption, secure e-mail, internet banking, mobile phone communication *etc.* For these applications there exists a range of algorithms that provide basic cryptographic services: confidentiality, data integrity, authentication and non-repudiation [155]. Cryptographic algorithms include stream and block ciphers, hash functions, digital signatures, public-key encryptions *etc.* These algorithms are usually divided into secret-key and public-key algorithms. Although the former allow for a fast encryption of a large amount of data, effective information protection against eavesdropping and modifications in open systems can only be achieved using public-key algorithms.

1.1.1 The Basic Model for Cryptography

Cryptology is the science of secure communication; it is divided into two parts: cryptography and cryptanalysis. Cryptography is dealing with the design and implementation of algorithms that facilitate cryptographic services. On the other hand, the goals of cryptanalysis are to examine the security of a cryptographic algorithm.

The foundations of cryptography originate from Claude Shannon [208] and the basic model is given by Fig. 1.1. Alice and Bob (or any two parties) want to transmit messages over an insecure channel in such a way that an adversary Eve is not able to learn the contents of their communication. For that purpose they use a secret key that was a priori exchanged. In modern cryptography Kerckhoffs's principle is assumed, which states that only the secret key k is not known to an

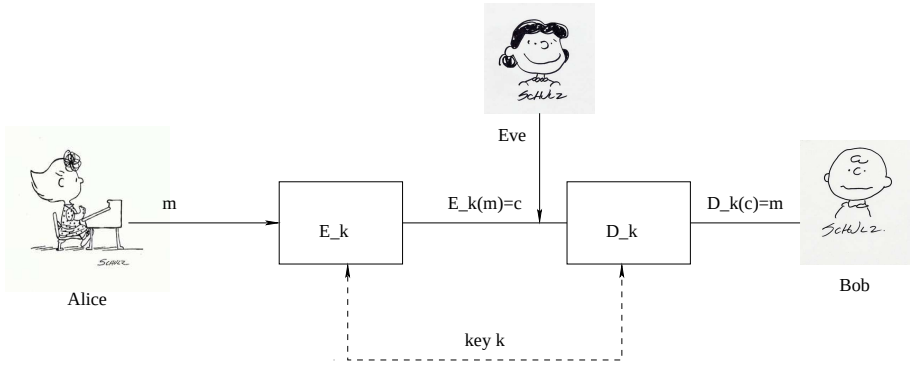


Figure 1.1: The basic model for a cryptosystem.

adversary. This rule was established already in the 19th century by A. Kerckhoffs.

Alice wants to send a message m to Bob, which is called the *plaintext*. It can be any element of a finite set of messages and here we assume that it was converted to a bit string. Alice is using the *secret key* k to encrypt the message by an injective mapping E_k to a string c, which is called the *ciphertext* c. Therefore, we can write $E_k(m) = c$ and this mapping E_k is called the *encryption* operation. Since E_k is injection, the inverse of it exists *i.e.*, the mapping D_k which is called the *decryption* operation. The same key k will be used by Bob for decryption of c *i.e.*, we can write $D_k(c) = D_k(E_k(m)) = m$. In this model, the role of an adversary can be *passive* or *active*. A passive adversary eavesdrops the channel and tries to find m or k from c. An active adversary tries to send her own messages, re-send some old messages *etc.* In this case we talk about tampering or active eavesdropping.

The system depicted in Fig. 1.1 is the model for *symmetric-key* (or secret-key) cryptography. This scheme for two parties, who want to communicate securely, is based on a shared secret key. Although symmetric cryptosystems allow for large amounts of data to be transferred efficiently, key management and key distribution problems do not scale well in the case of a large number of users. Some examples of symmetric-key algorithms are DES, triple-DES, AES and RC4.

1.1.2 Public-Key Cryptography

Diffie and Hellman introduced the idea of *public-key cryptography* [62] in the mid 70's. They showed that one can eliminate the need for prior agreement of a key in order to exchange some confidential data. The system is shown in Fig. 1.2. There exists a pair (E_e, D_d) for every user. Here E and D are respectively encryption and decryption mappings and $(e,\ d)$ are called the public and the secret key. The pair (E_e, D_d) should be easy to generate; in order to achieve secret communication

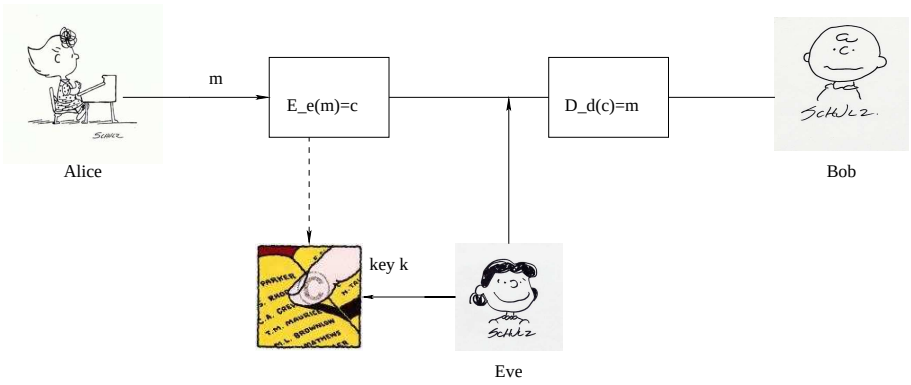


Figure 1.2: A Public-key encryption algorithm.

it is required that: $D_d(c) = D_d(E_e(m)) = m$ and that it is hard to derive d from e. Diffie and Hellman suggested to use a trapdoor one-way function for the encryption E_e. A *one-way function* is a function $f : X \to Y$ such that [237]:

- it is "easy" to calculate $f(x)$ for all $x \in X$;
- knowing $y = f(x)$ it is "computationally infeasible" to compute $x = f^{-1}(y)$ for almost all $y \in Y$;

A *trapdoor one-way function* is a one-way function that satisfies also the following condition:

- $x = f^{-1}(y)$ is easy to compute if some additional (trapdoor) information is provided.

The setting of a public-key cryptosystem also allows for the digital signatures; they were introduced by Diffie and Hellman in order to uniquely bind a message to the sender. This system is shown in Fig. 1.3. In this protocol Alice signs a message m with her private key d and send it to Bob. To verify the signature, Bob has to look up the public key e of Alice and compute $E_e(D_d(m)) = m$.

Public-key cryptosystems are present today in almost all spheres of digital communication *e.g.* for financial, governmental and medical applications; they form an essential building block for network security protocols (*e.g.* SSL/TLS, IPsec, SSH).

When comparing public-key cryptosystems, the following evaluation criteria need to be considered: security, key lengths, speed, implementation issues *etc.* Considering security, the hardness of the underlying mathematical problem is very important. None of these systems has been *proven* to be intractable.

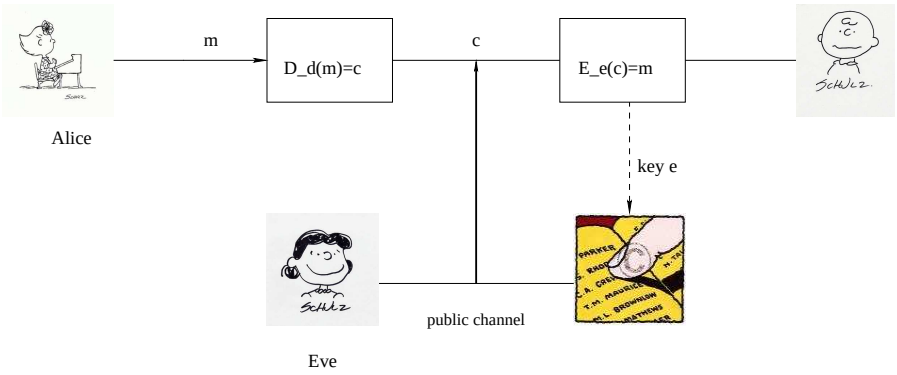


Figure 1.3: A Public-key signature algorithm.

The best-known and most commonly used public-key cryptosystems are based on factoring (RSA) and on the discrete logarithm problem in a large prime field (Diffie-Hellman, ElGamal, Schnorr, DSA) [155]. The RSA public-key cryptosystem is named after its inventors Rivest, Shamir and Adelman [193]. They allow secure communications over insecure channels without prior exchange of a secret key and they also enable digital signatures. Elliptic Curve Cryptography (ECC), which was proposed in the mid 80's by Miller [163] and Koblitz [126], is based on a different algebraic structure. It is important to point out that ECC offer equivalent security as RSA for much smaller key sizes. Other benefits include higher speed, lower power consumption and smaller certificates which is especially useful in constrained environments (smart cards, mobile phones, PDAs, *etc.*).

Hyperelliptic Curve Cryptography (HECC) was proposed in 1988 by Koblitz [127] as a generalization of ECC. The field size for HECC is at least a factor of two smaller than the one of ECC, with the same level of security. This makes HECC a very good choice for platforms with limited resources.

Algebraic tori based cryptosystems are another alternative for PKC. Their benefits are that they allow for shorter transmissions in discrete log-based cryptography over finite fields or on elliptic curves. Torus based cryptography assumes using algebraic tori to construct a group on which the discrete logarithm problem is defined. This idea was first introduced by Silverberg and Rubin in 2003 [196] and they proposed the name of CEILIDH.

Until today numerous public-key cryptosystems have been proposed. Most of the schemes which are used today base their security on a small number of mathematical problems:

- **The RSA problem:** Let a positive integer N is given (that is a product of

two distinct primes p and q), a positive integer e such that $gcd(e, \lambda(n)) = 1$ and an integer c; find m such that $m^e \equiv c \pmod{N}$. Here, we denote $\lambda(n) = lcm(p-1, q-1)$. The RSA problem is the problem of finding e^{th} roots modulo a composite number N. It is related to the Integer Factoring problem, in this case the problem of factoring a composite number N, which is the product of two large primes p and q. It can be shown that if the factors of N are known, the RSA problem can be easily solved [155].

- **Discrete logarithm problem (DLP):** This problem is based on the difficulty of computing logarithms in a large finite field. Some examples are: the Digital Signature Algorithm (DSA), the Diffie-Hellman key agreement protocol, ElGamal encryption and digital signature, Schnorr signature scheme *etc.* It is explained in a more formal way in Chapter 2.

- **Elliptic curve discrete logarithm problem (ECDLP):** This problem is the DLP defined over the group of points on an elliptic curve. A precise definition is given in Chapter 2. Examples include the elliptic curve analogues of: DSA (so-called ECDSA), Diffie-Hellman key agreement, ElGamal encryption and digital signature *etc.*

  The best algorithms known for the integer factorization problem and DLP take the sub-exponential time. On the other hand, the best algorithms known for the ECDLP take fully exponential time.

- **Hyperelliptic curve discrete logarithm problem (HECDLP):** This problem is the DLP defined over the group of Jacobian on a hyperelliptic curve. Examples include the hyperelliptic curve analogues of: DSA, Diffie-Hellman key agreement, ElGamal encryption and digital signature *etc.*

- **Discrete logarithm problem on Algebraic Tori:** This problem is related to the DLP in a finite field $\mathbb{F}_q$. Basically, the DLP on an algebraic torus $T_n(\mathbb{F}_q)$ is equivalent to the DLP in the subgroup of order $\Phi_n(q)$. Here $\Phi_n(q)$ stands for the n-th cyclotomic polynomial. More details are given in Chapter 2.

Here we give some details on the RSA cryptosystem and an intuitive explanation of all cryptosystems that are mentioned in this thesis. Elliptic/Hyperelliptic curve cryptosystems and Tori-based cryptosystems are introduced in Chapter 2 in more detail.

Before explaining the idea of Rivest, Shamir and Adleman we remind here on the requirements for asymmetric ciphers as introduced by Diffie and Hellman. Alice must have a public key e_A, which could be published such that Bob (or anybody else) can use it to encrypt a message to her. Because the public-key operation is a one-way function, it is not feasible for anybody else to decrypt her

messages except Alice, because she owns special trap-door information that allows her to decrypt messages that were sent to her.

The one-way function of RSA is based on modular exponentiation and the details are explained below. Here we stress how the composite number N makes this one-way function invertible. In order to set-up her keys, Alice first chooses two primes p and q such that $N = p \cdot q$. Then, Alice can publish N as her public value and keep the knowledge of p and q as a secret. This is the results of the fact that factoring long integers is a hard problem. Bob now has a one-way function related to her public-key, so he can look it up and send an encrypted message to Alice. As Alice is the only person who knows the factorization of N, she and only she can decrypt messages that were sent to her. More intuitive explanations of various ancient and modern cryptographic ciphers can be found in the popular book by Simon Singh [211].

Most of the public-key algorithms require the structure of an algebraic group. When Diffie and Hellman invented public-key cryptography the group they used consisted of the numbers modulo n. In the case of ECC, another group is used *i.e.*, the group of points on an elliptic curve. Public and private keys are defined as points on a curve and the one-way function is the multiplication of a point with a scalar. Therefore, Alice having as a private-key some integer e_A can send to Bob a multiple of a point P, so $e_A \cdot P$, which is also a point on the same curve as P lies on, due to the properties of a group. Nobody can recover her key by knowing $e_A P$ and P due to the difficulty of the ECDLP. Similar situation holds for HECC as an elliptic curve can be seen as a special case of a hyper-elliptic curve.

On the other hand, torus-based cryptography gives a possibility to work in a subgroup, while maintaining the security of a bigger group. More precisely, let $\mathcal{G}$ be a group and let $\mathcal{F}$ be a subgroup of $\mathcal{G}$. In the case of finite fields we can consider the multiplicative group $\mathbb{F}_{p^n}^*$ and the subgroup $\mathbb{F}_p^*$. It is clear that the representation of $\mathbb{F}_p^*$ can be n times more efficient w.r.t. memory and bandwidth. However, one has to be careful in order to keep the security level *i.e.* the DLP from the $\mathbb{F}_{p^n}^*$. To achieve that property the norm function has to be used. Namely, taking the norm "down to" a subfield could provide the solutions modulo $p-1$, which would make a system n times less secure. The solution is to take only those points from the field $\mathbb{F}_{p^n}^*$ for which the norm to all subfields of it is 1. In that way no extra information can be obtained from the norm that could allow the adversary to solve the DLP. This property results in the definition of torus T_n in which one can perform cryptographic protocols by transmitting only $\varphi(n)$ of $\mathbb{F}_p^*$ instead of n. In this way the obtained factor of compression for $n = 6$ is $\frac{6}{\varphi(6)} = 3$. More mathematical details on algebraic tori are given in Chapter 2 and on torus T_6 in Chapter 8.

The RSA Algorithm: The private key of a user consists of two large primes p and q and an exponent d. The public key consists of a pair (N, e), where $N = p \cdot q$

is the modulus (at least 1024 bits) and an exponent e is such that

$$e = d^{-1} \bmod \lambda(N)\,. \tag{1.1}$$

The corresponding p, q and d are kept secret. To encrypt a message M the user computes $C = M^e \bmod N$ and decryption is described by: $M = C^d \bmod N \equiv M^{1+k\varphi(N)} \equiv M \bmod N$. The previous equality follows by Fermat's theorem and the fact that λ is a divisor of $\varphi(N) = (p-1)(q-1)$. The RSA function is the modular exponentiation with the public exponent e and the private exponent d is referred to as the trapdoor to invert the function.

RSA was carefully examined since its invention and no serious attack has been found in the sense of breaking RSA theoretically. Some choices have to be avoided mainly w.r.t. required bit-lengths of exponents but it seems that proper implementation can be trusted. It is known that textbook RSA is insecure [33]. Secure RSA implementation requires padding of a plaintext before encryption and signing. There exist several standards that explain these schemes such as PKCS #1. Boneh [33] gives a nice overview of attacks on RSA.

Basic cryptographic services such as key agreement, digital signatures and encryption of data can be achieved by means of public-key protocols. Here, we mention the most important protocols.

The **Diffie-Hellman key agreement** protocol gives the first practical solution to the key exchange problem for two parties. The basic version of this protocol is as follows [155]:

1. *One time setup:* A prime p such that $(p-1)/2 = p'$, where p' is also a prime and a generator α of $\mathbb{Z}_p^*$ $2 \leq \alpha \leq p-2$ are selected and published.
2. *Protocol messages:*
$$A \to B : \alpha^x \bmod p \quad (1)$$
$$B \to A : \alpha^y \bmod p \quad (2)$$
3. *Protocol actions:* Perform the following steps each time a shared key is required.
   (a) A chooses a random secret x, $1 \leq x \leq p-2$ and sends B message (1).
   (b) B chooses a random secret y, $1 \leq y \leq p-2$ and sends A message (2).
   (c) B receives α^x and computes the shared key as $K = (\alpha^x)^y \bmod p$.
   (d) A receives α^y and computes the shared key as $K = (\alpha^y)^x \bmod p$.

The **ElGamal encryption** scheme can be also described in the group $\mathbb{Z}_p^*$ or in any finite cyclic group $\mathcal{G}$ as follows [155]. Here α is a generator of a group $\mathcal{G}$ and A's public-key is (α, α^a) and A's private key is a.

1. *Encryption:* In order to send a message m, B should perform the following steps:

(a) Obtain A's authentic public key (α, α^a).

(b) Represent the message as an integer $m \in \mathcal{G}$.

(c) Select a random integer k, $1 \leq k \leq n-1$. Here, n is the order of the group $\mathcal{G}$.

(d) Compute $\gamma = \alpha^k \bmod p$ and $\delta = m \cdot (\alpha^a)^k$.

(e) Send the ciphertext $c = (\gamma, \delta)$ to A.

2. *Decryption:* To recover the plaintext m from c, A should do the following:

(a) Use the private key a to compute γ^{-a}.

(b) Recover m by computing $(\gamma^{-a} \cdot \delta)$.

To complete the list of the basic public-key schemes we also add the **Digital Signature Algorithm (DSA)** [155]. By means of this algorithm a party A signs a binary message m and any other party B can verify this signature by use of A's public key as shown in Fig. 1.3. Prior to the algorithm, DSA primes p and q have to be generated. The prime q is selected first and then p is chosen such that $q|(p-1)$. The DSA algorithm itself consists of two parts: signature generation and verification. It also uses a hash function h that maps binary strings of arbitrary length to binary strings of a fixed length. For the sake of completeness we give a definition of hash function.

Definition 1 [155] *A hash function is a function h which has, as a minimum, the following two properties:*

1. *compression: h maps an input x of arbitrary finite bitlength, to an output h(x) of fixed bitlength.*

2. *ease of computation: given h and an input x, h(x) is easy to compute*

   An important class of hash functions are message authentication codes (MACs) that are often used to protect the message integrity. More precisely, a MAC algorithm is a family of functions h_k parameterized by a secret key k, with the properties above satisfied and also the following property.

3. *computation-resistance: given zero or more text-MAC pairs $(x_i, h_k(x_i))$, it is computationally infeasible to compute any text-MAC pair $(x, h_k(x)$ for any new input $x \neq x_i$ (including for $h_k(x_i) = h_k(x)$)*

We use MACs in Chapter 6. Next, we give all steps of the DSA algorithm.

1. *Signature generation:* In order to sign a message m, A should perform the following steps:

(a) Select a random secret integer k, $0 < k < q$.

(b) Compute $r = (\alpha^k \bmod p) \bmod q$.

(c) Compute $k^{-1} \bmod q$.

(d) Compute $s = k^{-1}(h(m) + ar) \bmod q$.

(e) A's signature for m is the pair (r, s).

2. *Verification:* To verify A's signature on m, B performs the following steps:

(a) Obtain A's system parameters (p, q, α, y). Here, α is a generator of the cyclic group of order q in $\mathbb{Z}_p^*$, $g \in \mathbb{Z}_p^*$ such that $\alpha = g^{(p-1)/q} \bmod p$ and $y = \alpha^a \bmod p$.

(b) Verify that $0 < r < q$ and $0 < s < q$; if not reject the signature.

(c) Compute $w = s^{-1} \bmod q$ and $h(m)$.

(d) Compute $u_1 = w \cdot h(m) \bmod q$ and $u_2 = r \cdot w \bmod q$.

(e) Compute $v = (\alpha^{u_1} y^{u_2} \bmod p) \bmod q$.

(f) Accept the signature if and only if $v = r$.

1.1.3 Attacks on Cryptosystems

An encryption algorithm is considered to be broken when a cryptanalyst recovers the secret key or the plaintext. Attacks on cryptography can be divided into two groups: *mathematical attacks* (more traditional type of attacks which are sometimes purely theoretical) and *implementation attacks* (more practical type which pose a growing threat today).

The mathematical approach is very useful for detecting fundamental weaknesses in a cryptographic algorithm; as a straightforward example we can mention the *brute-force attack*. In a brute-force attack on an asymmetric or a symmetric cipher, the cryptanalyst tries every possible key until the correct key is found. A brute-force attack on DES has recently become much faster as hardware and computers became more powerful. The book [55] describes a machine which was actually built to crack DES. In an update that was given by Wiener [248] is elaborated that the key search machines of $10,000\,\$$ or $1,000,000\,\$$ cost would break DES in 2.5 days or 35 minutes respectively. A good update on the latest progress on special purpose hardware for cryptanalysis can be found in the contributions to SHARCS workshop [233].

So-called *short-cut attacks* against a cryptosystem try to solve the difficult mathematical problem such as the integer factorization problem or the DLP. Examples are the Number Field Sieve algorithm and the Quadratic Sieve algorithm for the factorization; and the Pohlig-Hellman method and the index calculus methods for the DLP [155]. The best algorithm known to factor integers of more than 100 digits and to solve the DLP in $\mathbb{F}_p^*$ is the General Number Field Sieve

(GNFS) [67]. An integer N can be factored in subexponential time w.r.t. its size *i.e.* the complexity of factoring is [67]:

$$L_N(\alpha, \beta) := e^{(\beta+o(1))(logN)^{\alpha}(log\, logN)^{1-\alpha}}$$

In this notation, $f(n) = o(g(n))$ is equivalent to $n/g(n) \to 0$ as $n \to \infty$. In particular, for GNFS it holds heuristically:

$$L_N\left(\frac{1}{3}, \sqrt[3]{\frac{64}{9}}\right) .$$

Implementation Attacks

Implementation attacks exploit weaknesses in specific implementations of a cryptographic algorithm. Sensitive information, such as secret keys, a plaintext can be obtained by observing the power consumption, the electromagnetic radiation, *etc.* This class of attacks are called *side-channel attacks*. Another type of attacks, so-called *fault attacks*, reveal secret information by inserting faults into the device.

In 1996 Kocher *et al.* introduced the concept of timing attacks by showing that secret information can be extracted through measurements of the execution time of cryptographic algorithms [133]. Timing attacks are applicable to all implementations that have a non-constant execution time which depends on the bits of the secret key. It is evident that constant time-implementations should remove all vulnerabilities of cryptographic applications with respect to timing attacks. A great deal of work has been devoted to this topic [242, 170].

Two years later Kocher *et al.* performed successful attacks by measuring the power consumption while the cryptographic circuit is executing the implemented algorithm [134, 135]. The most straightforward power analysis, called *Simple Power Analysis* (SPA), uses a single measurement to reveal the secret key by searching for patterns in the power trace. However, implementations that are resistant against SPA attacks, can still be broken by using a more advanced technique, namely *Differential Power Analysis* (DPA). In this case many power measurements are evaluated using statistical analysis. As power-analysis based attacks and their prevention are discussed later in this thesis, we explain our assumptions on power models for CMOS circuits in more detail.

Basics of CMOS Power Consumption

The power consumption of a CMOS gate can be written as:

$$P = P_{SW} + P_{SC} + P_{LK} ,$$

where P_{SW}, P_{SC} and P_{LK} denote switching (or dynamic) power, short-circuit power and leakage (or stand-by) power respectively [149]. It is known that in

older technologies (0.13 μm and above), the largest contribution comes from P_{SW}. On the other hand, in deep sub-micron processes, P_{LK} becomes critical. In this thesis we deal only with the CMOS technology and it was shown that the CMOS power consumption model is valid for the FPGA platforms used in this thesis [177]. Leakage power becomes an issue at 90 nm and below. For the experiments in this thesis, FPGA platforms are used as in [178]. It was clear that the side-channel attacks are possible based on the switching power model and even if leakage power is an issue, there is a lot of information in the switching power. Hence, we focus on switching power and assume that it influences the total power the most of all components. In more detail the power dissipation in a CMOS gate can be written as:

$$P = (0.5C_L \cdot {V_{DD}}^2 + Q_{SC} \cdot V_{DD}) f_{Clock} E_{SW} + I_{LK} \cdot V_{DD} \,,$$

where V_{DD} is supply voltage, Q_{SC} is the short-circuit charge f_{Clock} is the operating frequency, C_L is the load capacitance and E_{SW} is the switching activity factor. The second term represents the static power dissipation due to the leakage current I_{LK}. The leakage current is directly determined by the number of gates and the process technology. In general, the leakage current consists of two major contributions [149]:

$$I_{LK} = I_{sub} + I_{gate} \,,$$

where I_{sub} stands for sub-threshold current caused by low threshold voltage and I_{gate} is gate current caused by the reduced thickness of gate oxide. The former one is dominant but grows only 5 times per generation while I_{gate} grows up to 500 times per generation [149].

It is notable that considering only switching power as the model for power consumption is not precise but for our purposes it is shown to be good enough. It is not our goal to deal with these models in detail, but future research will have to take also other components of power consumption into account especially looking into future technologies. To a first order we assume that leakage power is a constant. As such it contributes to the total power consumption, but it does not affect our ability to perform side-channel attacks. Indeed, side-channel attacks are based on power variations not the total power. Leakage power depends on the state of the gate. As such in future technologies it might be an extra source of side-channel information. This is a topic of future research. A good update on the present state of the art in process variation one can get from the International Solid-State Circuits Conference (ISSCC).

We also mention here that there exist some transitions in circuits that consume power but they do not change the functionality. Namely, due to timing skew through the logic, the output node and the internal nodes can make extra transitions *i.e.* glitches [190]. Glitches are also a source of side-channel information. It has been shown that circuits can be broken based on glitches [185]. Gates have also different delays and therefore the chances to have unbalanced signals which propagate through the circuit are very high.

For low-power consumption as imposed by many new applications, it is necessary to minimize E_{SW}, the circuit size (for the current CMOS technology) and the operating frequency. This can be achieved among others by architectural decisions as well, which will be discussed more in Chapter 6 as one of the key-issues for RFID technology. More precisely, the metric that is typically used and that should be minimized accordingly is energy per processed bit E. This value can be calculated as:

$$E = \frac{P}{throughput} \left[\frac{J}{bit}\right] .$$

Countermeasures for Side-channel Attacks

In the literature, both software and hardware countermeasures to prevent power analysis based attacks were proposed. The main software countermeasures against PA attacks include masking techniques and time randomization. Time randomization were proposed by Daemen and Rijmen [59] and also by Messerges [160]. In this type of countermeasure, operations occur after random time intervals. Therefore, DPA is likely to fail as power consumption traces cannot be easily aligned for statistical analysis. Masking techniques are the most common countermeasures and they were proposed on both hardware and software level [96, 56, 3, 95, 90]. In this method all the intermediate data in an algorithm are masked with a random value (*e.g.* by means of XORing). As a consequence, the first-order power analysis attacks become unfeasible. To overcome this type of countermeasures higher-order attacks were proposed [161].

The most simple hardware countermeasure against power analysis attacks is increasing the measurement noise by use of a random number generator [59]. On the other hand several novel circuit designs have also been suggested, among first by Tiri and Verbauwhede [224]. This method employs logic gates with a power consumption that is independent of the data signals and therefore the technique removes the foundation for DPA. More precisely, it is known that the power consumption of a circuit depends on the all transitions of the gates. If these transitions depend on secret information, the security of the implementation can be compromised.

Hence, in order to make a circuit resistant against power analysis attacks, the power consumption should be independent of the secret information. Asynchronous circuits can be also used [77]. A nice surveys on countermeasures can be found in the work of Borst *et al.* [37, 38] and Mangard [150].

Another side-channel attack is based on the electromagnetic radiation [80, 189]. An *ElectroMagnetic Analysis* (EMA) attack can be performed using similar methodology as for power attacks. Analogously, the terms Simple EMA (SEMA) and Differential EMA (DEMA) are commonly used. However, EMA attacks can extract much more information than power attacks. The reason is that the latter measure the complete power consumption, while EMA attacks can zoom in on

smaller parts of the circuit and it can also measure effects of the so-called "far field".

As EMA attacks are still under intensive development the countermeasures are an active research area. It is obvious that PA countermeasures can also be applied to EMA. However, because the EM signal contains more information than the power signal, other techniques have to be applied to defeat EMA attacks. Examples are asynchronism, dual-line logic, *etc.* [189, 80].

The second kind of implementation attacks are fault attacks. They are based on hardware faults, which are usually caused by some unexpected condition or defect. However, they can also be applied deliberately by the use of fault insertion techniques. The damage they can cause was first elaborated by Boneh *et al.* [34]. Their target application was RSA with the CRT. Biham and Shamir introduced differential fault attacks (DFAs) on the DES algorithm [28] and they showed that DFAs are applicable to any secret key cryptosystem.

Considering feasibility of glitches-based attacks, the first practical result was given by Aumüller *et al.* in [6]. They introduced a *spike*, *i.e.* a certain deviation from a standard interval for a smart card parameter *e.g.* the supply voltage. They performed two attacks on a smart card with an RSA coprocessor and proved that these type of attacks are feasible even if some countermeasure [206] is deployed.

Usual sources of faults are glitches (voltage, clock, temperature, radiations) or beams (UV light, X-rays, laser, camera flash, *etc.*). They can be semi-invasive or invasive. For the former type of fault insertions the best results were achieved by depackaging the chip (as for invasive attacks), but the passivation layer of the chip *i.e.* the shield covering the chip remains intact. More precisely, unlike invasive attacks they do not require electrical contact to the metal surface. As an example of a cheap semi-invasive attack optical fault induction attacks were introduced by Skorobogatov and Anderson [213]. They showed that it is possible to change the state of any individual bit in a microcontroller by using a camera flash and a laser pointer. Their target was an SRAM cell, but this type of attack is applicable to any other kind of memory as well. Another type of semi-invasive attacks are performed by electromagnetic induction *i.e.* eddy currents [198]. The follow-up work of Samyde *et al.* [199] summarizes all kinds of possibilities to read secret data from memories. They compare the effects caused by optical induction on one side and eddy currents on the other side. The conclusion is that for the latter the resolution is not so high as for optical fault attacks. Skorobogatov and Anderson suggested to use self-timed dual-rail logic as a countermeasure. The idea behind this approach is not just to have a logical 0 or 1 encoded as HL and LH (or vice versa), but also to prohibit the HH state. Namely, this combination signals an alarm, which will typically reset the processor. This property is present for example in WDDL logic of Tiri and Verbauwhede [225].

Most of the techniques for fault insertion target the secret data that is manipulated during the execution of a cryptographic algorithm, but some are directed

on faults on memories. Neve *et al.* [166] summarize these issues for memories and their secure use.

All attacks mentioned above are considered to be semi-invasive. Invasive fault induction includes probing the circuit with active or energy probes as mentioned by Weingart [246].

As for countermeasures against fault attacks, Samyde *et al.* [199] mention the importance of hardware countermeasures and list the necessary requirements for the protection of modern smart cards. Suggested measures to achieve minimal physical security against fault attacks include: implementing the CPU using random place-and-route to avoid the visibility of the registers, memory encryption, use of self-timed logic with built-in alarm propagation and alarm sensors for temperature, X-rays, *etc.* We have to underline here again that all these countermeasures have to be constantly re-evaluated due to progress in process technology. For example, random place-and-route may not be possible in deep submicron technologies. As a direct consequence, a general recommendation is that the use of special techniques should be minimized (*e.g.* to key manipulations). Therefore, a special care is required also for memories as they might become the weakest components in a design. More precisely, memories require special attention as they have regular structures. Randomization is not an option. But they have an advantage that one can focus on the side-channel security of the individual components (the cell, word line, bit line, decoder *etc.*) and the techniques can then be "coppy-pasted" to the complete memory.

1.1.4 Implementations of Cryptographic Algorithms

Due to all previously mentioned threats, but also as a result of various constraints that are imposed by security applications, special care is required when implementing a cryptographic algorithm. Especially implementations of public-key cryptography present a challenge for most application platforms varying from software to hardware. The reason is that one has to deal with very long numbers (up to 2048 bits) in conditions that can be quite severe in costs, area and power. Emerging examples are RFID tags [120] and sensor networks [84]. For implementations of cryptographic protocols to achieve various security applications it is not enough to come up with an efficient implementation; it has to be also secure against side-channel attacks. With respect to this, it is well known that although software platforms offer a cost effective and flexible solution, only hardware implementations provide a suitable level of security related to side-channel attacks.

Smart cards are cryptographic tokens that are used more and more nowadays in financial applications, e-government, health insurance, *etc.* A smart card is an example of a platform where the core part, *i.e.* the most computationally intensive part, is hardware-based and some other building blocks of cryptographic protocols can run on a processor.

For the choice of implementation platform, several factors have to be taken into account. Hardware solutions provide the speed and more physical security, but the flexibility of hardware implementations is quite limited. For that property software solutions are needed, but on the other hand a pure software solution is not a feasible option for the implementations of public-key cryptography (PKC) in most constrained environments. Therefore, for embedded implementations, a desired trade-off in costs/flexibility and speed can be achieved by hardware-software co-design.

1.2 Our Contributions

Security is as strong as the weakest link, therefore protecting cryptographic systems should be done at all levels of abstraction, which are depicted in the security pyramid in Fig. 1.4 [18]. Each abstraction layer represents specific modelling, design and implementation issues that must be covered for secure system operation [203]. In this thesis we deal with the top three levels of the pyramid, *i.e.* with

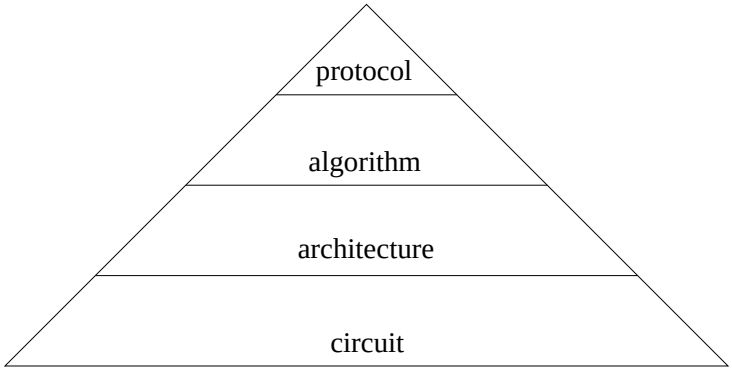


Figure 1.4: Security pyramid showing the different levels of abstraction.

secure implementations of cryptographic protocols and algorithms required for realization of these protocols. We also consider architectures that facilitate both efficiency and security. However, hardware architectures in the thesis have a purpose mainly to prove our higher-level choices. Below we explain our contributions in more detail.

Currently, the most widely used public-key cryptosystems are RSA and Elliptic Curve Cryptosystems. In the first part of our research, we have designed a hardware architecture for modular multiplication, which is efficient for bit-lengths suitable for both commonly used types of Public Key Cryptography. The challenge is to achieve both efficiency and security while processing large integers (160-2048

bits). RSA, still the most popular public key cryptosystem, has at its root the modular exponentiation operation. Modular exponentiation consists of repeated modular multiplications, which is also the basic operation for ECC. We have developed an efficient hardware implementation of Montgomery's Modular Multiplication (MMM) on an FPGA. The follow-up of this work was the development of a one common architecture for RSA and ECC. Figure 1.5 gives a graphical explanation of our results in the light of existing requirements. We achieved a more compact solution by using the same basic blocks for the arithmetic in a large prime field and we improved the algorithms to eliminate the conditional operations. Namely, it is known that conditional instructions facilitate side-channel attacks, especially timing and power attacks [100, 205, 244]. Removing the conditional instruction from the algorithm of Montgomery speeded up exponentiation that is required for RSA algorithm and it also eliminated one typical target of side-channel attacks. Our first goal was to improve the performance but it also resulted in a more secure solution. This is one rare case where providing side-channel security does not imply penalties in performance or area complexity. However, to resist more advanced attacks countermeasures are needed. Our results were published in [20, 174, 175, 9, 176].

Fields of characteristic two (binary fields) were also addressed. We have published an FPGA implementation of a new multiplier for binary fields [15]. A complete ECC implementation that deploys this new multiplier and for which point operations were adapted to obstruct simple side-channel attacks was also completed. We followed the recommendations of Chevallier-Mames *et al.* in order to achieve so-called "side-channel aware design" [51]. Our results were published in [20, 17, 16]. Starting from an unprotected implementation we achieved some simple side-channel protection by merely tweaking the algorithms for point operations of López and Dahab [148] in order to make them fully balanced. In this way the area complexity was preserved and we made a small trade-off between speed and security (Fig. 1.6).

We describe the first HECC implementation using a hardware/software co-design method on the 8051 micro-processor, which uses a small hardware co-processor to optimize the performance. We started from a pure software solution and afterwards we moved the hardware/software border-line in order to explore possible partitions between the two. We present various options for hardware/software co-design on our platform (Fig.1.7). Furthermore, we compared on the same platform HECC and ECC and results were for the first time in favor of HECC. Our results were published in [11, 106, 107].

Radio frequency identification is an example of an emerging technology which requires authentication as a cryptographic service. This property can be achieved by symmetric as well as asymmetric primitives. Previously known work considered mainly symmetric key algorithms *e.g.* AES and recently the publication of Wolkerstorfer [251] considered also ECC for these low-cost applications. The suit-

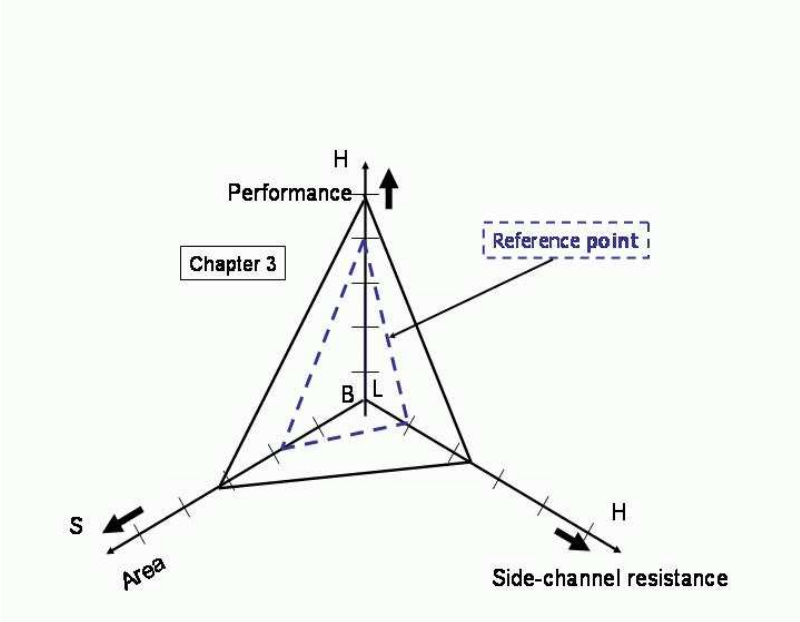


Figure 1.5: Contributions of Chapter 3. The reference point were the current state of the art algorithms for hardware implementations. We improved the algorithms and we developed one unique platform for RSA and ECC.

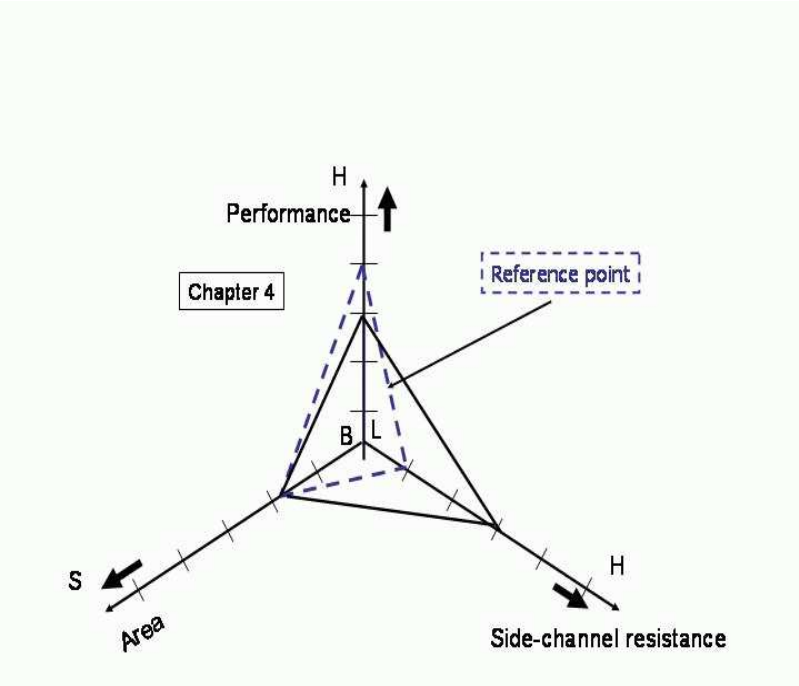


Figure 1.6: Contributions of Chapter 4. The reference point were the algorithms of López and Dahab [148]. We balanced the point operations to obtain a resistance to SPA.

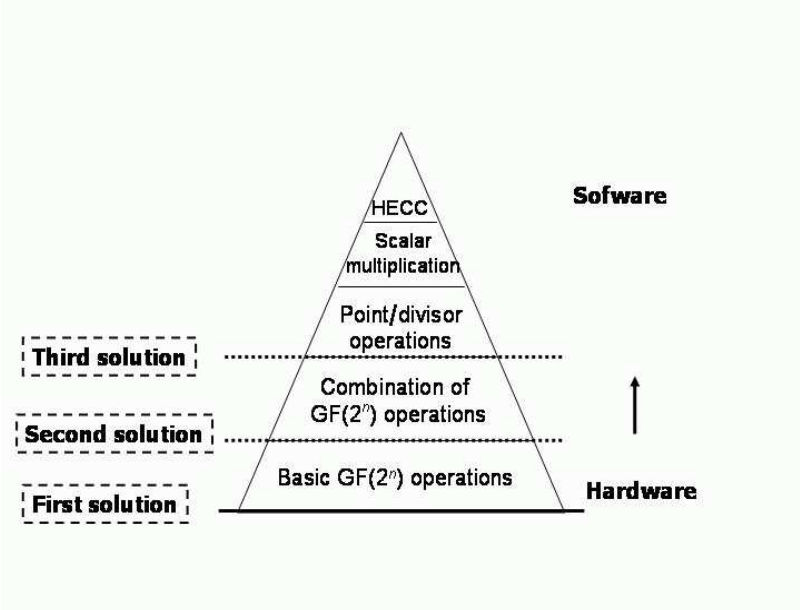


Figure 1.7: Contributions of Chapter 5. We started with a pure software implementation and afterwards we moved the HW/SW border-line in order to achieve the best partitioning.

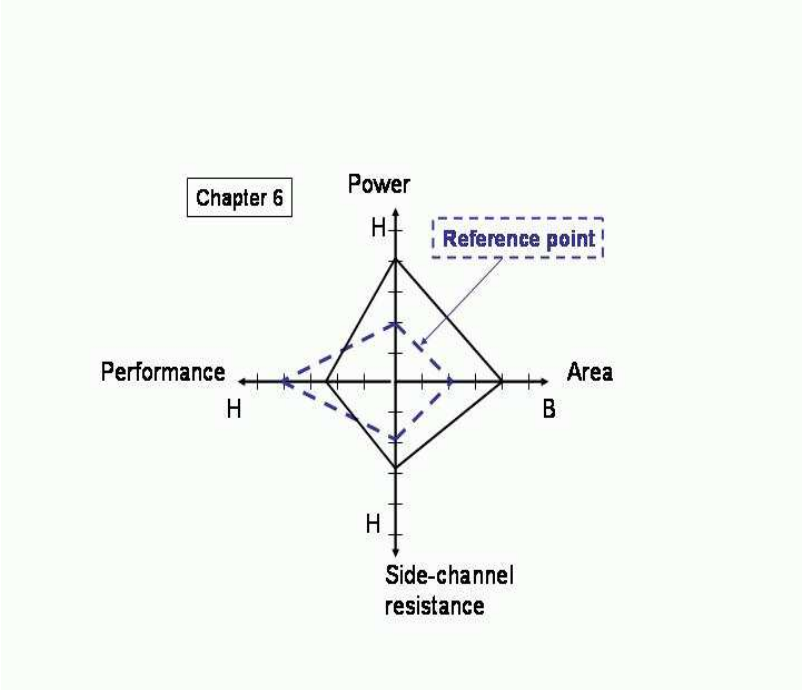


Figure 1.8: Contributions of Chapter 6. As a starting point we took the algorithms for ECC that are introduced in Chapter 4. We increased slightly the latency in order to obtain a more compact solution.

ability of PKC is an open research problem as constraints on cost, area and power are very challenging. We have investigated how an RFID-tag can be made unclonable by linking it inseparably to a Physical Unclonable Function (PUF). We present the security protocols that are needed for the detection of the authenticity of a product when it is equipped with such a system. We distinguish hereby between off and-on line authentication. It was shown that a PUF based solution for RFID-tags is feasible both in the on-line and the off-line case. We also discuss the feasibility of such a low-cost system that is one of current challenges of hardware implementations [229]. In this case we introduce another axes in our design space *i.e.* power (see Fig. 1.8). Namely, new application areas of RFID and sensor networks have firm requirements on low-power.

Besides practical research about the feasibility of side-channel attacks, this thesis also considers more theoretical models. A typical question is about the amount of secret information that some side-channel leakage provides. We have

considered timing information leakage that is derived about a secret exponent when modular exponentiation algorithms are deployed in an information theoretic setting. This so-called random exponent model (REM) has subsequently been used to obtain data on the exponent leakage, showing that the amount of information is too small to be useful in practice. A further refinement of the model, considering the observation of multiple parts of the exponent, has also been given. In the next step we consider the RSA case *i.e.* we assume exponents to be random but co-prime to some even composite number. The result is given by an upper bound on the information leakage in this refined model. Hence, one contribution of this thesis is also this information theory approach to side-channel leakage and our results were published in [12, 13].

To our knowledge, this work presents the first hardware architecture for efficient implementations of tori based cryptosystems. In this first instance we focus on efficient hardware architecture for CEILIDH. Side-channel investigation is left for future work. We compare our results with ECC and RSA implementations.

To conclude, this work makes the connection between algorithms on one side and hardware realizations of those algorithms on the other side. The criterium to do that is mainly performance, compactness and low-power for embedded systems and increased resistance against side-channel analysis (SCA).

1.3 Thesis Outline

In Chapter 2 we introduce the notation and mathematical background on elliptic and hyperelliptic curves and both types of cryptosystems that are based on discrete logarithm problem in the group of algebraic curves over a finite field, namely ECC and HECC. We also give an overview of the mathematical background for torus based cryptography.

Chapter 3 details the unique RSA/ECC architecture and suitable algorithms that also feature simple side-channel resistance. Furthermore, we explore some options for parallelism at the algorithmic level in order to optimize the ECC performance.

In Chapter 4 balanced algorithms for curve based cryptography (*i.e.* ECC and HECC) are given. The results were published in [17, 16]. We made the ECC point addition and doubling balanced, *i.e.* they are implemented as identical sequences of operations.

Chapter 5 presents results that show for the first time that even on a small 8-bit processor one can implement hyperelliptic curve cryptography efficiently. It is also the first result showing that on some platforms HECC outperforms ECC. A small hardware module that we designed results in a significant speed-up compared with a software-only solution. This hardware/software co-design approach offers a new alternative for low-power and low-footprint devices. The results of this chapter were published in [11, 106, 107].

In Chapter 6 we propose a solution for anti-counterfeiting based on RFID-tags and Physical Unclonable Functions (PUFs) [232, 212]. Our solution withstands physical cloning attacks as well as cloning attacks and it is based on analysis of the verification protocols. We investigate both the on-line and the off-line situation and we show that the solutions that we propose are feasible on a constrained device such as an RFID-tag. The results of this chapter will apear in [229].

In Chapter 7 we introduce an information-theoretic approach to side-channel analysis where the actual leakage is correlated to Hamming weight. We use a combinatorial approach to develop a simple model for Hamming weight leakage. To obtain information on the exponent leakage a so-called Random Exponent Model (REM) is used and afterwards we examine a more realistic situation, namely the RSA algorithm. The results prove that REM is a good way to model the information leakage due to side-channel analysis. The results were published in [12, 13].

Finally, in Chapter 8 we consider hardware implementations of recently proposed tori-based cryptosystems. One option for hardware implementation is described and the preliminary results show that this is also a viable solution for efficient PKC implementations.

Chapter 2

Curve-based Cryptography and Algebraic Tori

In this chapter, we introduce some notation and we give an overview of the mathematical background of elliptic/hyperelliptic curves and of algebraic tori. We also introduce the cryptosystems based on these mathematical objects. Most of the material on ECC/HECC presented here can be found in books [110, 209, 158, 130, 29, 103]. Most of the details for tori are from Lidl and Niederreiter [145], otherwise references are given.

2.1 Elliptic Curve Cryptography (ECC)

2.1.1 Definitions

Let $\mathbb{F}$ be any field *i.e.* a set on which two binary operations are defined, so-called addition and multiplication. That means that there exist the zero-element 0 (for the addition) and the identity e (for the multiplication) such that $0 \neq e$. Also $\mathbb{F}$ is an additive abelian group (with 0 as athe neutral element) and all elements except 0 form a multiplicative group (with the identity element e). A field $\mathbb{F}$ is said to be an extension field of $\mathbb{K}$ if $\mathbb{K}$ is a subfield of $\mathbb{F}$. In this case $\mathbb{F}$ is a vector space over $\mathbb{K}$.

Theorem 1 ([110], p. 258) *The following conditions on a field $\mathbb{F}$ are equivalent:*

1. *Every nonconstant polynomial $f \in \mathbb{F}[x]$ has a root in $\mathbb{F}$.*
2. *Every nonconstant polynomial $f \in \mathbb{F}[x]$ splits over $\mathbb{F}$.*
3. *Every irreducible polynomial in $\mathbb{F}[x]$ has degree one.*

4. *There is no algebraic extension of* $\mathbb{F}$ (*except* $\mathbb{F}$ *itself*).
5. *There exist a subfield* $\mathbb{K}$ *of* $\mathbb{F}$ *such that* $\mathbb{F}$ *is algebraic over* $\mathbb{K}$ *and every polynomial in* $\mathbb{K}[x]$ *splits in* $\mathbb{F}[x]$.

Definition 2 *A field that satisfies the conditions of Theorem 1 is said to be* ***algebraically closed****. An extension field* $\mathbb{F}$ *of a field* $\mathbb{K}$ *that satisfies the equivalent conditions of Theorem 1 is called an* ***algebraic closure*** *of* $\mathbb{K}$.

Throughout this chapter $\mathbb{F}_q$ denotes the finite field containing q elements, where q is a prime power. For a field $\mathbb{F}_q$, $\overline{\mathbb{F}}_q = \bigcup_{k\in\mathbb{N}_0} \mathbb{F}_{q^k}$ is the algebraic closure of $\mathbb{F}_q$.

Definition 3 ([209], p. 5) *Let* $\mathbb{F}$ *be a field. The set*

$$\mathbb{A}^2 = \mathbb{A}^2(\overline{\mathbb{F}}) = \{P = (x_1, x_2) : x_i \in \overline{\mathbb{F}}\}$$

is called the ***affine plane*** *over the field* $\mathbb{F}$. *The set of* $\mathbb{F}$-***rational points*** *in* $\mathbb{A}^2$ *is the set*

$$\mathbb{A}^2(\mathbb{F}) = \{P = (x_1, x_2) : x_i \in \mathbb{F}\}.$$

The ***projective plane****, denoted* $\mathbb{P}^2$ *or* $\mathbb{P}^2(\overline{\mathbb{F}})$ *is the set of all triples*

$$(x_0, x_1, x_2) \in \mathbb{A}^3$$

such that $\exists i$ *for which* $x_i \neq 0$, *modulo the equivalence* $\sim$ *acting on the set of all nonzero vectors of* $\mathbb{F}^3$. *The equivalence relation* $\sim$ *is given by* $(x_0, x_1, x_2) \sim (y_0, y_1, y_2)$ *if there exists a* $\lambda \neq 0 \in \overline{\mathbb{F}}$ *such that* $(x_0, x_1, x_2) = \lambda(y_0, y_1, y_2)$. *An equivalence class containing* (x_0, x_1, x_2) *is sometimes denoted by* $[x_0, x_1, x_2]$ *and* x_0, x_1, x_2 *are called* ***homogeneous coordinates*** *for the corresponding point in* $\mathbb{P}^2$. *The set of* $\mathbb{F}$-***rational points*** *in* $\mathbb{P}^2$ *is the set:*

$$\mathbb{P}^2(\mathbb{F}) = \{[x_0, x_1, x_2] \in \mathbb{P}^2 : \ x_i \in \mathbb{F}\}.$$

There exist a map between points in $\mathbb{A}^2(\mathbb{F})$ and points of $\mathbb{P}^2(\mathbb{F})$ that is given by: $(x, y) \mapsto [(x, y, 1)]$. This map is called the *homogenization map*. The reverse map $[(x, y, 1)] \mapsto (x, y)$ is called the *dehomogenization map*. It only operates on points in $\mathbb{P}^2(\mathbb{F})$ with z-coordinate for which $z \neq 0$. The projective plane can be seen as the affine plane with some extra points added, for which $z = 0$ (the points at infinity). The projective plane can be also defined as

$$\mathbb{P}^2 = \mathbb{A}^2 \cup \mathbb{P}^1.$$

Here, $\mathbb{P}^1$ represents the projective line which can be seen as the set of all directions in $\mathbb{A}^2$, where *direction* has a non-oriented notion.
Next, we define a curve in both cases because projective coordinates have some advantages for implementations of elliptic curves based protocols [111, 29].

Definition 4 [158] *An **algebraic curve** in the affine plane $\mathbb{A}^2$ is set of solutions to a polynomial equation in two variables: $f(x, y) = 0$. A polynomial $F(X, Y, Z)$ is called a **homogeneous polynomial of degree d** if it satisfies the identity $F(tX, tY, tZ) = t^d F(X, Y, Z)$. A **projective curve** C in the projective plane $\mathbb{P}^2$ is defined as a set of solutions to a polynomial equation $C : F(X, Y, Z) = 0$, where* F *is a non-constant homogeneous polynomial. The **degree of the curve** C is the degree of the polynomial* F*.*

Let $\mathrm{C} \subset \mathbb{P}^2$ be a curve given by a homogeneous polynomial of degree d,

$$C : F(X, Y, Z) = 0\,.$$

If $P = [a, b, c] \in C$ is a point of C with $c \neq 0$, then according to the mapping $\mathbb{P}^2 \hookleftarrow \mathbb{A}^2 \cup \mathbb{P}^1$, the point $P \in C \subset \mathbb{P}^2$ corresponds to the point

$$(\frac{a}{c}, \frac{b}{c}) \in \mathbb{A}^2 \subset \mathbb{A}^2 \cup \mathbb{P}^1.$$

Starting from $F(a, b, c) = 0$ with the fact that F is homogeneous of degree d results in: $0 = \frac{1}{c^d} F(a, b, c) = F(\frac{a}{c}, \frac{a}{c}, 1)$. Then for a new (non-homogeneous) polynomial $f(x, y)$ defined by $f(x, y) = F(x, y, 1)$, we get a map:

$$\{[a, b, c] \in C : c \neq 0\} \to \{(x, y) \in \mathbb{A}^2 : f(x, y) = 0\}$$
$$[a, b, c] \mapsto (\frac{a}{c}, \frac{b}{c})\,.$$

It is easy to see that this map is one-to-one, since if $(r, s) \in \mathbb{A}^2$ satisfies the equation $f(x, y) = 0$, then $[r, s, 1] \in C$. The curve $f(x, y) = 0$ in $\mathbb{A}^2$ is called the *affine part* of the projective curve C.
On the other hand consider the points $[a, b, c]$ on C with $c = 0$. The points $[a, b, 0]$ on C satisfy the equation $F(X, Y, 0) = 0$, and they are mapped to the points at infinity $[a, b] \in \mathbb{P}^1$ in $\mathbb{A}^2 \cup \mathbb{P}^1$. It can be shown that these points, which are directions in $\mathbb{A}^2$, correspond to the limiting tangent directions of the affine curve $f(x, y) = 0$.

2.1.2 Elliptic Curves

Definition 5 *A projective **Weierstrass equation** is a homogeneous equation of degree 3 of the form*

$$E : Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3, \tag{2.1}$$

*where $a_1, a_2, a_3, a_4, a_6 \in \overline{\mathbb{F}}$. The Weierstrass equation is said to be **smooth** or **non-singular** if for all projective points $P=(X,Y,Z) \in \mathbb{P}^2(\overline{\mathbb{F}})$ satisfying*

$$F(X, Y, Z) = Y^2Z + a_1XYZ + a_3YZ^2 - X^3 - a_2X^2Z - a_4XZ^2 - a_6Z^3 = 0, \tag{2.2}$$

at least one of three partial derivatives $\frac{\partial F}{\partial X}, \frac{\partial F}{\partial Y}, \frac{\partial F}{\partial Z}$ *is non-zero at P. If all three partial derivatives are equal to zero in some point P, then P is called a* ***singular point****, and the Weierstrass equation is said to be* ***singular****.*

Definition 6 *An* ***elliptic curve*** *E is the set of all solutions in* $\mathbb{P}^2(\overline{\mathbb{F}})$ *of a smooth Weierstrass equation. There is exactly one point in E with Z-coordinate equal to 0, namely (0:1:0). We call this point the* ***point at infinity*** *and denote it by* ∞*.*

The Weierstrass equation for an elliptic curve using non-homogeneous (affine) coordinates $x = \frac{X}{Z}, y = \frac{Y}{Z}$, is:

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6. \tag{2.3}$$

An elliptic curve E is then the set of solutions to equation (2.3) in the affine plane $\mathbb{A}^2(\overline{\mathbb{F}}) = \overline{\mathbb{F}} \times \overline{\mathbb{F}}$, together with the extra point at infinity ∞. If $a_1, a_2, a_3, a_4, a_6 \in \mathbb{F}$, then E is said to be *defined over* $\mathbb{F}$, and we denote this by $E/\mathbb{F}$. If E is defined over $\mathbb{F}$, then the set of $\mathbb{F}$-rational points of E, denoted $E/\mathbb{F}$, is the set of points both of whose coordinates lie in $\mathbb{F}$, together with the point ∞.
Example. Figure 2.1 represents the graphs over $\mathbb{R}$ of the curves $y^2 = x^3 - 4x + 1$ and $y^2 = x^3 - 5x + 5$.

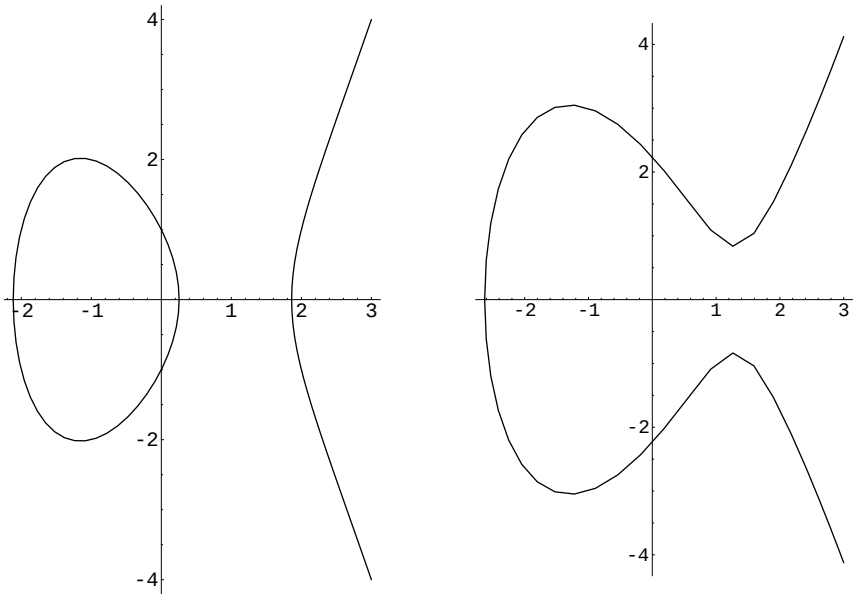


Figure 2.1: Graphs of elliptic curves $y^2 = x^3 - 4x + 1$ (on the left) and $y^2 = x^3 - 5x + 5$ (on the right) over $\mathbb{R}$.

Two elliptic curves are said to be *isomorphic* if they are isomorphic as projective varieties [158]. In short, two projective varieties V_1, V_2 defined over a field $\mathbb{F}$ are isomorphic over $\mathbb{F}$ if there exist morphisms $\Phi : V_1 \to V_2$, $\Psi : V_2 \to V_1$ (both defined over $\mathbb{F}$) such that $\Phi \circ \Psi$ and $\Psi \circ \Phi$ are the identity maps on V_2, V_1 respectively. For a more detailed mathematical introduction of varieties see the book of Silverman [209].
This notion of isomorphism is related to the coefficients of the Weierstrass equation that defines the curve by following result:

Theorem 2 ([158], p. 16) *Two elliptic curves $E_1/\mathbb{F}$ and $E_2/\mathbb{F}$ given by the equations*

$$E_1 : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$
$$E_2 : y^2 + b_1xy + b_3y = x^3 + b_2x^2 + b_4x + b_6$$

are isomorphic over $\mathbb{F}$, denoted $E_1/\mathbb{F} \cong E_2/\mathbb{F}$, if and only if there exists $u, r, s, t \in \mathbb{F}, u \neq 0$, such that the change of variables

$$(x, y) \to (u^2x + r,\ u^3y + u^2sx + t) \tag{2.4}$$

transforms equation E_1 to equation E_2.

The change of variables (2.4) is called an *admissible change of variables*. These changes of variables imply that all results for one type of curve immediately follow for all curves isomorphic to it. Therefore it is possible to simplify the equation (2.3) for some specific fields which are mostly used in elliptic curve cryptography. The following quantities are associated with the curve E (2.3):

$$\begin{aligned} d_2 &= a_1^2 + 4a_2 \\ d_4 &= 2a_4 + a_1a_3 \\ d_6 &= a_3^2 + 4a_6 \\ d_8 &= a_1^2a_6 + 4a_2a_6 - a_1a_3a_4 + a_2a_3^2 - a_4^2 \\ c_4 &= d_2^2 - 24d_4 \\ \Delta &= -d_2^2d_8 - 8d_4^3 - 27d_6^2 + 9d_2d_4d_6 \\ j(E) &= c_4^3/\Delta\,. \end{aligned}$$

The quantity Δ is called the *discriminant* of the Weierstrass equation and $j(E)$ is called the *j-invariant* of E if $\Delta \neq 0$ [158].

Theorem 3 ([158], p. 19) *E is an elliptic curve, i.e., the Weierstrass equation is non-singular, if and only if $\Delta \neq 0$.*

Theorem 4 ([158], p. 19) *If two elliptic curves $E_1/\mathbb{F}$ and $E_2/\mathbb{F}$ are isomorphic over $\mathbb{F}$, then $j(E_1) = j(E_2)$. The converse is also true if $\mathbb{F}$ is an algebraically closed field.*

In the following we consider several admissible changes of variables that result in simplified curve equations for characteristic other than 2 or 3 (so suitable for a prime field $\mathbb{F}_p$) and for characteristic 2 (for a field $\mathbb{F}_{2^n}$). We follow the approach as in the book by Menezes [158].

Let $E/\mathbb{F}$ be an elliptic curve given by equation (2.3). If $char(\mathbb{F}) \neq 2$, then the admissible change of variables

$$(x, y) \rightarrow (x, y - \frac{a_1}{2}x - \frac{a_3}{2})$$

transforms $E/\mathbb{F}$ to the curve

$$E'/\mathbb{F} : y^2 = x^3 + b_2x^2 + b_4x + b_6\,,$$

which is isomorphic to $E/\mathbb{F}$.
If $char(\mathbb{F}) \neq 2, 3$, then the admissible change of variables $(x, y) \rightarrow (\frac{x-3b_2}{36}, \frac{y}{216})$ transforms E' to the curve: $E''/\mathbb{F} : y^2 = x^3 + ax + b$. And again, $E' \cong E''$ which implies $E \cong E''$ over $\mathbb{F}$ because the relation of isomorphism is an equivalence relation. Therefore, for $char(\mathbb{F}) \neq 2, 3$, we can assume that $E/\mathbb{F}$ has the form:

$$E : y^2 = x^3 + ax + b, \quad a, b \in \mathbb{F}\,. \tag{2.5}$$

The condition $4a^3 + 27b^2 \neq 0$ is required for E to be non-singular. Hence, an elliptic curve E over $\mathbb{F}$ (where $char(\mathbb{F}) \neq 2, 3$) is the set of all solutions to the equation (2.5) where $a, b \in \mathbb{F}$ and $4a^3 + 27b^2 \neq 0$, together with a special point ∞ called the point at infinity.
Let $\mathbb{F}$ be a field of characteristic 2. After applying a similar admissible change of variables we end up with one of two possible equations.

- An elliptic curve E of zero j-invariant over $\mathbb{F}$ is the set of all solutions $(x, y) \in \overline{\mathbb{F}} \times \overline{\mathbb{F}}$ of the equation:

  $$E : y^2 + cy = x^3 + ax + b\,, \tag{2.6}$$

  where $a, b, c \in \mathbb{F}, c \neq 0$, together with the point at infinity.

- An elliptic curve E of non-zero j-invariant over $\mathbb{F}$ is the set of all solutions $(x, y) \in \overline{\mathbb{F}} \times \overline{\mathbb{F}}$ of the equation:

  $$E : y^2 + xy = x^3 + ax^2 + b\,, \tag{2.7}$$

  where $a, b \in \mathbb{F}, b \neq 0$, together with the point at infinity.

In both cases, E is an additive abelian group with the point at infinity as neutral element. The curves that satisfy Equation (2.6) are called the *supersingular curves*.

2.1.3 Group Law

Before we introduce a public key cryptosystem based on elliptic curves, we describe the elliptic curve group and define a discrete logarithm problem which is similar to the DLP for an arbitrary cyclic group.
Let E be an elliptic curve given by a Weierstrass equation, $E \subset \mathbb{P}^2$ consisting of the points $P = (x, y)$ together with the point at infinity $\infty = [0, 1, 0]$. In order to define a group operation we use the fact that any line $p \subset \mathbb{P}^2$ intersects E at exactly 3 points (counting multiplicities). This fact is a special case of Bézout's theorem (see Fulton [78]).
Composition Law ([209], p. 55): Let P,Q be any two points (not necessarily distinct) on E. Let p be the line through P and Q and R be the third point on p. Let $\overline{p}$ be the line through R and ∞ and let $\overline{R}$ be the third point of intersection on $\overline{p}$. Define the law of composition $+_E$ to be:

$$P +_E Q = \overline{R}.$$

Remark: This composition law is referred to as the "tangent-chord method"; it is demonstrated in Figs. 2.2 and 2.3 for an elliptic curve over the set of real numbers $\mathbb{R}$.

Proposition 1 *The operation $+_E$ is well-defined and makes E into an abelian group with neutral element ∞. The inverse of an element is the third point of intersection of the line through that element and ∞.*

Proofs using geometry or algebra are both available in the book of Silverman [209]. From now on, we write the group operation using ordinary '+'instead of '$+_E$'. So, we refer to the group, set of points, curve and equation, all as E. For a point P on a curve E over $\mathbb{F}_q$, $\langle P \rangle$ denotes a cyclic group generated by P.

2.1.4 Elliptic Curves over Finite Fields

For cryptography we need a finite cyclic group in which the group operation is efficiently computable, but the discrete logarithm problem is very difficult to solve. Elliptic curve groups appear to meet these criteria when the underlying field is finite.
Elliptic curves that are used in most applications are defined over $\mathbb{F}_q$ with $q = p^m$ where p is a prime number. In standards such as IEEE [111] and ANSI [5] fields for $q = p$ and $q = 2^n$ where $p \approx 2^n$ and $n \geq 160$ are recommended. Also, it appears that elliptic curve systems over both prime ($\mathbb{F}_p$) and binary ($\mathbb{F}_{2^n}$) fields provide the same level of security but fields $\mathbb{F}_{2^n}$ have some implementation advantages. Namely, arithmetic in $\mathbb{F}_{2^n}$ can be implemented more efficiently than arithmetic in $\mathbb{F}_p$, at least on platforms whithout specialized arithmetic co-processors. However, if an RSA accelerator is available, it makes sense to choose a prime field ECC.

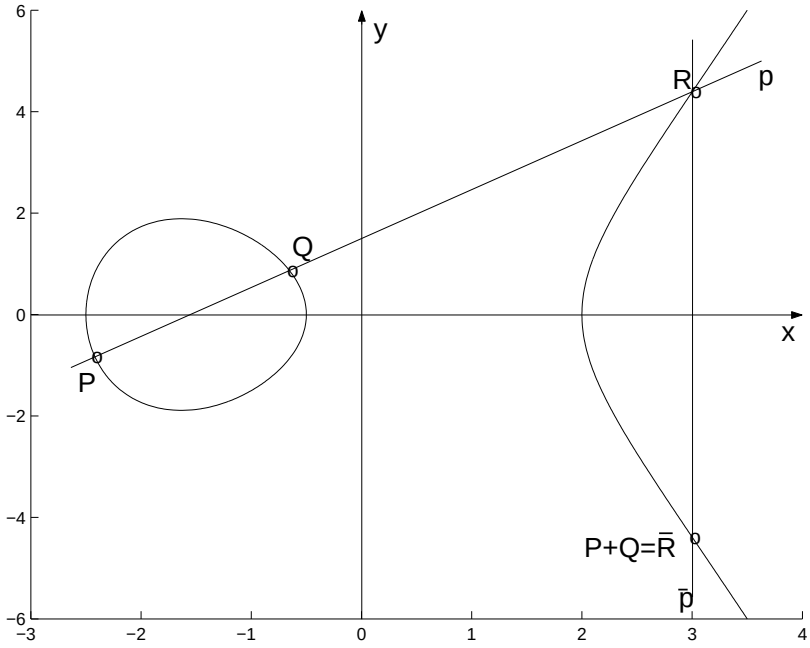


Figure 2.2: Addition of two different two points on an elliptic curve.

Namely, in this case both cryptosystems rely on integer modular arithmetic and hardware components can be shared as we show in Chapter 3.

The choice of a "suitable" parameter n in $\mathbb{F}_{2^n}$ for ECC is also an active research topic. The so-called composite fields have some advantages for implementations. For example, a part of the arithmetic in a subfield can be precalculated and stored in a memory to boost performance. However, with respect to cryptographic security it is typically recommended to use fields $\mathbb{F}_{2^p}$ where p is a prime. For example, it was shown by Gaudry *et al.* [87] that composite fields are "weaker" in the sense that in that case the ECDLP is sometimes easier than in a general case. Also, some recent results of Menezes *et al.* [154] exhibit weaknesses for the cases when $n \equiv 0 \pmod 5$ and $n \equiv 0 \pmod 6$.

2.1.5 Elliptic Curve Cryptosystems

The security of many public-key cryptosystems relies on the presumed intractability of the discrete logarithm problem in some group $\mathcal{G}$. Diffie and Hellman proposed already in 1976 the use of discrete logarithms in finite fields as a one-way function (cf. Chapter 1).

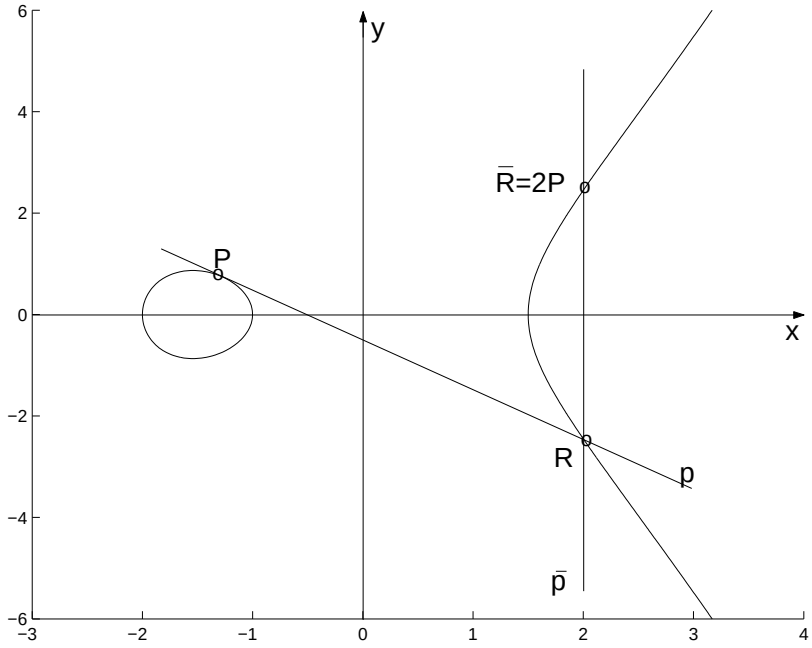


Figure 2.3: Doubling of a point on an elliptic curve.

In order to provide a suitable group for cryptography, the discrete logarithm problem should be computationally infeasible but the group operation and the group representation should be efficient. The set of rational points on an elliptic curve is an example of a group that satisfies these criteria.

Definition 7 *Consider the cyclic group $\mathbb{F}_q^*$ where α is a generating element of the group. For x, $0 \leq x < q-1$, let $\beta = \alpha^x \in \mathbb{F}_q^*$. The problem of finding the logarithm x for given α and β is called the* ***discrete logarithm problem (DLP)****. (Here $\mathbb{F}_q^*$ denotes the multiplicative group of a field $\mathbb{F}$.)*

For the group of points on an elliptic curve we have the following problem that is believed to be hard [29].

Definition 8 *Let E be an elliptic curve over $\mathbb{F}_q$ and let $P \in E$ be a point of order k i.e. ord(P) = k. Let $Q \in \langle P \rangle$ and $Q = \alpha P$ for α, $0 \leq \alpha < k$. The problem of finding the logarithm α for given P and Q is called the* ***elliptic curve discrete logarithm problem (ECDLP)****.*

The number of rational points on a curve E over a finite field $\mathbb{F}_q$ is denoted by $\#E(\mathbb{F}_q)$. This number is finite as follows from the following theorem.

Theorem 5 ***(Hasse)*** *Let E be an elliptic curve over $\mathbb{F}_q$. Then*

$$q + 1 - 2\sqrt{q} \leq \ \#E(\mathbb{F}_q) \leq q + 1 + 2\sqrt{q} \tag{2.8}$$

The quantity t, defined by $\#E(\mathbb{F}_q) = q+1-t$ is called the *trace of Frobenius* ([29], p. 34). Hasse's theorem implies $|t| \leq 2\sqrt{q}$.
The algorithms that attempt to solve the ECDLP are exponential in time for a general curve. However, an elliptic curve E for a cryptosystem needs to satisfy some conditions. In short, an elliptic curve E whith group order $\#\ E(\mathbb{F}_q) = n$, should meet the following requirements for cryptographic applications [29]:

1. The group should have a subgroup of large prime order r, where the exact size is determined by the the current progress in square root attacks. This means that one can choose a prime r of more than 160 bits, which is comparable to the security of conventional public-key cryptosystems with key lengths of at least 1000 bits.

2. The curve E should not be an anomalous curve. An elliptic curve defined over the field $\mathbb{F}_q$ is called *anomalous curve* if it has exactly q points ($t = 1$ in this case).

3. The smallest l for which $q^l \equiv 1 (\mathrm{mod}\, n)$, where $gcd(n, q) = 1$, should be large. This excludes curves of traces $t = 0$ and $t = 2$ and the other supersingular curves.

In short, one should always consider the case where the point P has a sufficiently large prime order and a curve should not fall into any of those weak cases. Then, the only algorithms to solve the ECDLP time are Shanks' baby-step-giant-step method and the methods of Pollard [29, 155]. For an elliptic curve E that is defined over a finite field $\mathbb{F}_q$ the complexity of those algorithms is of about $\sqrt{q}$ elliptic curve operations *i.e.* one usually writes that the complexity is of $\mathcal{O}(\sqrt{\#E(\overline{\mathbb{F}}_q)})$.

Discrete-log cryptosystems can be modified to work in the group of points on an elliptic curve. For example, the Diffie-Hellman key agreement can be adapted as follows [30].
We note first that any random point of sufficiently high order on an elliptic curve E can be used as a key. It is however required that both parties agree in advance on a conversion method, which maps a point to an integer. For example, they can choose a map $f : \mathbb{F}_q \to \mathbb{N}$ that maps just one coordinate of a point to an integer (see Koblitz *et al.* [131]).
EC Diffie-Hellman Key Exchange: The protocol involves the following messages:

Alice A		Bob B
A	$\xrightarrow{x\,G}$	B
A	$\xleftarrow{y\,G}$	B

1. *One time setup:* Alice and Bob wish to agree on a secret key over an insecure channel. They first agree on the set of domain parameters ($\mathbb{F}_q$, E, n, h, G). Here, E is an elliptic curve over $\mathbb{F}_q$, G is a generating (publicly known) point in the elliptic curve group of order n and the integer h is called the cofactor. For the cofactor it holds:
$$\#E(\mathbb{F}_q) = h \cdot n\,.$$
Due to the security of the ECDLP as discussed before, one usually selects a curve for which $h \leq 4$.

2. *Protocol messages:*
$$\begin{array}{ll} A \rightarrow B : x\,G & (1) \\ B \rightarrow A : y\,G & (2) \end{array}$$

3. *Protocol actions:* Perform the following steps each time a shared key is required.
   (a) A chooses a random secret x, $1 \leq x \leq n-1$ and sends B message (1). A's public key is xG; A's private key is x.
   (b) B chooses a random secret y, $1 \leq y \leq n-1$ and sends A message (2).
   (c) B receives x and computes the shared key as $K = (x)\,y\,G = x\,y\,G$.
   (d) A receives y and computes the shared key as $K = (y)\,x\,G = x\,y\,G$.

So, they both end with the same point which is then the common key: $K = xy\,G$. An adversary Eve who's spying on Alice and Bob may have knowledge of $G, x\,G, y\,G$ but not of x or y. She wants to determine number $K = xy\,G$ and her task is called the "Diffie-Hellman problem for elliptic curves".

The idea of ElGamal encryption transforms to ECC as follows: Suppose that the set of messages has been represented by the set of the points on elliptic curve. Bob wants to send Alice a message $M \in E$ after they have already exchanged $x\,G$ and $y\,G$ as in Diffie-Hellman protocol.

1. *Encryption:* In order to send a message m, B should perform the following steps:
   (a) Obtain A's authentic public key $x\,G$.
   (b) Represent the message m as a point M on a curve.
   (c) Select a random integer l, $1 \leq l \leq n-1$. Here, $n = ord(G)$.
   (d) Compute $c = (lG, M + l(x\,G))$.
   (e) Send the ciphertext c to A.

2. *Decryption:* In order to decipher the plaintext m from c, A should do the following:
   (a) Use the private key x to compute $x\,(l\,G)$.

(b) Recover M by subtracting $x\,l\,G$ from the second point in the pair.

Another protocol which provides a digital signature, is the ECDSA protocol, which is performed as follows [116]:

ECDSA protocol:

The ECDSA is specified by an elliptic curve E defined over $\mathbb{F}_q$ and a publicly known point $G{\in}E$ of prime order n. A private key of Alice is a scalar x and the corresponding public key is $Q{=}x\,G{\in}E$. The ECDSA requires a hash function which is a computationally efficient function mapping binary strings of arbitrary length to binary strings of some fixed length, called hash-values as defined in Chapter 1.

A signature for message M is generated as follows:

1. *ECDSA Signature Generation:* In order to sign a message m, A should perform the following steps:
   (a) Select a random integer k, $1 \leq k \leq n-1$.
   (b) Compute $kG = (x_1, y_1)$.
   (c) Compute $r = x_1 \mod n$. If $r = 0$ then go back to the step 1.
   (d) Compute $s = k^{-1}(h(M) + x \cdot r) \mod n$, where h is a hash algorithm. If $s = 0$, then go to step 1.
   (e) A's signature for the message M is the pair (r,s).
2. *ECDSA Signature Verification:* In order to verify A's signature on m, B performs the following steps:
   (a) Obtain an authenticated copy of A's public key Q.
   (b) Verify that r and s are integers in the interval $[1, n-1]$.
   (c) Compute $w = s^{-1} \mod n$ and $h(M)$.
   (d) Compute $u_1 = h(M)\,w \mod n$ and $u_2 = rw \mod n$.
   (e) Compute $u_1G + u_2Q = (x_0, y_0)$ and $v = x_0 \mod n$.
   (f) Accept the signature if and only if $v = r$.

The parameter m has to be at least 160 bits long; with that size a security level similar to that of the DSA is obtained. In this case DSA and ECDSA signatures have the same bitlength (320 bits).

2.2 Hyperelliptic Curve Cryptography (HECC)

We now present the mathematical background on hyperelliptic curves including the algorithms for efficient arithmetic in the Jacobian group. Most of the material presented here can be found in [157]. For other references on hyperelliptic curves the reader is referred to the work of Koblitz [128, 130]. The theory of algebraic curves can be found in the book of Fulton [78].

2.2.1 Basic Definitions and Properties

Hyperelliptic curves are a special class of algebraic curves; they can be viewed as as generalization of elliptic curves. Namely, a hyperelliptic curve of genus $g = 1$ is an elliptic curve, while in general, hyperelliptic curves can be of any genus $g \geq 1$.

Definition 9 *Let $\mathbb{F}$ be a field and $\overline{\mathbb{F}}$ be the algebraic closure of $\mathbb{F}$. A **hyperelliptic curve C of genus g** over $\mathbb{F}$ ($g \geq 1$) is given by an equation of the form*

$$C : v^2 + h(u)v = f(u) \quad in \quad \mathbb{F}[u, v]\,, \tag{2.9}$$

where $h(u) \in \mathbb{F}[u]$ is polynomial of degree at most g ($deg(h) \leq g$), $f(u) \in \mathbb{F}[u]$ is a monic polynomial of degree 2g+1 ($deg(f) = 2g + 1$), and there are no singular points.
*A **singular point** on C is a solution $(u, v) \in \overline{\mathbb{F}} \times \overline{\mathbb{F}}$ that simultaneously satisfy the equation (2.9) and the equations:*

$$\begin{array}{ll} 2v + h(u) & = 0 \\ h'(u)v - f'(u) & = 0. \end{array} \tag{2.10}$$

Definition 10 *Let $\mathbb{L}$ be an extension field of $\mathbb{F}$. The set of **$\mathbb{L}$-rational points on C**, denoted $C(\mathbb{L})$, is the set of all points $P = (x, y) \in \mathbb{L} \times \mathbb{L}$ which satisfy the equation (2.9) of the curve C, together with a special **point at infinity** denoted ∞. The set of points $C(\overline{\mathbb{F}})$ is denoted by C. The points in C other than ∞ are called **finite points**.*

Definition 11 *Let $P = (x, y)$ be a finite point on a curve C. The **opposite** of P is the point $\widetilde{P} = (x, -y - h(x))$. If a finite point P satisfies $P = \widetilde{P}$ then the point is said to be **special**; otherwise, the point is said to be **ordinary**. The opposite of ∞ is defined to be $\infty = \widetilde{\infty}$ itself.*

Example. Consider a hyperelliptic curve $C : y^2 = x(x-1)(x-2)(x-4)(x-5)$ over the set of real numbers. It intersects the x-axes in exactly five points and the graph is shown in Fig. 2.4.

2.2.2 Polynomial and Rational Functions on a Hyperelliptic curve

Definition 12 *The **coordinate ring of C over** $\mathbb{F}$, denoted $\mathbb{F}[C]$ is the quotient ring*

$$\mathbb{F}[C] = \mathbb{F}[u, v]/(v^2 + h(u)v - f(u))\,,$$

*where $(v^2 + h(u)v - f(u))$ denotes the ideal in $\mathbb{F}[u, v]$ generated by the polynomial $v^2 + h(u)v - f(u)$. Similarly, the **coordinate ring of C over** $\overline{\mathbb{F}}$ is defined as*

$$\overline{\mathbb{F}}[C] = \overline{\mathbb{F}}[u, v]/(v^2 + h(u)v - f(u))\,.$$

*An element of $\overline{\mathbb{F}}[C]$ is called a **polynomial function** on C.*

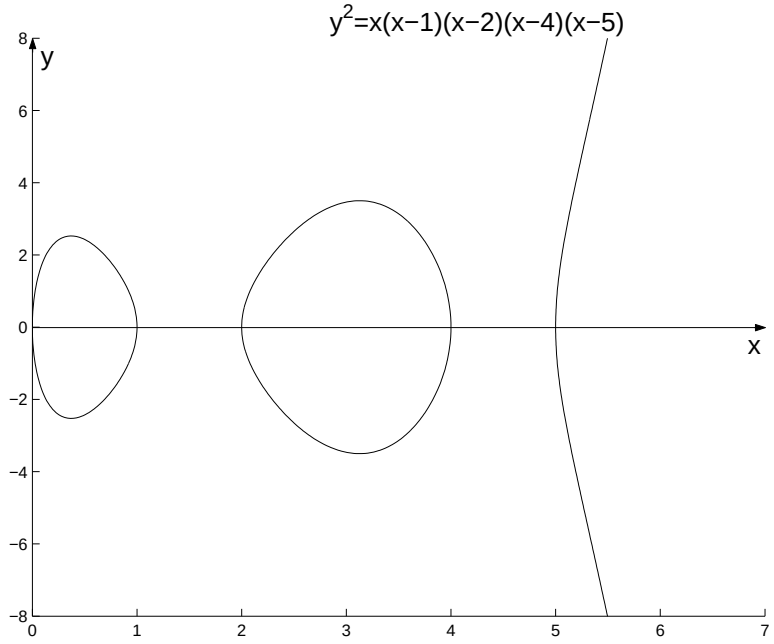


Figure 2.4: Hyperelliptic curve $y^2 = x(x-1)(x-2)(x-4)(x-5)$ over $\mathbb{R}$.

Lemma 1 [157] *The polynomial $r(u,v) = v^2 + h(u)v - f(u)$ is irreducible over $\overline{\mathbb{F}}$, and hence $\overline{\mathbb{F}}[C]$ is a commutative ring without divisors of zero i.e. an integral domain. (In other words a ring is called an integral domain if it is a commutative ring with identity $e \neq 0$ in which $ab = 0 \Longrightarrow a = 0 \vee b = 0$.)*

For each polynomial function $G(u,v) \in \overline{\mathbb{F}}[C]$, v^2 can be replaced by $f(u) - h(u)v$, to obtain a representation

$$G(u,v) = a(u) - b(u)v\,, \tag{2.11}$$

where $a(u)$, $b(u) \in \overline{\mathbb{F}}[u]$. This representation is also unique.

Definition 13 *Let $G(u,v) = a(u) - b(u)v$ be a polynomial function in $\overline{\mathbb{F}}[C]$. The **conjugate** of $G(u,v)$ is defined to be the polynomial function $\overline{G}(u,v) = a(u) + b(u)(h(u) + v)$.*

Definition 14 *The **function field** $\mathbb{F}(C)$ ($\overline{\mathbb{F}}(C)$) of C over $\mathbb{F}$ ($\overline{\mathbb{F}}$) is the field of fractions of $\mathbb{F}[C]$ ($\overline{\mathbb{F}}[C]$). The elements of $\overline{\mathbb{F}}[C]$ are called **rational functions** on C.*

It is easy to see that $\overline{\mathbb{F}}[C]$ is a subring of $\overline{\mathbb{F}}(C)$, *i.e.* every polynomial can be viewed as a rational function.

2.2.3 Zeros and Poles of Rational Functions

Definition 15 *Let $R \in \overline{\mathbb{F}}(C)^*$ (a non-zero rational function) and $P \in C$. If $R(P) = 0$ then R is said to have a **zero** at P. If R is not defined at P then R is said to have a **pole** at P; in this case we write $R(P) = \infty$.*

Theorem 6 *Let $P \in C$. Then there exists a function $U \in \overline{\mathbb{F}}(C)$ with $U(P) = 0$ such that the following property holds: for each polynomial function $G \in \overline{\mathbb{F}}(C)^*$, there exist an integer d and function $S \in \overline{\mathbb{F}}(C)$ such that $S(P) \neq 0, \infty$ and $G = U^d S$. Furthermore, d does not depend on the choice of U. The function U is called a **uniformizing parameter** for P.*

Definition 16 *Let $G \in \overline{\mathbb{F}}[C]^*$ and $P \in C$. Let $U \in \overline{\mathbb{F}}(C)$ be a uniformizing parameter for P, and write $G = U^d S$ where $S \in \overline{\mathbb{F}}(C)$, $S(P) \neq 0, \infty$. The **order of G at P** is defined to be $ord_P(G) = d$.*

Lemma 2 *Let $G \in \overline{\mathbb{F}}[C]^*$ and $P \in C$. Then the order of G at P is equal to the order of its conjugate $\overline{G}$ in $\widetilde{P}$ (the opposite of P) i.e. $ord_P(G) = ord_{\widetilde{P}}(\overline{G})$.*

Theorem 7 *Let $G \in \overline{\mathbb{F}}[C]^*$. Then G has a finite number of zeros and poles. Moreover, $\sum_{P \in C} ord_P(G) = 0$.*

Definition 17 *Let $R = G/H \in \overline{\mathbb{F}}(C)^*$ and $P \in C$. The order of R at P is defined as $ord_P(R) = ord_P(G) - ord_P(H)$.*

2.2.4 Divisors

Here we introduce the divisors and the Jacobian of a hyperelliptic curve.

Definition 18 *A **divisor D** is a finite formal sum of points in C*

$$D = \{D = \sum_{P_i \in C} m_i P_i, \quad m_i \in \mathbb{Z}, m_i = 0 \quad \text{for almost all} \quad i\}.$$

*The **degree of D**, denoted $\deg D$, is the integer $\deg D = \sum_{P_i \in C} m_i$. Also, $ord_{P_i}(D) = m_i$ is called the order of the point P_i for divisor D.*

Definition 19 *Let $R \in \overline{\mathbb{F}}(C)^*$ be a non-zero rational function. The **divisor of** R is*

$$div(R) = \sum_{P_i \in C} (ord_{P_i} R) P_i .$$

The set of all divisors, denoted $\mathbb{D}$, forms an additive group under the addition rule. The set of all divisors of degree 0, denoted $\mathbb{D}^0$ is a subgroup of $\mathbb{D}$.

Definition 20 *A divisor $D \in \mathbb{D}^0$ is called a **principal divisor** if $D = div(R)$ for some rational function $R \in \overline{\mathbb{F}}(C)^*$. The set of all principal divisors, denoted $\mathbb{P}$, is a subgroup of $\mathbb{D}^0$. The quotient group $\mathbb{J} = \mathbb{D}^0/\mathbb{P}$ is called the **Jacobian** of the curve C. We call D_1 and D_2 equivalent divisors and write $D_1 \sim D_2$ if $D_1 - D_2 \in \mathbb{P}$; for $D_1, D_2 \in \mathbb{D}^0$.*

The Jacobian $\mathbb{J}$ is a finite quotient group and every element on the Jacobian is an equivalence class of divisors. It can be shown that every element of the Jacobian can be uniquely represented by a so-called reduced divisor [157]. In practice, the Mumford representation according to which each divisor is represented as a pair of polynomials $[u, v]$ is usually used. Here, u is monic of degree 2, $deg(v) < deg(u)$ and $u|f - hv - v^2$ (so-called reduced divisors). More details can be found in the paper of Lange [138].

2.2.5 Hyperelliptic Curve Cryptosystems

Definition 21 *The following problem is called the Discrete Log Problem (DLP) on $\mathbb{J}(\mathbb{F})$: given two divisors $D_1, D_2 \in \mathbb{J}(\mathbb{F})$, determine an integer m, where $D_2 = mD_1$, if such an integer exists. This problem is also sometimes called **Hyperelliptic Curve Discrete-Log problem (HECDLP)**.*

The main operation in any hyperelliptic curve based primitive is scalar multiplication, *i.e.* mD where m is an integer and D is a reduced divisor in the Jacobian of some hyperelliptic curve C. The first algorithm for arithmetic in the Jacobian was invented by Cantor [46]. However, until "explicit formulae" were invented, the HECC was not considered a suitable alternative to EC based cryptosystems. For geni 2 and 3, there was substantial work on those formulae and algorithms for computing the group law on the Jacobian have been optimized. Several cases for coordinates from affine to various projective options have also been worked out. For references see papers from Lange [137, 138] and the work of Wollinger *et al.* [252, 182]. Algorithms for the group operation for the case of genus 2 hyperelliptic curves, which we used are due to Lange [138] and Byramjee and Duquesne [45]; they are given in Chapter 5 of the thesis.

As the HECDLP can be viewed as a generalization of ECDLP, almost all attacks on the ECDLP can be applied the HECDLP. However, there is also an instance of index-calculus attacks that is applicable to HECDLP [30]. As already mentioned a genus 2 curve allows for bit-lengths of HEC parameters that are a factor 2 shorter than those of ECC while obtaining the same security level. Higher-order genuses imply even shorter bit-lengths, which is very attractive for embedded implementations in constrained environments. However, some care with

the choice of a genus is necessary. The work of Thériault [223] showed that for genus 4 curves, the index-calculus provides a faster attack than a general case and the attack appears to be effective even for genus 3 curves. Even higher genuses were excluded already before for security reasons due to the work of Gaudry [86]. In this thesis we consider only hyperelliptic curves of genus 2 over binary fields.

2.3 Algorithms for Hyper/Elliptic Curve Cryptography

Here, we consider algorithms for elliptic curve cryptography over prime and binary fields; for hyperelliptic curves we are interested only in genus 2 curves over binary fields. We give the basic algorithms and the structure of the operations for curve-based cryptography; our modifications and improvements are explained in corresponding chapters (cf. Chapter 1).

The main operation in any curve-based primitive is the scalar multiplication. The hierarchical structure for operations required for implementations of curve-based cryptography is given in Fig. 2.5. Point/Divisor multiplication is at the top level. At the next (lower) level are the point/divisor group operations. The lowest level consists of finite field operations such as addition, subtraction, multiplication and inversion required to perform the group operations. The only difference between ECC and HECC is in the middle level that in this case consists of different sequences of operations. Those for HECC are a bit more complex when compared with the ECC point operation, but they use shorter operands.

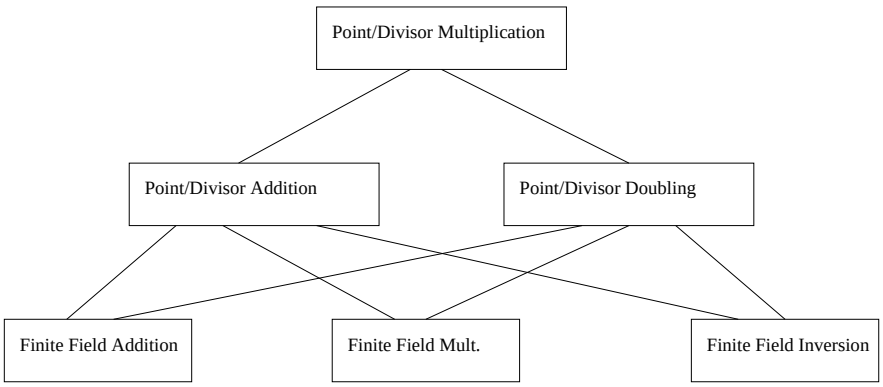


Figure 2.5: Scheme of the hierarchy for ECC/HECC operations.

2.3.1 Point/Divisor Multiplication

The point/divisor scalar multiplication is achieved by repeated point/divisor addition and doubling. All algorithms for modular exponentiation can also be applied for point multiplication. More details on algorithms for exponentiation are given in Chapter 3.

2.3.2 ECC Point Operations in $\mathbb{F}_p$

When E is a curve defined with equation (2.5), inverse of the point $P = (x_1, y_1)$ is $-P = (x_1, -y_1)$. The sum $P + Q$ of points $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ (assume that $P, Q \neq \infty$, and $P \neq \pm Q$) is point $R = (x_3, y_3)$ where:

$$\begin{aligned} \lambda &= \tfrac{y_2 - y_1}{x_2 - x_1} \\ x_3 &= \lambda^2 - x_1 - x_2 \\ y_3 &= (x_1 - x_3)\lambda - y_1. \end{aligned} \tag{2.12}$$

For $P = Q$, we get the following, "doubling" formulae:

$$\begin{aligned} \lambda &= \tfrac{3x_1^2 + a}{2y_1} \\ x_3 &= \lambda^2 - 2x_1 \\ y_3 &= (x_1 - x_3)\lambda - y_1. \end{aligned} \tag{2.13}$$

The point at infinity ∞ plays a role analogous to that of the number 0 in ordinary addition. Thus, $P + \infty = P$ and $P + (-P) = \infty$ for all points P.

There are many types of coordinates in which an elliptic curve may be represented. In the equations above affine coordinates are used, but so-called projective coordinates have some implementation advantages. The main conclusion is that point addition can be done in projective coordinates using only field multiplications, with no inversions required. Thus, inversions are deferred, and only one needs to be performed at the end of a point multiplication operation. A projective point (X, Y, Z) on the curve satisfies the homogeneous Weierstrass equation:

$$Y^2 Z = X^3 + aXZ^2 + bZ^3 \tag{2.14}$$

and, when $Z \neq 0$, it corresponds to the affine point $(X/Z, Y/Z)$. It appears that other projective representations result in more efficient implementations of the group operation [53]. In particular, a weighted projective representation (also referred to as Jacobian representation) is preferred in the sense of faster arithmetic on elliptic curves [29, 111]. In this representation a triplet (X, Y, Z) corresponds to the affine coordinates $(X/Z^2, Y/Z^3)$ for $Z \neq 0$. In this case we have a weighted projective curve equation of the form:

$$E : Y^2 = X^3 + aXZ^4 + bZ^6. \tag{2.15}$$

Weighted projective coordinates provide faster arithmetic than the "normal" projective coordinates. Conversion from projective to affine coordinates costs 1 inversion and 4 multiplications, while vice versa is trivial. If one implements addition and doubling in a way specified in the IEEE standard [111], the total costs for general addition is $1I + 3M$ in affine coordinates and $16M$ in projective coordinates ($11M$ if $Z_1 = 1$ *i.e.*, one point is given in affine coordinates, and the other one in projective coordinates). Here, I and M are denoting the modular inversion and multiplication operations, respectively. In the case of doubling (with $a = p - 3$), this relation is $1I + 4M$ in affine coordinates against $8M$ in projective coordinates. Thus, the choice of coordinates is determined by the ratio $I : M$. Therefore, multiplication in finite field is the most important operation to focus on when working with projective coordinates. On the other hand, the extra inverter is required for affine coordinates' representation because one inversion has to be performed for every point operation.

2.3.3 ECC Point Operations in $\mathbb{F}_{2^n}$

Here we consider a finite field of characteristic 2, *i.e.* $\mathbb{F}_{2^n}$. For the case of $char(\mathbb{F}) = 2$ we are only interested in curves that are non-supersingular. A non-supersingular elliptic curve E over $\mathbb{F}_{2^n}$ is defined as the set of solutions $(x, y) \in \mathbb{F}_{2^n} \times \mathbb{F}_{2^n}$ of the equation:

$$y^2 + xy = x^3 + ax^2 + b \tag{2.16}$$

where $a, b \in \mathbb{F}_{2^n}, b \neq 0$, together with ∞.

The point addition in affine coordinates is performed according to the following formulae. Let $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ be two points on an elliptic curve E. Assume $P_1, P_2 \neq \mathcal{O}$ and $P_1 \neq -P_2$. The sum $P_3 = (x_3, y_3) = P_1 + P_2$ is computed as follows ([29], p. 57):
If $P_1 \neq P_2$,

$$\begin{aligned} \lambda &= (y_2 + y_1) \cdot (x_2 + x_1)^{-1} \\ x_3 &= \lambda^2 + \lambda + x_1 + x_2 + a \\ y_3 &= \lambda(x_1 + x_3) + x_3 + y_1 \,. \end{aligned}$$

If $P_1 = P_2$,

$$\begin{aligned} \lambda &= y_1/x_1 + x_1 \\ x_3 &= \lambda^2 + \lambda + a \\ y_3 &= (x_1 + x_3)\lambda + x_3 + y_1 \,. \end{aligned}$$

Projective coordinates can be used also in this case to avoid the inversion in a binary field. We elaborate this in Chapter 4 in more detail.

2.3.4 HECC Divisor Arithmetic in $\mathbb{F}_{2^n}$ for Genus 2 Curves

The formulae for the addition and doubling are given in Table 2.1 and Table 2.2 respectively. The numbers in parenthesis correspond to the case of mixed addition.

The curve used is of a special form ($y^2 + xy = x^5 + f_1x + f_0$) that allows for more optimized formulae in the case of doubling [45].

2.3.5 Finite Field Arithmetic

The bottom level in the hierarchy of curve-based cryptography consists of finite field operations (see Fig. 2.5). Again, we distinguish between the two cases *i.e.* fields of odd characteristic ($\mathbb{F}_{p^m}$) and fields of characteristic 2 ($\mathbb{F}_{2^n}$). We discuss addition/subtraction and multiplication in $\mathbb{F}_p$ in Chapter 3 as the basic operations for both RSA and ECC over $\mathbb{F}_p$. On the other hand, binary field operations are explained in Chapter 4 which deals with ECC/HECC over $\mathbb{F}_{2^n}$. Furthermore, $\mathbb{F}_{p^m}$ operations are included in Chapter 8 as the basis for arithmetic of algebraic torus. Here we only give the way to calculate the inversion operation by means of Fermat's theorem. We will use these facts later in the remainder of the thesis.

Theorem 8 ([104], p. 63) *If p is prime, then*

$$a^p \equiv a(\mathrm{mod}p)\,. \tag{2.17}$$

Theorem 9 *Fermat's little theorem* ([104], p. 63) *If p is prime and $p \nmid a$, then*

$$a^{p-1} \equiv 1(\mathrm{mod}p)\,. \tag{2.18}$$

The previous two congruences are equivalent when $p \nmid a$; for $p|a$ equation (2.17) is trivial.

Theorem 10 *Fermat-Euler theorem* ([104], p. 63) *If $gcd(a, m) = 1$, then*

$$a^{\varphi(m)} \equiv 1(\mathrm{mod}m)\,. \tag{2.19}$$

It is easy to observe that Theorem 9 is a special case of Theorem 10.

2.4 Algebraic Tori

Here we give mathematical background for algebraic tori. Other details about public-key cryptosystems based on tori are given in Chapter 8.

The main purpose of torus based cryptography is to provide better efficiency for the same security. Namely, by shortening transmissions by a factor of $\frac{n}{\varphi(n)}$, one could achieve many benefits in hardware, as well as software implementations. More precisely, due to the fact that we are working in a subgroup one could use shorter exponents and shorter representations. In this way, actual savings could be achieved not just in bandwidth, but also in speed, memory, power usage, *etc.*

The XTR cryptosystem uses the subgroup of $\mathbb{F}^*_{p^6}$ of order $p^2 - p + 1 = \varphi_6(p)$. Transmissions consist of only two elements of $\mathbb{F}_p$, instead of six. These subgroups

can be regarded as algebraic tori and are the basis of the cryptosystems LUC and XTR.

Here we review how to construct public key cryptosystems based on the group of rational points of an algebraic torus defined over a finite field. Algebraic torus based cryptosystems result from an idea to compress elements of subgroups of $\mathbb{F}_{q^n}^*$. More precisely, the discrete log-based cryptography is performed by transmitting only $\varphi(n)$ elements of $\mathbb{F}_q$, instead of n elements of $\mathbb{F}_q$ [197].

Here, we denote a large finite field with $\mathbb{F}_q$, $(q \geq 2^{1024})$, where $q = p^n$ and p is a prime. Then $\mathbb{F}_q$ can be viewed as the extension field of $\mathbb{F}_p$ *i.e.* we can write: $\mathbb{F}_q = \mathbb{F}_p[t]/(f(t))$, where $f(t)$ is an irreducible polynomial of degree n with coefficients from $\mathbb{F}_p$. In other words $\mathbb{F}_q$ is an n-dimensional vector space over $\mathbb{F}_p$.

Let $g \in \mathbb{F}_p^*$ be an element of a large multiplicative order l $(l \geq 2^{160})$. The multiplicative group $\mathbb{F}_p^*$ of nonzero elements of $\mathbb{F}_p$ is cyclic and a generator of the cyclic group $\mathbb{F}_p^*$ is called a primitive element of $\mathbb{F}_p$. For such g we write: $< g >= \mathbb{F}_p^*$.

Lemma 3 *Let $\mathbb{F}_{q^k}$ be an extension field of the field $\mathbb{F}_q$. For $a \in \mathbb{F}_{q^k}$, we have $a \in \mathbb{F}_q$ iff $a^q = a$.*

Definition 22 *For $x \in \mathbb{F}_{q^k}$ the trace $Tr_{\mathbb{F}_{q^k}/\mathbb{F}_q} : \mathbb{F}_{q^k} \longrightarrow \mathbb{F}_q$ of x over the field $\mathbb{F}_q$ is defined by:*

$$Tr_{\mathbb{F}_{q^k}/\mathbb{F}_q}(x) = x + x^q + \ldots + x^{q^{k-1}}. \tag{2.20}$$

This linear mapping is the sum of the conjugates of x with respect to $\mathbb{F}_q$ and it maps every element of $\mathbb{F}_{q^k}$ to the subfield $\mathbb{F}_q$. Another interesting mapping from a finite field to a subfield is the norm function.

Definition 23 *For $x \in \mathbb{F}_{q^k}$ the norm function $N_{\mathbb{F}_{q^k}/\mathbb{F}_q} : \mathbb{F}_{q^k} \longrightarrow \mathbb{F}_q$ of x over field $\mathbb{F}_q$ is defined by:*

$$N_{\mathbb{F}_{q^k}/\mathbb{F}_q}(x) = x \cdot x^q \cdot \ldots \cdot x^{q^{k-1}} = x^{\frac{q^k-1}{q-1}}. \tag{2.21}$$

The following property of the norm also holds:

$$N_{\mathbb{F}_{q^k}/\mathbb{F}_q}(xy) = N_{\mathbb{F}_{q^k}/\mathbb{F}_q}(x) \cdot N_{\mathbb{F}_{q^k}/\mathbb{F}_q}(y)$$

for all $x, y \in \mathbb{F}_{q^k}^*$.

Definition 24 *Let $\mathbb{F}_q$ be a field and n be a positive integer. The splitting field of $x^n - 1$ over a field $\mathbb{F}_q$ is called the **n-th cyclotomic field** over $\mathbb{F}_q$. The roots of $\zeta_n = e^{\frac{2\pi i}{n}}$ of $x^n - 1$ in the cyclotomic field are called the **n-th roots of unity** over $\mathbb{F}_q$.*

Let p be the characteristic of $\mathbb{F}_q$, such that $p \nmid n$ and ζ_n a primitive n-th root of unity over $\mathbb{F}_q$. Then the polynomial

$$\Phi_n(x) = \prod_{s=1,\ gcd(s,n)=1}^{n} (x - {\zeta_n}^s)$$

*is called the **n-th cyclotomic polynomial** over $\mathbb{F}_q$. The degree of $\Phi_n(x)$ is $\varphi(n)$. In this case when $p \nmid n$ the set of n-th roots of unity forms a cyclic group of order n with respect to the multiplication in the n-th cyclotomic field. It holds that $\Phi_n(x)$ is an irreducible polynomial in $\mathbb{Z}[x]$ and therefore the minimal polynomial of ζ_n.*

Theorem 11 *Let $\mathbb{F}_q$ be a field and n be a positive integer such that $p \nmid n$. Then:*

$$x^n - 1 = \prod_{k|n} \Phi_k(x),$$

and the coefficients of $\Phi_n(x)$ are from the prime subfield of $\mathbb{F}_q$.

Let $< g >= \mathbb{F}_q^*$, where $q = p^n$ and $h = g^x$, where $0 \leq x < q - 1$. In that case we get:

$$N_{\mathbb{F}_{p^n}/\mathbb{F}_p}(g) = g^{\frac{p^n-1}{p-1}} \tag{2.22}$$

From the property of the norm function it holds that $N_{\mathbb{F}_{p^n}/\mathbb{F}_p}(h) = (N_{\mathbb{F}_{p^n}/\mathbb{F}_p}(g))^x$. Hence, because $ord(N_{\mathbb{F}_{p^n}/\mathbb{F}_p}(g)) = p - 1$ (eq. (2.22)), using the norm can give solutions mod $p-1$. Therefore, to obtain the required level of security one should ensure that the equation $N_{\mathbb{F}_{p^n}/\mathbb{F}_p}(h) = (N_{\mathbb{F}_{p^n}/\mathbb{F}_p}(g))^x$ has only a trivial solution *i.e.* the norm of a generator g should be 1 with respect to every subfield. This condition leads to the term of **cyclotomic subgroup** $G_{p,n} \subseteq \mathbb{F}_{p^n}^*$. More precisely, we can write:

$$G_{p,n} = \{x \in \mathbb{F}_{p^n}^* | N_{\mathbb{F}_{p^n}/\mathbb{F}_{p^k}}(x) = 1,\ k|n,\ k \neq n\},$$

for all subfields $\mathbb{F}_p \subseteq \mathbb{F}_{p^k} \subset \mathbb{F}_{p^n}$. It also holds: $\#G_{p,n} = \Phi_n(p)$.
Now we give the definition of an algebraic torus as in [97]:

Definition 25 *Let $K = \mathbb{F}_p$ and $L = \mathbb{F}_{p^n}$. The **torus** T_n is the intersection of the kernels of the norm maps $N_{L/F}$, for all subfields $K \subseteq F \subset L$:*

$$T_n(K) := \bigcap_{K \subseteq F \subset L} Ker[N_{L/F}].$$

One can also define an algebraic torus as an algebraic group that over some larger field is a product of multiplicative groups. A field over which a torus becomes isomorphic to a product of multiplicative groups is called a splitting field for the

torus *i.e.* in that case the torus splits over that field [197]. In other words, the torus $T_n(\mathbb{F}_p)$ is the group of elements of $\mathbb{F}^*_{p^n}$ that have norm 1 down to every intermediate subfield $\mathbb{F}^*_{p^d}$, where $d \neq n$ [197].

We also introduce birational isomorphism between algebraic varieties (over some field $\mathbb{K}$), which we can here consider as the solution set of polynomial equations with coefficients from the field $\mathbb{K}$.

Definition 26 [197] *A **rational** map between algebraic varieties is a function defined by polynomials or quotients of polynomials that is defined almost everywhere. A **birational isomorphism** between algebraic varieties is a rational map that has a rational inverse. A d-dimensional variety is **rational** if it is birationally isomorphic to $\mathbb{A}^d$. (Here $\mathbb{A}^d$ is an affine d-space.)*

2.5 Security of Public-Key Cryptosystems

In this section we compare key-lengths for public-key cryptosystems that we mention in the thesis. Then we can also easily compare performances of different cryptosystems on the same platform.

Recommended key-lengths for RSA are at least 1024 bits currently. This estimate depends on the difficulty to factor integers and the current progress of factorization efforts (*e.g.* NFSNET). The security of 1024 bit RSA is usually compared to 160 bit ECC and with 80 bit of a symmetric key algorithm such as AES. However, Lenstra and Verheul estimated that w.r.t. computationally equivalent security 1024 and 1375 bit RSA are comparable to 139 and 160 bit ECC respectively [141]. On the other hand, for cost equivalent security they suggested slightly different corresponding bit-lengths. In [68] the numbers as in Table 2.3 were suggested. Here, the security level n means that $\mathcal{O}(2^n)$ operations are needed by the best known algorithms to break the system.

For the cryptosystems based on the DLP in finite field one should use the same key lengths as for RSA. For HECC of genus 2 over $\mathbb{F}_p$ the situation is the same as for ECC over finite fields of the form $\mathbb{F}_p$ or $\mathbb{F}_{p^2}$. This means that a security of 2^n operations is achieved with a key of $2n$ bits [67]. For torus based cryptosystems, required key lengths of 160 bits correspond to 1024 RSA keys.

Table 2.1: Formulae for the divisor addition.

Input	$D_1 = [U_{11}, U_{10}, V_{11}, V_{10}, Z_1]$, $D_2 = [U_{21}, U_{20}, V_{21}, V_{20}, Z_2]$	
Output	$2D = [U_1', U_0', V_1', V_0', Z']$	
Step	Calculations	# mult.
1	*Precomputation and resultant r:* $Z = Z_1 \cdot Z_2,\ \widetilde{U}_{21} = Z_1 \cdot U_{21}, \widetilde{U}_{20} = Z_1 \cdot U_{20},$ $\widetilde{V}_{21} = Z_1 \cdot V_{21}, \widetilde{V}_{20} = Z_1 \cdot V_{20},$ $t_1 = U_{11} \cdot Z_2 + \widetilde{U}_{21},\ t_2 = U_{10} \cdot Z_2 + \widetilde{U}_{20},$ $t_0 = U_{11} \cdot t_1 + t_2 \cdot Z_1,\ r = t_0 \cdot t_2 + t_1^2 \cdot U_{10}$	$12M(6M)$
2	*Compute almost s:* $t_4 = V_{10} \cdot Z_2 + \widetilde{V}_{20},\ t_5 = V_{11} \cdot Z_2 + \widetilde{V}_{21},$ $w_2 = t_0 \cdot t_4, w_3 = t_1 \cdot t_5;$ $s_1 = (t_0 + Z_1 \cdot t_1) \cdot (t_4 + t_5) + w_2 + w_3 \cdot (Z_1 + U_{11});$ $s_0 = w_2 + U_{10} \cdot w_3$	$8M(7M)$
3	*Precomputations:* $R = Z \cdot r, s_0 = s_0 \cdot Z, s_3 = s_1 \cdot Z,\ \widetilde{R} = R \cdot s_3;$ $S_3 = {s_3}^2, S = s_0 \cdot s_1,\ \widetilde{S} = s_3 \cdot s_1, \widetilde{\widetilde{S}} = s_0 \cdot s_3, \widetilde{\widetilde{R}} = \widetilde{R} \cdot \widetilde{S};$	$9M$
4	*Compute l:* $l_2 = \widetilde{S} \cdot \widetilde{U}_{21}, l_0 = S \cdot \widetilde{U}_{20}, l_1 = (\widetilde{S} + S) \cdot (\widetilde{U}_{21} + \widetilde{U}_{20})$ $+l_2 + l_0,;\ \ l_2 = l_2 + \widetilde{\widetilde{S}};$	$3M$
5	*Compute U':* ${U_0}' = {s_0}^2 + {s_1}^2 \cdot t_1 \cdot (t_1 + \widetilde{U}_{21}) + t_2 \cdot \widetilde{S} +$ $+R \cdot [t_1 \cdot r + s_1 \cdot Z];$ ${U_1}' = \widetilde{S} \cdot t_1 + R^2,\ l_2 = l_2 + {U_1}',$ $t_4 = {U_0}' \cdot l_2 + S_3 \cdot l_0,;\ \ t_5 = {U_1}' \cdot l_2 + S_3 \cdot ({U_0}' + l_1);$ $Z' = \widetilde{R} \cdot S_3, {U_1}' = \widetilde{R} \cdot {U_1}',\ \ {U_0}' = \widetilde{R} \cdot {U_0}';$	$17M$
6	*Compute V':* ${V_0}' = t_4 + \widetilde{\widetilde{R}} \cdot \widetilde{V}_{20};$ ${V_1}' = t_5 + \widetilde{\widetilde{R}} \cdot (\widetilde{V}_{21} + Z);$	$2M$
total		$51M(44M)$

Table 2.2: Formulae for divisor doubling in projective coordinates.

Input	$D = [U_1, U_0, V_1, V_0, Z]$	
Output	$2D = [U_1', U_0', V_1', V_0', Z']$	
Step	Calculations	# mult.
1	*Precomputation and resultant r:* $t_0 = Z^2, \;\; t_1 = {U_1}^2, \;\; r = U_0 \cdot Z$	$3M$
2	*Compute k:* $k_1 = f_3 \cdot t_0 + t_1, \;\; k_0 = U_1 \cdot k_1 + Z \cdot [t_0 + V_1 \cdot (Z + V_1)]$	$4M$
3	*Compute s:* $t_2 = k_0 \cdot U_1, s_1 = k_0 \cdot Z, s_0 = k_1 \cdot r + t_2$	$3M$
4	*Compute l:* $t_0 = t_0 \cdot r$, $r = t_0 \cdot s_1$, $t_1 = s_1 \cdot k_0$, $t_3 = U_0 \cdot k_0$, $l_2 = s_1 \cdot t_2$, $l_0 = s_0 \cdot t_3$, $l_1 = (t_2 + t_3) \cdot (s_0 + s_1) + l_2 + l_0$;	$7M$
5	*Compute U':* ${U_0}' = {s_0}^2 + r, \;\; {U_1}' = {t_0}^2$;	$2M$
6	*Precomputation:* $l_2 = l_2 + s_0 \cdot s_1 + {U_1}'$, $s_1 = {s_1}^2$, $t_2 = r \cdot t_1$, $t_0 = {U_1}' \cdot l_2 + l_0 \cdot s_1, \;\; t_1 = {U_1}' \cdot l_2 + s_1 \cdot ({U_0}' + l_1)$;	$7M$
7	*Adjust:* $Z' = s_1 \cdot r, \;\; {U_1}' = {U_1}' \cdot r$, ${U_0}' = {U_0}' \cdot r$;	$3M$
8	*Compute V':* ${V_0}' = t_0 + t_2 \cdot V_0$, ${V_1}' = t_1 + t_2 \cdot V_1 + Z'$;	$2M$
total		$31M$

Table 2.3: Comparison of the key lengths for RSA and ECC.

Security level	RSA	ECC
80	1248	160
112	2432	224
128	3248	256

Chapter 3

Hardware Design for RSA and Elliptic Curve Cryptosystems

This chapter presents a scalable hardware implementation of both commonly used public key cryptosystems, RSA and Elliptic Curve Cryptosystem on the same platform. The introduced hardware accelerator features a design which vary from very small targeting wireless applications, up to a very big design used for network security. The latter option can include a few dedicated large number arithmetic units; each of which is a systolic array performing the Montgomery Modular Multiplication (MMM). The bound on the Montgomery parameter R has been optimized to facilitate more secure ECC point operations. Furthermore, we present a new implementation option for the CRT (Chinese Remainder Theorem) scheme which we believe to be less vulnerable to side-channel attacks.

Our contribution deals with an FPGA implementation of RSA and ECC cryptosystems over a field of prime characteristic. The results were published in [19, 20, 9]. We used the systolic array to achieve arbitrary precision in bits [173, 176], hence easily bridging the gap between the bit-lengths for ECC from 160 bits to 2048 (or higher) bit long moduli for RSA. We use modular exponentiation based on Montgomery's method without any modular reduction achieving the best possible bound for the Montgomery parameter R. We also introduce the new bound on the Montgomery parameter for ECC which allows us to perform a very secure yet efficient point addition and doubling. We show that in the case of two or more arithmetic units a high level of parallelism can be achieved altering ECC operations between these units. The parallelism between more units and also between cells of the systolic array is beneficial for side-channel resistance. Moreover, we introduce a new variation of Garner's scheme for CRT decryption, which has a

built-in countermeasure against timing and power analysis based attacks.

Since the architecture was already used for RSA applications, it was natural to implement elliptic curve arithmetic in $\mathbb{F}_p$. All required components were already available, as ECC in $\mathbb{F}_p$ is based on ordinary modular arithmetic. If one uses projective coordinates, modular multiplication remains as the most time consuming operation for ECC. Hence, an efficient implementation relies on efficient modular multiplication, as is the case for RSA. Nevertheless, it is also important to focus on time-constant algorithms which are less likely to leak side-channel information. To sumarize, in this chapter we want to introduce a secure combined RSA-ECC implementation which as well meets high demands in speed implied by state of art for RSA hardware implementation.

3.1 RSA Cryptosystem

The security of the RSA cryptosystem is based on the difficulty of the RSA problem (cf. Chapter 1). It is still the most popular cryptosystem, especially for high-end devices that are typically used in e-commerce and Virtual Private Network (VPN) servers.

It is evident that modular exponentiation and also modular multiplication are the most important operations which have to be considered in detail.

3.1.1 Modular Exponentiation

The dominant cost operation in the RSA cryptosystem is modular exponentiation, namely computing $M^e \bmod N$. The basic technique for exponentiation is repeated square and multiply (see Knuth [125], p. 461). In [155] this method is called left-to-right binary exponentiation (Alg. 3.1). An exponent e is given here in the MSB form and by the radix 2 representation *i.e.* $e = \Sigma_{i=0}^{t-1} e_i 2^i$, $e_i \in \{0,1\}$. A similar algorithm is also used for point/divisor multiplication in ECC/HECC. In this case the analogous scheme is called double-and-add or the binary method (Alg. 3.2) [29].

Algorithm 3.1: Modular exponentiation

Require: $0 \leq M < N$, $0 < e < N$, $e = (e_{t-1}, e_{t-2}, \cdots e_0)_2$, $e_i \in \{0,1\}$ $e_{t-1} = 1$ and N
Ensure: $M^e \bmod N$
1: $A \leftarrow M$
2: **for** i from $t-2$ to 0 **do**
3: $A \leftarrow A \cdot A \bmod N$
4: If $e_i = 1$, then $A \leftarrow A \cdot M \bmod N$
5: **end for**
6: Return A

Algorithm 3.2: Point multiplication: Binary method

Require: A point P, a t-bit integer k, $k = (k_{t-1}, k_{t-2}, \cdots k_0)_2$, $k_i \in \{0, 1\}$
Ensure: $Q = kP$
1: $Q \leftarrow O$
2: **for** i from $t-1$ to 0 **do**
3: $\quad Q \leftarrow 2Q$
4: $\quad$ If $k_i = 1$, then $Q \leftarrow Q + P$
5: **end for**
6: Return Q

Numerous methods for speeding-up exponentiation and scalar multiplication have been proposed in the literature; for a survey see Gordon [93]. Recently, side-channel security is also considered to be an important factor for the choice of a suitable exponentiation algorithm. These methods are divided into three large groups: generic methods *i.e.*, methods which can be applied to any finite abelian group, exponent recoding techniques and special methods developed for particular operations (as in Gallant *et al.* [79]). We give a short description of these methods as they are also relevant for ECC/HECC scalar multiplication.

Generic Methods for Exponentiation

Here, we discuss the techniques: windowing, comb techniques and addition chains. The most frequently mentioned are various **windowing techniques**; they generalize the basic algorithm by processing more than one bit of the exponent is processed per iteration. The basic idea is as follows: the exponent is divided into digits (windows). Algorithm 3.1 can thus be considered as a special case where the window size is equal to 1. For the sliding window methods, these windows do not necessarily have the same length. The exponent is divided into zero and non-zero windows. In the work of Koç [47] different variants of the sliding window method are proposed. These methods also include a precomputation phase in which some powers of M are precalculated. The number of pre-calculations varies according to the window size. Considering different parameters various trade-offs are possible. A similar idea is used in the **comb techniques**, which precompute tables that depend on M (see Lim and Lee [146]). These methods are very useful for applications with fixed base M such as digital signature schemes. When the exponent is fixed, it is very useful to use **addition chains** [155]. In particular, for ECC the cost of point subtraction is the same as the cost for point additions, so it is possible to combine both operations for point multiplication on elliptic curves. The purpose of addition chains is to minimize the number of multiplications required for an exponentiation. A comprehensive and detailed discussion can be found in Knuth [125].

Exponent Recoding Techniques

Exponent recoding techniques replace the binary representation of an exponent with a representation which has fewer non-zero terms (see Gollmann *et al.* [91]). Many techniques for exponent recoding have been proposed in the literature. Here we mention the **signed-digit representation**. Consider an integer representation of the form $k = \sum_{i=0}^{l} s_i 2^i$, where $s_i \in \{-1, 0, 1\}$. This is called the (binary) *signed digit* (SD) representation (see Menezes *et al.* [155]). The representation is redundant. For example, the integer 3 can be represented as $(011)_2$ or $(10\bar{1})_2$, where $\bar{1} = -1$. It is said that an SD representation is *sparse* if it has no adjacent non-zero digits. A sparse SD representation is also called a *non-adjacent form* (NAF). Every integer k has a unique NAF which has the minimum weight of any signed digit representation of k.

Special Methods for Exponentiation

Many special techniques have been proposed which take advantage of some special property of the underlying field, or the elliptic curve etc. As an example, Walter introduced the MIST algorithm [241, 243], which is claimed to offer a good protection against power analysis-based attack without losing too much on efficiency. His method selects at random between different widths of windows. This idea seems to be a good solution for side-channel resistance but it is hard to implement it in hardware.

3.1.2 Montgomery's Arithmetic

Modular multiplication forms the basis of modular exponentiation which is the core operation of the RSA cryptosystem. It is also present in many other cryptographic algorithms including those based on ECC and HECC. The most popular algorithm for modular multiplication is Montgomery's method [164]. The approach of Montgomery avoids the time consuming trial division that is the common bottleneck of other algorithms. His method is proven to be very efficient; it is the basis of many implementations of modular multiplication in hardware as well as software. For the sake of completeness we give here all details for Montgomery's arithmetic, although only multiplication is required for RSA. However, ECC implementations benefit from other properties as well.

Let N be a modulus. For a word base $b = 2^\alpha$, the Montgomery radix (or parameter) R is typically chosen such that $R = 2^r = (2^\alpha)^n > N$. Let x be an odd integer represented by its radix b representation $x = \sum_{i=0}^{n-1} x_i {b_i}^i$. There is a one-to-one correspondence between each element $x \in \mathbb{Z}_\mathbb{N}$ and its representation $X = xR \bmod N$. This representation is usually referred to as the *Montgomery representation*. Sum and subtraction of two elements in Montgomery representation is again an element in Montgomery representation. This fact is also used for

our implementation of ECC in $\mathbb{F}_p$. For efficient implementation of modular multiplication the crucial operation is modular reduction. Therefore, we first introduce the Montgomery reduction.

Montgomery Reduction

Let N and R be as above, so $R > N$ and $gcd(N, R) = 1$ and let T be an integer such that $0 \leq T < NR$. A method to compute $TR^{-1} \bmod N$ that is given in Algorithm 3.3 is called the Montgomery reduction. We will first prove the following fact:

Lemma 4 ([155], p. 601) *Let R and N be an integers, such that $gcd(N, R) = 1$ and let $N' = -N^{-1} \bmod R$ and let T be an integer such that $0 \leq T < NR$. If $m = TN' \bmod R$ then it follows*

$$\frac{T + mN}{R} \equiv TR^{-1} \bmod N$$

and $\frac{T+mN}{R}$ is an integer.

Proof: It holds $T + mN \equiv T \bmod N$ and hence, $(T + mN)R^{-1} \equiv TR^{-1} \bmod N$, so the first claim is proved. From the assumptions above, we can write

$$\begin{aligned} m &= TN' + kR \\ NN' &= -1 + lR\,, \end{aligned}$$

for some integers k and l. Then it follows

$$\begin{aligned} \frac{T + mN}{R} &= \frac{T + (TN' + kR)N}{R} = \frac{T + TN'N + kNR}{R} \\ &= \frac{T + T(-1 + lR) + kNR}{R} = \frac{lTR + kRN}{R} = lT + kN\,, \end{aligned}$$

which is an integer. □

Montgomery's Modular Multiplication (MMM)

We consider here the Montgomery's product of two elements X and Y that are written in the Montgomery representation. The product of X and Y is the element Z for which $z = (xy) \bmod N$. More precisely, for $X = xR \bmod N$ and $Y = yR \bmod N$ the following equality holds:

$$\begin{aligned} Z &= zR \bmod N = [(xy) \bmod N]R \bmod N \\ &= (xR \bmod N)(yR \bmod N) \\ &= XYR^{-1} \bmod N\,. \end{aligned}$$

Algorithm 3.3: Montgomery Reduction

Require: Integers $N = (N_{n-1}, \ldots, N_1, N_0)_b$, with $gcd(N, b) = 1$, $R = b^n$ $T = (t_{2n-1}, \ldots, t_1, t_0)_b < NR$ and $N' = -N^{-1} \bmod b$

Ensure: $TR^{-1} \bmod N$

1: $A \leftarrow T$. (Notation: $A = (a_{2n-1}, \ldots, a_1, a_0)_b$.)
2: **for** i from 0 to $n-1$ **do**
3: $\quad m_i \leftarrow a_i \, N' \bmod b$
4: $\quad A \leftarrow A + m_i N b^i$
5: **end for**
6: $A \leftarrow A/b^n$
7: If $A \geq N$, then $A \leftarrow A - N$
8: Return A.

We denote this product as follows:

$$\text{Mont}(X, Y) = XYR^{-1} \bmod N = \frac{XY + mN}{R}\,, \tag{3.1}$$

which holds by Lemma 4 and we call it *the Montgomery product.*

So, the method of Montgomery requires conversions of x and y to the N-residue domain and conversion of the calculated result back to the integer domain. The procedure is as follows. To compute $z = xy \bmod N$, one first has to compute the Montgomery products of x and y with $R^* = R^2 \bmod N$ to get X and Y. Then $\text{Mont}(X, Y) = Z = xyR \bmod N$ followed by $\text{Mont}(Z, 1)$ gives the desired result:

$$\text{Mont}(Z, 1) = xyRR^{-1} \bmod N = xy \bmod N\,. \tag{3.2}$$

The schematic of all required conversions from normal to Montgomery domain and back is given in Figure 3.1.

In the original algorithm of Montgomery the requirements on the parameter R were as follows: $R > N$ and R^{-1} and N' are satisfying $0 < R^{-1} < N$, $0 < N' < R$ and $RR^{-1} - NN' = 1$. For the computation of the Montgomery product $T = XYR^{-1} \bmod N$, Algorithm 3.4 was proposed by Montgomery [155].

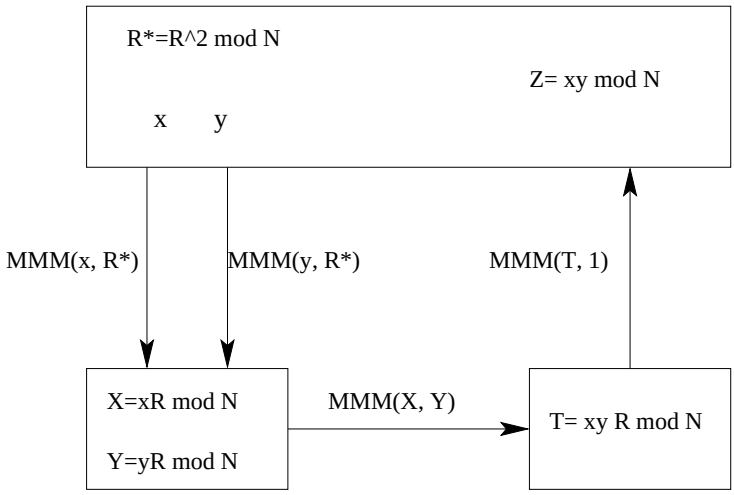


Figure 3.1: Conversions required in order to perform the algorithm of Montgomery. Here MMM stands for Mont.

Let us verify equation (3.1).

$$\begin{aligned} MMM(X,Y) &= \frac{XY+mN}{R} \\ &= \frac{XY+(XYN'+kR)N}{R} \\ &= \frac{XY+XYNN'+kRN}{R} \\ &= \frac{XY(1+NN')+kNR}{R} \\ &= \frac{XYRR^{-1}+kNR}{R} \\ &= XYR^{-1} \end{aligned}$$

(Because $1+NN' = RR^{-1}$ and m is defined as $m = (XY)N' \bmod R$.)
This also implies:

$$T = \frac{XY+mN}{R} < \frac{NN+RN}{R} < \frac{RN+RN}{R} = 2N. \tag{3.3}$$

Namely, in the original notation of Montgomery after each multiplication a reduction was needed (step 5 in the Algorithm 3.4). The input had the restriction $X, Y < N$ and the output T was bounded by $T < 2N$. Hence, if $T > N$, N must

Algorithm 3.4: Montgomery Modular Multiplication

Require: Integers $N = (N_{n-1}, \ldots, N_1, N_0)_b$, $x = (x_{n-1}, \ldots, x_1, x_0)_b$, $y = (y_{n-1}, \ldots, y_1, y_0)_b$ with $0 \leq x, y < N$ with $gcd(N, b) = 1$, $R = b^n$, and $N' = -N^{-1} \bmod b$

Ensure: $xyR^{-1} \bmod N$

1: $T \leftarrow 0$. (Notation $T = (t_l, t_{l-1}, \ldots, t_0)_b$).
2: **for** i from 0 to $n-1$ **do**
3: $\quad m_i \leftarrow (t_0 + x_i y_0)\, N' \bmod b$
4: $\quad T \leftarrow (T + x_i y + m_i N)/b$
5: $\quad$ If $T \geq N$, then $T \leftarrow T - N$
6: **end for**
7: Return T

be subtracted so that the output can be used as input of the next multiplication. This step slows down the algorithm and introduce a vulnerability to side-channel attacks.

We want to find a bound on R such that with $X, Y < 2N$ the output of the Montgomery multiplication $T < 2N$. We have proved the following theorem in [20]:

Theorem 12 *The result of a Montgomery multiplication* $XYR^{-1} \bmod N < 2N$ *when* $X, Y < 2N$ *and* $R > 4N$.

Proof: Write $R \geq kN$, then:

$$T = \frac{XY + mN}{R} = \frac{XY}{R} + \frac{m}{R}N < \frac{4}{k}N + N\,, \tag{3.4}$$

where $m = (XY \bmod R)N' \bmod R$.

Hence, $T < 2N$ for $k \geq 4$, implying: $4N \leq R$. To guarantee the existence of the modular inverse of R, R and N should be relatively prime. This excludes $4N = R$. The final round in the modular exponentiation is the conversion to the integer domain, *i.e.*, calculating the Montgomery function of the last result and 1. The same arguments as above prove that this final step remains within the following bound: $\mathrm{Mont}(T, 1) \leq N$. □

This is the same result as in the paper of Walter [242]. The work of Walter offers many other useful results for Montgomery's techniques. Some other results considering constant time implementations which is presumed to be a first step towards secure hardware solutions are proposed by Hachez and Quisquater [101].

CRT Based Implementations

By means of the Chinese Remainder Theorem (CRT), the speed for the RSA decryption scheme can be increased up to 4 times (see the book by Koblitz [129]).

This possibility is very attractive for practical applications. However, it includes some pitfalls on security, so it has to be carefully implemented. Here, we explain about the CRT implementation of RSA which is an efficient way to reduce the work factor of the decryption process. Use of CRT for RSA was proposed in 1982 by Quisquater and Couvreur [188].

Let us consider again the RSA protocol [237]. Alice wants to send a message M to Bob, so she looks up the public exponent e_B of Bob. She will send the cipher text C computed from:

$$C = M^{e_B} \bmod N\,. \tag{3.5}$$

Bob can recover M from C by using his private key d_B (here, let $e_B d_B = 1 (\bmod \lambda(N))$):

$$C^d \equiv M^{ed} \equiv M^{1+l\cdot\lambda(N)} \equiv M \cdot M^{\lambda(N)} \equiv M \bmod N. \tag{3.6}$$

If the factors of N *i.e.* p and q are known to Bob, he can do calculations $\bmod p$ and $\bmod q$ instead of mod$N$. He computes $M_p \equiv C_1^d (\bmod p)$ and $M_q \equiv C_2^d (\bmod q)$, (where $C_1 \equiv C \,(\bmod p)$ and $C_2 \equiv C \,(\bmod q)$). All these calculations are performed modulo integers p and q that are typically half of the length of N. The linear combination of M_p and M_q is the original message M. More precisely, Bob first precomputes integers a and b satisfying:

$$\begin{array}{ll} a & \equiv 1\,(\mathrm{mod} p) \\ a & \equiv 0\,(\mathrm{mod} q) \end{array} \tag{3.7}$$

and

$$\begin{array}{ll} b & \equiv 0\,(\mathrm{mod} p) \\ b & \equiv 1\,(\mathrm{mod} q) \end{array} \tag{3.8}$$

By the Chinese Remainder Theorem, M is now given by: $M = aM_p + bM_q (\bmod N)$. This method is known in the literature as the Gauss algorithm [155]. A similar efficient algorithm for reconstructing the message M is Garner's algorithm [155]. It reconstructs M as follows.

$$M \equiv M_q + q \cdot (s \cdot (q^{-1} \bmod p)) \bmod p \tag{3.9}$$

where $s \equiv (M_p - M_q) \bmod p\,.$

These computations can be performed in $O((lg\, n)^2)$ bit operations. (Here, lg denotes the base 2 logarithm.) Altogether, this way of decryption can reduce the workload by a factor of 4 if cubic complexity of exponentiation is assumed.

In April 2000 Compaq and RSA Security Inc. announced a new patented technology $MultiPrime^{TM}$ [54] as a generalization of standard RSA scheme. Instead of a modulus $N = pq$, as in traditional RSA system, N is a product of three or more (distinct) prime numbers. The idea was that increasing the number of factors and using CRT with parallel exponentiators increases the performance. The dependence of the performance of modular exponentiation and the length of modulus

is not linear: it is approximately cubic or quadratic depending on the implementation [20]. However, a high level of security has to be preserved. The length of N is not the only relevant factor. Smaller factors make some factoring methods more efficient, for example the Elliptic Curve factoring Method (ECM). The current record is 66 digits for smallest factor [66], which results in the use of not more than 4 prime factors of N for the modulus lengths of 2048 bits and up to three factors for 1024 bit modulus. In conclusion, *MultiPrime* does not require a larger modulus compared to the standard $N = pq$ scheme. For example, a 2048-bit modulus $N = pq$ offers approximately the same level of security against factoring as a 2048-bit modulus $N = pqr$ [140]. For a 1024-bit modulus A. Lenstra, estimated that factoring a 1024-bit RSA modulus $N = pq$ is computationally equivalent to finding a 341-bit factor of a 3-prime modulus N of the same size [140]. The reason is that the best known algorithm for factoring is the number field sieve which has complexity dependent on the size of the number to be factored (cf. Chapter 1). On the other hand, the complexity of ECM is a function of the size of the least factor. Consequently, with a proper choice of parameters, this variant of RSA offer a reasonable security level. Even more, most of the known attacks on RSA [33] take advantage of the fact that the modulus is a product of exactly two primes. In this respect the *MultiPrime* appears to be not only faster, but also a more secure variant than $N = pq$ [105].

Figure 3.2 represents the performance for different encryption exponent e for the ASIC architecture that we presented in [20] for the three RSA options ($N = pq$ with/without CRT and $N = pqr$).

Another improvement in efficiency is expected when not all factors differ. Instead of $N = pqr$, we consider for example $N = p^2q$. This is a special case of the RSA scheme where $N = p^nq$, which was introduced in 1997 by Takagi in [219, 220]. He estimates the running time of decryption for $N = p^2q$ by use of this method as three times faster than for $N = pqr$ when the standard CRT scheme is used. However, those were estimates for a 768-bit modulus. Cryptosystems based on the difficulty of factoring p^2q do not necessitate a larger modulus than the standard RSA cryptosystem with a modulus $N = pq$ [65].

Another performance improvement was suggested by Boneh and Shacham [36]. They named this variant *MultiPower* RSA and it takes advantage of small encryption exponents by use of a method known as Hensel lift (see Theorem 13). For example, in the case of 3 factors of N, *i.e.* $N = p^2q$, decryption via CRT is performed in such a way that exponentiation $\mathrm{mod}p^2$ is replaced with exponentiation $\mathrm{mod}p$ followed by Hensel lift.

1. Compute $Z = M_p = C^d(\mathrm{mod}p)$ and $M_q = C^d(\mathrm{mod}q)$.
2. Then $M_{p^2} = Z + Yp$. (According to Hensel's lemma, the solution $\mathrm{mod}\,p$ can be lifted in this way to the solution $\mathrm{mod}p^2$.)
3. Note that now: $M_{p^2}{}^e(\mathrm{mod}p^2) = (Z + Yp)^e \equiv Z^e + eYpZ^{e-1}(\mathrm{mod}p^2)$.

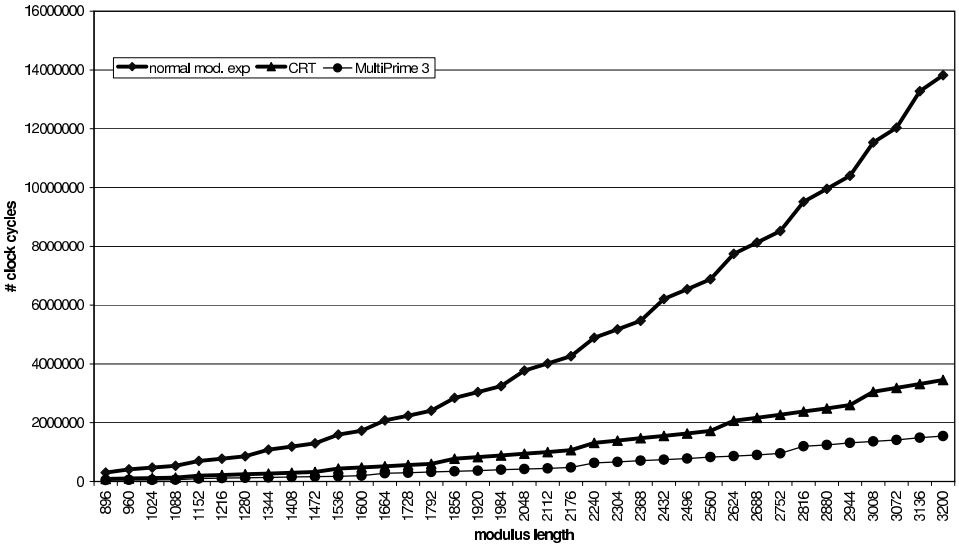


Figure 3.2: Performances of modular exponentiation for three different options of RSA technology as in Batina and Muurling [20].

4. Also, $M = C^d (\mathrm{mod} N) = aM_{p^2} + bM_q^e (\mathrm{mod} N)$ implies the following:
   $C \equiv a(Z^e + eYpZ^{e-1}) + bM_q^e (\mathrm{mod} p^2)$

5. From the previous equations:
   $C - Z^e \equiv eYpZ^{e-1} (\mathrm{mod} p^2)$.

6. The value Y, can be recovered now easily, by using:
   $C - Z^e = t$ and t is divisible by p, so the previous modular equation can be reduced by p, resulting in:
   $eYZ^{e-1} \equiv \frac{t}{p} (\mathrm{mod} p)$.

From this equation we get Y and then M_{p^2}. To conclude this observation, we introduce Hensel's lemma [167], according to which the previous steps can be derived.

Theorem 13 (***Hensel's lemma***). *Let $f(x) \in \mathbb{Z}_p[x]$ be a polynomial with coefficients in $\mathbb{Z}_p$. If $f(a) \equiv 0 (\mathrm{mod} p^j)$ and $f'(a) \neq 0 (\mathrm{mod} p)$ then there is a unique $t \in \{1, 2, ...p-1\}$ such that $f(a + tp^j) \equiv 0 (\mathrm{mod} p^{j+1})$.*

Proof: Write the Taylor's series for the polynomial f:

$$f(a + tp^j) = f(a) + tp^j f'(a) + t^2 p^{2j} \frac{f''(a)}{2!} + \ldots + t^n p^{nj} \frac{f^{(n)}(a)}{n!} . \tag{3.10}$$

Then, it follows:

$$f(a + tp^j) \equiv f(a) + tp^j f'(a) (\mathrm{mod} p^{j+1}) . \tag{3.11}$$

So, $f(a + tp^j) \equiv 0 \, (\mathrm{mod} p^{j+1})$, if and only if:

$$tf'(a) \equiv -\frac{f(a)}{p^j} (\mathrm{mod} p). \tag{3.12}$$

Since, $f'(a) \neq 0 \, (\mathrm{mod} p)$, congruence (3.12) has a unique solution. □

The number of modular operations required for decryption by use of the formulae for *MultiPower* can be found in Table 3.1. In Figure 3.3 timings are shown

Table 3.1: Total number of operations for *MultiPower* based decryption.

step 1	1 exponentiation mod$p$ 1 exponentiation mod$q$	M_p M_q
step 2	0	
step 3	0	
step 4	1 modular reduction 1 exponentiation modp^2 1 exponentiation mod$p$	$C \bmod p^2$ $Z^e \bmod p^2$ $eZ^{e-1} \bmod p$
step 5	1 division with p 1 inversion mod$p$	t/p get Y from $eZ^{e-1}Y \equiv (t/p) \bmod p$

for various options of *MultiFactor*. The $MultiPower$ appears to be more efficient than $MultiPrime$ with all different primes only when the public exponent e is relatively small compared to the modulus length. As we can conclude, for exponents which are less than approximately 290 bits *MultiPower* is faster than standard RSA. If a public exponent is smaller than approximately 130 bits the performance of *MultiPower* is better than both, *MultiPrime* and standard RSA. However, the conclusions may be different for other architectures.

In Table 3.2 possible speed-ups in software and hardware are given for two options of *MultiFacor RSA*. The fact that the numbers for our hardware solution are smaller than the ones for software is due to the specific hardware architecture. The numbers for software implementation are obtained on a 750 MHz Pentium III by use of the GMP library for arbitrary precision arithmetic as given in the work by Boneh and Shacham [36].

3.2 Previous Work

This section reviews some of the most relevant previous work in hardware implementations for RSA and ECC. The vast majority of published work that is

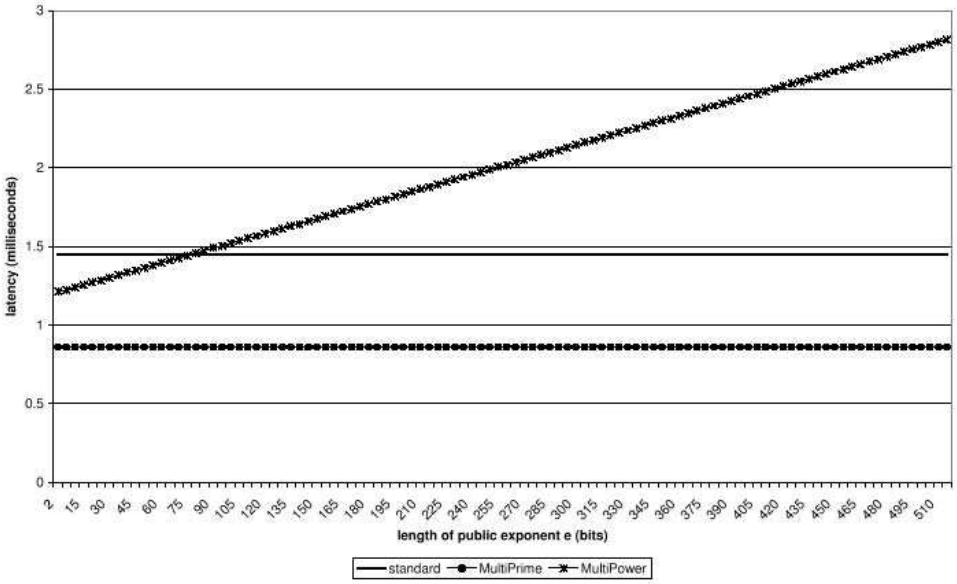


Figure 3.3: MultiPower versus MultiPrime related to the length of public exponent e as in [20].

Table 3.2: Speed-ups for *MultiFactor* RSA.

			Hardware [20]		
	Theory	Software [36]	n=1024 $L(e)$=16	n=1024 $L(e)$=140	n=2048 $L(e)$=16
MultiPrime	2.25	1.73	1.20	1.20	1.69
MultiPower	3.375	2.3	1.49	1.25	2.02

considering implementations of PKC deals with software platforms. Some of the work is done on FPGAs and only very few implementations are presenting an ASIC implementation of ECC in the field of prime characteristic. Some authors have considered dual field implementations *i.e.* ECC in prime and binary field.

3.2.1 RSA Hardware Implementations

There is a long history of RSA hardware implementations. The "first" RSA chip was designed by Rivest, Shamir and Adleman in 1980 [192]. In contained approximately 40,000 transistors but failed to work reliably. However, a fast development of VLSI RSA circuits was forecasted. R. Rivest reviewed the issues involved in building a special-purpose chip for performing RSA encryption/decryption [192]. In the 80s already several methods for implementing modular reduction were known. In the work by Brickell [41] the quotient digits are approximated using only the high order bits of the divisor and the current remainder. In 1985 Kochanski [132] proposed a RSA chip in which the modular exponentiation operation was divided between a CMOS array and a microprocessor. A full length modular multiplication took place in CMOS array by the commands from a microcontroller. The complete modular exponentiation was performed by microcontroller. In 1986, Rankine [191] introduced an implementation of a 512-bit modulus exponentiator for RSA applications. A wide range of applications including smart cards was claimed. This device was able to execute a 512-bit exponentiation in less than a second. Various speed-up techniques in hardware and software have been analyzed and compared by Shand and Vuillemin [207]. These include: Chinese remainders, MMM, addition chains, quotient pipelining etc. Various solutions for systolic arrays were proposed in the past ten years, for example [69, 239, 113, 240, 227], but no implementations have been reported to our knowledge (see also the work by Blum and Paar [31]).

One of the early chips was presented by Hoornaert *et al.* [108]. It was implemented as an 120-bit bit-slice processor, which could be interconnected without any additional circuitry to obtain arbitrary word lengths. Hence, it was the first one dealing with scalability. By that hardware quality we mean the ability of hardware platform to handle arbitrary bit-lengths and to be easily scaled to a desired size. In that case, the performance should also be scaled linearly. The work of Tenca and Koç defines precisely the notion of scalable hardware. The authors have described a pipelined Montgomery multiplier, which has the ability to work on any given operand precision and is adjustable to the available area and/or performance. The authors propose an algorithm in which the operand Y (multiplicand) is scanned word-by-word and the operand X (multiplier) is scanned bit by bit. This organization allowed some parallelism which has maximal degree of $p_{max} = \lceil \frac{e+1}{2} \rceil$, where for operands with m bits of precision, $e = \lceil \frac{m+1}{w} \rceil$ words are required. If less than p_{max} processing units (PUs) are available, it causes the pipeline to stall. In that case some buffering is necessary and the total execution time increases. The certain number of pipeline stages is also a parameter considered in the design. Yet, having only few stages yields very poor performance for the high precision range (512-1 024 bits) due to pipeline stalls. However, the flexibility of this design provides various design trade-offs. In this work the radix-2 algorithm is addressed and higher radix was dealt with in Tenca *et al.* [222].

Savaş *et al.* [201] used the same design methodology to obtain a dual-field multiplier for both fields of use in cryptography *i.e.*, $\mathbb{F}_p$ and $\mathbb{F}_{2^n}$, without compromising scalability. The basic observation to provide the unified Montgomery multiplier is that an adder module, equipped with the property of performing addition with or without carry, is available. This dual-field adder is basically a full adder which is capable of performing both types of addition. This so-called dual multiplier would have obvious benefits for many applications of public key cryptography. More on contributions considering dual-field arithmetic can be found in Chapter 4.

Koç *et al.* discussed five different Montgomery multiplication algorithms in [50]. For all five methods the space and time requirements have been analyzed in detail. For a general class of processors the most efficient method is identified. The algorithms differ on the basis of two factors. The first factor is whether multiplication and reduction are *separated* or *integrated*. In the separated approach, reduction is done after multiplication. In the integrated approach, the algorithm alternates between multiplication and reduction which can be either *coarse-grained* or *fine-grained*, depending on how often one switches between these two operation. The second factor is the type of scanning which can be either *operand scanning* or *product scanning*. As the most efficient method, the authors identified the Coarsely Integrated Operand Scanning.

More recent work on hardware implementation of RSA includes the work by McIvor *et al.* [153]. They use Carry Save Adders (CSAs) to perform the large word length additions required for MMM. The obtained performance for one 1024 bit RSA decryption on the Xilinx Virtex2 board was 2.63 ms.

The work of Crowe *et al.* [58] also proposed a single architecture for RSA and ECC. A hardware optimized version of MMM is used for modular multiplication. The so-called dual processor could operate in parallel for ECC or in a pipelined series for RSA.

Our contribution presented in [20] is combining a systolic array architecture with a Montgomery based RSA implementation, achieving the same notion of scalability as introduced in Tenca and Koç [221]. The optimal bound for Montgomery's parameter R is achieved which, with some savings in hardware, omits completely all reduction steps that are presumed to be vulnerable to side-channel attacks.

3.2.2 ECC Implementations over $\mathbb{F}_p$

Still plenty of the work in ECC over $\mathbb{F}_p$ deals with software implementations, where there exist many hardware implementations over binary field. It appears that the arithmetic in characteristic 2 is easier to implement and area and power consumption are smaller than in the case of $\mathbb{F}_p$. This is believed to be true, but only for platforms where specialized arithmetic coprocessors for finite field arithmetic are not available. On the other hand, an advantage of prime field is their suitability for both RSA and ECC with sharing of hardware like in our implementation.

To the best of our knowledge, the only documented ECC processor before our work [175] over fields $\mathbb{F}_p$ was proposed in Orlando and Paar [172]. This so-called Elliptic Curve Processor (ECP) is scalable in terms of area and speed and especially suited for FPGAs. The ECP is also best suited for projective coordinates and it is using a new type of high-radix precomputation-based Montgomery multiplier. It consists of three main components: the main controller (MC), the arithmetic unit controller (AUC) and the arithmetic unit (AU). The MC is controlling the point multiplication. The AUC is responsible for point operations and it controls the AU. The AU is in charge for $\mathbb{F}_p$ operations. It consists of a register file, an adder and a multiplier as the most critical component. This multiplier performs a generalized version of the Montgomery multiplication algorithm with quotient pipelining introduced in Orup [179]. This version supports positive and negative operands and features Booth recoding and precomputation. Positive and negative numbers appear often in ECC algorithms and both are equally treated as subtraction has the same cost as addition. The proposed ECP architecture was verified on an example of the field $\mathbb{F}_{2^{192}-2^{64}-1}$ which is one of the field recommended by various standards. The scalability of the multiplier to larger fields was also verified in the field of size 521 bits. The authors have estimated eventual timing of 3 ms for computing one point multiplication.

3.3 One Hardware Platform for RSA and ECC

After giving mathematical background, some history and relevant previous work in this section we explain our contribution. Here we discuss how an FPGA implementation of Montgomery multiplication that was originally designed for RSA can efficiently be used to perform prime field ECC operations. This design consists of a Modular Montgomery Multiplier (MMM), designed as a systolic array. This array is one-dimensional and consists of a fixed number of Processing Cells (PCs). The MMM performs Montgomery modular multiplication that features the following operation: $\text{Mont}(X, Y) = XYR^{-1} \bmod N$.
As explained above for a word base $b = 2^\alpha$, R should be chosen such that $R = 2^r = (2^\alpha)^l > N$. When computing the Montgomery product $T = Mont(x, y) = xyR^{-1} \bmod N$, the procedure shown in Algorithm 3.5 is performed [155]. Here, $T < 2N$ for $k \geq 4$, implying: $4N \leq R$. We use $4N < R = 2^{l+2}$.

3.3.1 Systolic Array

Figure 3.4 shows a schematic of the systolic array that was implemented in the MMM [176]. A PC contains adders and multipliers that can process in general α bits of X and Y in one clock cycle. Here X and Y are the multiplicand and

Algorithm 3.5: Montgomery Modular Multiplication w/o final subtraction

Require: $N = (n_{l-1} \cdots n_1 n_0)_{2^\alpha}$, $x = (x_{l-1} \cdots x_1 x_0)_{2^\alpha}$, $y = (y_{l-1} \cdots y_0)_{2^\alpha}$, $x, y \in [0, N-1]$, $R = (2^\alpha)^l$, $gcd(N, 2^\alpha) = 1$, $N' = -N^{-1} \mod 2^\alpha$

Ensure: $xyR^{-1} \mod 2N$

1: T $\leftarrow 0$.
2: **for** i from 0 to $(l-1)$ **do**
3: $\quad m_i \leftarrow (t_0 + x_i y_0)\, N' \bmod 2^\alpha$
4: $\quad T \leftarrow (T + x_i y + m_i N)/2^\alpha$
5: **end for**
6: Return (T)

multiplier. Each PC calculates the temporary result

$$T_i = \frac{T_{i-1} + 2^\alpha x_i Y + m_{i-1} N}{2^\alpha} \tag{3.13}$$

in each clock cycle.

Performance results for the multiplier are given in Section 4.7.4 and a more extensive description is given in following publications [173, 174, 175, 176] and in the doctoral thesis of Sıddıka Berna Örs [178].

3.3.2 Montgomery Modular Multiplication

The processing cells in the systolic array shown in Fig. 3.4 perform Equation (3.14). Because the critical path of the systolic array is the same as the critical path of one PC, the clock frequency of the Montgomery multiplier will be the same for all bit-lengths. This property gives the advantage of using the circuit for both, RSA and ECC.

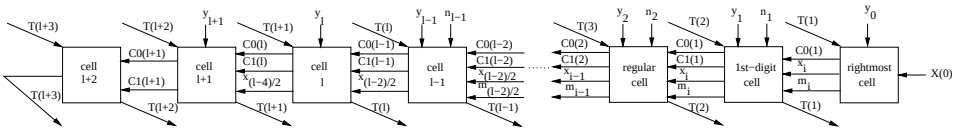


Figure 3.4: Schematic of the Modular Montgomery Multiplier.

When $\alpha = 1$, the ith iteration of Step 2 in Algorithm 3.5 computes the temporary result $T_i = \frac{T_{i-1} + x_i Y + m_i N}{2}$ where $i = 0, \ldots, l+1$ and $T_1 = 0$. In this case the jth bit of T_i is obtained using the following recurrence equation:

$$2^2 \times c1_{i,j} + 2 \times c0_{i,j} + t_{i,j} = t_{i-1,j+1} + x_i \times y_j + m_i \times n_j + 2 \times c1_{i,j-1} + c0_{i,j-1}\,. \tag{3.14}$$

Table 3.3: The performance of the FPGA implementation of the Montgomery multiplier for $\alpha = 4$.

Number of clock cycles	$3\frac{n}{4} + 7$
Clock period	19 ns
Clock frequency	53 MHz
Total latency	$14.25n + 133$ ns
Number of gates	4547

Here $c1_{i,j}$ and $c0_{i,j}$ denote the carry chain on the array. One processing cell of the systolic array is shown in Figure 3.5.

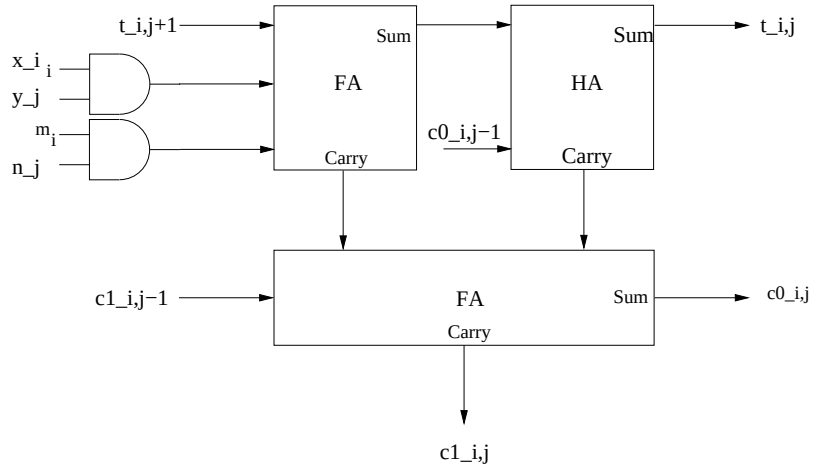


Figure 3.5: The regular PC of the systolic array for $\alpha = 1$.

Parameter $\alpha = 4$ for our implementation. Table 3.3 shows the performance of the FPGA implementation of the Montgomery multiplier. Parameter n is the bit-length of N, l in Figure 3.4 is $\frac{n}{4}$.

3.3.3 ECC Implementation

In this section we present our FPGA implementation for ECC point operations for prime fields.

ECC Processor

The MMM can be used for fast RSA implementation but also for ECC point operations in the prime field. Due to the scalability of the design, the FPGA architecture can perform both, *i.e.* efficient exponentiations on large operands (for RSA) and modular multiplication on the smaller ECC operands. In Fig. 3.6 a schematic of an FPGA implementation for ECC is given. One or two MMMs are used to perform the modular (Montgomery) multiplications. A Large Number Co-Processor (LNCP) is added to the design to perform the additions and subtractions. These units have their own RAMs and are connected with a data bus.

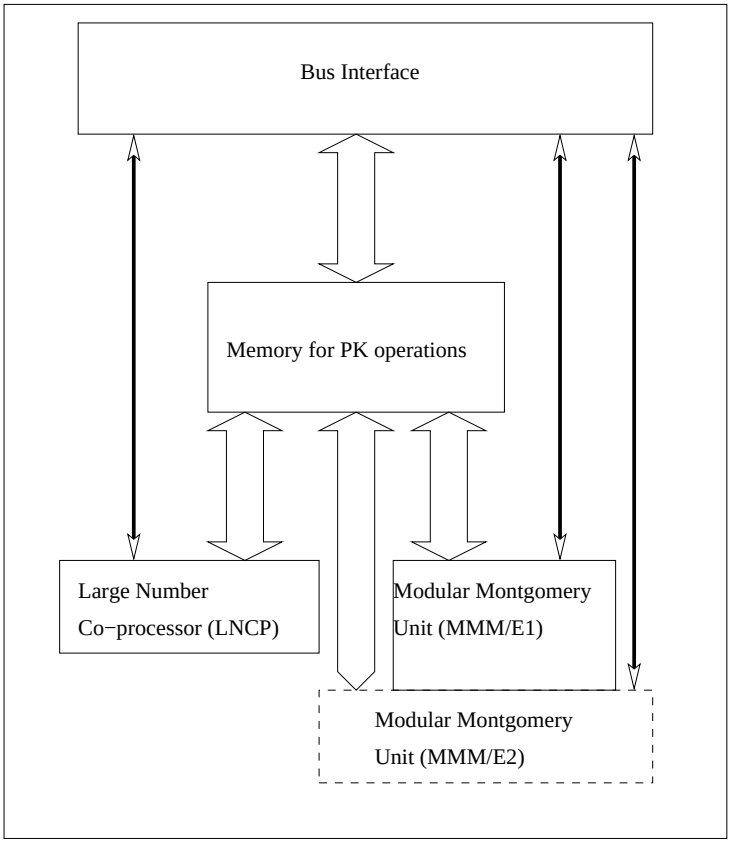


Figure 3.6: Schematic of the RSA/ECC processor.

As already explained, the performance of an elliptic curve cryptosystem is determined by the efficient realization of the arithmetic operations (addition, mul-

tiplication and inversion) in the underlying finite field. If projective coordinates are used the inversion operation becomes negligible. Therefore, coprocessors for elliptic curve cryptography are designed to accelerate the field multiplication. Considering multiplication in the prime field *i.e.* $\mathbb{F}_p$, the whole work which is done for the RSA implementation is relevant. The only difference is that shorter bit-lengths are used *i.e.*, 160-300 bits (cf. Chapter 2).

Point Addition

Point addition and doubling can be performed according to algorithms as in Chapter 2 (see also Blake *et al.* [29]). Here we assume that the two points that will be added *i.e.* $P = (X_1, Y_1, Z_1)$ and $Q = (X_2, Y_2, Z_2)$ are already transformed to the Projective coordinates and Montgomery representation. The result is point $R = P + Q = (X_3, Y_3, Z_3)$.
Scheduling of point addition. Point addition can be even more efficiently performed if two MMM units are used. The operations can be conveniently divided between the two units. (Modular) addition and subtraction will be done on a Large Number Co-processor (LNCP). Those operations can be performed in the same time as the Montgomery multiplication. The following scheduling as shown in Table 3.4 can be used. Table 3.4 shows that the performance can almost be doubled by using two MMM units.

Point Doubling

Here we discuss a special case of point addition *i.e.* point doubling, where the points P and Q are respectively given as: $P = (X_1, Y_1, Z_1)$ and $R = 2P = (X_3, Y_3, Z_3)$.
Scheduling of point doubling. In Table 3.5 a possible schedule for point doubling over the 2 MMMs and the LNCP is given. The difficulty in the scheduling of point doubling lies in the operations scheduled in MMM2 and the LNCP, which are all depending on the answer of the previous operation.

Modular Addition and Subtraction

Montgomery modular multiplication, modular addition and modular subtraction are the basic operations for point addition. MMM is performed on our highly scalable Montgomery based multiplier. Modular addition and modular subtraction can be implemented as a repeated addition. However, the number of additions and subtractions would be data dependent. Let us take a better look at these two operations. As proven in Section 3.3.1, the result of an operation on our multiplier will always be smaller than twice the modulus ($2N$). All modular additions and subtractions in the point addition scheme are with two outputs of the Montgomery multiplier.

Table 3.4: Scheduling of point addition.

MMM1	MMM2	LNCP
$Z_2{}^2$	$Z_1{}^2$	
$\lambda_1 = X_1 Z_2{}^2$	$\lambda_2 = X_2 Z_1{}^2$	
$Z_2{}^3$	$Z_1{}^3$	$\lambda_3 = \lambda_1 - \lambda_2$
		$\lambda_7 = \lambda_1 + \lambda_2$
$\lambda_4 = Y_1 Z_2{}^3$	$\lambda_5 = Y_2 Z_1{}^3$	
	$\lambda_3{}^2$	$\lambda_6 = \lambda_4 - \lambda_5$
		$\lambda_8 = \lambda_4 + \lambda_5$
$\lambda_6{}^2$	$\lambda_7 \lambda_3{}^2$	
$\lambda_3{}^3$	$Z_1 Z_2$	$X_3 = \lambda_6{}^2 - \lambda_7 \lambda_3{}^2$
		$\lambda_9 = \lambda_7 \lambda_3{}^2 - 2X_3$
$\lambda_8 \lambda_3{}^3$	$\lambda_9 \lambda_6$	
$Z_3 = Z_1 Z_2 \lambda_3$		$Y_3 = \frac{\lambda_9 \lambda_6 - \lambda_8 \lambda_3{}^3}{2}$

Table 3.5: Scheduling of point doubling.

MMM1	MMM2	LNCP
$3X_1{}^2$	$Z_1{}^2$	
$Y_1{}^2$	$Z_1{}^4$	
$\lambda_2 = 4X_1 Y_1{}^2$	$aZ_1{}^4$	
$\lambda_3 = 8Y_1{}^4$		$\lambda_1 = 3X_1{}^2 + aZ_1{}^4$
	$\lambda_1{}^2$	
$Z_3 = 2Y_1 Z_1$		$X_3 = \lambda_1{}^2 - 2\lambda_2$
		$\lambda_2 - X_3$
	$\lambda_1(\lambda_2 - X_3)$	
		$Y_3 = \lambda_1(\lambda_2 - X_3) - \lambda_3$

For example:

$$\begin{aligned} \lambda_1 = &\quad X_1 {Z_2}^2 < 2p \quad \text{and} \quad \lambda_2 = X_2 {Z_1}^2 < 2p \\ \lambda_3 = &\quad \lambda_1 - \lambda_2 \bmod p \\ \lambda_7 = &\quad \lambda_1 + \lambda_2 \bmod p \end{aligned}$$

The result of the modular addition and subtraction is again the input of another Montgomery multiplication and can therefore be larger than the modulus but should be positive. If it would be possible to calculate the previous calculations as "normal" *i.e.* non-modular addition and subtraction, this would make the operations very efficient but also time constant.

Keeping in mind the "$2p$" bound for the operands as a result of the bound for the Montgomery parameter, we get:

$$\begin{aligned} 0 < \lambda_1 + \lambda_2 < 4p - 1 \\ 0 < \lambda_1 + 2p - \lambda_2 < 4p \end{aligned} \tag{3.15}$$

We try to determine a bound for the Montgomery parameter such that we can use these non-modular addition and subtraction instead of the modular forms. To achieve this we must ensure that the inputs X and Y of the Montgomery multiplier that are smaller than $4p$ result in a Montgomery product that is smaller than $2p$.

As already mentioned, in the original implementation of our MMM the inputs of a Montgomery multiplication should be smaller than $2p$. We will use the following lemma.

Lemma 5 *If the Montgomery parameter R satisfies the following inequality $R > 16N$, then for inputs $X, Y < 4N$ the result T of the MMM will satisfy $T < 2N$ (as required).*

Proof: The Montgomery multiplication as implemented in the MMM calculates the following:

$$T = \frac{AB + mN}{R} = \frac{AB}{R} + \frac{m}{R}N\,, \tag{3.16}$$

where m is calculated modulo R. Filling in the bounds for the inputs and $R > 16N$ we get

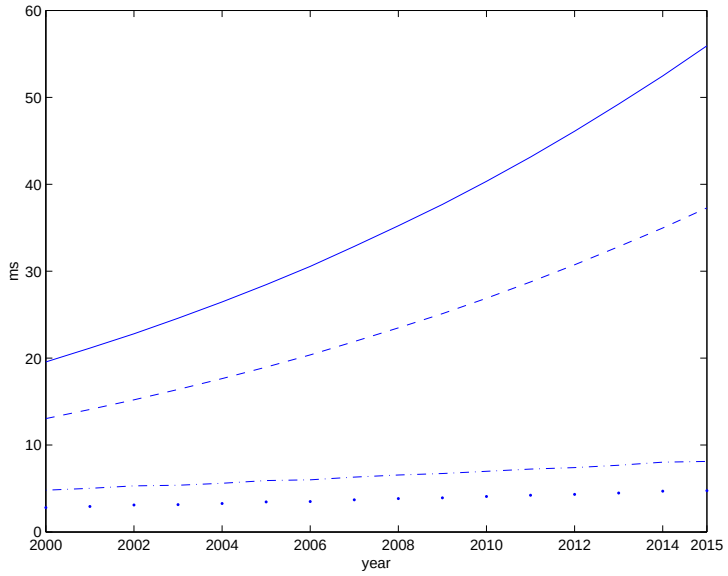
$$T = \frac{AB}{R} + \frac{m}{R}N < \frac{4N \cdot 4N}{R} + N \leq 2N\,. \tag{3.17}$$

If n is the length of modulus N in bits then the following is valid: $N < 2^n$ and $16N < 2^{n+4}$. With $R = 2^r$, we get $r \geq n + 4$. □

We have shown that for all modulus lengths, inputs smaller than $4p$ will result after a Montgomery multiplication on the MMM in a value which is smaller than $2p$. Therefore we can use time constant implementation of modular addition and subtraction. This results in a more efficient implementation with respect to the latency and the hardware resources are minimized. Furthermore, there is no loss in efficiency caused by this enlarged bound because R is usually already bigger than this bound (especially for $\alpha > 1$).

3.4 Results and Timings

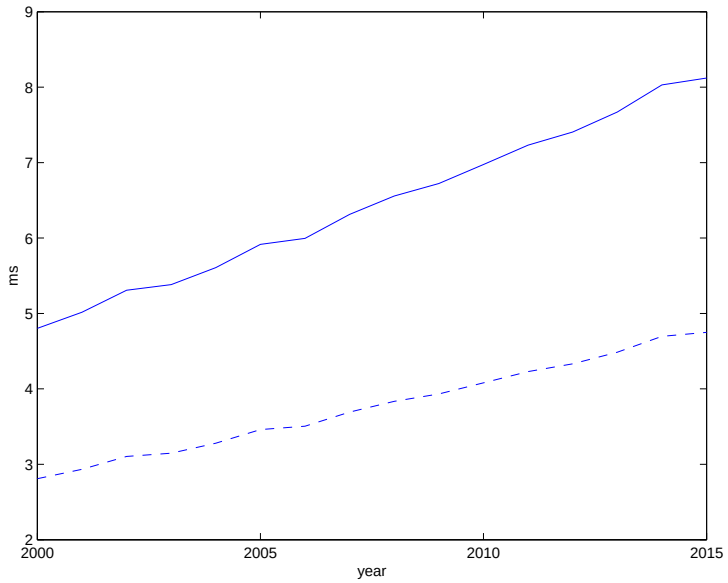
As mentioned in Chapter 2 Lenstra and Verheul made a security comparison between RSA and ECC key lengths in [141]. They introduced a table that included corresponding key bit-lengths assuring minimal security in the years to come for the two Public Key systems (see Table 3.6). In Figure 3.7 the performances for ECC and RSA are given according to the key sizes that were given in their paper. The figures show also that especially for the future applications the performance of ECC is more attractive than the performance of RSA.



—RSA with 1 MMM, - - RSA with 2 MMMs,
-. ECC with 1 MMM, .. ECC with 2 MMMs

Figure 3.7: The years correspond to the required bit-lengths assuring minimal security for RSA and ECC. On the y-axis the performance of one RSA exponentiation or one ECC point multiplication is given in ms.

Figure 3.8 shows the performance for an ECC implementation with one and two MMMs. The implementation with 2 MMMs is scheduled according to the schedule given in Table 3.4 and Table 3.5. The figure shows a speed-up of a factor of 2 for the two MMMs variant. At the same time the hardware is enlarged with a factor less than 2. The detailed performance results are given in Table 3.6.



—ECC with 1 MMM, - - ECC with 2 MMMs.

Figure 3.8: Performance of ECC with 1 or 2 MMM. On the y-axis the performance of one ECC point multiplication is given in ms.

3.5 Secure Implementations of CRT

We now briefly review some benefits of Montgomery's Multiplication Method, which are also evident for CRT implementations.

Especially implementations of CRT schemes are found to be very sensitive to side-channel attacks. For example, recently a new SPA-based attack was introduced by Novak [168], which is targeting the algorithm of Garner (Algorithm 3.6) [155]. This scheme is often used in many applications, including smart cards. It is usually implemented as shown in Algorithm 3.6.

The third step is the critical one. Novak observed that if the modular subtraction is implemented in the common way it may leak information. More precisely, to perform subtraction (mod p) one has to check the sign of $s-t$ and conditionally add p if $s-t<0$ ($p>q$ is required). Novak managed to build a successful attack based on this observation. An implementation of Algorithm 3.6 can produce the optional pattern in a power trace as a result of the conditional addition.

We propose the following solution. Instead of the subtraction mod$p$, one can compute the following:

$$x = s + p - t. \tag{3.18}$$

Table 3.6: RSA and ECC performance in ms at 53 MHz.

year [141]	RSA			ECC		
	size	1 MMM	2 MMM	size	1 MMM	2 MMM
2002	1024	22.8	15.2	139	5.3	3.1
2014	1536	52.5	35	172	8	4.7
2023	2048	90.6	60.4	197	10.5	6.1
2051	4096	350.9	234	275	11.4	6.7

Algorithm 3.6: Garner's algorithm for CRT

Require: ciphertext C, $N = p \cdot q$, $(p > q)$ and precomputed values $d_1 \equiv d \mod (p-1), d_2 \equiv d \mod (q-1)$ and $U = q^{-1} (\mathrm{mod} p)$ $(C_1 \equiv C \,(\mathrm{mod} p)$, $C_2 \equiv C \,(\mathrm{mod} q))$

Ensure: $R = M \equiv t + q \cdot (s \cdot (q^{-1} \bmod p)) \bmod p$

1: $s = M_p \equiv {C_1}^{d_1} \bmod p$,
2: $t = M_q \equiv {C_2}^{d_2} \bmod q$
3: $x = (s - t)(\mathrm{mod} p)$
4: $R = t + q \cdot ((x \cdot U)(\mathrm{mod} p))$
5: Return R

For $p > q$ the result stays within the following bounds $0 < x < 2p$ which can be handled easily if Step 4 is implemented by use of the algorithm of Montgomery. Namely, the algorithm as proposed in [101, 242] for Montgomery modular multiplication takes two inputs $0 < X, Y < 2p$ and the result is also within the same interval, if the proper bound for Montgomery parameter R is chosen. This result is converted from the Montgomery domain to the usual domain by a Montgomery multiplication with 1. Changing Garner's scheme in this way results in the algorithm that is not data-dependent. We prove this in more detail.

Lemma 6 *The result of $x = s + p - t$ is always smaller than $2p$, for the parameters s, p, t defined as above, i.e. $s = M_p \equiv {C_1}^{d_1} \bmod p$, $t = M_q \equiv {C_2}^{d_2} \bmod q$ and $p > q$.*

Proof: It is shown that with the use of Montgomery's algorithm and $R > 4N$, the final result of modular exponentiation is bounded by the modulus N. (See Theorem 12.)
Now, we can prove the claim. It is obvious that $0 \leq s < p$ and $0 \leq t < q$. We assume $p > q$ with which this proof does not lose its generality, the other case is almost the same. Then we get $x = s + p - t < p + p - 0 = 2p$. Hence, if the

multiplication in the Algorithm 1 is implemented as the one of Montgomery, no conditional subtraction is required as in original algorithm. This concludes the proof. □

3.6 Security Remarks

In this section we address side-channel security *i.e.* resistance to timing [133, 100] and power analysis based attacks [135] (cf. Chapter 1). These types of attacks, together with fault-analysis based attacks [34, 117, 6] electromagnetic analysis attacks (EMA) [189, 80] and other physical attacks such as probing attacks [4] are a major concern especially for wireless applications. Here we discuss the first two with respect to our RSA/ECC hardware implementation, which are also believed to be the most practical .

As already mentioned in Chapter 1, computations performed in non-constant time *i.e.* computations which are time-dependent on the values of the operands, may leak secret key information. This observation is the basis for timing attacks. On the other hand, power analysis based attacks use the fact that the power consumed at any particular time during a cryptographic operation is related to the function being performed and data being processed. The attack can be usually performed easily for smart cards, since they receive the power externally and an attacker can easily get to hold on the source of this side-channel information.

In our implementation all modular reductions are excluded. The weaknesses in the conditional statements of the algorithm (used for realization of the reduction step) are time variations and therefore these should be omitted. By use of an optimal upper bound the number of iterations required in the algorithm based on Montgomery's method of multiplication can be reduced [244]. Another timing information leakage that was observed by Quisquater *et al.* [100] and Walter [244] was the timing difference between "square" and "multiply". This information can be used to attack RSA, even where advanced exponentiation methods are used. In our architecture, this weakness is removed, because the same systolic array is performing squarings and multiplications, which are therefore indistinguishable with respect to timing.

Besides that, when considering power analysis attacks, some other precautions have also been introduced. The fact that all of the processing elements operate in parallel makes these types of attacks far less likely to succeed. Both, RSA and ECC can benefit from this fact.

As already mentioned, this architecture can be an option for wireless devices, although we have chosen to introduce a network security product as an example.

With respect to suitability of an FPGA platform as a playground for examination of side-channel leakage we mention here the work of Örs *et al.* [177]. In that work they characterize the power consumption of a XILINX Virtex 800 FPGA and showed that it is possible to draw conclusions about vulnerability of an ordinary

ASIC in CMOS technology by performing power-analysis attacks on an FPGA-implementation. With respect to this, an FPGA design can serve as a good model for an ASIC platform not just for usual hardware related properties but also for security at least for technologies above 130 nm. We use this also in Chapter 4 where we give more details on our measurement set-up and the experiments performed on it.

3.7 Conclusions

We have presented a hardware implementation on a systolic array architecture that is scalable in all parameters and ideally suitable for RSA and ECC algorithms. We have also introduced a bound on Montgomery parameter R, which allows us to perform the most efficient point addition and doubling for ECC, as well as modular exponentiation. Furthermore, we explored the parallelism between the both elliptic curve point operation in order to improve the performance. By using this optimal scheduling for our architecture we have obtained a substantial speed-up for ECC when compared with RSA implementation on the same platform. More precisely, ECC can be faster with a factor between 4 and 30, depending on bit-lengths and the number of multipliers used.

Finally we evaluated side-channel resistance of our implementations. Even in the case of CRT the Montgomery's algorithm appears to be the best option for side-channel resistance.

Chapter 4

Balanced Algorithms for a Side-Channel Aware Design

As already mentioned (cf. Chapter 1), another alternative to RSA for PKC is Elliptic Curve Cryptography (ECC), which was proposed in the mid 1980s by Miller [163] and Koblitz [126]. Furthermore, in 1988 Koblitz suggested to use the generalization of EC for cryptography, so-called Hyperelliptic Curve Cryptography (HECC) [127]. While ECC applications are highly developed in practice, the use of HECC is still of pure academic interest. However, recent work of Pelzl *et al.* [182] showed that HECC performance can be compared to the ECC performance.

For ECC (or HECC) two types of finite fields are being considered, *i.e.* binary and prime fields. A field $\mathbb{F}_{2^n}$ offers far more options as there are many choices for bases, irreducible polynomials, composite fields, etc. There exist several ways to accelerate this curve-based arithmetic. Following a bottom-up approach these are: speeding-up the finite-field arithmetic (especially multiplication and inversion), choosing a "good" representation (*i.e.* coordinates that are more efficient) and accelerating a scalar multiplication operation.

This chapter deals with algorithms for hardware implementations of curve based cryptography (*i.e.* ECC and HECC). The results were published in [17, 16]. We also made the point addition and doubling balanced, *i.e.* they are implemented as identical sequences of operations. As an example we implemented an ECC point multiplication algorithm, using the approach of Montgomery, for which a single power trace does not expose the Hamming weight nor the bits of the secret key. Nevertheless, our FPGA implementation is also compact and efficient. The proposed multiplier for the finite field operations is digit-serial and scalable to arbitrary bit-lengths. It calculates the result by splitting the multiplication into two separate processes. The presented architecture compares favorably to designs presented in the literature. The performance is improved with a factor 2 and the area

is increased with a smaller factor. Furthermore, the power consumption graphs show an improved side-channel resistance of the new implementation.

4.1 Elliptic Curves over $\mathbb{F}_{2^n}$

The basic algorithms for ECC are given in Chapter 2. For point operations in either case *i.e.* the doubling or the addition, the computation requires one field inversion (I), two field multiplications (M) and one squaring (S), or $1I + 2M + 1S$. As we are interested in hardware implementations we count squarings and multiplications together as they are both executed on the same multiplier.

We introduce some more notation. Let $P_4 = (x_4, y_4) = P_2 - P_1$ and $P_5 = (x_5, y_5) = 2P_1$ with $P_3 = P_1 + P_2$ as defined before (cf. Chapter 2). The point P_4 is included because the method for point multiplication, as introduced by Montgomery, is defined by the fact that to add two points their difference should be known (while y-coordinate is not needed).

We rewrite the formulae for point operations from Chapter 2 using that $P_1 = (x_1, y_1) \in E$. For the x-coordinate of the point doubling, we get:

$$x_5 = (\frac{y_1}{x_1} + x_1)^2 + \frac{y_1}{x_1} + x_1 + a = \frac{{y_1}^2 + {x_1}^4 + y_1 x_1 + {x_1}^3 + a{x_1}^2}{{x_1}^2} = {x_1}^2 + \frac{b}{{x_1}^2}\,.$$

Therefore, for $P_1 \neq P_2$:

$$\begin{array}{ll} x_3 & = (\frac{y_1+y_2}{x_1+x_2})^2 + \frac{y_1+y_2}{x_1+x_2} + x_1 + x_2 + a\,, \\ y_3 & = (\frac{y_1+y_2}{x_1+x_2})(x_1 + x_3) + x_3 + y_1\,. \end{array} \tag{4.1}$$

If $P_1 = P_2$,

$$\begin{array}{ll} x_5 & = {x_1}^2 + \frac{b}{{x_1}^2} \\ y_5 & = {x_1}^2 + (x_1 + \frac{y_1}{x_1})x_3 + x_3\,. \end{array}$$

For P_4 we get from Blake *et al.* [29]:

$$x_4 = (\frac{y_1 + y_2 + x_2}{x_1 + x_2})^2 + \frac{y_1 + y_2 + x_2}{x_1 + x_2} + x_1 + x_2 + a\,. \tag{4.2}$$

We will use the observation that the x-coordinate of P_5 does not include the y-coordinate of P_1. Also the x-coordinate of the product and the sum of P_3 and P_4 can be expressed with the x-coordinate only. More precisely, we have:

Lemma 7 [148] *The x-coordinates of the points $P_3 = (x_3, y_3) = P_1 + P_2$ and $P_4 = (x_4, y_4) = P_2 - P_1$ on an elliptic curve given by equation $y^2 + xy = x^3 + ax^2 + b$ satisfy:*

$$x_3 + x_4 = \frac{x_1 \cdot x_2}{(x_1 + x_2)^2}\,. \tag{4.3}$$

Proof: The formula for addition (4.1), and the curve equation imply:

$$\begin{aligned} x_3 &= (\tfrac{y_1+y_2}{x_1+x_2})^2 + \tfrac{y_1+y_2}{x_1+x_2} + x_1 + x_2 + a = \\ &= \tfrac{{y_1}^2+{y_2}^2+(y_1+y_2)(x_1+x_2)+{x_1}^3+a({y_1}^2+{y_2}^2)}{(x_1+x_2)^2} = \\ &= \tfrac{x_2y_1+x_1y_2+(x_1+x_2)x_1x_2}{(x_1+x_2)^2} \,. \end{aligned}$$

Similarly,

$$x_4 = \frac{x_1x_2(x_1+x_2) + x_1y_2 + x_1x_2 + x_2y_1}{(x_1+x_2)^2} \,.$$

Summing up the previous two equations gives formula (4.3). □

The formula above follows also from Montgomery's work [165] in which he also observed that for some curves the sum (and the product) of the x-coordinates of the sum (or the difference) of two elliptic curve points can be expressed by the x-coordinates only (of these two points).

As already mentioned in Chapter 2 the inversion operation is very costly in hardware and can be avoided by choosing one of many options for projective coordinates. However, the number of multiplications is increased in this case, which makes the choice of a multiplier even more crucial for an efficient implementation.

Besides the choice of coordinates there is also a choice of basis. We consider polynomial bases as they are believed to facilitate flexible and generic implementations of ECC. In the polynomial basis the basis elements have the form $1, \omega, \omega^2, \ldots, \omega^{n-1}$ where ω is a root in $\mathbb{F}_{2^n}$ of an irreducible polynomial $f(x)$ of degree n over $\mathbb{F}_2$. In this basis the elements of $\mathbb{F}_{2^n}$ are polynomials of degree at most $n-1$ over $\mathbb{F}_2$, and arithmetic is carried out modulo the irreducible polynomial $f(x)$ of degree n over $\mathbb{F}_2$. According to this representation an element of $\mathbb{F}_{2^n}$ is a polynomial of length n and can be written as: $a(x) = \sum_{i=0}^{n-1} a_i x^i = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \ldots + a_1x + a_0$ where $a_i \in \mathbb{F}_2$.

To summarize, we consider squaring as a special case of multiplication and inversion is ignored because of projective coordinates. The addition of two field elements requires the modulo 2 addition of the coefficients of the elements. In hardware, a bit-parallel adder requires n XOR gates, hence the sum can be computed in one clock cycle. Field multiplication is discussed in Sect. 4.

4.2 Previous Work

This section lists some relevant previous work on ECC architectures for binary fields. There are many papers [217, 171, 92, 99, 124] dealing with this topic but very few efficient hardware implementations present a completely generic solution which allows an arbitrary choice for all parameters: field size, digit-length, irreducible polynomial, elliptic curve parameters, coordinates etc. We chose for a completely generic implementation as security criteria are changing frequently. Nevertheless,

we did not need any optimization to boost performance, which would be possible by fixing some parameters *e.g.* special curves, sparse polynomials, *etc.*

In 1989 Agnew *et al.* reported the first result for performing the elliptic curve operations on hardware [1]. They used the normal basis multiplier as arithmetic unit and a Motorola M68008 as control unit. The system could calculate about 9 elliptic curve points per second for k with Hamming weight of 30. The same authors also used Motorola M68030 as control unit and implemented a $\mathbb{F}_{2^{155}}$ processor in [2]. Sutikno *et al.* also proposed a VLSI design and implementation of arithmetic processor over $\mathbb{F}_{2^{155}}$ in [217]. The performance was 23 field multiplications for $m = 155$. Sutikno *et al.* also presented the use of non-supersingular elliptic curve group over $\mathbb{F}_{2^{155}}$ and a VLSI implementation of an ElGamal ECC processor in [218]. Gao *et al.* proposed an elliptic curve cryptosystem coprocessor with variable key size, which utilizes the internal SRAM/registers in an FPGA in [81, 82]. The scalar is decomposed as a NAF and the scalar multiplication is done with a series of addition/subtractions of elliptic curve points. Leung *et al.* described a Xilinx Virtex based FPGA implementation of an elliptic curve processor in [144]. It consists of an arithmetic logic unit (ALU), register file, a microcode sequencer and microcode storage. In ALU a Massey-Omura Multiplier is used. Another FPGA implementation for 270-bit operations similar to the previous was proposed by Ernst *et al.* in [72]. They used XC4085XLA FPGA for implementation and reported that 180,000 system gates were used. The clock speed was 34 MHz and resulting performance is 146 point multiplications per second. Schaumont and Verbauwhede introduced an Elliptic Curve Processor over $\mathbb{F}_{2^n}$ in [202, 204]. The architecture has a layered structure with the layers corresponding to the operations described in the security pyramid. The authors propose a language and simulation environment that allows to explore the design of security domain specific processors at a high abstraction level. Bednara *et al.* gave a comparison of several ways for hardware implementation of elliptic curve cryptosystems in [22]. They especially focused on FPGA based implementations.

As field multiplication is the most crucial for efficient hardware implementations the previous work on finite fields multipliers should be also considered. The first bit-serial multiplier for finite fields was discussed by Beth and Gollmann [27]. This multiplier uses convolution and reduction modulo an irreducible polynomial and takes n clock cycles to compute a multiplication. Relevant algorithms and architectures for multiplication in $\mathbb{F}_{2^n}$ have been proposed in [49, 27, 245, 228, 214, 159, 256]

In 2000 Orlando and Paar proposed a scalable elliptic curve processor architecture which operates over finite fields $\mathbb{F}_{2^n}$ in [171]. The architecture is scalable with a separated squarer (bit-parallel). Goodman and Chandrakasan proposed a cryptographic processor in [92], which performs a variety of algorithms for PKC applications. Multiplication is performed with a bit-serial multiplier using the Montgomery Modular Multiplication (MMM) [164]. Gura *et al.* [99] have intro-

duced a programmable hardware accelerator for ECC over $\mathbb{F}_{2^n}$, which can handle arbitrary field sizes up to 255. The multiplier they used is a digit-serial shift-and-add multiplier. For a detailed survey on finite fields multipliers and processors for PKC see [21].

There exists also some related work on so-called dual field ECC, which deals with processors that can support both type of fields. Wolkerstorfer [249] proposed the unique arithmetic unit that supports addition and multiplication for prime and binary fields. In his more recent work [251] this idea was developed further for light-weight implementations of ECC targeted for RFID applications (cf. Chapter 6). A scalable dual-field ECC processor is described in the work by Satoh and Takano [200]. They proposed a high-speed version of the processor and another, compact one.

4.3 A New Hardware Implementation

In this section we describe our new hardware implementation for EC cryptosystems. We follow the top-down approach and for each step we elaborate our choice. We consider the approach of Montgomery [165] for scalar multiplication. It uses a representation where computations are performed on the x-coordinate only. According to López and Dahab [148], the Montgomery representation requires less memory and offers a better protection against side-channel attacks. The same conclusion with respect to side-channel protection was drawn by Joye and Yen [119]. They also observed its benefit for a parallel computation. Menezes and Vanstone observed that the benefit in storage is at considerable expense of speed [156]. From an algorithmic point of view, Stam concludes that it is less efficient than other known methods and that in the binary case it can hardly be recommended [215]. However, our conclusion is just the opposite, at least for hardware implementations. This method can benefit from independent calculations for point operations that can be therefore performed fully in parallel by means of two multipliers. Furthermore, we have optimized the formulae for the point operations to have exactly the same number of field multiplications for point addition and doubling. The field multiplications are performed in corresponding steps in both point operations. Similar work has been done by Fischer *et al.* [75] and Izu and Takagi [114] for the $\mathbb{F}_p$ case. However their point operations are not fully balanced as in our case. We are convinced that our approach also offers an improved resistance against side-channel attacks compared to other unbalanced methods. We provide some evidence for it in the form of power consumption graphs for what are considered to be "side-channel vulnerable" operations.

4.3.1 Montgomery Method for Point Multiplication in $\mathbb{F}_{2^n}$

For point multiplication we chose the method of Montgomery that maintains the relationship $P_2 - P_1$ as invariant [165]. The idea of Montgomery dealt with speeding up the calculation of only the x-coordinate of the result. More precisely, to add two points their difference is used as an input parameter while the y-coordinate is not used in the algorithm. This fact is justified by cryptographic applications that rarely use the y-coordinate. The algorithm used (Algorithm 4.1) is a variant of the binary method and was considered by López and Dahab [148]. They have also introduced an option for recovering the y-coordinate.

Algorithm 4.1: Algorithm for point multiplication

Require: an integer $k > 0$ and a point P
Ensure: $x(kP)$
$k \leftarrow k_{l-1}, ..., k_1, k_0$
$P_1 \leftarrow P$, ;
$P_2 \leftarrow 2P$.
for i from $l-2$ downto 0 **do**
If $k_i = 1$ then
$x(P_1) \leftarrow x(P_1 + P_2)$, $x(P_2) \leftarrow x(2P_2)$
Else
$x(P_2) \leftarrow x(P_1 + P_2)$, $x(P_1) \leftarrow x(2P_1)$
end for
Return $x(P_1)$

The advantage of this algorithm is that it calculates one point addition and one doubling in each step. In this way the loop operations do not depend on the exponent, which could offer an increased resistance against timing and other side-channel attacks. Besides that we noticed that the algorithm requires less registers compared to other hardware solutions. Nevertheless, the performance is not much affected.

4.3.2 Point Addition and Doubling

In this part we move one level lower *i.e.* to point operations. This is where our design is improved with respect to other proposals. Namely, point operations (add and double) are in principle different, which can be explored from the point of side-channel analysis. Some authors have tried already before to balance these two operations in order to improve side-channel resistance. Chevallier-Mames *et al.* [51] presented a balanced algorithm for ECC over binary fields in the case of affine coordinates. We mention here the work of Brier and Joye [43] who suggested two approaches to achieve uniformity of point operations. However, both approaches

result in substantial penalty in speed. For the formulae of López and Dahab in $\mathbb{F}_{2^n}$ the operation count is $A : D = 5M : 6M$. Here, A and D are the point operations and M is a field multiplication. We remind the reader that field addition in hardware for $\mathbb{F}_{2^n}$ is just a simple bit-wise XOR operation and therefore is not taken into account.

As already mentioned, we deal with projective coordinates to avoid expensive inversions in hardware. Let us consider the formulae for point operations in the case of simple projective coordinates *i.e.* $x_i = (X_i/Z_i), i = 1, 2$. The results of point doubling and point addition, *i.e.* $X_5 = X(P_5) = X(2P_1)$ and $X_3 = X(P_3) = X(P_1 + P_2)$ respectively, are calculated as:

$$\begin{aligned} X_5 &= {X_1}^4 + b \cdot {Z_1}^4 \\ Z_5 &= {X_1}^2 \cdot {Z_1}^2 . \end{aligned} \tag{4.4}$$

$$\begin{aligned} X_3 &= x_4 \cdot Z_3 + (X_1 \cdot Z_2) \cdot (Z_1 \cdot X_2), \\ Z_3 &= (X_1 \cdot Z_2 + X_2 \cdot Z_1)^2. \end{aligned}$$

It is easy to see that point doubling and addition would require 6 and 5 multiplications respectively. We slightly rewrote the formulae in order to have 6 multiplications for both point operations. We had to add one more multiplication in the point addition, so we used the following formula:

$$X_1Z_2 + X_2Z_1 = (X_1 + X_2)(Z_1 + Z_2) - X_1Z_1 - X_2Z_2 ,$$

that follows from the Karatsuba-like approach. In the case of Karatsuba's algorithm as explained in the book by Knuth [125] a formula for multiplication reduces the problem of multiplying $2n$-bit numbers to three multiplications of n-bit numbers. Here we need one extra multiplication so we compute $X_1Z_2 + X_2Z_1$ with 3 multiplications. The next property we want is to have also balanced field multiplications in each step of the point operation algorithms (this is not the case for the algorithm of [148]). In this way the two multipliers will work fully in parallel while the exponent is scanned bit by bit. Then for each bit 1 addition and 1 doubling are performed, with fully synchronized multipliers. The algorithms for point addition and doubling are given in Algorithm 4.2.

Each point operation requires exactly 6 multiplications which are also balanced with respect to the 9 steps in Algorithm 4.2. In Step 5 of the point doubling a redundant operation is inserted to balance even the field additions. The required number of intermediate n-bit registers is 3 for both cases. More precisely, the following lemma holds:

Lemma 8 *When Algorithm 4.1 deploys algorithm 4.2 we get the following number of operations in* $\mathbb{F}_{2^n}$*:*

$$\begin{aligned} \#inversions &= 1 \\ \#multiplications &= 12\lfloor log_2 k \rfloor + 13 \\ \#additions &= 6\lfloor log_2 k \rfloor + 7 \end{aligned}$$

Algorithm 4.2: EC point addition and doubling

Require: X_i, Z_i for $i = 1, ..., 4$; $x_i = \frac{X_i}{Z_i}$, $x_4 = x(P_1 - P_2)$
Ensure: $X(P_1 + P_2) = X(P_3) = X_3$, Z_3

1. $X_3 \leftarrow X_1 + X_2$, $Z_3 \leftarrow Z_1 + Z_2$
2. $Z_3 \leftarrow X_3 \cdot Z_3$
3. $T_2 \leftarrow X_1 Z_1$
4. $X_3 \leftarrow X_2 Z_2$
5. $Z_3 \leftarrow Z_3 + T_2 + X_3$
6. $Z_3 \leftarrow {Z_3}^2$
7. $T_1 \leftarrow x_4 Z_3$
8. $X_3 \leftarrow T_2 X_3$
9. $X_3 \leftarrow X_3 + T_1$

Require: $c \in \mathbb{F}_{2^n}$, $c = b^{2^{n-1}}$, X_1, Z_1 where $x_1 = \frac{X_1}{Z_1}$
Ensure: $X(2P_1) = X(P_5) = X_5$, Z_5

1. $T_1 \leftarrow c + Z_1$, $T_1 \leftarrow T_1 + Z_1$
2. $Z_5 \leftarrow {Z_1}^2$
3. $X_5 \leftarrow {X_1}^2$
4. $T_1 \leftarrow Z_5 T_1$
5. $T_1 \leftarrow T_1 + X_1 + X_1$
6. $Z_5 \leftarrow X_5 Z_5$
7. $T_1 \leftarrow {T_1}^2$
8. $X_5 \leftarrow {X_5}^2$
9. $X_5 \leftarrow X_5 + T_1$

Note that the 12 multiplications require only the time for 6 multiplications since two multiplications are performed in parallel in every iteration of the main loop. In this way we improved the formulae of López and Dahab in order to have fully balanced point operations to render simple side-channel attacks.

4.3.3 An Algorithm for Field Multiplication

The standard way to compute the product $c(x) = a(x) \cdot b(x) \mathrm{mod} f(x)$ is the one that is using convolution and to which we refer to as the classical algorithm. Another way to calculate the product of two polynomials in $\mathbb{F}_{2^n}$ is Montgomery's multiplication algorithm as proposed in [49]. Here, we define the Montgomery Modular Multiplication (MMM, cf. Chapter 3) as: $\mathrm{MMM}[a(x), b(x)] := a(x) \cdot b(x) \cdot r^{-1}(x) \mathrm{mod} f(x)$. Before a sequence of operations can be started, all operands have to be converted to the form $a(x)r(x) \mathrm{mod} f(x)$, the so-called M-residue of the operand by multiplication with $r(x) = x^n$. The classical and the Montgomery modular multiplication algorithms can be combined into one algorithm. The main idea is to perform both algorithms in parallel, as proposed by Potgieter in [186]. The detailed algorithms were described in our paper [14]. The bit-serial version of the algorithm is given in Algorithm 4.3.

Our circuit implements Algorithm 4.4. The combined modular multiplication algorithm includes two parts, classical and Montgomery, each of which is a systolic array. The parts look quite similar as their cells are performing similar operations *i.e.* multiplication and XOR. The difference is that they shift in opposite directions and they start from opposite parts of the loop. While the classical multiplier starts the shift-and-add process from the MSB of one of the operands, and shifts the cumulative result left, the Montgomery based multiplier starts at the LSB, and shifts the result right. Those two arrays process the operand $a(x)$ from different sides and they stop after exactly $\lceil s/2 \rceil$ cycles (here s is the number of digits). The classical part still has to perform a shift over $\lceil s/2 \rceil$ bits but this is taken care of

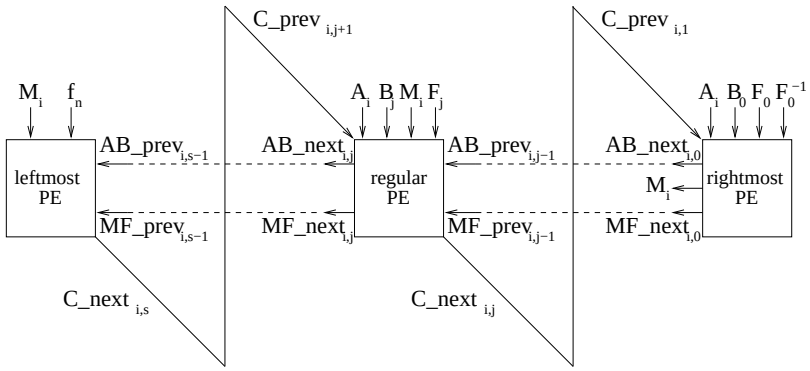


Figure 4.1: Schematic of the multiplier. The classical and the Montgomery part are calculated in parallel and the result is XOR-ed afterwards.

Algorithm 4.3: Bit-serial Modular Mult. in $\mathbb{F}(2^n)$

Require: polynomials $a(x)$, $b(x)$ and $f(x)$,
Ensure: $Res(x) = a(x) \cdot b(x) \cdot x^{-\lfloor \frac{n}{2} \rfloor} \bmod f(x)$
1: $Res(x) = 0, Res_C(x) = 0, Res_M(x) = 0$,
2: **for** i from 0 to $\frac{n}{2} - 1$ **do**
3: $\quad Res_C(x) \leftarrow Res_{C_{n-1}} \cdot f(x) + Res_C(x) \cdot x + a_{n-1-i} \cdot b(x)$
4: $\quad Res_M(x) \leftarrow Res_M(x) + a_i \cdot b(x)$
5: $\quad Res_M(x) \leftarrow Res_M(x) + Res_{M_0} \cdot f(x)$
6: $\quad Res_M(x) \leftarrow Res_M(x)$ div x
7: $\quad Res(x) = Res_C(x) + Res_M(x)$
8: **end for**
9: Return $Res(x)$

by the conversion of the M-residue of the result. More precisely, the M-residue is of the form $a(x)b(x)r^{-1}(x) \bmod f(x)$ where $r(x) = x^{\lceil s/2 \rceil}$. The multiplication is completed by calculating $MMM\,[Res(x), 1]$.

A schematic of the multiplier is given in Fig. 4.1. A_i, B_i and F_i are the coefficients of $a(x)$, $b(x)$ and $f(x)$ respectively. The outputs $c(x)$ become inputs to the systolic arrays in the next clock cycle. Finally, the result of the multiplication is obtained by XOR-ing the outputs of both systolic arrays.

In Fig. 4.2 one processing element (PE) of the array is depicted. This is the so-called regular PE, while the boundary PEs are called leftmost and rightmost PE. They have a slightly different structure which is shown in Fig. 4.3 (leftmost PE) and 4.4 (rightmost PE). We zoomed into the PEs to show the operations inside in Fig. 4.5, 4.6, 4.7, 4.8 and 4.9.

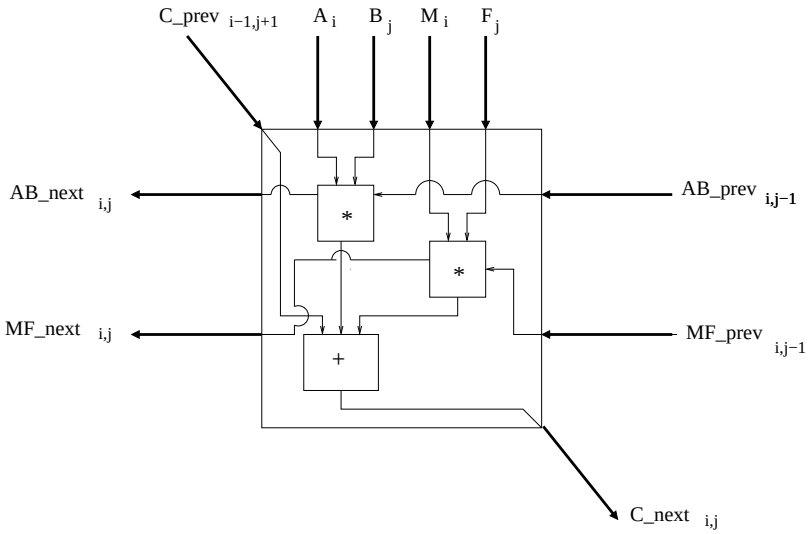


Figure 4.2: The regular PE of the systolic array.

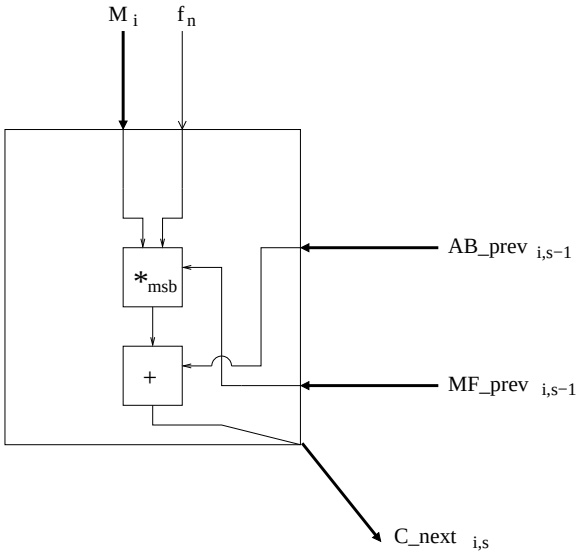


Figure 4.3: The leftmost PE of the systolic array.

Algorithm 4.4: Digit-serial Modular Multiplication in $\mathbb{F}_{2^n}$

Require: polynomials $a(x)$, $b(x)$ and $f(x)$, $F_0{}'(x)$
Ensure: $Res(x) = a(x) \cdot b(x) \cdot x^{-\lceil \frac{n}{2} \rceil} \bmod f(x)$
1: $Res(x) = 0, Res_C(x) = 0, Res_M(x) = 0$,
2: **for** i from 0 to $\lceil \frac{s}{2} \rceil - 1$ **do**
3: $Res_C(x) \leftarrow Res_{C_{n-1}} \cdot f(x) + Res_C(x) \cdot x + A_{s-i} \cdot b(x)$
4: $Res_M(x) \leftarrow Res_M(x) + A_i \cdot b(x)$
5: $M(x) \leftarrow Res_{M_0} \cdot F_0{}'(x) \pmod{x^w}$
6: $Res_M(x) \leftarrow Res_M(x) + M(x) \cdot f(x)$
7: $Res_M(x) \leftarrow Res_M(x) \text{ div } x^w$
8: **end for**
9: $Res(x) = Res_C(x) + Res_M(x)$
10: Return $Res(x)$

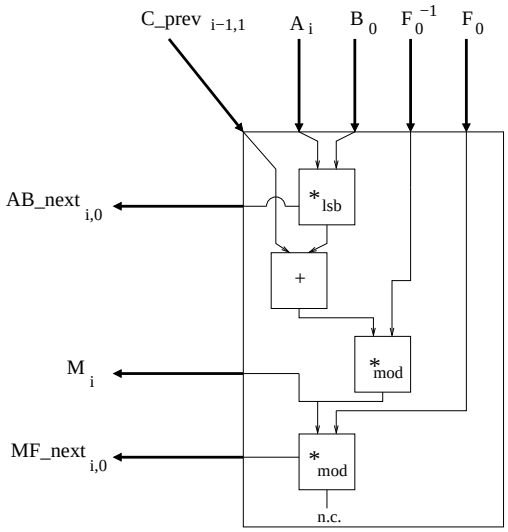


Figure 4.4: The rightmost PE of the systolic array.

After conversion from the M-residue to the normal representation, we get the result. Namely, the following lemma holds.

Lemma 9 *The result of Algorithm 4.4 is in the Montgomery domain i.e.* $Res(x) = a(x)b(x)r^{-1}(x)$ *where* $r(x) = x^{\lceil \frac{s}{2} \rceil}$.

Proof: Here $n = sw$, *i.e.* the bit representation can be written in s words of length w. After $\lceil \frac{s}{2} \rceil$ number of steps in Algorithm 4.4, the left (Res_C) part calculated

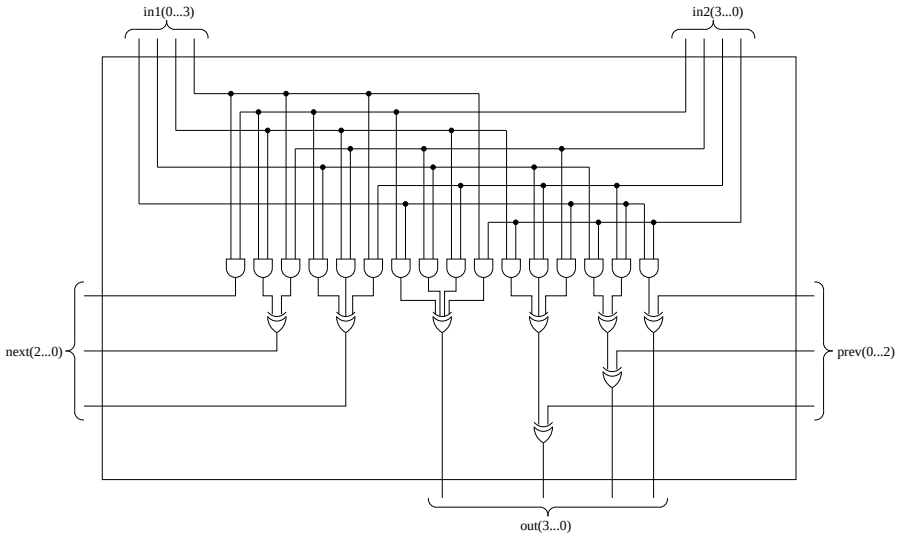


Figure 4.5: The multiplication operation in the regular PE ($w = 4$).

has been shifted to the right for $\lceil \frac{s}{2} \rceil \cdot w$ bits. At the same time, the right (Res_M) part was shifted to the left for the same number of bits due to the division with x^w. That resulted in a partial result of the form $Res_M = a(x)b(x)x^{-\lceil \frac{s}{2} \rceil \cdot w}$. The Res_C part still has to be shifted over the remaining $\lceil \frac{n}{2} \rceil$ bits, which is adjusted due to the remaining conversion from Montgomery to normal representation. □

In Algorithm 4.4 $A_i(x)$ represents one digit of the polynomial $a(x)$. The addition step is performed by multiplication of $A_i(x)$ with $b(x)$. In order to be able to perform Step 7 in Algorithm 4.4 a multiple of $f(x)$ has to be added which results in $Res_M(x)$ being divisible by x^w. So, for $Res_M(x) \neq 0 \mod x^w$, we need to find $M(x)$ such that $Res_M(x) + M(x)f(x) = 0 \mod x^w$. $M(x)$ is calculated from the following equation: $M(x) = Res_{M_0} \cdot {F_0}^{-1}(x) \mod x^w$. It is easy to see that ${F_0}^{-1}(x) \mod x^w \equiv {F_0}'(x)$ which follows from the relations for Montgomery's parameter $r(x)$ [49]. Also, here $Res_{M_0}(x)$ and $F_0(x)$ are the least significant words of $Res_M(x)$ and $f(x)$ respectively.

4.3.4 A Prototype FPGA Architecture

Our Elliptic Curve Processor (ECP) is shown in Fig. 4.10. The operation blocks on each level from top to bottom are as follows:

- Level 1: Main Controller
- Level 2:

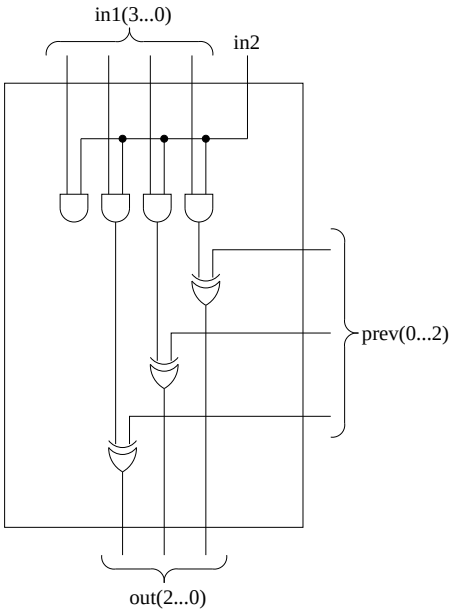


Figure 4.6: The multiplication operation in the leftmost PE ($w = 4$).

1. Normal to Montgomery representation conversion (NtoM)
2. Affine to Projective coordinates conversion (AtoP)
3. EC Point Multiplication (PM)
4. Projective to Affine coordinates conversion (PtoA)
5. Montgomery to Normal representation conversion (MtoN)

- Level 3: EC Point Doubling (PD) and EC Point Addition (PA)
- Level 4: Modular Addition (MA), Modular Multiplication (MM) and Modular Inversion (MI)

The main control Finite State Machine (FSM) first commands the NtoM to convert all the inputs from normal to Montgomery representation and the AtoP to convert the coordinates from affine to projective. The NtoM and the AtoP both use the MM to perform these conversions. The AtoP also needs the MA to do some precalculations (see Eq. (4.4)). When all conversions are finished the Main Controller orchestrates the PM block to start the point multiplication by invoking the PD and the PA in parallel. It writes the resulting X1 and Z1 to its output registers. The FSM inside the PM orchestrates these operations. Due to

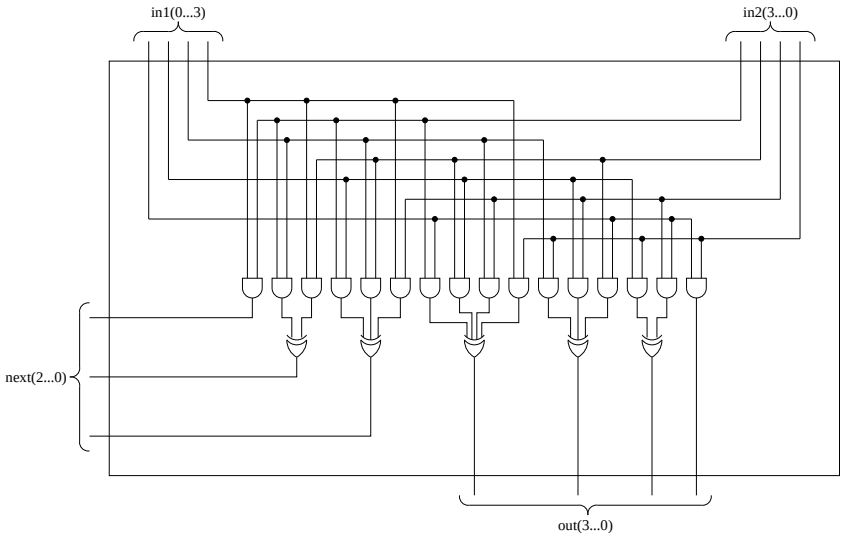


Figure 4.7: The multiplication operation in the rightmost PE ($w = 4$).

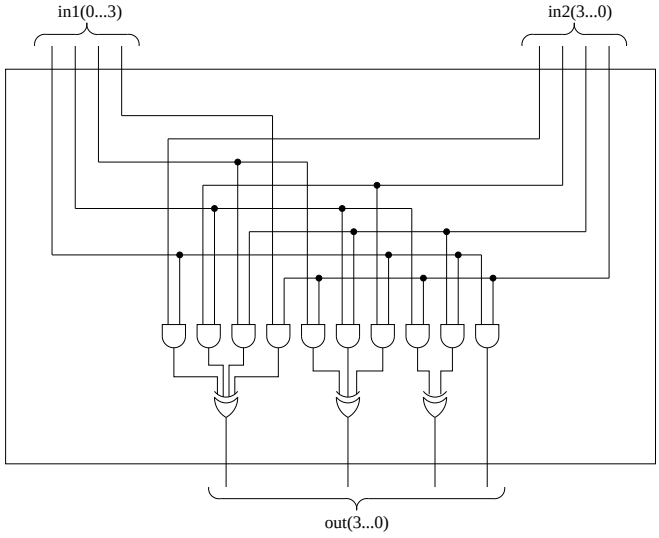


Figure 4.8: The modular multiplication operation in the rightmost PE ($w = 4$).

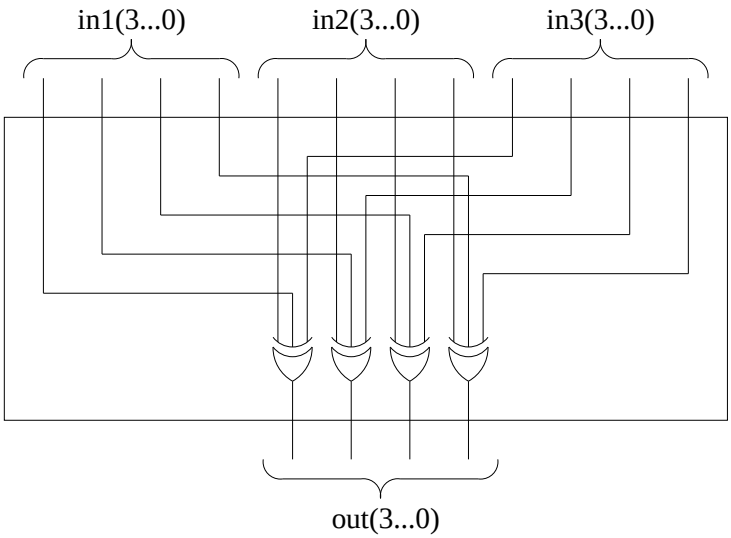


Figure 4.9: The addition operation in all PEs.

the parallelism of the point operations both the PD and the PA use an MA and an MM. The next step after point multiplication is conversion from projective to affine representation using the MI, which invokes the MM to perform the modular inversion using Fermat's theorem. Finally, the affine coordinates are converted from Montgomery to normal representation using the MM.

The flowchart of the FSM inside the point multiplication block is shown in Fig. 4.11. When the START signal is set, the bits of k are evaluated from MSB to LSB resulting in the assignment of new values for P1 and P2. These values depend on the key-bit k_i. When all bits have been evaluated, an internal counter gives an END signal. The result of the last P1 calculation is written to the output register and the VALID output is set.

4.4 Results

The results of our design on a Xilinx Virtex XCV800 FPGA are given in Table 4.1. The formula for the latency that is used in Table 4.1 is:

$$\text{latency} = [21s] + [6s(n-2) + 3s] + [18sn + 4n]\,,$$

where the three parts of the formula correspond to the calculations of the main conversion operations, the modular inversion and the point multiplication respectively.

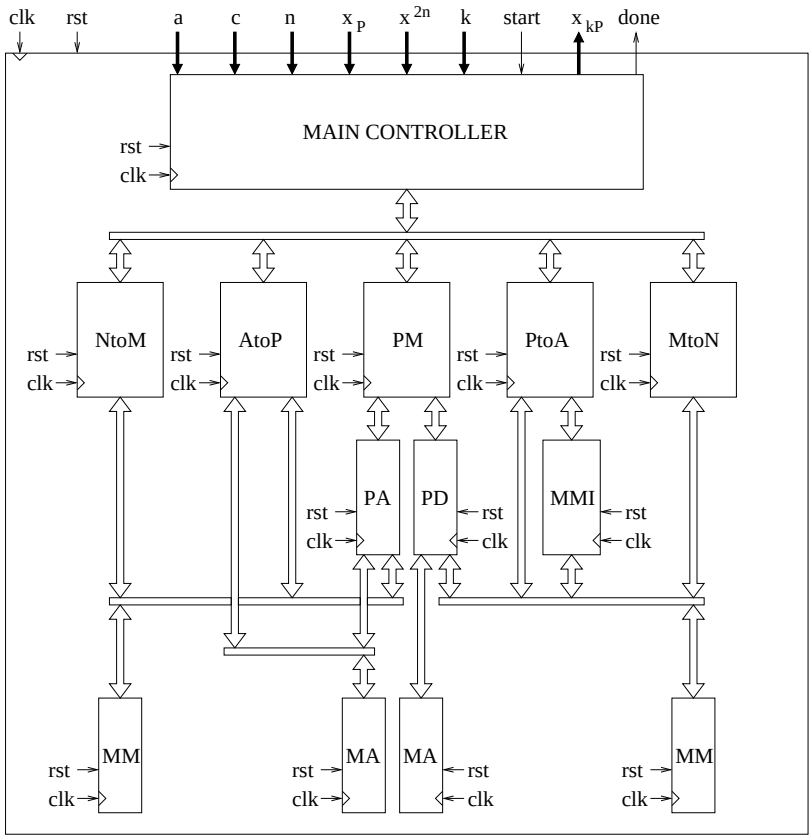


Figure 4.10: Architecture of the elliptic curve processor.

One of the consequences of the scalability of the design is that the minimal clock period does not depend on the bit-length n; it only depends on the digit-length w. This can be observed in Table 4.1.

Table 4.2 presents a broader comparison with other architectures. We found that comparing designs is hard since these designs have been optimized for different goals, implemented on different platforms and have chosen different options for bases, coordinates, irreducible polynomials etc. Moreover, some of these solutions are not scalable: in these designs, some parameters are fixed (such as a special polynomial, special reduction etc.) which boosts the performance. Therefore, we have included only the solutions that are either scalable [92, 99] or that are believed to be the state of the art in ECC hardware implementations. The purpose of this

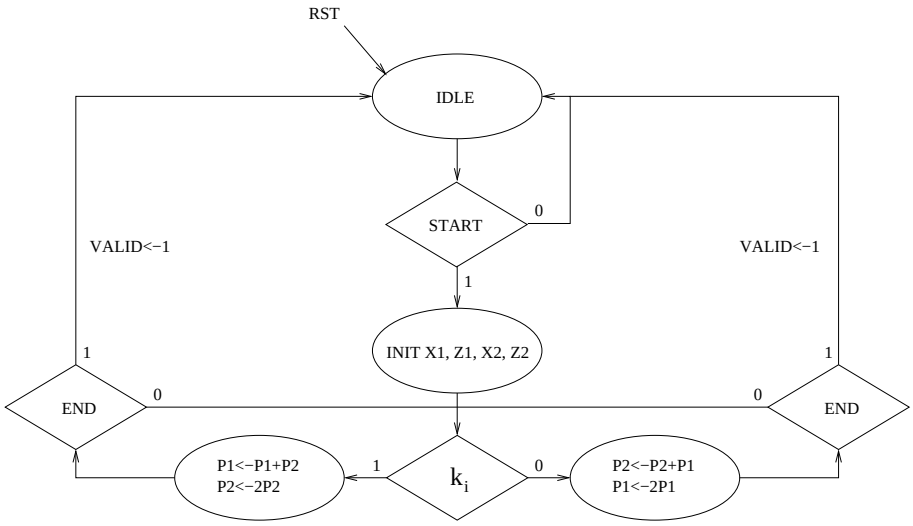


Figure 4.11: Flowchart of the FSM inside the ECPM.

table is mainly to reference the prior art in this area. We give this comparison as a proof that a scalable and side-channel secure design can also lead to a solution that is competitive in performance.

4.5 Side-channel Security

We consider again side-channel attacks as listed in Chapter 1. These attacks present a realistic threat for wireless applications and have been demonstrated to be very effective against smart cards without specific countermeasures. The latency of the proposed algorithm does not depend on the Hamming weight of the exponent, which makes it very suitable to defend against timing attacks. Indeed, timing attacks can explore all steps with non-constant execution time; conditional instructions are a typical target. Also in that case power traces may reveal some secret information that would allow the attacker to perform the Simple Power Analysis *i.e.* an SPA attack. For example, a typical double-and-add algorithm 3.2, which executes the point doubling and addition operations if the i-th bit of the exponent *i.e.* $k_i = 1$ and otherwise (for $k_i = 0$) only doubling is performed, is not SPA-resistant. Figure 4.12 shows clearly different patterns for specific bits. On the other hand, Algorithm 1 for point multiplication is more uniform w.r.t. the Hamming weight of the exponent (Figure 4.14).

Also, the computational difference between the point operations is a typical tar-

Table 4.1: Implementation results for bit-lengths of $n = 179$ and $n = 211$ and bit-serial and digit-serial multipliers are compared.

	Results for ECC point multiplication					
	# slices		period (ns)		latency (ms)	
n	179	211	179	211	179	211
$w = 1$	10626	13863	19.165	19.159	2.479	3.438
$w = 4$	11433	14660	19.298	19.301	1.886	2.614
$w = 8$	11622	14950	20.050	20.043	1.008	1.391
$w = 16$	11881	15367	20.961	20.971	0.557	0.763

Table 4.2: Comparison of different architectures for multiplication in $\mathbb{F}_{2^n}$

Impl.	Fin. field	FPGA	Frequency (MHz)	Latency (ms)	Hardware compl.
[92]	$\mathbb{F}_{2^{160}}$	ASIC	50.0	est. 5	n. a.
[171]	$\mathbb{F}_{2^{167}}$	XCV400E	76.7	est. 0.84	6 513 gates, 501 reg.
[99][a]	$\mathbb{F}_{2^{160}}$	XCV2000E	66.4	est. 0.586	14 241 LUTs, 2 990 FFs
[124]	$\mathbb{F}_{2^{163}}$	XCV1000E	22.1	7.4	not known
[*][b]	$\mathbb{F}_{2^{179}}$	XCV800	47.7	0.991	7 788 slices

[a] The performance of [99] is for so-called known curves because of a special reduction. For generic curves the performance is much slower.

[b] Our result

get of an attacker [43, 226]. To prevent that, cryptographic algorithms should be implemented as sequences of operations that are indistinguishable through simple side-channel analysis. Chevallier-Mames *et al.* define this property as the side-channel atomicity [51]. According to the authors, the SPA-resistant algorithms should consist of so-called side-channel atomic blocks, which are algorithm specific. In our implementation, there exist side-channel atomic blocks on different level of ECC hierarchy. Following the top-down approach those are point addition/doubling and multiplication/squaring. For that purpose we use the same multiplier for both, multiplication and squaring, although there exist more efficient architectures for squaring. Here we propose a fully balanced point multiplication

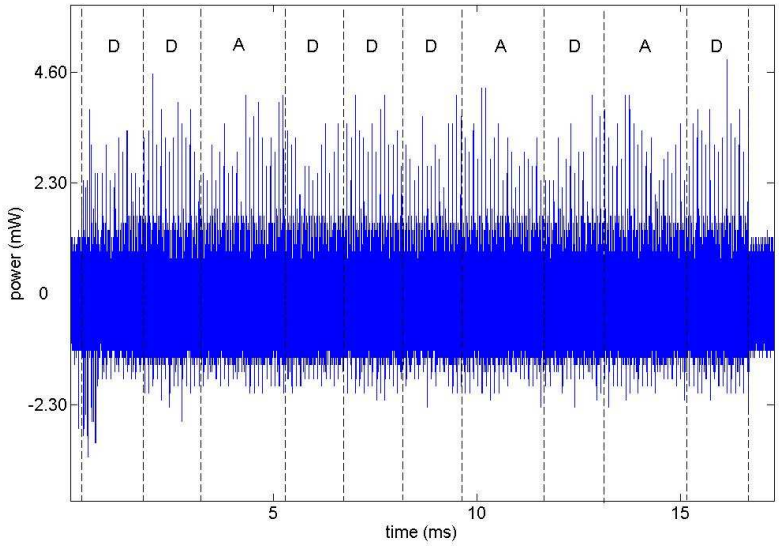


Figure 4.12: Power consumption trace of EC point multiplication by a simple double-and-add algorithm.

that performs the same operations for every loop of the algorithm. More precisely, Algorithm 4.2 for point operations executes exactly one field multiplication in corresponding steps. It is not the case in general, as doubling and addition are two different operations as in most of the standard references on ECC [111, 29]. Figures 4.15 and 4.16 show separate patterns for these two operations implemented as in most of the standard references on ECC [111, 29]. In this case the addition takes 14 and the doubling 10 multiplications, which is visible from the power traces. Considering our new Algorithm 4.2 point operations result in two not so distinguishable patterns (Figure 4.17 and 4.18). In both figures the total of 6 field multiplications can be observed. Furthermore, both operations (double and add) are performed in parallel for each step in Algorithm 4.1.

Next we describe our measurement setup on which those experiments are performed. Our setup essentially consists of two boards as visible in Fig. 4.13. The main board is responsible for interfacing the PC via the parallel port. It is connected with the XILINX parallel cable in order to program the FPGA and it provides some LEDs, switches and buttons for testing purposes. The daughter board itself just carries the FPGA, it allows to access some pins for triggering and to measure the power consumption of the FPGA in a convenient way. A protocol

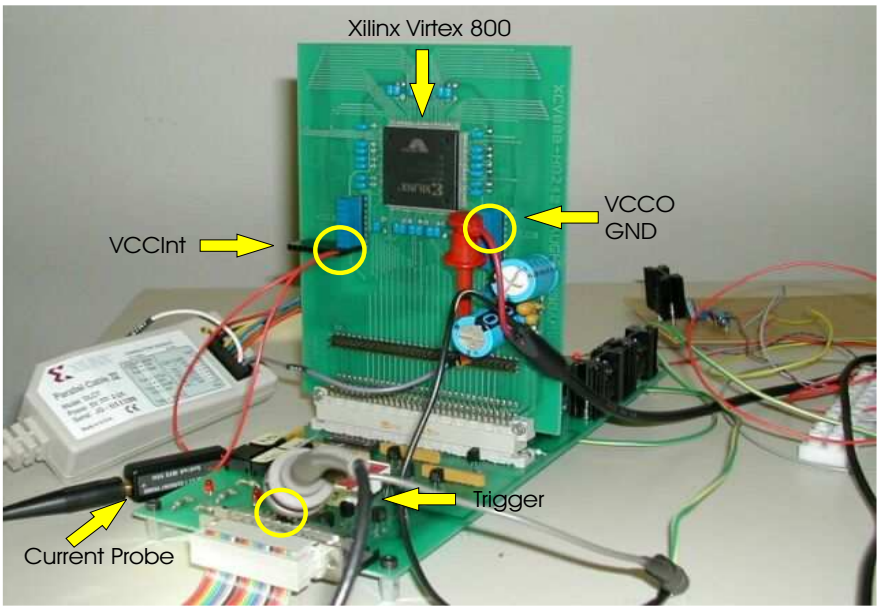


Figure 4.13: The measurement setup. On the daughter board the current probe is connected [178].

was designed to send and receive data to and from the FPGA. When the FPGA communicates with the PC, it uses the three most significant bits of the status lines to indicate its status. The two remaining bits of status lines are used for sending the result from the FPGA to the PC. The protocol is independent of the operation executed in the FPGA. Only the length of the data which is communicated can be modified by the PC. This provides a flexible setup where experiments with different algorithms can be performed in a coherent manner. The measurement setup was developed by Sıddıka Berna Örs and more details can be found in her thesis [178].

We measured the power consumption of the FPGA during the operations mentioned above. The clock frequency of the FPGA board was 300 kHz and the sampling frequency of the oscilloscope was 500 MHz.

The systolic array architecture also introduces more parallelism in the multiplication process which will make power analysis more difficult. These first results, although proving SPA and timing resistance, are just the first step towards a side-channel resistant design. As a future work Algorithm 4.2 should be more carefully examined w.r.t. security against side-channel attacks; however, we are convinced that our design approach offers an increased resistance to these attacks. Moreover,

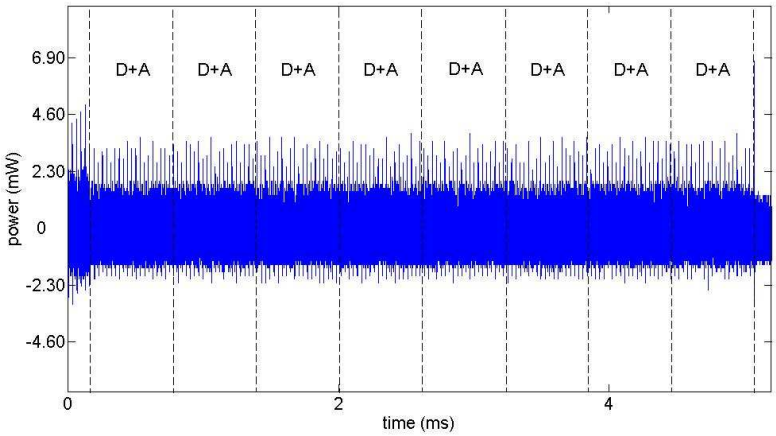


Figure 4.14: Power consumption trace of EC point multiplication by Algorithm 4.1.

for more advanced attacks such as (first- or higher-order) Differential Power Analysis (DPA), Differential ElectroMagnetic Analysis (DEMA) other countermeasures are required as well. Due to some problems with our measurement set-up these investigations are left for future work.

4.6 Hardware Implementation of Hyperelliptic Curve Cryptography

When one considers the hierarchy of operations for ECC and HECC, it is clear that the efficiency of both is heavily dependent on finite field arithmetic. Therefore, a comparison of both types of curve-based cryptosystems on the same FPGA platform is of interest. FPGAs are becoming more popular due to their flexibility and performance. While experts agree that bit-sizes for HECC parameters (with curves of genus 2) offer the same security as two times longer bit-sizes for ECC, it is still unclear whether HECC could outperform ECC. The reason is that HECC, although with shorter bit-lengths, requires more complex operations. Here we attempt to answer this question for an FPGA platform. The results show that HECC implementation, although larger for almost 30% in area, features 35% decrease in latency.

Here we describe an efficient implementations of HECC on the same FPGA platform. We specify all details of an FPGA architecture and we provide a fair comparison with ECC implementation. Namely, we used the same hardware blocks

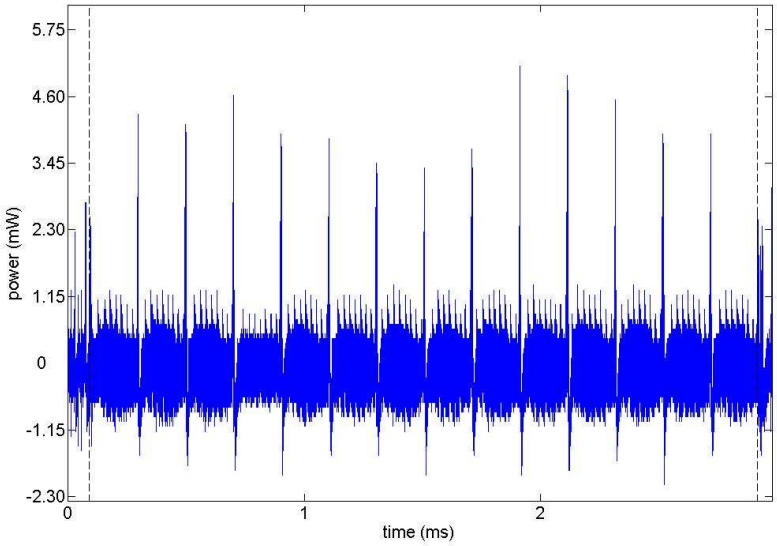


Figure 4.15: Power consumption trace of an EC point addition as in the IEEE standard. The total number of steps is 14 as visible from the graph.

for the field operations and we compared the latency for equal security levels. We concluded that HECC is a promising solution for public key cryptography.

4.6.1 Algorithms for ECC/HECC

Point/Divisor multiplication: The divisor scalar multiplication is implemented as the basic "double-and-add" algorithm [155].
Point/Divisor addition and doubling: Let the quintuple $[U_1, U_0, V_1, V_0, Z]$ stand for $[x^2 + u_1x + u_0, v_1x + v_0] = [x^2 + \frac{U_1}{Z}x + \frac{U_0}{Z}, \frac{V_1}{Z}x + \frac{V_0}{Z}]$. This form allows us to complete both point operations without inversion. Only one inversion and four multiplications are required at the end to convert back from projective to affine coordinates. We used the formulae from [45] for the doubling and we used the same approach to get the formulae for the addition in the case of mixed coordinates. The addition for type II curve has the same complexity as the one of Lange [138], *i.e.*, it takes $44M$, but the doubling has been further optimized to $31M$ (here M denotes number of multiplications/squarings).
Finite field arithmetic: We used the polynomial basis representation with irreducible polynomials in $\mathbb{F}_{2^{163}}$ and $\mathbb{F}_{2^{83}}$. The field addition of two vectors in

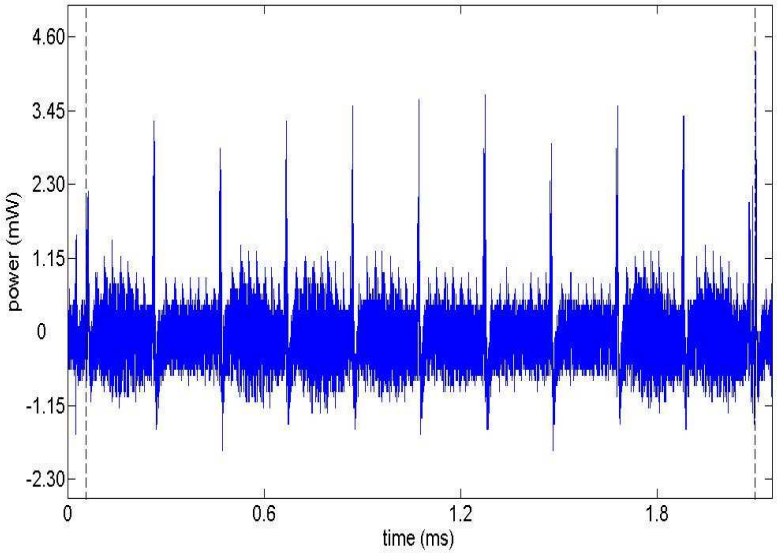


Figure 4.16: Power consumption trace of an EC point doubling that features 10 steps.

hardware or software in $\mathbb{F}_{2^n}$ is simply the bit-wise xor of the two vectors. While the inversion algorithm is actually more complex, it is only performed a single time (for the case of projective coordinates) and hence it is not the bottleneck in the implementation. The field multiplication is the most costly operation in our system.

4.7 FPGA Implementation

In this section we describe the architecture of the HECC and the ECC designs in a top-down manner. In Sect. 4.7.1 the top-level architecture and the main controller are described. Sections 4.7.2, 4.7.3 and 4.7.4 show the details of the top-level components, which are activated by the main controller.

4.7.1 Top-level Architecture

The top-level architecture is similar for the ECC and the HECC designs. They both use two finite field operation blocks (a Montgomery multiplier and an adder) and four Finite State Machines (FSMs). Figure 4.19 shows the schematics of the

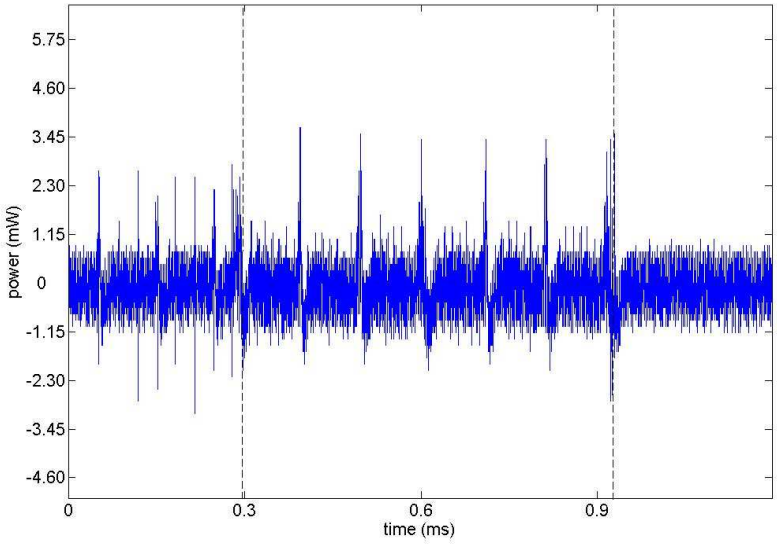


Figure 4.17: Power consumption trace of an EC point addition as in Algorithm 4.2 (6 field multiplications are easily detected).

top-level architecture. The main controller FSM consists of 13 states for ECC and 17 states for HECC. The Algorithmic State Machine (ASM) chart of the FSM looks similar for both implementations. It is depicted in Fig. 4.20. The following happens in the different states:

Conversions (CONV1): The curve parameters and the coordinates of the points and divisors are converted to the Montgomery representation using the Montgomery multiplier. For ECC this state consists of 4 Montgomery multiplication steps (MUL) and for HECC of 5 steps.

Point doubling (DOUBLE) and point addition (ADD): In these states the doubling and the addition FSMs are executed, respectively. They are explained in more detail in Sect. 4.7.2.

Inversion (INV): In order to convert back from projective to affine coordinates a modular inversion of the Z-coordinate needs to be performed. This is done using a separate FSM, as explained in Sect. 4.7.3.

Conversions (CONV2): First, a conversion to affine coordinates is performed using the output of the inversion block. This takes 3 and 4 multiplications

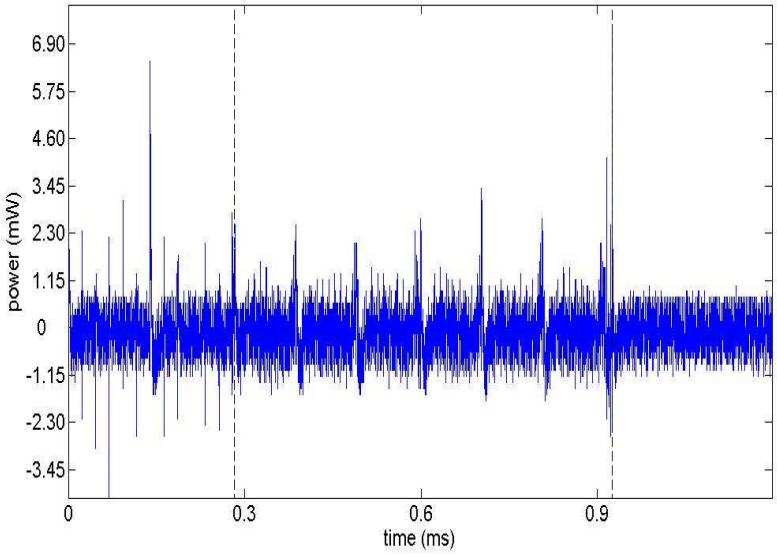


Figure 4.18: Power consumption trace of an EC point doubling as in Algorithm 4.2 includes also 6 multiplications.

for the ECC and the HECC implementations, respectively. Next, the affine coordinates are transformed from the Montgomery to the normal representation. For this, 2 multiplications need to be performed for ECC and 4 for HECC.

4.7.2 Doubling and Addition

The doubling and addition FSMs execute the operations of the point/divisor doubling and addition. Both FSMs use the Montgomery multiplication and addition blocks of the top-level architecture several times. Considering the fact that a Montgomery multiplication and an addition can be performed in parallel, we get the results that are listed in Table 4.3.

4.7.3 Modular Inversion

The modular inversion is performed using Fermat's little theorem (cf. Chapter 2). In the modular inversion FSM (see Figure 4.21) one multiplication is performed in

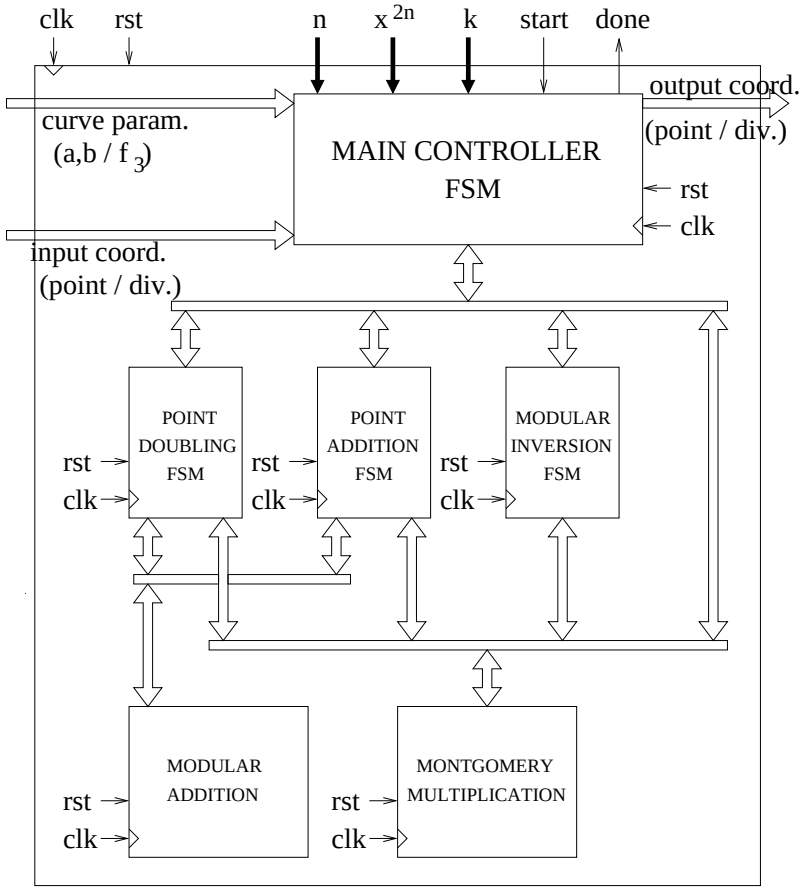


Figure 4.19: Architecture of the ECC/HECC processor.

every state. This corresponds to $2n - 3$ multiplications for one modular inversion.

4.7.4 Montgomery Multiplication and Modular Addition

The Montgomery multiplier is depicted in Fig. 4.22. The circuit is digit-serial and is described with more details in [159]. The picture only shows the combinatorial part of the multiplier. Actually, the output of the systolic array is led back to the input on every rising clock edge.

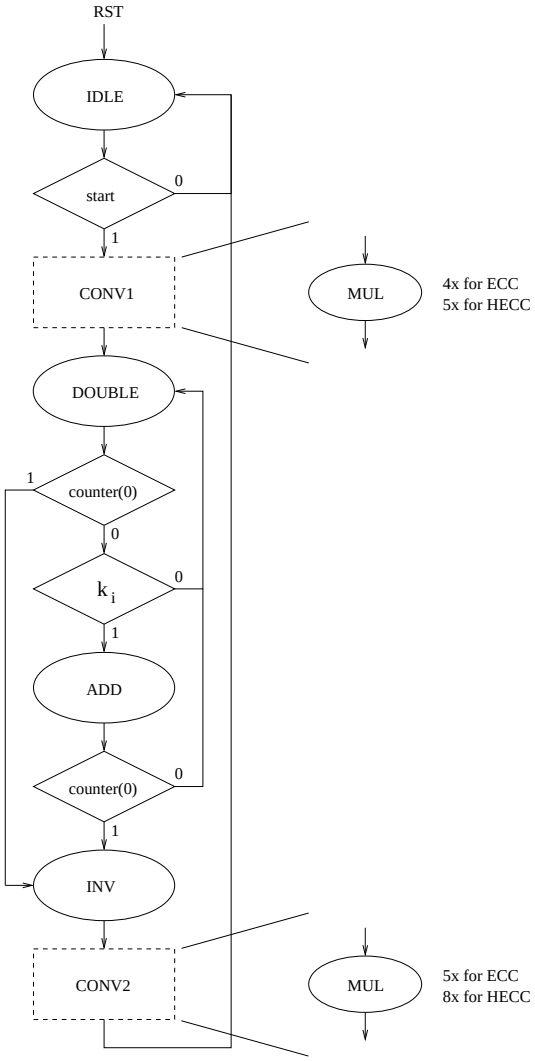


Figure 4.20: Flowchart of the main FSM. Here, counter(0) is a control bit checking wether all key-bits have been evaluated.

4.8 Results

We have implemented both algorithms on a Xilinx Virtex-2 Pro FPGA and we have used Xilinx tool for synthesis. Table 5.4 gives the performance results for

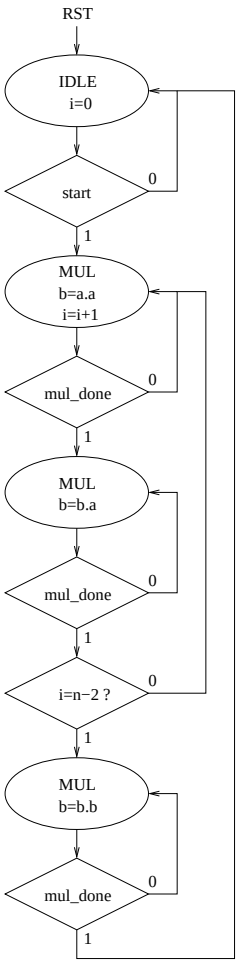


Figure 4.21: Flowchart of the FSM for the inversion.

ECC and HECC operations that were obtained after synthesis. The finite-field multiplier is smaller due to shorter bit-lengths of the parameters, but the parts for Addition and Doubling of points/divisors are larger in the case of HECC. The results in this table are obtained by use of the bit-serial multiplier *i.e.* $w = 1$ for the purpose of evaluating different blocks in the architecture.

We also compared our performance results with other relevant work on FPGA platforms (Table 5.6). These results are obtained by use of the digit-serial version of the Montgomery multiplier, where the digit-length w is $w = 16$. The work

Table 4.3: Comparison of the number of states and the number of temporary registers for the ECC and HECC point doubling and addition.

	number of states	number of temporary registers
ECC doubling	11	4
ECC addition	15	6
HECC doubling	33	15
HECC addition	48	28

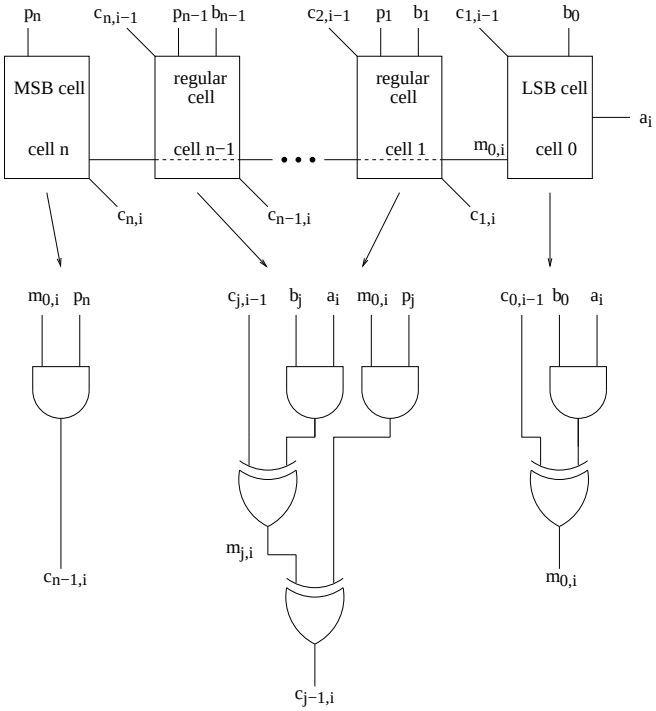


Figure 4.22: Schematic of the Montgomery multiplier.

of Kim *et al.* [123] presents a faster implementation, but the authors used affine coordinates with a dedicated hardware inverter. Furthermore, we focus on a fair comparison and both ECC/HECC could be further optimized. For example, we used the basic "double-and-add" algorithm for point/scalar multiplication, while some windowing method or even the NAF algorithm would lead to substantial

Table 4.4: Implementation results for operations in $\mathbb{F}_{2^{83}}$ and $\mathbb{F}_{2^{163}}$ for ECC and HECC respectively.

Operation	# slices	# in. LUTs	τ $[ns]$	freq. $[MHz]$
Multiplication: ECC	461	688	2.601	384.5
HECC	235	351	2.591	386
Div/P. Double: ECC	1994	3693	4.573	218.7
HECC	3720	6876	4.523	221.1
Div/P. Add: ECC	2740	5045	4.218	237.1
HECC	4808	8885	4.703	212.6
Scalar Mult.: ECC	8769	16293	5.521	181.1
HECC	11296	20999	4.693	213.1

improvement in performances. Yet, Table 5.6 shows that we have obtained a practical ECC/HECC implementations in hardware.

Table 4.5: Comparison of various implementation results for point/divisor multiplication on FPGAs.

Reference	PKC	Field	# slices	Freq. [MHz]	Perf. $[ms]$
[52]	HECC	$\mathbb{F}_{2^{83}}$	22000	-	10
[123]	HECC	$\mathbb{F}_{2^{89}}$	9950	62.9	0.436
this work	ECC	$\mathbb{F}_{2^{163}}$	8769	153	0.667
this work	HECC	$\mathbb{F}_{2^{83}}$	11296	166	0.496

4.9 Conclusions and Future Work

In this chapter a complete ECC processor for binary fields is presented. An FPGA implementation is described that includes a bit/digit-serial multiplier that combines two previously known multiplication methods. The proposed architecture is a systolic array that allows for good performance in speed and more parallelism in operation which is also beneficial for side-channel security. Furthermore, we propose a fully balanced point multiplication algorithm that performs the same operations for every loop of the algorithm. By using this approach we proved that this so-called side-channel aware design is the first step towards side-channel resistance. However, it is clear that additional countermeasures might be required.

This work is also the first one to provide the comparison of ECC and HECC on the same FPGA platform. The results are somewhat unexpectedly in favor of HECC. Yet, none of the previous comparisons was performed on the same hardware platform, but on an ARM7 [254] or some general purpose processors. The excellent HECC performance can be credited to the full scalability of the multiplier; the architecture for HECC which handles "only" 83-bit long parameters could benefit from that. On the other hand we did use a special case of a hyperelliptic curve, but it is likely that this special case will later be included in standards. This form of a curve also allows for more optimized doubling which improves the performance of HECC further. Future work will deal with the optimizations of this FPGA solution and will address the simple side-channel security in a form of balancing the divisor operations.

4.10 Related Work on HECC Hardware Architectures

There is not much previous work on hardware implementations of HECC. The first complete hardware implementation of HECC was given by Boston *et al.* [39]. They designed the coprocessor for genus two curves over $\mathbb{F}_{2^{113}}$ and implemented it on a Xilinx Virtex II FPGA. The algorithm of Cantor was used for all computation on Jacobians. On the other hand, the work of Elias *et al.* [70] used Lange's explicit formulae. The results reported were the fastest in hardware at the time. Wollinger *et al.* [253] investigated an HECC implementation on a VLSI coprocessor. They used projective coordinates and completed their research on VLSI platforms started in [25, 24]. They compared coprocessors using affine and projective coordinates and concluded that the latter should be preferred for hardware implementations. They used a curve of a special form ($y^2 + xy = x^5 + f_1x + f_0$), which allowed for more optimized formulae. In [123] three different architectures on an FPGA have been examined for a broad range of applications.

Our work was the first one to compare ECC and HECC on the same platform and using the same low-level hardware blocks. We show that, with improved HECC formulae, performances of both curve-based cryptosystems are similar. More precisely, if we use the "area-time product" as a metric for the efficiency of hardware implementation (as suggested by Wolinger [252]) we get almost the same figure for both ECC and HECC.

Chapter 5

HW/SW Co-design for Curve-based Cryptography on the 8051 μP

Implementing public-key cryptography on platforms with limited resources, such as microprocessors, is a challenging task. Hardware/software co-design is often the only answer to implement the computationally intensive operations with limited memory and power at an acceptable speed. Here we describe such a solution for Elliptic and Hyperelliptic Curve Cryptography (ECC/HECC).

The proposed hardware/software co-design of both systems was implemented and co-simulated using the GEZEL design environment [88]. As a low-cost platform, we chose an 8-bit 8051 microprocessor to which one small hardware co-processor was added for field multiplication. First we show that the point/Jacobian scalar multiplication for ECC/HECC can be computed in 3.97/2.488 sec at 12 MHz on this platform if a minimal hardware module is added *i.e.* a hardware multiply-add unit. These results prove that HECC is a more suitable choice than ECC on similar low-power and low-footprint architectures. The optimal hardware/software solutions provide in both cases a substantial speed-up over software-only solutions. Further on, we explore another option for a hardware/software partitioning and the parallelism for the divisor operations in the case of HECC. This solution allows for the speed-up in performance of almost a factor 4 with a small increase of hardware resources. The results were published in [11, 10, 106].

Almost all existing ECC and HECC implementations consider binary fields and for HECC mainly curves of genus two or three are deployed; the latter is motivated by security reasons [86, 223]. Software implementations were developed on general purpose processors and on embedded microprocessors *e.g.* on an ARM [182, 254] and some research has been performed on hardware implementa-

tions [39, 25, 24, 123, 253]. However, we describe the first HECC implementation using a hardware/software co-design. More precisely, we have implemented the HECC divisor multiplication on the 8051 microprocessor, which uses a small hardware co-processor to optimize the performance. To allow a fair comparison, we have implemented an EC cryptosystem on the same platform and for the first time the comparison heavily favors HECC. More precisely, performance results show that HECC allows a shorter latency while at the same time less extra hardware is required. This was just the first step towards exploring all possibilities for hardware/software co-designed ECC/HECC implementations. Such an investigation is of special interest as embedded devices are believed to be of vital importance for a broad area of pervasive computing such as sensor networks and wireless applications.

First we examined the pure software implementations. The results show that a software-only solution is not an option in these constrained environments. Therefore, some small extra hardware was added, which facilitates the field operations, in particular the inversion and multiplication in the binary field. We conclude that even with very limited hardware resources one can obtain an attractive performance. For the case of ECC we chose so-called López and Dahab projective coordinates [103] as they feature the smallest number of multiplications per point doubling. For HECC we used the formulae of Byramjee and Duquesne [45] to achieve an optimized divisor doubling operation as well. For the optimal hardware/software co-design we used GEZEL as a design environment. GEZEL is especially suitable for the exploration of domain-specific coprocessor and multiprocessor micro architectures as it can provide cycle-true hardware/software cosimulation with various embedded core instruction set simulators. More details on GEZEL are given in Section 5.3.

5.1 Hyperelliptic Curves Cryptograhy (HECC)

Here we consider a hyperelliptic curve C of genus $g = 2$ over $\mathbb{F}_{2^n}$, which is given with an equation of the form:

$$y^2 + (h_2x^2 + h_1x + h_0)y = x^5 + f_4x^4 + f_3x^3 + f_2x^2 + f_1x + f_0\,. \tag{5.1}$$

For our implementation we used so-called type II curves [45], which are defined with $h_2 = 0, h_1 \neq 1$. In particular, the authors of [45] recommended to use curves of the form: $y^2 + xy = x^5 + f_3x^3 + x^2 + f_0$, since they combine simpler arithmetic with a good security level.

A divisor D is a formal sum of points on the hyperelliptic curve C *i.e.* $D = \sum m_P P$ and its degree is $degD = \sum m_P$ as defined in Chapter 2. Let Div denote the group of all divisors on C and Div_0 the subgroup of Div of all divisors with degree zero. The Jacobian J of the curve C is defined as the quotient group $J = Div_0/P$. Here P is the set of all principal divisors. For implementations of

HECC, we need to implement the multiplication of an element of the Jacobian *i.e.* a divisor with some scalar.

5.2 Algorithms for HECC

Here we give some details of the algorithms that were used for our implementations.

5.2.1 Point/Divisor Multiplication

As explained in Chapter 2 the point/divisor scalar multiplication is achieved by use of point/divisor addition and doubling. We used the NAF algorithm [29] (Algorithm 5.1) to reduce the number of additions.

Algorithm 5.1: Left to right NAF for ECC/HECC scalar multiplication

Require: Divisor $D = [a, b]$, scalar k in NAF form $k = (k_l, k_{l-1}, \cdots, k_0)$, $k_i \in \{-1, 0, 1\}, k_l = 1$
Ensure: $R = [a', b'] = kD$
1: $R \leftarrow 0$
2: **for** i from l downto 0 **do**
3: $\quad R \leftarrow 2R$
4: $\quad$ **if** $k_i = 1$ **then** $R \leftarrow R + D$
5: $\quad$ **if** $k_i = -1$ **then** $R \leftarrow R - D$
6: return R

Point Addition and Doubling

We chose so-called López and Dahab projective coordinates [103]. In this case, the point double can be calculated by 9 field multiplications *i.e.* $9M$ by use of 2 temporary variables. The point addition can be calculated by $13M$ and by use of 3 temporary variables if we apply mixed coordinates. Here M denotes number of multiplications/squarings.

Divisor Addition and Doubling

For implementation of HECC we also used projective coordinates [138]. For affine coordinates a divisor is usually represented as $D = [u, v] = [x^2 + u_1x + u_0, v_1x + v_0]$. Let the quintuple $[U_1, U_0, V_1, V_0, Z]$ stand for $[x^2 + u_1x + u_0, v_1x + v_0] = [x^2 + \frac{U_1}{Z}x + \frac{U_0}{Z}, \frac{V_1}{Z}x + \frac{V_0}{Z}]$. This form allows us to complete both point operations without inversion. Only one inversion and four multiplication are required at the end to convert back from projective to affine coordinates. We used the formulae from [45] for doubling and we used the same approach to get the formulae for

addition in the case of mixed coordinates. The addition for type II curves has the same complexity as the one of Lange [138] *i.e.* it takes $44M$, but the doubling has been further optimized in [45] to $31M$. The formulae for the divisor addition and doubling are given in Chapter 2.

Finite Field Arithmetic

As examples we used the binary fields $\mathbb{F}_{2^{83}}$ and $\mathbb{F}_{2^{163}}$ for ECC, HECC respectively. We used the polynomial basis representation with the irreducible polynomial being a pentanomial in both cases. Each element of the field can be represented as an 11/21-byte word. The field addition of two vectors in hardware or software in $\mathbb{F}_{2^n}$ is simply the bitwise xor of the two vectors. The field multiplication is the most costly operation in our system, since it is performed thousands of times during the course of a single divisor multiplication. While the inversion algorithm is actually more complex, it is only performed a single time (for the case of projective coordinates) and hence it is not the bottleneck in our initial implementation. We discuss our choices for the field multiplication in more detail in Sect. 4.

5.3 Implementations

5.3.1 8051 Microprocessor

Here we give a brief overview of the 8051 microprocessor platform. An 8051 is an 8-bit microcontroller originally designed by Intel that consists of several components: a controller and instruction decoder, an ALU, 128 bytes of internal memory (IRAM), up to 64K of external RAM (XRAM) addressed by a 16-bit DPTR register and up to 64KB of external program memory or 4KB of internal program memory (ROM). The 8051 also has 128 bytes of special function registers (SFRs), which are used to store system values such as timers, serial port controls, input/output registers, etc. The architecture is shown in Fig. 5.1, which is based on the Dalton 8051 core from UC Riverside [61].

An external RAM module (XRAM) can be attached to the 8051 core when the 128 bytes of internal RAM are insufficient, which is often the case in public-key cryptosystems. The 8051 interfaces to the outside world via a serial port as well as four input/output register ports, labeled P0 through P3.

It should also be noted that the 8051 in its original form relies on a clock division principle. That is, the external clock entering into the device is actually divided by 12 to produce the system clock. Thus, a 12 MHz external clock would produce a 1 MHz machine clock frequency inside the 8051, with most instructions requiring 1 or 2 machine cycles. Newer 8051 cores attempt to reduce the clock division [60]. The clock division principle can serve as an advantage to co-designed systems in that the coprocessor circuitry can inherently operate at 12x the internal

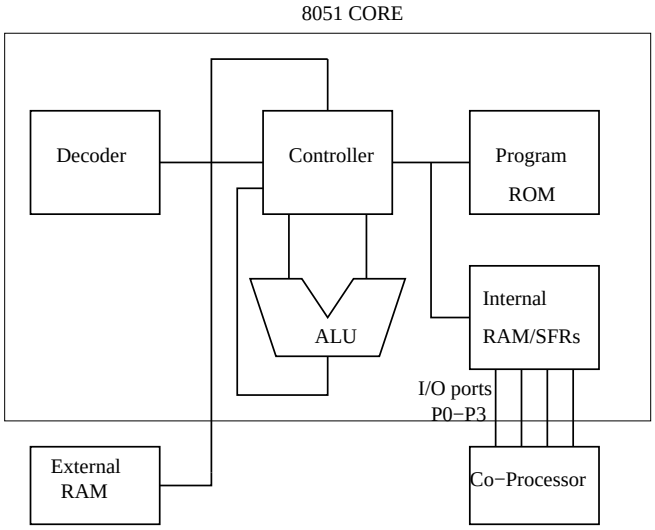


Figure 5.1: The architecture of the 8051 microprocessor.

8051 machine rate.

5.3.2 Implementation Options for ECC and HECC

We present two types of ECC/HECC implementations on the 8051 processor. The first type is a pure software implementation obtained as a C model operating on the 8051. The second type is a mixed hardware/software model in which some of the functions are performed in C while the $\mathbb{F}_{2^n}$ finite field operations (multiplication/addition/inversion) are performed in hardware. (In the case of ECC/HECC $n = 163/83$ respectively.) The hardware operators and the 8051 are connected by a memory-mapped interface, over the 8051's P0, P1, and P2 I/O port interfaces.

Software Implementation

The first C implementation was compiled onto the 8051 processor using the Keil suite.

Multiplication: In the software implementation, we used a modified form of Algorithm 4 of [102] to implement a fast multiplication. The algorithm is a fast comb-based multiplication method implemented for a 32-bit processor with a window size of 4. Based upon initial simulation results, for an 8-bit processor, we found that a window size of 2 provides faster performance.

Reduction: To reduce the multiplication result by the irreducible polynomial, a fast reduction technique was used. This technique was based on Algorithm 6 of [102]. We have used a similar approach but modified the algorithm to implement the reduction using our $\mathbb{F}_{2^n}$ pentanomial and a word-size of 8 bits.
Inversion: The inversion function for this case is implemented by means of Fermat's little theorem [129].

In addition, we present two different options for hardware/software ECC/ HECC implementations on the 8051 processor. For both solutions software routines were enhanced with binary field operations in hardware. In the first attempt we implemented a data path which includes a hardware $\mathbb{F}_{2^n}$ multiplier. For the second one the hardware multiplier was replaced with a $\mathbb{F}_{2^n}$ "multiply-and-add" data path. The general methodology is given in Fig. 5.2. Figure 5.3 shows the

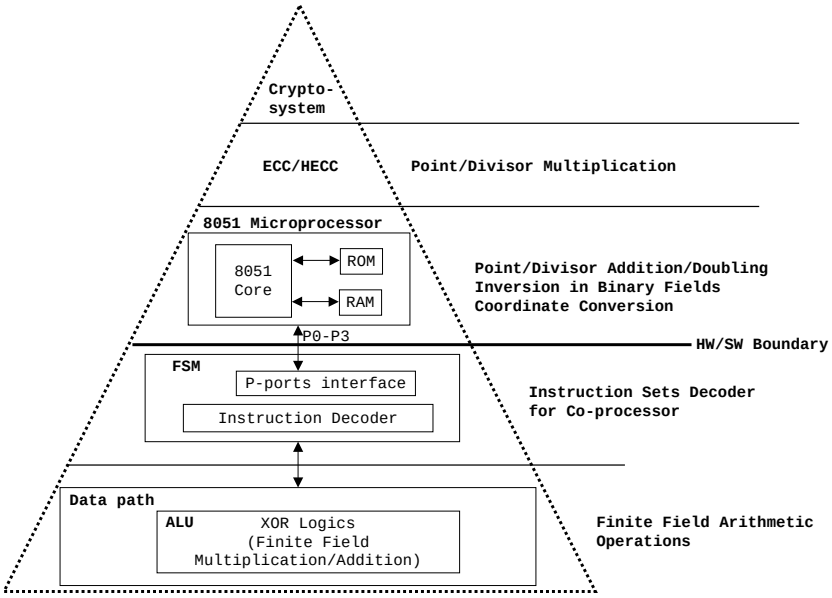


Figure 5.2: The general strategy for hardware/software co-design for curve-based cryptography.

data path. The data I/O ports from the 8051 processor are 8 bits long and the multiplication is performed on the $\mathbb{F}_{2^n}$ operands. There is an instruction register that controls the hardware data path from the 8051 processor. The supported instructions for the data path of Fig. 5.3 are as shown in Table 5.1:

Table 5.1: Instructions for the data path.

Instr.	Definition
`LOADA`	Load 8-bits of data from the 8051 to Reg. A of HW data p.
`LOADB`	Load 8-bits of data from the 8051 to Reg. B of HW data p.
`DOMULT`	Perform $\mathbb{F}_{2^n}$ mult. on A and B and put the results in C
`GETC`	Return 8-b. of data from Reg. C of HW data p. to the 8051

As the data is transferred back and forth from the CPU to the hardware multiplier there is a lot of I/O overhead. In order to optimize the total performance we tried to reduce the I/O transfers with minimum additional memory storage added to the data path. The key observation is that in the schedule of the divisor's double and add operations (see Table 2.2) there are many expressions of the following form: $d = a \cdot b + c$.

Initially for such expression, a and b were moved to the hardware multiplier, the multiplication was performed in the hardware, then the result was returned back to the CPU and the addition with c was performed in the software. In order to speed up this expression the hardware multiplier was replaced with a $\mathbb{F}_{2^n}$ "multiply-and-add" data path. For this purpose a hardware adder and a feedback line that can keep the result of the multiplication in hardware was added to the original data path and therefore, the number of I/O transfers decreased with not much of extra hardware. For the new data path (Fig. 5.4), the instructions shown in Table 5.2 were added.

Table 5.2: The new instructions for the data path.

Instruction	Definition
`MOVE_CTOB`	Move the data in Register C to Register B
`DOADD`	Perform $\mathbb{F}_{2^n}$ addition on A and B and put the res. in C

Moreover, in the software routines that implement the divisor's double and add operations, we moved the coprocessor's instructions up and down in the schedules of the divisor's operations, so that we do not have to repeatedly load the same values into the internal register A of the data path. The performance gain of these optimizations will be provided in the next section.

In addition, for the best performance in the final HW/SW implementation on the 8051 processor, the $\mathbb{F}_{2^n}$ inversion operation was performed in HW. The same HW data path is used to implement the inversion algorithm which consists of repeated multiplications. The details of the hardware $\mathbb{F}_{2^n}$ multiplication and

inversion are given after introducing our design environment.

Design Environment: At this stage we briefly introduce the design environment GEZEL in which we model the co-designed system. In our application, we used the Dalton 8051 ISS to perform cycle-accurate simulations for our software-only implementation. For the hardware/software system, we designed our co-processor multiplier using GEZEL's hardware description language. The language syntax is primarily used to describe the FSMD (finite state machine plus data path) system model. Thus, a data path for the co-processor was designed and its corresponding control logic was also designed in the GEZEL language.

After the design of the hardware co-processor, we have attached the co-processor to the input/output ports of the 8051 ISS (P0-P3) using the GEZEL design environment, and then performed timing and functional verification. GEZEL gave us the ability to co-simulate the 8051 with clock division circuitry as it interfaced with a 12 MHz hardware module in a cycle-exact manner. Upon verification of the functionality of the multiplier co-processor, the GEZEL code was automatically converted to RTL VHDL and input into Synplicity for FPGA synthesis.

Multiplier: In the first version, the multiplier implements a finite field multiplication and simultaneously a corresponding reduction in a bit-serial implementation. A bit-serial implementation was chosen for area compactness as well as to take advantage of the 12x increase in effective clock rate of the co-processor over the 8051 core. In the second version, the multiplier was enhanced with the additional "multiply-and-add" instruction and data path element, as described previously.

Inversion: Inversion in binary fields can be performed by a chain of multiplica-

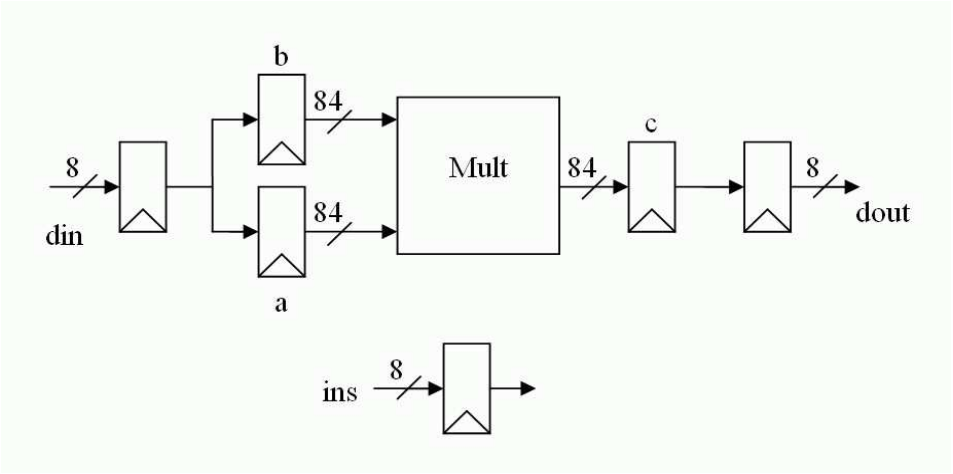


Figure 5.3: Data path for the initial HECC design.

tions (and squarings). It is of interest if squarings are faster than multiplication such as for normal bases. First by means of Fermat's little theorem we have: $a^{-1} = a^{2^n-2} = (a^{2^{n-1}-1})^2$, for all $a \in \mathbb{F}_{2^n}$. The technique to compute this in an optimal way is the basis for the idea of Itoh and Tsujii [112]. Their method is especially suited for normal bases but can be applied on polynomial bases as well. Here we consider the case for n odd, so $n-1$ is even. Then we can write:

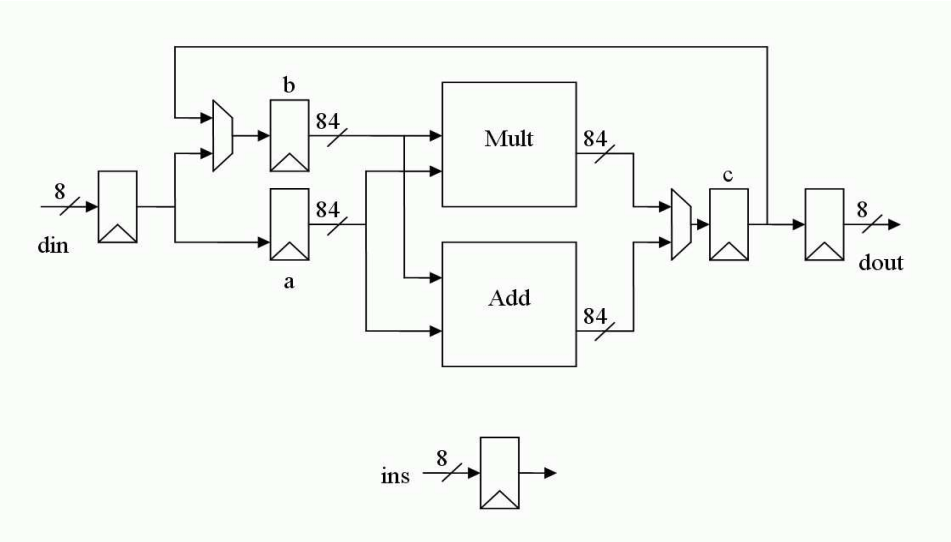


Figure 5.4: Data path of the new co-processor (HECC).

$$a^{2^{n-1}-1} = a^{(2^{\frac{n-1}{2}}-1)(2^{\frac{n-1}{2}}+1)} = (a^{2^{\frac{n-1}{2}}-1})^{2^{\frac{n-1}{2}}} a^{2^{\frac{n-1}{2}}-1}.$$

In our case for $\mathbb{F}_{2^{83}}$ we get:

$$a^{-1} = a^{2^{83}-2} = (a^{2^{82}-1})^2 = ((a^{2^{41}-1})^{2^{41}} a^{2^{41}-1})^2,$$

which means that we need to use formula for $a^{2^{n-1}-1}$, but now $n-1$ is odd. In this case:

$$a^{2^{n-1}-1} = aa^{2^{n-1}-2} = a(a^{2^{n-2}-1})^2.$$

By repeated use of these formulae we can compute the inverse by only $90M/171M$ respectively for ECC/HECC. The total number of multiplications (or squarings) required to compute an inverse in $\mathbb{F}_{2^n}$ is given by $\lfloor log_2(n-1) \rfloor + w(n-1) - 1$. Here $w(k)$ denotes the Hamming weight of the positive integer k. The example with a sequence of computations for the case of HECC is given in Table 5.3.

Table 5.3: Example of inversion in $\mathbb{F}_{2^{83}}$.

Calculation	Number of S/M
$a^{2^2-1} = a \cdot a^2$	1S+1M
$a^{2^4-1} = (a^{2^2-1})^{2^2} \cdot a^{2^2-1}$	2S+1M
$a^{2^5-1} = a \cdot (a^{2^4-1})^2$	1S+1M
$a^{2^{10}-1} = (a^{2^5-1})^{2^5} \cdot a^{2^5-1}$	5S+1M
$a^{2^{20}-1} = (a^{2^{10}-1})^{2^{10}} \cdot a^{2^{10}-1}$	10S+1M
$a^{2^{40}-1} = (a^{2^{20}-1})^{2^{20}} \cdot a^{2^{20}-1}$	20S+1M
$a^{2^{41}-1} = a \cdot (a^{2^{40}-1})^2$	1S+1M
$a^{2^{82}-1} = (a^{2^{41}-1})^{2^{41}} \cdot a^{2^{41}-1}$	41S+1M
$a^{-1} = (a^{2^{82}-1})^2$	1S
Total	82S+8M = 90M

5.4 Results

Here we give detailed results of our implementations and we discuss them further. In Table 5.4 the timings for all finite field operations are given for hardware and software. Timings for all basic operations are shown and in the last row, the "multiply-and-add" operation is also added. These figures do not include calculations only, but also numerous data transfers. Pure software figures are included to prove that the "software-only" option is not feasible. Namely, we concluded that although our software implementations could possibly be further optimized, it would still be difficult to achieve an efficient ECC/HECC implementation.

Another observation is that the numbers for addition and multiplication in hardware are the same. The reason for that is because the majority of the time for multiplication and addition on hardware is spent on the I/O transfers. Therefore, the time to perform a single multiplication (83/163 cycles) or an addition (1 cycle) is not more than just a few 8-bit I/O transfers from the 8051 to the accelerator or from the 8051 to the XRAM. Moreover, for the same reason, the time to compute a single multiplication is also very close to the time that is required to compute $ab + c$. However, the fact that this operation is used repeatedly allowed for a speed-up in the new data path. The sizes of XRAM and ROM are given in bytes (B).

The results for the scalar multiplication of a point/divisor for various implementation options are given in Table 5.5. The FPGA area is given in number of LUTs without XRAM and ROM which are specified separately. As can be seen in Table 5.5, there is a significant increase in performance when the field multiplication is moved into hardware. An additional timing reduction occurs after the

Table 5.4: Implementation results for operations in $\mathbb{F}_{2^{83}}$ for hardware and software routines.

Operation	Perf.[Cl. Cyc.]	Perf.[ms]	XRAM[B]	ROM[B]
Add. (SW)	38 K	3.2	54	608
Mult. (SW)	650 K	54.1	122	2065
Inv. (SW)	467.2 M	38.9 K	160	2383
Add. (HW)	28.2 K	2.3	53	934
Mult. (HW)	28.2 K	2.3	53	934
Inv. (HW)	788.5 K	65.7	75	1835
$ab + c$ **(HW)**	30.5 K	2.5	44	942

point operation signal flow graphs are analyzed and manipulated, and the new "multiply-and-add" operation is used. This reduction is a bit bigger for the case of HECC (40%) because the particular formulae for the divisor operations allowed to take more benefit of this option than for the case of ECC (28%).

Comparing ECC and HECC figures, we observe that HECC not only provides better performance, but also deploys a smaller hardware module. This result presents a unique observation among all previously published work. Namely, most of the researchers in the field concluded that, so far HECC does not give any advantage when compared with ECC. However, this platform proved that to be a wrong conclusion. HECC offers some benefits especially on embedded platforms mainly due to shorter operands' bit-lengths. Also that fact results in less extra hardware as the multiplier for HECC is twice as small as the one for ECC. The only figure that still favors ECC is the amount of ROM, which is due to more complex divisor operations.

Additionally, we have calculated figures for the co-processor's usage and the actual time distribution that is spent on I/O accesses. The co-processor's usages in total number of cycles are 0.170% and 0.486% for ECC and HECC respectively. This means that the performance is actually achieved with a very low hardware utilization. This fact showed that there are even more promising solutions to be explored. Namely, by adding some more hardware one could achieve a speed-up in performance and still maintain a low-cost and low-power solution. This observation led to our next step.

Now we compare our performance results with other work on embedded processors. Table 5.6 shows that our results feature practical implementations in constrained environments. First, it should be mentioned that it is extremely difficult to compare the performance of cryptographic primitives on different embedded processors, since each processor presents a unique architecture and memory structure. The discussion below is primarily to reference prior art.

Table 5.5: Implementation results for point/divisor multiplication.

Implementation	FPGA [LUTs]	XRAM [Bytes]	ROM [Bytes]	Perf.[s] @12MHz
ECC:SW	3300	980	7597	144.5
HECC:SW	3300	1186	13926	149.8
ECC:C+HW multiplier (Fig. 5.3)	3868	980	7597	5.52
ECC:C+HW multiplier (Fig. 5.4)	4210	910	8739	3.97
HECC:C+HW multiplier (Fig. 5.3)	3600	927	12789	4.1518
HECC:C+HW multiplier (Fig. 5.4)	3781	936	11524	2.488

The first two references for HECC relate to the ARM7, which is a 32-bit platform and features a completely different architecture than the 8051. Even so, the second reference is of the same order as ours using frequency scaling for rough normalization. The most suitable comparison to this work is [136] and [98]. Gura *et al.* achieve the shown performance using a "faster" 8051, *i.e.* an 8051 whose clock division was much less than 12x. They also demonstrate the well-known fact that the AVR is much faster than the 8051, due to the AVR RISC (vs. 8051 CISC) architecture and the fact that the AVR does not require clock division circuitry. This provides perspective when comparing to the ECC implementation of Kumar and Paar [136].

Table 5.6: Implementation results for divisor multiplication on various embedded platforms.

Ref.	PKC	Field	Platform	Freq.[MHz]	Perf.[ms]	Perf./F.
[184]	HECC	$\mathbb{F}_{2^{83}}$	ARM7	80	71.56	0.894
[7]	HECC	$\mathbb{F}_{2^{80}}$	ARM7	80	374	4.675
[136]	ECC	$\mathbb{F}_{2^{163}}$	AVR	4	113	28.25
[98]	ECC	$\mathbb{F}_p$	8051	12	4580	381.67
our	HECC	$\mathbb{F}_{2^{83}}$	8051+Acc.	12	2488	207.33
our	ECC	$\mathbb{F}_{2^{163}}$	8051+Acc.	12	3970	330.83

5.5 HW/SW Co-Design of a Hyperelliptic Curve Cryptosystem using a Microcode Instruction Set Coprocessor

As a next step in exploring various solutions for hardware/software partitioning, such as shown in Fig. 5.2, we decided to move the hardware/software boundary more up. Therefore, a microcode instruction set coprocessor was designed to work with an 8-bit 8051 microcontroller in order to implement HECC more efficiently. The Gezel design environment allows us to explore design options for co-processor design to the 8051 microprocessor. Communication between processor and co-processor in the case of the 8051 is a bottleneck. Therefore, the idea is to move even more functionality to the co-processor. This co-processor then becomes an Application-Specific Instruction Set co-processor suitable for secure embedded systems.

5.5.1 Choice of Algorithms

As in the previous case we used the NAF algorithm and the projective coordinates. We have re-written the formulae from [45] for the doubling to achieve almost full parallelism for field multiplications. We also used the same approach to get the formulae for the addition in the case of mixed coordinates. The formulae for both, the parallelized addition and doubling are given in Table 5.7 and Table 5.8 respectively. By using this approach the formulae for addition include $23M$ as the longest data path, when computations are performed on two separate multipliers. (In the case of single addition the total of multiplications is $44M$.) For the doubling operation the total of $17M$ is required in the case of two multipliers.

For the finite field arithmetic the same algorithms were used for field addition, multiplication and inversion as in our first hardware/software co-design case-study. The difference for this option is that dual-multiplier/dual-adder data path was used and a microcode instructions were added to perform various combinations of the field basic operations (addition and multiplications).

5.5.2 The Architecture

Our choice for a hardware/software partitioning is given in Fig. 5.5.

The hardware part of the architecture is given in Fig. 5.6. The coprocessor consists of a data path, RAM and the top controller. For communication between the microcontroller and the coprocessor four 8-bit ports are used. Two of them are for the input/output data and the other two are for coprocessors instruction and the address to access the RAM. Data transfers to the RAM are performed through the input-word and the output-word registers that are 84 bits wide. This bit-length is also used for the operations in the coprocessors datapath. As previously

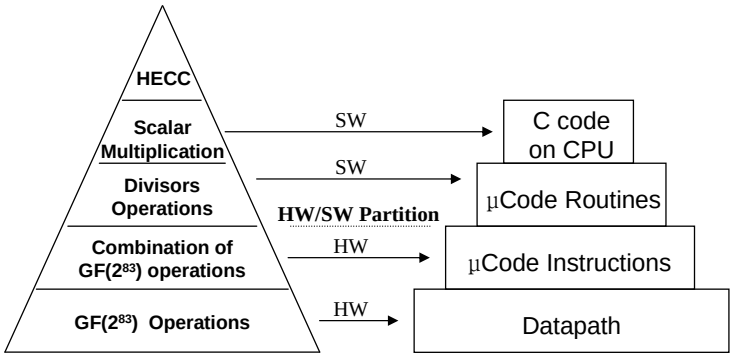


Figure 5.5: One option for a HW/SW partitioning for HECC implementations.

mentioned this option can be also viewed as an Application Specific Instruction Set Processor or coprocessor (ASIP).

The software part is divided into two parts. One part consists of routines of microcode instructions that implement the divisors operations and also the conversion of the final projective result to the affine coordinate. The second software part is the C code for scalar multiplication, which is compiled to microcontroller's assembly code. This partition of the software is implemented in C and calls the routines of divisors operations described above.

5.5.3 Results

The results of our implementations are given in Table 5.9. The total delay of $656\,ms$ ($7.8\,Mcycles$) is obtained, which present a speed-up of a factor almost 4, when compared with the first solution. This solution gives just one option for a division between hardware and software. By adding more software routines one can achieve a more flexible solution. Also, our parallelized formulae allowed for concurrent sequences of computations on two separate hardware units. In this way, inactivity of a hardware co-processor that was a bottleneck of our first solution, has been resolved. However, we believe that there are other options that could possibly improve this solution further with respect the variable of interest (*i.e.* latency, hardware complexity, power *etc.*).

5.6 Related Work on ECC/HECC Implementations over Binary Fields

Algorithms for ECC and HECC and implementations have been studied intensively in the past years. While ECC applications are highly developed in practice, the use of HECC has not reached that level yet. A significant amount of work has been performed on investigating the formulae for the group operation [138, 252, 183, 45]. Explicit formulae for genus 2 curves are given by Lange [138] for arbitrary fields and for various types of coordinates. There exist practical results for both software platforms (general purpose or embedded processor) [254, 182] and hardware devices, such as FPGAs [39, 123]. The most detailed and complete reference dealing with software as well as hardware implementations is [252].

For embedded processors, a large amount of work has been performed for the ARM platform [254, 184, 7, 182]. Pelzl *et al.* [182] have implemented the group operation of genus 2 and 3 for HECC on an ARM7 processor. They compared the results with an ECC implementation (with corresponding security) and showed that HECC performance is comparable to the one of ECC. The performance for divisor scalar multiplication on the ARM microprocessor for genus 2 was further optimized in [184] and compared to genuses 3 and 4. They proved that genus 3 is the fastest, requiring less than 70 *ms* on an ARM7 running at 80 MHz. The work of Wollinger *et al.* [254] considered not just the ARM7TDMI but also the ColdFire and a PowerPC. In addition, they provided the first thorough comparison of ECC and HECC on those platforms.

With respect to the platform, we mention here other relevant experiences with curve-based cryptography. Woodbury *et al.* [255] showed that EC point multiplication can be performed on an 8051 microcontroller in less than 2 sec as a pure software solution. However, they used a 134-bit OEF at a lower security level. Gura *et al.* [98] compared ECC and RSA on 8-bit CPUs and proved that Public-key Cryptography is viable on small devices.

For hardware/software co-design the only relevant work that we are aware of is the one of Kumar and Paar [136]. They implemented ECC on an 8-bit AVR microcontroller with some extra hardware for field multiplications. They show that a 163-bit point multiplication can be calculated in 0.113 sec with a microcontroller running at 4 MHz.

5.7 Conclusions and Future Work

This work is the first one showing that even on a small 8-bit processor one can implement hyperelliptic curve cryptography efficiently. It is also the first publication showing that on some platforms HECC outperforms ECC. A small hardware module that we designed results in a significant speed-up compared with

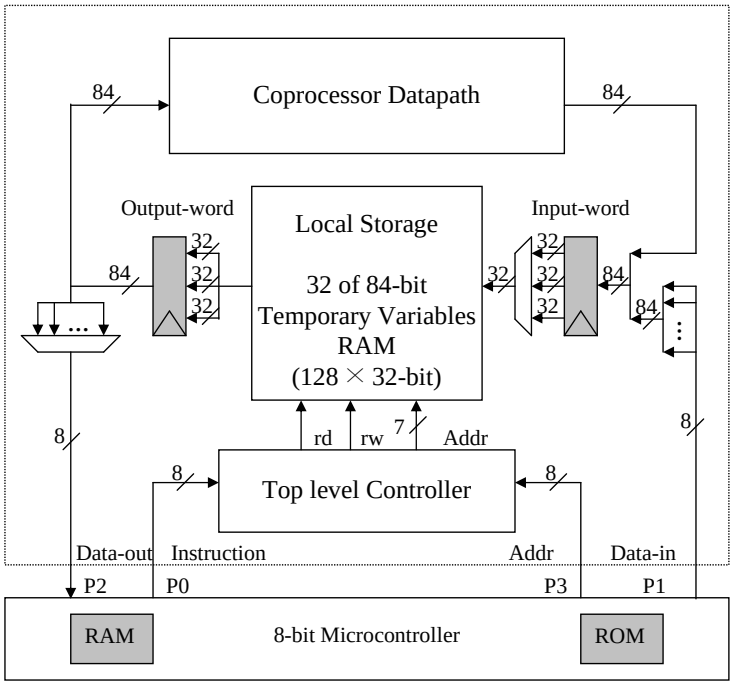


Figure 5.6: HW/SW partitioning of the HECC.

a software-only solution. Our results show that hardware/software co-design for HECC offers a new alternative for low-power and low-footprint devices. An 8051 micro-controller was chosen for its low-costs as such platforms are believed to be of vital importance for embedded security applications. The flexibility of the GEZEL environment allows us to experiment with other micro-controllers and different communication interfaces between processor and co-processor. The influence of the communication bandwidth and type of interface on the hardware/software boundary of Figure 5.5 is a topic of future research.

Table 5.7: Formulae for parallelized addition.

Step	Calculations		# mult.
1	*Precomputation and resultant r:*		$3M$
	$t_1 = U_{11} \cdot Z_2 + U_{21}$	$t_2 = U_{10} \cdot Z_2 + U_{20}$	
	$t_0 = U_{11} \cdot t_1 + t_2$	$a = {t_1}^2$	
	$r = t_0 \cdot t_2 + a \cdot U_{10}$		
2	*Compute almost s:*		$4M$
	$t_4 = V_{10} \cdot Z_2 + V_{20}$	$t_5 = V_{11} \cdot Z_2 + V_{21}$	
	$w_2 = t_0 \cdot t_4$	$w_3 = t_1 \cdot t_5$	
	$b = t_0 + t_1$		
	$b = b \cdot (t_4 + t_5)$	$a = w_3 \cdot (1 + U_{11})$	
	$b = w_2 + b$	$s_0 = w_2 + U_{10} \cdot w_3$	
	$s_1 = a + b$		
3	*Precomputations:*		$5M(4M)$[1]
	$R = Z_2 \cdot r$	$s_3 = s_1 \cdot Z_2$	
	$\widetilde{R} = R \cdot s_3$	$s_0 = s_0 \cdot Z_2$	
	$S = s_0 \cdot s_1$	${S_3} = {s_3}^2$	
	$\widetilde{S} = s_3 \cdot s_1$	$\widetilde{\widetilde{S}} = s_0 \cdot s_3$	
	$\widetilde{\widetilde{R}} = \widetilde{R} \cdot \widetilde{S}^{\,2}$		
4	*Compute l:*		$2M$
	$l_2 = \widetilde{S} \cdot U_{21}$	$l_0 = S \cdot U_{20}$	
	$l_1 = (\widetilde{S} + S) \cdot (U_{21} + U_{20})$		
	$l_2 = l_2 + \widetilde{\widetilde{S}}$	$l_1 = l_0 + l_1 + l_2$	
5	*Compute U':*		$9M$
	$a = t_1 \cdot r$	${b} = {s_1}^2$	
	$c = s_1 \cdot Z_2$	$b = b \cdot t_1$	
	$a = R \cdot (a + c)$	$b = b \cdot (t_1 + U_{21})$	
	$d = t_2 \cdot \widetilde{S} + {s_0}^2$		
	${U_0}' = b + d + a$		
	${U_1}' = \widetilde{S} \cdot t_1 + R^2$		
	$b = S_3 \cdot (U_0' + l_1)$		
	$l_2 = l_2 + {U_1}'$		
	$t_4 = U_0' \cdot l_2 + S_3 \cdot l_0$		
	$Z' = \widetilde{R} \cdot S_3$	$t_5 = {U_1}' \cdot l_2 + b$	
	${U_1}' = \widetilde{R} \cdot {U_1}'$	${U_0}' = \widetilde{R} \cdot {U_0}'$	
6	*Compute V':*		$1M$
	${V_0}' = \widetilde{\widetilde{R}} \cdot \widetilde{V}_{20}$;	${V_1}' = \widetilde{\widetilde{R}} \cdot \widetilde{V}_{21} + Z'$	
	${V_0}' = t_4 + {V_0}'$	${V_1}' = t_5 + {V_1}'$	
total			$24M(23M)$

Table 5.8: Formulae for parallelized doubling.

Step	Calculations		# mult.
1	*Precomputation and resultant r:*		$2M$
	$t_0 = Z^2$ $r = U_0 \cdot Z$	$t_1 = {U_1}^2$ $a = Z + V_1$	
2	*Compute k:*		$2M$
	$k_1 = f_3 \cdot t_0 + t_1$ $k_0 = U_1 \cdot k_1 + Z \cdot b$	$b = V_1 \cdot a + t_0$	
3	*Compute s:*		$2M$
	$t_2 = k_0 \cdot U_1$ $s_0 = k_1 \cdot r + t_2$	$s_1 = k_0 \cdot Z$	
4	*Compute l:*		$4M$
	$t_0 = t_0 \cdot r$ $r = t_0 \cdot s_1$ $l_2 = s_1 \cdot t_2$ $l_1 = (t_2 + t_3) \cdot (s_0 + s_1)$ $l_1 = l_1 + l_2 + l_0$	$t_1 = s_1 \cdot k_0$ $t_3 = U_0 \cdot k_0$ $l_0 = s_0 \cdot t_3$	
5	*Compute U':*		$1M$
	${U_0}' = {s_0}^2 + r$	${U_1}' = {t_0}^2$	
6	*Precomputation:*		$4M$
	$a = s_0 \cdot s_1 + {U_1}'$; $l_2 = l_2 + a$ $Z' = s_1 \cdot r$ $t_0 = {U_0}' \cdot l_2 + l_0 \cdot s_1$ $t_1 = {U_1}' \cdot l_2 + s_1 \cdot b$	${s_1} = {s_1}^2$ $b = {U_0}' + l_1$ $t_2 = r \cdot t_1$	
7	*Adjust:*		$1M$
	${U_1}' = {U_1}' \cdot r$	${U_0}' = {U_0}' \cdot r$	
8	*Compute V':*		$1M$
	${V_0}' = t_0 + t_2 \cdot V_0$	${V_1}' = t_1 + t_2 \cdot V_1 + Z'$	
total			$17M$

Table 5.9: HW/SW cryptosystem peformance.

HECC Operations	8051 Perf. number of Clocks [Mcycles]	8051 Perf. Delay [ms] At 12 MHz
I/O Data Transfer	0.089	7.4
Divisor's Double	0.11	9.9
Divisor's Addition	0.13	11.1
Coordinate Conversion	0.20	17.2
Scalar Multiplicaton	7.87	656

Chapter 6

Lightweight Implementations of Curve-based Cryptography

RFID tags are passive devices consisting of a microchip connected with an antenna. Typically, they have no battery, but they obtain power from the electromagnetic field produced by the RFID reader. Today they are mainly used for identification of products. The application areas for RFIDs vary from supply chain management, inventory management, preventing banknotes counterfeiting to vehicles tracking and security of newborn babies *etc.* In short, RFID tags are meant to be a ubiquitous replacement for bar codes with some added functionality.

An emerging application is the use of RFID-tags for anti-counterfeiting by embedding them into a product. However, there is a risk related to naively using those tags for several applications. In particular, if no appropriate cryptographic measures are taken, the privacy of a user carrying tagged items can be severely damaged. In order to enable these applications and at the same time minimize the risks, public key cryptography (PKC) offers attractive solutions.

Our contributions in this chapter are as follows. We identify the technological components of anti-counterfeiting technology and we propose a solution for anti-counterfeiting based on RFID-tags and physical unclonable structures that can be used for physical protection purposes. An example are so-called Physical Unclonable Functions (PUFs) [232, 212, 231, 180]. They are introduced in more detail in Sect. 6.3. Our solution withstands physical cloning attacks as well as cloning attacks based on analyzing the verification protocols. In particular, we present a solution based on PUFs that can be inseparably bound to an IC. We present protocols for both the on-line and the off-line situation. The construction for the off-line situation inherits its security from the underlying cryptographic algorithms

(signature and secure identification scheme) used. We show that the construction that we propose is feasible on a constrained device such as an RFID-tag. More precisely, we argue that curve-based cryptography might be a feasible solution for authentication of a tag.

6.1 Introduction and Previous Work

The use of RFID-tags for anti-counterfeiting purposes was proposed by Juels [122]. By locating an RFID-tag with specific product and reference information on a product, one aims to verify the authenticity of the product. Loosely speaking the verification is performed as follows. When a product passes a reader, the reader checks whether the necessary and authentic product and reference information is present on the tag. For this purpose it runs a protocol with the tag. If the necessary information is there and verified to be authentic, the product is declared to be genuine and otherwise not. However, by capturing the necessary authentication information (obtained *e.g.* by eavesdropping the protocol between the tag and the reader), and by storing it in a new chip, the attacker has effectively made a clone of the original tag that cannot be distinguished from an original tag by a reader. In order to make this cloning of the tag infeasible, it should not be possible to derive the tag secrets by active or passive attacks.

We stress however that it is rather easy to physically clone a tag. This means that an attacker can capture the RFID-tag, investigate it, read out its memory (with reasonable effort) and in particular its security related data such as identification number, reference information, keys, *etc.* Then she produces a new tag with exactly the same data in its memory. When this tag is embedded into a product, it will again be impossible for a reader to distinguish an authentic product from a fake one. In order to protect an RFID-tag against this type of cloning attack, one can of course attempt to prevent read out its memory by using several protective measures as introduced by Neve *et al.* [166]. However these measures will increase the price of the tag so much that it will become unacceptably high for its main application. Currently the prices of tags range from a few cents up to 1 $.

To summarize, RFID-based identification is an example of an emerging technology which requires authentication as a cryptographic service [76]. This property can be achieved by symmetric as well as asymmetric primitives. Most of the previous work dealt with implementations of symmetric ciphers. The most notable example is the work of Feldhofer *et al.* [74], which considered implementation of AES on an RFID tag. On the other hand, the suitability of PK algorithms for RFID is an open research problem as limitations in costs, area and power are quite severe. Recently, the work of Wolkerstorfer [251] showed that ECC based PKC is feasible on RFID-tags by implementing the ECDSA on a small IC. In this chapter we also investigate which PKC-based identification protocols can be implemented in this context. In particular, we discuss the feasibility of identification protocols

based on curve-based cryptography. Some of the results of this chapter will appear in [229].

6.2 Model

We investigate the following problem. As observed by Hopkins *et al.* [109], goods that have some value (pharmaceuticals, luxury goods, banknotes, *etc.*) are likely to be counterfeited. An attacker who has captured an authentic good, can produce a clone that is indistinguishable from the original one. In practice, she clones the packaging material and copies the necessary registration and verification information printed on the package. In order to thwart these attacks we propose to use physical unclonable structures *i.e.* PUFs. PUFs have been proposed as a cost-effective mean to produce uncloneable tokens for identification (see the works of Pappu [180] and Simmons [210]). They are realized as a physical system such that the function is easy to evaluate but hard to clone.

In order to protect a product against cloning (counterfeiting) a detection mark is embedded into the product or its packaging. This detection mark consists of a physical and a digital part. The mark is put there by a legitimate authority which is assumed to be trusted. We consider an active attacker that knows the position of the tag in the product or its package. We also assume that the attacker can (passively) eavesdrop the channel between a reader and the tag, or can install a fake reader that communicates with the tag (active attack). Finally, we assume that the attacker can physically attack the tag; *i.e.* she can try to read out its memory or use side-channel attacks like power analysis attacks. The goal of the attacker is to produce a fake RFID-tag containing reference information such that it can only with small probability be distinguished from a real tag. Clearly, by embedding such a fake tag into a fake product, the fake product is identified as an authentic one.

6.2.1 Components of Anti-Counterfeiting Technology

In order to protect a product against counterfeiting, technological means are needed to verify whether the product is authentic or not. In order to make an item unclonable, the following two components are needed.

1. *Physical protection.* This is obtained by using unclonable physical structures embedded in the package (removal of the PUF leads to its destruction). One or more unique *fingerprints* derived from the physical structure will be printed on the product for the verification of the authenticity of the product.

2. *Cryptographic protection* serving two goals. Firstly, cryptography provides techniques (digital signatures) to detect and prevent tampering with data (fingerprints) derived from a physical object. Secondly, it provides secure

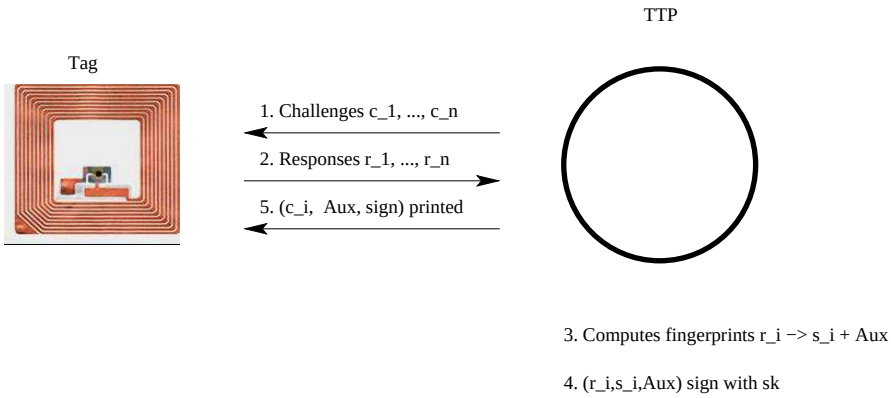


Figure 6.1: The basic model for an enrollment of an RFID tag.

identification protocols to identify a product. Those protocols do not leak any necessary identification information to an eavesdropper attacking (actively or passively) the communication channel.

6.2.2 A General Anti-Counterfeiting Protocol

We give an intuitive protocol for checking the authenticity of a product based on embedding a PUF in the product in combination with the use of cryptographic techniques. First there is an enrollment phase, which is performed by some trusted authority and is shown in Fig. 6.1. During this phase the following steps are performed.

1. Several fingerprints are derived from the PUF by challenging it with multiple challenges and recording the responses. These responses are then turned into binary fingerprints (and some auxiliary data are derived for use during the verification phase).

2. These challenges, fingerprints and auxiliary data are then signed with the secret key sk of the issuer of the product (the issuer is assumed to be trustworthy).

3. The signatures, the challenges (corresponding to the fingerprints) and maybe some auxiliary data (needed to perform processing during the authentication phase) are also printed on the product (and/or stored in a database).

During the verification phase, the authenticity is checked by running the following protocol (see Fig. 6.2).

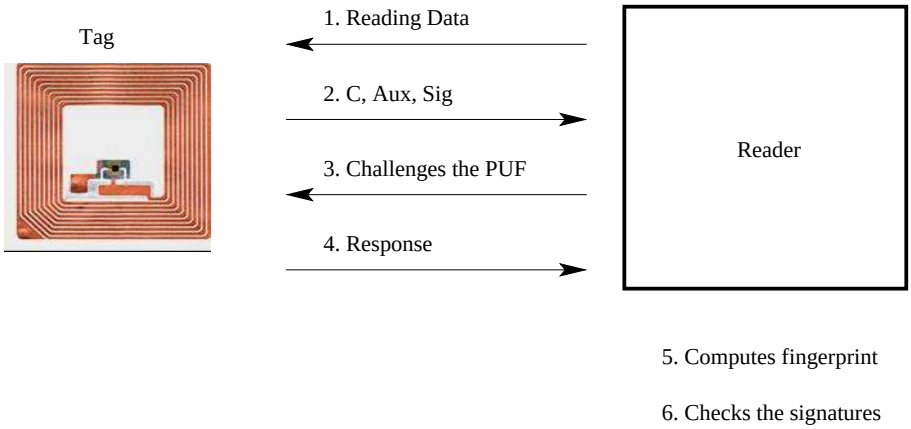


Figure 6.2: The basic model for an authentication of an RFID tag.

1. The verification device reads the challenges and auxiliary data.

2. The verification device challenges the physical structure with one of the challenges printed on the product. After having measured the responses, it derives the fingerprint from the response based on the auxiliary data.

3. Then, using the fingerprint derived in step 2., the verification device checks the signature to verify that the fingerprint, challenges and auxiliary data were printed on the product by a legitimate authority. If the signature is not correct, the product is not authentic.

We briefly analyze the security of this protocol. An attacker who wants to counterfeit the product has to embed a fake physical structure on the product that produces correct fingerprints to the challenges (with correct signatures). Under the assumption that the physical structure is unclonable, she cannot produce a clone of the originally embedded physical structure. More precisely, we assume that given some challenges $c_1, \ldots, c_n$ and corresponding fingerprints $s_1, \ldots, s_n$ she cannot produce a (fake) physical structure that produces the same fingerprints $s_1, \ldots, s_n$ given the original challenges $c_1, \ldots, c_n$. On the other hand she can produce another structure and create challenges, auxiliary data and fingerprints $s'_1, \ldots, s'_n$ according to the procedures used during enrollment. However, since she does not know the secret key sk and the responses of her fake structure will be different with very high probability, she will not be able to put the correct signatures on these data. The verification device will detect that the signatures are not correct and reject this as a fake product.

We note that the number of fingerprints that can be verified during a verification session is very limited by time and space constraints. Furthermore, the attacker can easily capture the required fingerprints (by measuring the responses according to the challenges printed on the product). Therefore the production of a clone only requires the fabrication of a physical structure producing the same fingerprints for a limited number of challenges.

6.2.3 RFID Systems

The PUF based solution for preventing counterfeiting of goods that was presented above can be improved with active components, that are inseparably linked with a PUF. An example consists of an RFID-tag equipped with a microchip that is inseparably bound to a PUF. Because of the presence of a microchip a secure identification protocol can be run without revealing any information on the fingerprint of the PUF. Additionally, by inseparably linking the chip and the PUF, it becomes possible to prevent leakage of the PUF measurement to the outside world.

Typical RFID systems consist of the following two components: the *RFID-tag* and a *reader* as shown in Figures 6.1 and 6.2. The reader will perform the verification to detect whether a tag is authentic or not. The RFID-tag consists of an antenna connected to a microchip that can store and read data and has possibly some dedicated hardware to perform a small amount of computations. Typically, the power for performing operations is obtained from the RF-field (by inductive coupling). A reader can read and write data from/on a tag. The reader is often linked with some system that can perform computations on the data that it receives from tags.

In order to use RFID-tags for anti-counterfeiting purposes, we proceed as follows. An RFID-tag is embedded in a product for this purpose. The (identification) data stored in the memory of the tag is signed with the secret key sk of the legitimate issuer. The tag communicates with a reader for verification purposes over a public channel. The ROM memory of the tag is accessible to the attacker. The reader has a certified public key pk corresponding to the issuer's secret key for verification of the digital signatures.

In Sec. 6.3, we explain how a physically unclonable RFID-tag is constructed. Additionally, we present protocols with which an RFID-tag can authenticate itself securely to the reader. We investigate the efficiency of these protocols and show that they can be executed on a low-cost tag.

6.3 Physical Unclonable Functions

Here we introduce a definition of a PUF as given in Gassend *et al.* [83].

Definition 27 *A Physical Unclonable Function is a function that maps challenges to responses and that is embodied in a physical object. It satisfies the following properties:*

1. *Easy to evaluate: the physical object can be evaluated in a short amount of time.*
2. *Hard to characterize: from a number of measurements performed in polynomial time, an attacker who no longer has the device and who only has a limited (polynomial) amount of resources can only obtain a negligible amount of knowledge about the response to a challenge that is chosen uniformly at random.*

More formally the PUF model is as follows. We denote the PUF response to a challenge C during the enrollment phase by $X \in \mathbb{R}^n$ and during the verification phase by $Y \in \mathbb{R}^n$ (the pair (C, X) is called a Challenge-Response pair or CRP). The PUF response according to a fake PUF is denoted by Z. The responses X, Y, Z are modeled as random variables with probability distribution $\mathbb{P}_{X,Y,Z}$.

Definition 28 [235] *Let $\delta, \epsilon_a, \epsilon_e \geq 0$. A joint distribution $\mathbb{P}_{X,Y,Z}$ on $(\mathbb{R}^n)^3$ is called $(\delta, \epsilon_a, \epsilon_e)$-reliable if it satisfies*
i) $Prob(d(Y, X) > \delta) \leq \epsilon_a$;
ii) $Prob(d(Z, X) \leq \delta) \leq \epsilon_e$;
here the probabilities are over the joint distribution $\mathbb{P}_{X,Y,Z}$.

This definition implies that if the enrollment and authentication measurements (according to the same challenge C) are performed on the same PUF, then these responses are with high probability very close to each other. When on the other hand the measurements are performed on different PUFs (modeling the fact that the PUF used during authentication might be fake), the responses are with high probability far apart.

We propose to equip the microchip on an RFID-tag with a PUF that is inseparably linked to the chip. More precisely we define this as follows.

Definition 29 *An Integrated Physical Unclonable Function (I-PUF) is a PUF that additionally satisfies the following properties.*

1. *The I-PUF is inseparably bound to a chip which means that any attempt to remove the PUF from the chip leads to the destruction of the PUF and the chip.*
2. *It is impossible to tamper with the communication (measurement data) between the chip and the PUF.*
3. *The output of the PUF is inaccessible to an attacker.*

In the following we use only I-PUFs, while we will often use just the abbreviation PUF. The two best known examples of such I-PUFs are silicon PUFs [73] and coating PUFs [231]. Silicon PUFs are based on the fact that manufacturing variations in a circuit cause substantial differences in circuit delays. These variations are caused by mask variations but also by temperature and pressure variations during manufacturing. The magnitude of the delay variations caused in this way is about 5%. A challenge for the silicon PUF is a digital input signal. The delay caused by the circuit between input and output is the response of the PUF.

In a coating PUF, the IC is covered with a coating, which is doped with random dielectric particles. By random dielectric particles we mean several kinds of particles of random size and shape with a relative dielectric constant ϵ_r differing from the dielectric constant of the coating matrix. The PUF consists of the combination of the coating with the dielectric material. In order to challenge the coating PUF, an array of metal sensors is laid down between the substrate and the passivation layer. Sufficient randomness is only obtained if the dielectric particles are smaller than the distance between the sensor parts. A challenge corresponds to a voltage of a certain frequency and amplitude applied to the sensors at a certain point of the sensor array. Because of the presence of the coating material with its random dielectric properties, the sensor plates behave as a capacitor with a random capacitance value. The capacitance value is then turned into a key. We note that the additional measurement circuit requires less than 1000 gates.

6.3.1 Key Extraction

In this work, the term key extraction always refers to key extraction from noisy data. Generally speaking a key extraction algorithm is built on a Secret Extraction Code (see Tuyls and Goseling [230]). This construction can be applied to discrete and continuous data. An equivalent construction for the discrete case, called Fuzzy Extractors, was developed by Dodis *et al.* in [64]. For the sake of simplicity we describe the algorithm in terms of a *shielding function* [147] or (G, W)-pair [235], which generates a special set of Secret Extraction Codes, while having all the necessary properties.

Definition 30 *A function $G(.,.) : \mathbb{R}^n \times \mathcal{W} \rightarrow \{0,1\}^k$ is called δ-contracting if for all X there exists* helper data $W \in \mathcal{W}$ *such that for all X' that lie within a sphere of radius δ of X (so $||X' - X|| \leq \delta$) it holds $G(X', W) = G(X, W)$ ($\mathcal{W}$ denotes the space of helper data. At this point it has to be considered as some abstract space.)*

We use δ-contracting functions to extract keys $S = G(X, W)$ from noisy data X using *helper data* W. A function $G(.,.)$ is called ϵ-revealing if the helper data W leaks less than ϵ bits on S (in the information theoretic sense), *i.e.* $\mathbf{I}(W; S) \leq \epsilon$. An (ϵ, δ)-shielding function $G : \mathbb{R}^n \times \mathcal{W} \rightarrow \{0,1\}^k$ is a function that is δ-contracting

and ϵ-revealing. It is used to extract a secret of length k from the PUF response as follows.

- **Enrollment Phase:** The PUF is subjected to a challenge C and the response X is measured. Then a random key S is chosen from $\{0,1\}^k$ and helper data W is computed by solving $G(X,W) = S$ for W. The quadruplet $(\mathrm{ID}_{\mathrm{PUF}}, C, W, S)$ is then stored in a CRP database.
- **Verification Phase:** When the PUF is inserted into the reader the PUF's identity is sent to the verifier. The verifier chooses a random challenge C from his database and sends it to the PUF together with the corresponding helper data W. Then the PUF is subjected to the challenge C and its response X' is measured. A key S' is then computed as $S' = G(X', W)$.

Notice that if $G(.,.)$ is δ-robust and if $\mathbb{P}_{X,Y,Z}$ is $(\delta, \epsilon_a, \epsilon_e)$-reliable, then we obtain $\mathrm{Prob}(G(Y,W) = S) \geq 1 - \epsilon_a$ and $\mathrm{Prob}(G(Z,W) = \perp) \geq 1 - \epsilon_e$, which expresses that FRR (False Rejection Rate) and FAR (False Acceptance Rate) are at most ϵ_a and ϵ_e respectively. In the case of a passive attacker, the extracted key S can then be used securely since $\mathbf{I}(W;S) \leq \epsilon$. Note that by adding a privacy amplification this can be guaranteed (if the Réniy entropy is sufficiently large). Also note that this procedure can be used to set up a shared secret key between an I-PUF and a verifier (reader).

The PUF responses are often analog data. More precisely, in the case of an optical PUF the PUF response is a speckle pattern which can be seen as an analog picture [212] . In the case of a coating PUF the responses are given by capacitance values which are analog signals. Therefore, the helper data typically consists of three parts. The first part W_1 allows to quantize the signal into a binary representation while the second part W_2 implements the error correction and the random key choice on the binary data. The third part is used for privacy amplification.

6.3.2 Example

We present a brief example of key extraction from noisy (binary) data. It shows that the required processing at the side of the RFID-tag is low. Assume for the sake of simplicity that the responses X are uniformly random binary strings of length k, *i.e.* $X \in \{0,1\}^k$. Furthermore, we assume that the authentication measurement performed during the verification phase can be modeled as a noisy observation over a binary symmetric channel with cross-over probability p. Let $\mathcal{C}$ be an error correcting code, with l codewords. Then, for a key $s \in_R \{0, \ldots, l-1\}$ the helper data $w(x,s) = x \oplus c_s$ is generated during the enrollment phase (where $c_s \in \mathcal{C}$). During the verification phase, the tag measures y and computes $G(y, w(x;s)) = \mathsf{Dec}(y \oplus w(x;s))$ (Dec denotes the decoding algorithm of the error-correcting code $\mathcal{C}$). Clearly, if y corresponds to the same challenge (and the same PUF), s is obtained after decoding while otherwise a random code-word is obtained or a

decoding error. Hence, the tag has to perform an XOR operation and a decoding operation. On a tag with some S-RAM (Static RAM) available (which most tags have), the decoding costs less than 1000 gates. The reason is that in the case of coating PUFs the codewords are relatively short (200 bits) and the information rate is high. In that case BCH codes can be efficiently used. On BCH codes more can be found in the books by Pretzel [187] and van Tilborg [236].

6.4 Unclonable RFID-Tags

6.4.1 Set-up

In order to make unclonable RFID-tags, we introduce RFID-tags whose microchips are equipped with an I-PUF. In our construction, the PUF is used as a secure memory for storing secret keys. The secret key s which is usually stored in (protected) ROM or EEPROM is derived from the PUF, when needed. In order to enable the generation of the secret key s during authentication, helper data w is stored in (publicly accessible) ROM (EEPROM). The key s is derived from the response X of the PUF by means of a key extraction algorithm (a Fuzzy Extractor and the helper data w are used here). It was mentioned in Sect. 6.3.1 that the public helper data w reveals only a negligible amount of information on the key s. Given our assumption on I-PUFs in Def. 29, it follows that the key s is securely stored in the PUF.

6.4.2 On-line Authentication

We assume that every reader is connected with a reference database through an authenticated channel. The reference database contains for each tag ID a list of CRPs of its corresponding PUF. We distinguish between two different situations, i) we assume that there is a *large* number of challenge response pairs available for the PUF; i.e. n is a large number. We refer to this case as *strong PUFs*. ii) The number of different challenge response pairs n is rather small. This case is referred to as *weak PUFs*. More precisely, a strong PUF is a PUF that has so many challenge-response pairs such that an attack (performed during a limited amount of time) based on exhaustively measuring challenge-response pairs only has a negligible probability of success [232]. This fact was elaborated by Tuyls *et al.* in [232].

Strong PUFs

During the **enrollment phase** the PUF is challenged by a Certification Authority (CA) with n independent challenges [232], say $c_1, \ldots, c_n$ and the corresponding responses $x_1, \ldots, x_n \in \{0,1\}^k$ are measured. The data $(c_i, x_i), i = 1, \ldots, n$ are

securely stored in the database (and unkown to an attacker). No additional information is stored in the (ROM) memory of the RFID-tag.
During the **authentication phase**, the following protocol is performed between the tag and the reader.

1. The reader asks the tag for its identification number, ID.
2. The reader gets from the database a random challenge response pair say (c_i, x_i) for this ID.
3. The reader sends the challenge c_i over a public channel to the tag.
4. The tag challenges its PUF according to the challenge c_i, measures y_i and sends y_i over the public channel to the reader.
5. The reader verifies whether $d_H(x_i, y_i) \leq \delta$, where δ is some predetermined threshold. If this condition is satisfied, the reader considers the tag to be authentic, in the other case it is decided that this is a counterfeited tag.
6. The database removes the pair (c_i, x_i) from the database.

Security:

It is clear that in order to have a secure system for RFID-tags with some reasonable life time, a large number of CRPs is needed (*e.g.* $\sim 10^9$ was computed by Tuyls *et al.* [232]). Since the various CRP pairs are independent, a passive attacker has a probability of guessing a response z_i with $d_H(z_i, r_i) \leq \delta$ to a challenge c_i equal to $\sum_{i=0}^{\delta} \binom{k}{i}/2^k \approx 2^{(h(\alpha)-1)k}$ when $\delta = \alpha k$ and h denotes the binary entropy function. Note that an active attacker will probe the PUF of the tag with a fake reader that sends well chosen challenges $c'_1, \ldots, c'_m$ to the tag. When the responses $y_1, \ldots, y_m$ are returned, he records those and uses them to make a model of the PUF and to guess the responses to other remaining (unused) challenge-response pairs. It was also shown in [232] that the number of responses that can be obtained in a limited amount of time is small compared to the total number of challenges, *i.e.* $m \ll n$ (Typically $m = 100$ and $n = 10^9$). Hence, the probability that $c_j \in \{c'_1, c'_2, \ldots, c'_m\}$ for some $j \in_R \{1, \ldots, n\}$ is $\mathcal{O}(\frac{m}{n})$. This implies that the verifier has to keep its database with CRP-pairs secret.

Complexity:

We note that from a computational point of view, this protocol is very inexpensive for the RFID-tag. It only has to measure responses to challenges. Note that no cryptographic operations have to be performed. In another variant of this protocol keys are derived from the responses of the PUF using the helper data scheme.

In order to prevent the attack a MAC can be used too (cf. Chapter 1). Then, the PUF and the reader first establish a shared secret (based on a challenge-response pair of the PUF) with which the challenges sent by the reader are MACed. Then, the tag checks the MAC. If it turns out that it is not correct, the tag will not react to any further query from the reader. We explain the idea in more detail below.

Weak PUFs

The Protocol: During the enrollment phase the PUF is challenged by a Certification Authority (CA) with n challenges, say $c_1, \ldots, c_n$ and the corresponding responses $x_1, \ldots, x_n$ are measured. Then, a secret $s_i \in_R \{0,1\}^k$ for $i = 1, \ldots, n$ is randomly chosen for each response x_i. The corresponding helper data $w_1, \ldots, w_n$ are computed by means of the *key extraction* algorithm explained in Sect. 6.3.1. In the (ROM) memory of the RFID-tag the challenges $c_1, \ldots, c_n$ are stored. In the database of the verifier, the triples (c_i, w_i, s_i) for $i = 1, \ldots, n$ are stored.
Let $\mathrm{MAC}_k(.)$ denote a MAC algorithm using the key k. Then the authentication protocol performs the following steps.

1. The reader contacts the tag which sends its ID.
2. The reader contacts the database and gets a random challenge response pair for this ID, say (c_i, s_i), together with the helper data w_i. It sends the data (c_i, w_i) together with $\mathrm{MAC}_{s_i}(c_i, w_i)$ to the tag. Additionally, the reader chooses a random nonce $m \in \{0,1\}^t$ and sends it to the tag too.
3. The tag challenges its PUF according to the challenge c_i, and measures the response y_i. Then, it uses the helper data w_i to derive $s_i' = G(y_i, w_i)$. Using s_i' and w_i the tag checks the MAC $\mathrm{MAC}_{s_i}(w_i)$ using s_i'. If the MAC is not correct, the tag stops. Otherwise it proceeds. Note that in this step we have in fact transformed the key-extraction function G into a robust key-extraction function G^* in a similar way as presented in [40].
4. Finally, the tag MACs the message m and sends $\mathrm{MAC}_{s_i'}(m)$ to the reader.
5. The reader checks the MAC $\mathrm{MAC}_{s_i'}(m)$. If the MAC is correct, it considers the tag as authentic and otherwise not.

Security: The authentication technique in step 2 prevents an active attack by a fake reader. If the challenge-response protocol carried out in steps 3 and 4 uses a secure MAC-ing algorithm, it reveals no information on the used key s_i.

Complexity: This protocol requires only symmetric crypto primitives which require a few thousands of gates.

6.4.3 Off-line Authentication

Here we focus on off-line authentication because it is very attractive from a practical point of view.

We introduce our PUF-Certificate-Identity-based Identification scheme (PUF-Cert-IBI) by following the definition of Certificate-based IBI by Bellare *et al* in [23]. Let $\mathcal{SI} = (K_g, P, V)$ denote a standard identification scheme (SI-scheme) where K_g denotes the key generation algorithm, and P, V denote the interactive protocols run by the prover and verifier respectively. Let $\mathcal{SS} = (\mathrm{SK}_g, \mathrm{Sign}, V_f)$ be a standard signature scheme (SS-scheme) [63] with SK_g denoting the key generation algorithm, Sign denoting the signing algorithm and V_f the verification algorithm run by a verifier. We assign to each tag an identity I (this might be the serial number or EPC-code of the tag or the serial number of the product in which it has been embedded). To the PUF, the $\mathcal{SI}$, the $\mathcal{SS}$ scheme and the identity I an Identity-Based Identification scheme $(\mathrm{MK}_g, \mathrm{UK}_g, \hat{P}, \hat{V})$ is associated as follows.

During **enrollment** the issuer uses SK_g as the master-key generation algorithm MK_g. This means that the master key msk is used for generating signatures and the corresponding public key mpk for verification of the signatures. The algorithm UK_g consists of the following steps. For each RFID-tag, having identity I, the issuer then creates a public-secret key pair (pk, sk) using the algorithm K_g on input 1^k. The couple (pk, sk) is the public-secret key pair for the SI-scheme. The issuer runs the following protocol with the tag.

- It requests the tag to challenge its PUF with a challenge c and to record the response $x(c)$.
- The tag sends $x(c)$ to the issuer.
- Based on the knowledge of $x(c)$ and sk, the issuer determines the helper data w such that $sk = G(x, w)$.
- The helper data w are written into the ROM (EEPROM) memory of the tag.

Finally, the issuer creates the following certificate that is also stored in the ROM of the tag $\mathrm{Cert} \leftarrow (pk, \mathrm{Sign}(msk, pk||I))$. The usk is then put to $usk \leftarrow (\mathrm{PUF}, \mathrm{Cert})$. During **authentication**, the tag (in the role of the prover) runs the following steps with a verifier.

- The tag runs the protocol $\hat{P}$ which consists of the following steps.
  - It challenges the PUF with c, measures the response $y(c)$ and computes $sk \leftarrow G(y(c), w)$.
  - Initialisation of the prover protocol P of the $\mathcal{SI}$ scheme with sk.
  - It includes the certificate Cert in the first step of the algorithm P.

- The verifier uses (mpk, I) as input for the verification algorithm $\hat{V}$.
- When the verifier receives Cert from the tag, it first verifies Cert by running $V_f(mpk, pk||I, \text{Sign}(msk, pk||I))$.
- If the certificate Cert is invalid the protocol is aborted.
- If Cert is valid, the verifier initializes V with pk and runs it.
- If V accepts, then the verifier accepts.

The security of our PUF-Certificate-Identity-based identification scheme follows from the following theorem. This theorem and its proof are very similar to Theorem 4.2 in [23].

Theorem 14 *Let $\mathcal{SI}$ be an SI-scheme and SS a uf-cma secure SS-scheme. Here uf-cma stands for: existential unforgeability under chosen message attack. Let PUF-Cert-IBI be the corresponding PUF-Certificate-Identity based Identification scheme presented above. If the scheme $\mathcal{SI}$ is impersonation-atk secure then PUF-Cert-IBI is impersonation-atk secure for atk $\in \{pa, aa, ca\}$ (pa: passive attack, aa: active attack, ca: concurrent attack).*

It follows from this theorem, that by choosing an appropriate SI-scheme (withstanding a *pa*, *aa* or *ca*) the PUF-Cert-IBI inherits the same property. If only resistance against passive attacks is needed, the Schnorr Identification scheme can be used as it is known that this scheme is secure against passive attacks under the discrete logarithm assumption. It is also secure against active attacks with a few more assumptions [42]. An alternative is to use Okamoto's identification scheme [155], which is secure against passive, active and concurrent attacks under the discrete logarithm assumption.

Schnorr Identification Protocol based on ECDLP

Here we specify the Schnorr identification protocol based on ECDLP that could be performed in the case of off-line authentication. In this case a tag proves its identity to a reader in a 3-pass protocol.

1. **Common Input:** The set of system parameters in this case consists of: (q, a, b, P, n, h). Here, q specifies the finite field, a, b, define an elliptic curve, P is a point on the curve of order n and h is the cofactor. In this case of tag authentication, most of these parameters are assumed to be fixed.
2. **Prover-Tag Input:** The prover's secret a such that $Z = -aP$.
3. **Protocol:** The protocol involves exchange of the following messages:

Prover P		Verifier V
$r \in_R \mathbb{Z}_n$		
$X \leftarrow rP$	$\xrightarrow{X}$	
	$\xleftarrow{e}$	$e \in_R Z_{2^t}$
$y = ae + r$	$\xrightarrow{y}$	
		If $yP + eZ = X$ then accept, else reject

More precisely, the steps of the protocol are:

- *Commitment by a Prover-Tag:* The tag picks $r \in_R \{0, \ldots, n-1\}$, and sends $X = rP$ to the reader.
- *Challenge from a Verifier-Reader:* The reader picks a number $e \in [1, 2^t]$ and sends it to the tag.
- *Response from a Tag:* The tag computes $y = ae + r$ and sends it to the reader.
- The verifier checks that $yP + eZ$ equals X. Check: $yP + eZ = (ae + r)P + eZ = aeP + rP + (-eaP) = rP = X$

The Okamoto Identification Protocol based on ECDLP

We can use the Schnorr identification protocol [155] as the SI in our our Cert-IBI as introduced above. Another option for secure identification is Okamoto's identification protocol. We give details of it and afterwards we investigate the efficiency of the implementations of both protocols.

1. **Common Input:** Common input is the set of system parameters consisting of (q, a, b, P_1, P_2, n, h) as before.
2. **Prover-Tag Input:** The prover's secret (s_1, s_2) such that $Z = -s_1P_1 - s_2P_2$.
3. **Protocol:** The protocol involves the exchange of the following messages:

Prover P		Verifier V
$r_1, r_2 \in_R \mathbb{Z}_n$		
$X \leftarrow r_1P_1 + r_2P_2$	$\xrightarrow{X}$	
	$\xleftarrow{e}$	$e \in_R Z_{2^t}$
$y_i = r_i + es_i$, $i = 1, 2$	$\xrightarrow{y_1, y_2}$	
		If $y_1P_1 + y_2P_2 + eZ = X$ then accept, else reject

More precisely, the steps of the protocol are:

- *Commitment by a Prover-Tag:* The tag picks $r_i \in_R \{0, \ldots, n-1\}$, $i = 1, 2$ and sends $X = r_1P_1 + r_2P_2$ to the reader.
- *Challenge from a Verifier-Reader:* The reader picks a number $e \in [1, 2^t]$ and sends it to the tag.
- *Response from a Tag:* The tag computes $y_i = r_i + es_i$, $i = 1, 2$ and sends those values to the reader.
- The verifier checks that $y_1P_1 + y_2P_2 + eZ$ equals X.
  Check: $y_1P_1 + y_2P_2 + eZ = (r_1 + es_1)P_1 + (r_2 + es_2)P + e(-s_1P_1 - s_2P_2)Z = r_1P_1 + r_2P_2 = X$

6.5 Implementation

In this section, we discuss implementation issues, *i.e.* efficiency and size of the hardware if the off-line RFID identification protocol is implemented on an RFID-tag. As examples we consider both, the Schnorr identification protocol and the one of Okamoto for both types of curve-based cryptography. They both allow a user to prove knowledge of x given the public information g^x in a group where the discrete log problem is difficult. We also compare the efficiency of implementation between implementations of ECC and HECC. Furthermore, we discuss all other hardware-related components that are involved *e.g.* storage, random number generation *etc.*

6.5.1 Implementation of Schnorr's Scheme based on ECDLP

We investigate here the efficiency of this protocol on an elliptic curve over $\mathbb{F}_{2^{163}}$. As can be observed from the protocol, the critical operation is the point multiplication. The easiest way to calculate the point or scalar multiplication is by means of the basic double-and-add algorithm as explained in Chapter 2. The expected number of ones in the binary representation of k is $n/2$, hence the expected number of point additions and doublings, *i.e.* A and D respectively is: $\frac{n}{2}A + nD$. On the other hand, the expected number of operations when the NAF method for point multiplication [29] is used is $\frac{n}{3}A + nD$, but in this case extra logic is required to compute the NAF form of a scalar k. Here, A and D are the point operations.

In either case of point addition and doubling, the computation requires one field inversion (I), two field multiplications (M) and one squaring (S), or $1I + 2M + 1S$. As we are interested in hardware implementations, we count squarings and multiplications together as they are both executed on the same multiplier.

The inversion operation can be avoided by choosing one of many options for projective coordinates. The addition of two field elements requires the modulo 2 addition of the coefficients of the elements. In hardware, a bit-parallel adder for

a binary field requires n XOR gates and the sum can be computed in one clock cycle.

Another option for scalar or point multiplication is to use so-called "Montgomery ladder"[119] as mentioned in Chapter 4. As we also showed there, the Montgomery representation requires less memory and offers a better protection against side-channel attacks. These both facts are very useful in this case as memory *i.e.* registers are very "expensive" in hardware implementations. Also, side-channel attacks are an issue on RFID tags and also some cheap protection *i.e.* by means of balanced implementations is desirable.

For point operations (addition and doubling) we consider again the formulae of López and Dahab in $\mathbb{F}_{2^n}$ where the operation count is $A : D = 5M : 6M$ (cf. Chapter 4, Algorithm 4.2).

ECC Operations

In this section we describe ECC operations at each level by following the top-down approach. For the point multiplication we chose the method of Montgomery that maintains the relationship $P_2 - P_1$ as invariant. It uses a representation where computations are performed on the x-coordinate only.

We start from Algorithm 4.2 as in Chapter 4, but we want to save some registers, as it is known that this part is usually the largest portion of the total area. As the previous formulae require 3 intermediate registers (2 for addition and 1 for doubling), we eliminate 2 intermediate registers and we rewrite the formulae. Therefore, we get the following sequences for point operations that require only one intermediate variable (T) (see Algorithm 6.1). Moreover, this value is manipulated only twice for addition and it could be even stored in some RAM. In this way we made a trade-off between performance and area as point operations require now 7 and 8 multiplications for addition and doubling (instead of 5 and 6 M respectively). Furthermore, point operation can be also easily balanced to achieve some simple side-channel protection such as in Chapter 4.

Algorithm 6.1: EC point operations that minimizes # intermediate registers

Require: $X_1, Z_1, X_2, Z_2, x_4 = x(P_2 - P_1)$
Ensure: $X(P_1 + P_2) = X(P_3) = X_3,\ Z_3$

1. $Z_3 \leftarrow X_2 \cdot Z_1$
2. $X_3 \leftarrow X_1 \cdot Z_2$
3. $Z_3 \leftarrow X_3 + Z_3$
4. $Z_3 \leftarrow {Z_3}^2$
5. $X_3 \leftarrow X_1 \cdot Z_2$
6. $X_3 \leftarrow X_3 \cdot X_2$
7. $X_3 \leftarrow X_3 \cdot Z_1$
8. $T \leftarrow x_4 \cdot Z_3$
9. $X_3 \leftarrow X_3 + T$

Require: $b \in \mathbb{F}_{2^n},\ X_1, Z_1$
Ensure: $X(2P_1) = X(P_5) = X_5,\ Z_5$

1. $Z_5 \leftarrow {Z_1}^2$
2. $Z_5 \leftarrow {Z_5}^2$
3. $Z_5 \leftarrow b \cdot Z_5$
4. $X_5 \leftarrow {X_1}^2$
5. $X_5 \leftarrow {X_5}^2$
6. $X_5 \leftarrow X_5 + Z_5$
7. $Z_5 \leftarrow {X_1}^2$
8. $Z_5 \leftarrow Z_5 \cdot Z_1$
9. $Z_5 \leftarrow Z_5 \cdot Z_1$

For the field multiplication we choose the most compact solution. The standard way to compute the product $c(x) = a(x) \cdot b(x) \bmod f(x)$ is the one that uses convolution and to which we refered to as the classical algorithm [27]. Classical modular multiplication is typically based on the following equation:

$$\begin{aligned} a(x) \cdot b(x) &= (a_{n-1}x^{n-1} + \cdots + a_1 x + a_0) \cdot b(x) \bmod f(x) \\ &= (\cdots (a_{n-1}b(x)x + a_{n-2}b(x))x + \cdots \\ &\quad + a_1 b(x))x + a_0 b(x) \bmod f(x)\,, \end{aligned} \tag{6.1}$$

which illustrates the Horner scheme for multiplication. More precisely, modular multiplication can be calculated via Algorithm 6.2.

Algorithm 6.2: Classical Modular Mult. in $\mathbb{F}_{2^n}$

Require: polynomials $a(x)$, $b(x)$ and $f(x)$,
Ensure: $Res(x) = a(x) \cdot b(x) \bmod f(x)$
1: $Res(x) \leftarrow 0$,
2: **for** i from n-1 to 0 **do**
3: $\quad Res(x) \leftarrow Res_{n-1} \cdot f(x) + Res(x) \cdot x + a_i \cdot b(x)$
4: **end for**
5: Return $Res(x)$

The most compact architecture for this multiplication is the classical bit-serial multiplier (the MSB or the LSB multiplier) as introduced by Beth and Gollmann [27].

A Prototype Elliptic Curve Processor

The Elliptic Curve Processor (ECP) is shown in Fig. 6.3. The operational blocks are as follows:

- Control Unit(CU)
- Arithmetic Unit (ALU)
- Registers
- Memory: RAM

The Control Unit takes care of scalar multiplication, point operations and all conversions to suitable representation. It also commands the ALU which performs field multiplication, addition and inversion. The largest part of the ALU is the finite field multiplier. The inversion operation is also performed by the multiplier using Fermat's theorem.

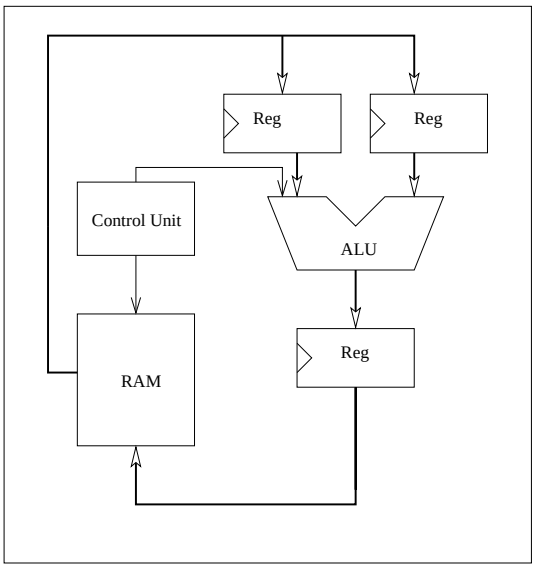


Figure 6.3: Architecture of the elliptic curve processor.

Estimated Results

Here we estimate the performance of the ECC processor for the field $\mathbb{F}_{2^{163}}$. The irreducible polynomial is the pentanomial $f(x) = x^{163} + x^7 + x^6 + x^3 + 1$. One point multiplication takes $163 \cdot 15M = 2445M$. Conversion of coordinates $A \rightarrow P$ and $P \rightarrow A$ takes respectively $2M$ and $I + 2M$. Assuming that inversion is done by means of Fermat the total for conversion is around 300 M. This all together results in around 3000 M. One field multiplication (M) takes 163 cycles, which results in 489000 cycles for point multiplication. With a clock frequency 1 MHz one point multiplication would take less than half a second, which is reasonably fast.

The estimated area complexity for the bit-serial multiplier is $16n$ [71], so for $n = 163$ we get around 2.6 kgates. Modular addition takes 163 XOR gates, so it sums up to close to 3 kgates. The complexity of the FSMs used is hard to estimate, but as those are only some control logic it should not be too large. However, the registers that are required might take quite large area as 1FF is at least 6 NANDs. This is the most crucial aspect of the design. However, as 3 registers are absolutely necessary for ALU, we believe that this hardware component can be of the order of 5 kgates, depending on the technology. We assume that EC parameters as well as other pre-calculated input values can be stored in memory blocks. It may further slow-down the performance but there is certainly enough margin for that

according to the RFID specifications [122]. Another option to minimize hardware complexity would be to decrease the field size. Namely, according to the work of Lenstra and Verheul [141] (cf. Chapter 2) 163-bit long key sizes correspond to RSA keys that are much longer than 1024 bits in the sense of computationally equivalent security. This means that one could use around 130 bits long ECC keys to achieve the security of 1024-bit RSA. Consequently, scaling down ECC parameters would result in a roughly linear decrease of hardware complexity at expense of security level. That ECC are a suitable technology for RFIDs was also concluded in the work of Wolkerstorfer [251]. That work is the first complete ECC low-power and compact implementation that meets the constraints imposed by the EPC standard. Yet, our solution can be even smaller as our off-line authentication do not require full ECDSA algorithm to be executed on a single tag. That allows for further optimization with respect to area.

6.5.2 Light-weight Implementations of ECC/HECC

The only difference the hierarchy for ECC/HECC operations between ECC and HECC is the middle level that in this case consists of different sequences of operations as shown in Figure 2.5, Chapter 2. Those for HECC are a bit more complex when compared with the ECC point operation, but they use shorter operands. We conclude that the same algorithms and by analogy the same hardware components can be used at the top and bottom level in both variants of curve-based cryptography. We compare the number of operations and required resources for scalar multiplication in both cases.

HECC Operations

The point/divisor scalar multiplication is achieved by repeated point/divisor addition and doubling. The point addition/doubling costs expressed in field multiplications (denoted with M) for standard projective coordinates is $14M/10M$ respectively [103]. On the other hand costs for the HECC divisor operations are 44 M and 31 M for addition and doubling respectively [45].

The most crucial operation for implementations of both ECC and HECC is field multiplication. The irreducible polynomials chosen are the pentanomial $f(x) = x^{163} + x^7 + x^6 + x^3 + 1$ (for ECC) and the trinomial $f(x) = x^{79} + x^9 + 1$ (for HECC). Therefore, the area complexity for the n bit-serial multiplier for HECC two times smaller than in the case of HECC. In both cases and it takes n cycles to complete one multiplication (M).

ECC/HECC Comparison

Now we can give estimates for the latency and the area complexity for both types of cryptosystems. As we are interested in implementations of identification protocols

(*e.g.* Schnorr, Okamoto) the operation required is one point multiplication (or multiple-point multiplication in the case of Okamoto's scheme). Having in mind the factor 2 to maintain the same level of security, we compare ECC over $\mathbb{F}_{2^{163}}$ with HECC over $\mathbb{F}_{2^{79}}$. Here we chose for finite fields of a form $\mathbb{F}_{2^p}$, where p is a prime as recommended by standards. However, to allow for a better comparison we also examined the case of ECC for the composite field $\mathbb{F}_{(2^{79})^2}$. In this case an ALU is exactly the same as the field arithmetic is performed in the field $\mathbb{F}_{2^{79}}$. Yet, we have to add that the composite fields are not recommended by standards due to security reasons as explained in Chapter 2.
The estimates for required hardware resources and the latency in all three cases are given in Table 6.1.

Table 6.1: Performance comparison and estimated area complexity for both types of curve based cryptography.

Cryptosyst.	Lat. [# cyc.]	ALU	# 80-bit regs.	Tot. area
ECC ($\mathbb{F}_{2^{163}}$)	495520	3 kgates	16	10 kgates
ECC ($\mathbb{F}_{(2^{79})^2}$)	720480	1.5 kgates	16	8.5 kgates
HECC ($\mathbb{F}_{2^{79}}$)	695200	1.5 kgates	20	11 kgates

As an example for $n = 163$ in the case of ECC we get around 2.6 kgates for the multiplier. Modular addition takes 163 XOR gates, so it sums up to around 3 kgates. This amount is two times smaller for HECC but it takes more cycles to compute one group operation and also more memory *i.e.* RAM is required. The expected number of point/divisor additions and doublings, is $\frac{n}{2}A + nD$. This gives for both, ECC and HECC the total of $80A + 160D$. Adding one inversion in both cases (by means of Fermat) results in around 3000 field multiplications for ECC and almost 9000 field multiplications for HECC. However, in the latter the field elements have bit-lengths that are two times shorter than in the case of ECC. Therefore, using a simple bit-serial multiplier requires around twice as many cycles for ECC field multiplication than for HECC. When we compare ECC with ECC over $\mathbb{F}_{(2^{79})^2}$, we get the factor 1.5 as the latency ratio. The reason is that in composite field ($\mathbb{F}_{(2^{79})^2}$) each element is represented as $c = c_1 t + c_0$ where $c_0, c_1 \in \mathbb{F}_{2^{79}}$ and the multiplication in this field takes 3 multiplications in $\mathbb{F}_{2^{79}}$ [145] (plus 4 additions that we do not take into account here). On the other hand each multiplication in this smaller field takes two times less cycles and therefore we get this factor of 1.5.

The expected number of cycles required to perform field multiplication, which is the most time consuming operation, is 495 520 vs 695 200 for ECC, HECC respectively. At the same time the ALU for HECC is twice as small as the one for

ECC. However, an important impact on the size of implementations is the amount of registers. ECC, HECC curve parameters could be stored in ROM so we do not take these into account. For the storage of variables, we can estimate almost the same amount of memory as the divisor requires more coordinates but those are at the same time shorter. The crucial impact on the size of the hardware is the amount of intermediate registers required. When the function to be optimized is amount of gates, the strategy is to sacrifice performance in order to minimize the area. We explained the idea above for the case of ECC.

We apply the same strategy to the case of HECC. As the group operations are much more complex, it is very hard to eliminate intermediate registers but nevertheless possible. We explain here the idea briefly. If the calculation $t_0 = Z^2$ from Step 1 in Table 2.2 is delayed to Steps 3 and 5 we do not have to store intermediate variable t_0 but to add 4 M instead. In this way we perform a memory-performance trade-off in order to minimize the total area. By following this strategy, the estimated impact on performance is around 50%, which is similar to the case of ECC. Our conclusion is that both types of curve-based cryptography are feasible for RFID systems. We also assume here that numbers for latency given in Table 6.2 allow for this sort of trade-offs as the total number of cycles would keep one point multiplication below 500 ms with an assumed frequency of 1 MHz.

6.5.3 Implementation of Okamoto's Scheme

In our previous work, we investigated feasibility of the ECC version of the Schnorr identification protocol in an RFID system [229]. Here, we investigate the feasibility of the scheme of Okamoto as it provides even more security than the scheme of Schnorr.

As can be seen from Okamoto's scheme, the required computation on a tag is of a form $kP + lQ$ *i.e.* so-called multiple-point multiplication. For the purpose of speeding-up this computation one uses Shamir's trick [103]. The scalars k and l are stored in a 2-row matrix in which each row contains binary representation of one of the scalars. All values of the form $iP + jQ$, $0 \leq i, j < 2^w$ are precalculated and stored where w is given width of the window. The algorithm to perform this so-called simultaneous point multiplication is computing at each of $\lceil \frac{t}{w} \rceil$ steps w doublings and 1 addition from the list of the precalculated values of the form $iP + jQ$. As a width of the window w is a variable that allows some trade-off, we chose the smallest window *i.e.* $w = 1$. In this way, the memory requirements are minimized as only 3 points have to be stored: $P, Q, P+Q$. The exact computation is given in Algorithm 6.3 [103].

Algorithm 6.3: Simultaneous point multiplication

Require: $k = (k_{t-1}, \ldots, k_0)_2$, $k = (k_{t-1}, \ldots, k_0)_2$, P, Q points on the curve
Ensure: $R = kP + lQ$
1: Compute $P + Q$
2: $R \leftarrow \infty$
3: **for** i from $t - 1$ downto 0 **do**
4: $R \leftarrow 2R$
5: $R \leftarrow R + (k_i P + l_i Q)$
6: **end for**
7: Return(R)

The expected running time of Algorithm 6.3 ([103], p. 109) is:

$$\left[\left(3 \cdot 2^{2(w-1)} - 2^{w-1} - 1\right) A + \left(2^{2(w-1)} - 2^{w-1}\right) D\right] + \left[\left(\frac{2^{2w} - 1}{2^{2w}} d - 1\right) A + (d-1) w D\right],$$

and it requires storage of $2^{2w} - 1$ points. As our goal is to minimize the area *i.e.* storage, we choose the minimal window width, so $w = 1$. In that case the running time of the algorithm is: $(\frac{3}{4}t - 1)A + (t - 1)D$.

Generation of Randomness

Often a source of randomness is needed on the tag; this can be derived from thermal noise, shot noise, jitter, etc. Here we will derive that randomness from the PUF. In general this can be done by applying a random challenge to the PUF *e.g.* in a range out of its specification. The random challenge can be generated by the reader. For a construction of a random number generator based on a PUF, we refer to O'Donnel *et al.* [169].

6.5.4 Estimated Results

If we compare the performances for implementations of two identification schemes, the Schnorr's and Okamoto's scheme, we get the Table 6.2. The conclusion is that the scheme of Okamoto in the case of ECC takes around 30% more hardware resources and the latency is at the same time longer for around 20%. With respect to the most compact solution, as required due to low gate-count and low-power requirements, implementing curve-based protocols with shorter bit-lengths appears to be an attractive option. For example, in the case of ECC one could use 130 bits long parameters. This solution would still maintain a suitable level of security [141], especially for low-cost RFIDs, and the gate complexity would scale-down accordingly resulting in even more acceptable solution for RFID standards [76].

Table 6.2: ECC performance comparison for two identification schemes.

Ident. scheme	# multiplications (M)	Storage (# 160-bit regs.)
Schnorr	3040	8 (P, k)
Okamoto	3580	11 $(P, Q, P+Q, k, l)$

6.6 Side-Channel Attacks

In the area of RFID as well as for sensor networks applications there is a big concern related to all kinds of side-channel and tamper attacks due to the accessibility of tags and nodes in general to the attacker. Therefore, some precautions are needed related to the design for cryptographic protocols and even some cheap countermeasures are desired.

Here, we briefly discuss the ability of an ECC implementation on an RFID tag to withstand simple side-channel attacks, such as the SPA. To prevent that, cryptographic algorithms should be implemented as sequences of operations that are indistinguishable through simple side-channel analysis. For example, by use of the approach of Montgomery [165] for scalar multiplication with balanced point operations such as in Chapter 4, one can achieve exactly the same number of field multiplications for point addition and doubling. The penalty for performance implies a small increase in the number of cycles, but the impact on the area is unrelated to this performance-security trade-off. We believe that this approach provides a low-cost side-channel resistance which is crucial for such applications.

6.7 Related Previous Work

The two most related papers to ours are [122] and [121]. Both deal with the cloning problem of RFID-tags and hence with the problem of using RFID-tags for anti-counterfeiting purposes. The focus of these papers is on efficient protocols for authenticating these tags. In these papers, one focuses on authentication of RFID-tags in the on-line situation; *i.e.* the reader shares a secret with the RFID-tag that is being authenticated. Clearly, when RFID-tags will become widely used, this is not a reasonable assumption. Considering the on-line case the light-weight protocol is obtained by applying the so-called HB^+ protocol (developed in [122])to the case of an RFID-tag equipped with a PUF. However, an attack to this protocol is recently announced by Gilbert *et al.* [89].

There is not much previous work dealing with hardware implementations of PKC on RFID tags or other low-power application platforms *e.g.* sensor nodes. Gaubatz *et al.* [85] showed that RSA is not a feasible solution while on the other

hand NtruEncrypt can be implemented in not more than 3000 gates. More recent work of Wolkerstorfer [251] is the first complete ECC low-power and compact implementation that meets the constraints imposed by the EPC standard. However, our solution can be even smaller as the off-line authentication in our case do not require full ECDSA algorithm to be executed on a single tag. That allowed for further optimization with respect to area.

6.8 Conclusions

In this chapter we have shown that by equipping RFID-tags with I-PUFs, the tags become unclonable and hence suitable for anti-counterfeiting purposes. Using our protocols, both the physical cloning attack as well as the cloning attack based on (actively or passively) attacking the protocol between the tag and the reader can be prevented. It has been shown that the required protocols are feasible on an RFID-tag, even in the off-line situation.

We have also discussed the feasibility of public key based secure identification protocols for RFID-tags. As an example we investigated the implementation of Schnorr's and Okamoto's identification protocols in detail. It was shown that Okamoto is just slightly more expensive than Schnorr's identification protocol. Furthermore, it is argued that HECC is feasible as well on an RFID tag. In order to compare its efficiency with ECC implementations more research is needed. We stress here again the fact that the memory part is the crucial one for our implementations and there is no other way to foresee the real impact of it before the actual implementations are completed. This part of our research is still work in progress. Also, the important issue here is low-power with respect to the $\left[\frac{J}{bit}\right]$ figure of merit as explained in Chapter 1. The power consumed should be below 10 μA with a voltage supply of 1.5 V [250], which makes this a challenging task. Therefore, with respect to these issues there is some more work to be done before one can completely answer the question about feasibility of curve-based cryptography for RFID tags.

Chapter 7

Side-Channel Entropy for Modular Exponentiation Algorithms

As discussed in Chapter 1, there exist many analysis methods which are used to attack implementations in hardware or software of cryptographic algorithms. Rather than attacking the algorithms themselves, specific properties of the implementation such as the timing behaviour [100], [133] and the power consumption [135] during execution of the algorithm are exploited to gain information about the secret key. These side-channel attacks can be very effective in breaking the cryptosystem, necessitating a careful implementation and the development of countermeasures.

In this chapter we focus on the information leakage from the secret exponent in modular exponentiation algorithms (like those used in Discrete-log or RSA cryptosystems). This leakage from the secret exponent is due to obtaining information on the number of square and multiply operations performed. More precisely, obtained information is Hamming weight of the secret exponent.

We use the combinatorial approach to develop a simple model for Hamming weight leakage. Using the so-called random exponent model (REM) in an information theoretic approach, the exponent leakage is shown to be too small to be exploited in practice, *e.g.* 6.06 bits for a 1024-bit exponent. An extension of the model to observation of multiple parts of the exponent is also given. The general relation between information leakage and exponent length is proven to be logarithmic.

In this first order approximation it was assumed that the secret exponents are chosen at random from the set of odd numbers. For this random exponent model the exact probability distribution is derived, and the entropy $H(K)$ of the number of operations K is determined. Furthermore, the validity of this model is shown to

hold when the exponent is relatively prime to the Euler (Totient) φ function [129] of the modulus. We focus on RSA keys, *i.e.* triplets (d, p, q), where pq is the secret factorization of the modulus N, (so $N = pq$) and d is the private RSA exponent. The probability of such an exponent depends on the choice of the modulus N, *i.e.*, on the type of primes p and q. In particular, when considering the use of strong primes we conclude that the entropy does not significantly differ from the one that we derived in the REM. Moreover, we prove that all options for primes *i.e.* from safe to random primes, are compliant with the developed model. The results show that this information is far from exploitable for the practical bit-lengths in nowadays applications. This work was published in [12, 13].

7.1 Assumptions for our Model

The first question we ask ourselves is how much information we obtain (in the Shannon sense) when observing the total number of operations, *i.e.* modular multiplications and squarings whether in total for a complete modular exponentiation, or in a certain part of the secret exponent. The simplest instance of this problem is to consider a left-to-right square-and-multiply algorithm for exponentiation in which we cannot distinguish between multiplication and squarings. In a first order approximation it is assumed that the secret exponents are chosen at random from the set of odd numbers $3, 5, ..., 2^n - 1$. To summarize, we have as follows.
Assumptions:

1. The secret exponent d is an arbitrary odd number $d \in \mathbb{Z}_N^*$.

2. Timings for the operations of multiplying and squaring of two numbers are the same. This assumption is realistic considering the required constant-time implementations as a necessary condition for side-channel security. A vast majority of hardware implementations nowadays meet this condition.

3. Each user chooses her/his private key independently and uniformly in the set of all odd numbers $\{3, 5, ..., 2^n - 1\}$. In particular, for all $d \in \{3, 5, ..., 2^n - 1\}$, $P(X_A = d) = P(X_B = d) = \frac{1}{2^{n-1}-1}$ where $P(X_A = d)$ gives the probability that user chooses the number d as her/his private exponent.

Here we focus on modular exponentiation which is used in the RSA cryptosystem [193]. Modular exponentiation *i.e.* the calculation of $M^e \bmod N$ is usually performed by use of the standard square-and-multiply algorithm *i.e.* Algorithm 3.1 as already mentioned in Chapter 3.

The question is how much information can be obtained about the secret exponent when observing the total number of operations, *i.e.* modular multiplications and squarings. Here, a computationally unrestrained adversary is considered as we discuss the side-channel security in the sense of Shannon Information Theory.

In this case, it is assumed that the adversary has unlimited time and computing power at her/his disposal. The security against such a computationally unbounded adversary is usually referred to as *unconditional security* [152], or *theoretical security* [208].

7.2 The Random Exponent Model

We aim to compute the number of exponents e of length l, which is at most $n\,(l \leq n)$. This means that if length $l < n$ then the first $n - l$ MSBs are zero. Let k denote the number of multiplications (or squarings) that have to be computed for an exponent e. We will compute this in a number of steps.

First, we determine the number of (different) exponents of length exactly n (so with MSB equal to 1), which require exactly k operations. The least significant bit (LSB) is also 1, because the exponent, say e, is odd, and therefore we have to count all different sequences of 0's and 1's for the remaining $n-1$ bits. Let us first calculate the number of operations for computing M^e by use of Algorithm 3.1.

The number of operations, which are squarings (S) and multiplications (M) are determined by the values of the bits on positions from $n-2$ to 0. More precisely, according to Algorithm 3.1, there will be exactly $n-1$ squarings and the number of multiplication is equal to the number of 1's in the binary representation of an exponent. Let the total number of operation be denoted with k, and w_H be its binary Hamming weight. Then, we conclude:

$$k = n - 1 + w_H(e) - 1 = n - 2 + w_H(e). \tag{7.1}$$

We can now count how many odd exponents exist with some fixed number of operations k, which have bit-length exactly n. Here, the LSB is also not taken into account (being fixed to 1). It is easy to see that the number of different exponents $g(n,k)$ for which $e_{n-1} = 1$ and $e_0 = 1$ that imply the total number of k multiplications is:

$$g(n,k) = \binom{n-2}{k-n} \tag{7.2}$$

where $k = n, n+1, ..., 2n-2$.

Secondly, we aim to determine the number of odd exponents of length up to n with exactly k operations involved in exponentiation. We get the following lemma:

Lemma 10 *Let $G(n,k)$ denote the number of different odd exponents of length at most n that imply exactly k operations when Algorithm 3.1 is performed. Then the following formula holds:*

$$G(n,k) = \sum_{i \geq \frac{k+2}{2}}^{n} \binom{i-2}{k-i}. \tag{7.3}$$

Proof: According to previous consideration we have to count different exponents of all possible bit-lengths i, where i is related to the number of operations k. The following equality holds: $i \leq k \leq 2i-2$. Therefore, we have to count $g(i,k)$ for all $\frac{k+2}{2} \leq i \leq n$. This concludes the proof. □

Example. To illustrate this formula let us find all odd exponents of length up to $n = 7$, for which $k = 8$. The possible lengths are $n = 5$: 11111; $n = 6$: 100111, 101011, 101101, 110011, 110101, 111001; and $n = 7$: 1000011, 1000101, 1001001, 1010001, 1100001. According to formula (7.3), we can easily check:

$$G(7,5) = \sum_{i=5}^{7} \binom{i-2}{8-i} = \binom{3}{3} + \binom{4}{2} + \binom{5}{1} = 1 + 6 + 5 = 12.$$

Let us now rewrite the formula (7.3) in a different way to simplify the calculations. We consider first the case when the number of operations is even, so $k = 2m$. In this case $G(n,k)$ becomes:

$$G(n,k) = \sum_{i=m+1}^{n} \binom{i-2}{2m-i} = \binom{m-1}{m-1} + \binom{m}{m-2} + \ldots + \binom{n-2}{2m-n}. \quad (7.4)$$

In order to compute numbers $G(n,k)$ we use the following theorem where the Fibonacci numbers appear [195]. The Fibonacci numbers F_n are integers that satisfy the recurrence relation $F_n = F_{n-2} + F_{n-1}$, for $n = 3, 4, \ldots$ where $F_1 = F_2 = 1$.

Theorem 15

$$F_{n+1} = \sum_{j=0}^{\lfloor \frac{n}{2} \rfloor} \binom{n-j}{j} \qquad \forall n \geq 0. \quad (7.5)$$

Proof: Follows directly as sums of diagonals in Pascal's triangle [195]. □

It is easy to see that this equality can be applied on the equation (7.3) if and only if $n = k$ and in that case we get $G(n,n) = F_{n-1}$. This holds trivially for $n > k$. So, we conclude: $G(n,k) = F_{n-1}$, for all n, $n \geq k$, where k is fixed.
We still have to add these cases for which $k > n$. Consider again the formula (7.5) (for k even):

$$\begin{aligned} G(n,k) &= \sum_{i=\frac{k+2}{2}}^{n} \binom{i-2}{k-i} = \sum_{i=\frac{k+2}{2}}^{n} \left[\binom{i-3}{k-i} + \binom{i-3}{k-i-1} \right] \\ &= \sum_{i=\frac{k}{2}}^{n-1} \binom{i-2}{k-i-1} + \sum_{i=\frac{k}{2}}^{n-1} \binom{i-2}{k-i-2} \\ &= \sum_{i=\frac{k-1+2}{2}}^{n-1} \binom{i-2}{(k-1)-i} + \sum_{i=\frac{k-2+2}{2}}^{n-1} \binom{i-2}{(k-2)-i} \\ &= G(n-1,k-1) + G(n-1,k-2)\,. \end{aligned} \tag{7.6}$$

We obtain the recurrence, which can be further expanded:

$$\begin{aligned} G(n,k) &= G(n-1,k-1) + G(n-1,k-2) \\ &= G(n-2,k-4) + 2G(n-2,k-3) + G(n-2,k-2) \\ &= \ldots = \sum_{i=0}^{\frac{k}{2}-1} \binom{\frac{k}{2}-1}{i} G(n-\frac{k}{2}+1, 2+i) \end{aligned} \tag{7.7}$$

The following theorem directly follows:

Theorem 16 *The numbers $G(n,k)$, for k even, satisfy the following equation:*

$$G(n,k) = G(n-1,k-2) + G(n-1,k-1) = \sum_{i=0}^{\frac{k}{2}-1} \binom{\frac{k}{2}-1}{i} G(n-\frac{k}{2}+1, 2+i). \tag{7.8}$$

Proof: Using equations (7.6) and an induction argument the proof follows immediately. □

Using (7.8) we can easily compute $G(n,k)$ for arbitrary n and k. The numbers $G(n,k)$ are listed in Table 7.1, for $n \leq 12$ and $k \leq 20$. The previous conclusions are easily verified. On the main diagonal we notice the Fibonacci numbers. Also, the numbers under this diagonal are the same within each column. Namely, we proved that $G(n,k)$ does not depend on k, for all $k \leq n$ and that this number is equal to $(n-1)^{\text{st}}$ Fibonacci number. We also note that the recurrence (7.8) holds.
Example. We are interested in $G(10,14)$ *i.e.*, how many exponents exist in the interval $[1, 2^{10}]$ which involve 14 operations while performing exponentiation. From Eq. (7.8) it is easy to check:

$$\begin{aligned} G(10,14) \quad &= \textstyle\sum_{i=0}^{6} \binom{6}{i} G(4,2+i) = G(4,2) + 6G(4,3) + 15G(4,4) + \\ &+ 20G(4,5) + 15G(4,6) = 1 + 6 \cdot 1 + 15 \cdot 2 + 20 \cdot 2 + 15 \cdot 1 = 92 \end{aligned}$$

Table 7.1: The behavior of numbers $G(n, k)$.

k / n	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	1	0	0	...															0
3	1	1	1	0	...														0
4	1	1	2	2	1	0	...												0
5	1	1	2	3	4	3	1	0	...										0
6	1	1	2	3	5	7	7	4	1	0	...								0
7	1	1	2	3	5	8	12	14	11	5	1	0	...						0
8	1	1	2	3	5	8	13	20	26	25	16	6	1	0	...				0
9	1	1	2	3	5	8	13	21	33	46	51	41	22	7	1	0	0	0	0
10	1	1	2	3	5	8	13	21	34	54	79	97	92	63	29	8	1	0	0
11	1	1	2	3	5	8	13	21	34	55	88	133	176	189	155	92	37	9	1
12	1	1	2	3	5	8	13	21	34	55	89	143	221	309	365	244	247	46	1

Let K denote the random variable associated to the number of operations k in an exponentiation with a randomly selected exponent. The probability distribution of K is given by:

$$Pr[K = k] = (2^{n-1} - 1)^{-1} G(n, k). \tag{7.9}$$

As mentioned before, the exponents are selected at random from the set of odd numbers $\{3, 5, \ldots, 2^n - 1\}$ with equal probability $(2^{n-1} - 1)^{-1}$.

As can be seen, even for very high values of n there is always a non-zero probability of a very small number of operations. In the remainder we will discuss these results from the point of view of side-channel leakage of the Hamming weight. This type of leakage typically occurs in our model *i.e.* when the modular exponentiation is performed as a simple square-and-multiply algorithm and the adversary has the precise timing information. However, this type of leakage occurs also for power analysis where the behavior of power consumption is sometimes proportional to Hamming weight of secret data that are being processed [162]. Another common type of model for power consumption is based on the leakage of Hamming distance.

7.3 Entropy of Operations and Exponent Leakage

We give here definitions for entropy, relative entropy and mutual information as in the book by Cover and Thomas [57]. Let X be a discrete random variable with alphabet $\mathcal{X}$ and with probability distribution function $p(x) = Pr\{X = x\}, x \in \mathcal{X}$. In this case we also write $X \sim p(x)$.

Definition 31 *The entropy H(X) of a discrete random variable X is defined by:*

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log p(x) .$$

The log is with the base 2 and entropy is expressed in bits.

Definition 32 *The joint entropy H(X,Y) of a pair of discrete random variables (X,Y) with a joint distribution p(x,y) is defined as:*

$$H(X, Y) = -\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) \log p(x, y) .$$

We also define the conditional entropy of a random variable with respect to another random variable.

Definition 33 *If $(X, Y) \sim p(x, y)$, then the conditional entropy $H(Y|X)$ is defined as:*

$$\begin{aligned} H(Y|X) &= -\sum_{x \in \mathcal{X}} p(x) H(Y|X = x) \\ &= -\sum_{x \in \mathcal{X}} p(x) \sum_{y \in \mathcal{Y}} p(y|x) \log p(y|x) . \end{aligned}$$

The following rule holds:

Theorem 17

$$H(X,Y) = H(X) + H(X|Y)$$

We introduce now the mutual information, which is a measure of the amount of information that one random variable contains about another random variable.

Definition 34 *Let X and Y two random variables with a joint probability distribution p(x,y) and their probability distributions p(x) and p(y). The mutual information I(X;Y) is the relative entropy between the joint distribution and the product distribution p(x)p(y), i.e.,*

$$\begin{aligned} I(X;Y) = \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log\frac{p(x,y)}{p(x)p(y)} \\ =& H(X) - H(X|Y) \\ =& H(Y) - H(Y|X) \\ =& H(X) + H(Y) - H(X,Y)\,. \end{aligned}$$

In the light of side-channel cryptanalysis, the previous consideration is tightly related to the following question: if the attacker possesses the exact information on the number of operations k performed while exponentiation has been executed, what is the probability for him to successfully guess the secret exponent e? We assume, as usual, that he has detailed knowledge on the algorithm of exponentiation that is used and the relevant bit-lengths. So, we are interested in the expected amount of information that K gives about E. Formulated in an information theoretic sense: we derive an expression for the mutual information $I(K;E)$ between the random variables K and E, where E is the random variable corresponding to the secret exponent e. This notion of given-away information of the key was previously introduced in [115].
For E and K random variables, we have:

$$\begin{aligned} I(K;E) &= H(K) - H(K|E) \\ &= H(E) - H(E|K)\,. \end{aligned}$$

Here $H(K)$, $H(E)$ denote the Shannon entropies of K and E respectively, and $H(K|E)$ and $H(E|K)$ are conditional entropies of K with given E and vice versa. $H(E|K)$ is usually interpreted as the uncertainty about E given K.

Note that for any given exponent, the number of operations is exactly determined by (7.1). Hence, the uncertainty in K, given E is zero, resulting in:

$$\begin{aligned} I(K;E) &= H(K), \text{ and} \\ H(E|K) &= H(E) - H(K). \end{aligned} \tag{7.10}$$

Previous equations clearly show that the information leakage is exactly equal to the entropy $H(K)$ of the number of operations. Having the exact probability distribution of this random variable from the previous section, this exponent leakage can precisely be determined.

By assuming a uniform probability distribution on K and by observing that K takes on values $k \in \{2, 3, ..., 2n-2\}$, we can easily bound $H(K)$ from above:

$$H(K) \leq \log_2(2n-3) \text{ bits.} \tag{7.11}$$

Approximating $H(E) \approx n-1$ we obtain the following lower bound on the conditional exponent uncertainty:

$$H(E|K) = H(E) - H(K) \geq n - 1 - \log_2(2n-3) \text{ bits.} \tag{7.12}$$

From equations (7.11) and (7.12) the logarithmic relation between information leakage and maximum exponent length can be observed. Applying (7.12) for $n = 1024$ bits, we get $H(E|K) \geq 1012$, so the average amount of information that K gives about E is not more than 12 bits. However, from Table 7.2 it can be seen that the exact exponent leakage in this case amounts to 6.06 bits.

Table 7.2: Data for the entropy and upper bound for two models.

		Ran. exp. model		Binomial Model	
N	$log_2(N)$	$H(K)$	Bound	$H(K)$	Bound
2	1	0	0	0	0
4	2	2.24	2.32	1.50	1.58
8	3	3.09	3.70	2.33	2.80
16	4	3.46	4.86	2.95	3.90
32	5	3.18	5.93	3.50	4.95
64	6	4.11	6.97	4.02	5.98
128	7	4.63	7.98	4.54	6.99
256	8	5.09	8.99	5.04	7.99
512	9	5.57	10.00	5.54	9.00
1024	10	6.06	11.00	6.05	10.00
2048	11	6.54	12.00	6.55	11.00

A slightly different model is one in which the most significant bit of the exponent is always taken to be 1. The justification of this model comes from the fact that in practice key generation algorithms may force the secret exponent to be of maximum length. Hence, for this model the length n is fixed and the Hamming weight of the exponent is a binomially distributed random variable. Consequently,

the number of operations is a binomially distributed random variable, K, which takes on values $k \in \{n-2, \ldots, 2n-2\}$. Use of this distribution, yields the following upper bound on the entropy of K:

$$H(K) \leq \log_2(n+1) \text{ bits.} \tag{7.13}$$

The two bounds for $H(K)$ differ by ≈ 1 bit. For long exponent lengths, the contribution of the short exponents tends to become negligible and the two models give identical results for exponent leakage. Figure 7.1 shows the exponent entropy loss for both models and their corresponding upper bounds.

For this random exponent model the exact formula for the probability distribution is derived, and the entropy $H(K)$ of the number of operations K is determined. The results show that this information is far from exploitable for the practical bit-lengths in current applications.

This observation can be furthermore extended in observing the situation in which the attacker is able to attack some specific part(s) of the exponent. The relation between exponent information leakage and the number of exponent parts is also given.

7.4 Observation in Parts

In this section we generalize the results from the previous section by considering the possibility that the attacker is observing the random exponent in two or more parts (*i.e.* group of bits) of the secret exponent. The assumption here is that the number of operations (total of squares and multiplies when performing an exponentiation) in each part is known to the adversary. Obviously, if each part comprises only one bit of the exponent, then she knows the value of that bit in each part exactly from the corresponding one or two operations observed. If, on the other hand, the number of parts is less, with more bits in each part, then the uncertainty for the attacker will be higher, as there is no longer a one-to-one correspondence between the values of the bits and the number of operations for each part. We consider the fixed exponent length model in which the number of operations has a binomial distribution. It is assumed that the exponent of length n bits is partitioned into parts of equal length l bits, with the exception of the most significant part, which contains the remaining $n - \lfloor \frac{n}{l} \rfloor l$ bits.

Let $h_B(m)$ denote the entropy (in bits) of a binomially distributed random variable B, representing the number of operations in an exponent part of length m bits. Let $H(K^t)$ denote the total amount of information obtained by observing the operations from all parts. Then if $n - \lfloor \frac{n}{l} \rfloor l > 0$, the following relation holds:

$$H(K^t) = h_B(\ell - 1) + \left(\left\lfloor \frac{n}{\ell} \right\rfloor - 1 \right) h_B(\ell) + h_B\left(n - \left\lfloor \frac{n}{\ell} \right\rfloor \ell - 1 \right) \quad \text{bits.} \tag{7.14}$$

This can be seen from the independence of the parts and the fact that the least and most significant bits are always 1. More precisely, the first part corresponds to

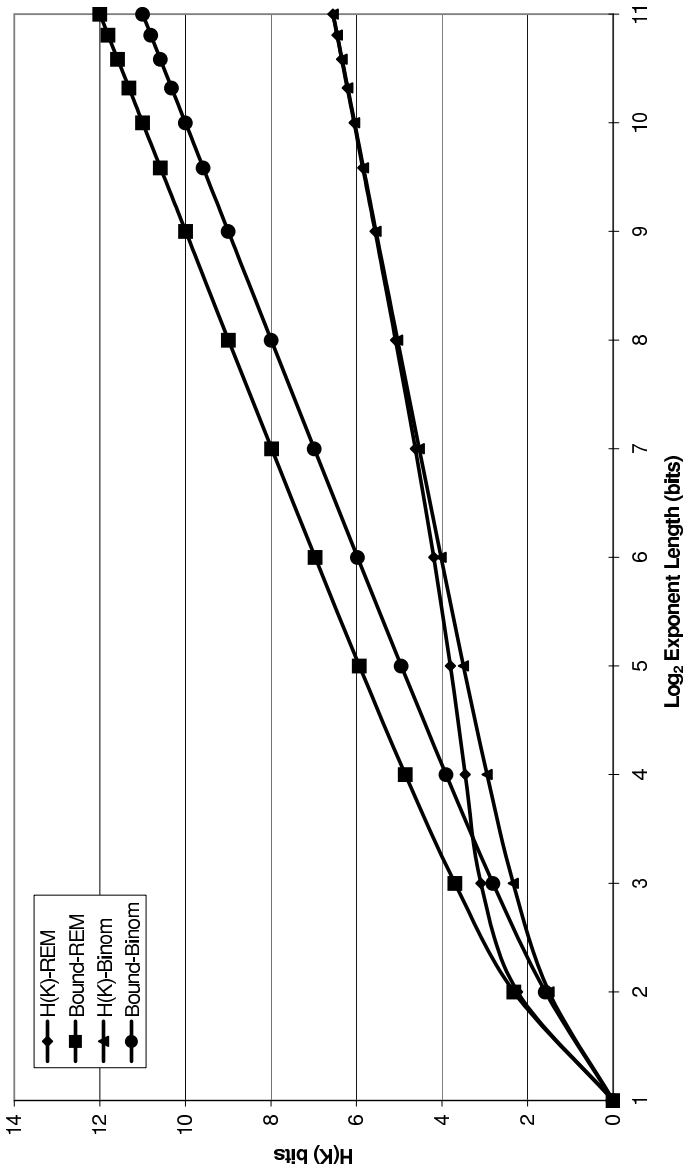


Figure 7.1: Exponent entropy loss for two different models (Random Exponent Model (REM) and binomial model.

the first l bits from the LSB. The entropy is influenced by all bits but 1 (the LSB). The largest (middle) part contains $\left\lfloor \frac{n}{\ell} \right\rfloor - 1$ parts of length l and the corresponding entropy is therefore $h_B(\ell)$. Finally, the last part is related to the first $n - \lfloor \frac{n}{l} \rfloor l$ where we know that $e_n = 1$.

Figure 7.2 shows $H(K^t)$ as a function of the part-length for a 1024 exponent on a log-log scale. The steps in the curve are caused by the sudden change in the number of parts $\lfloor \frac{n}{l} \rfloor$.

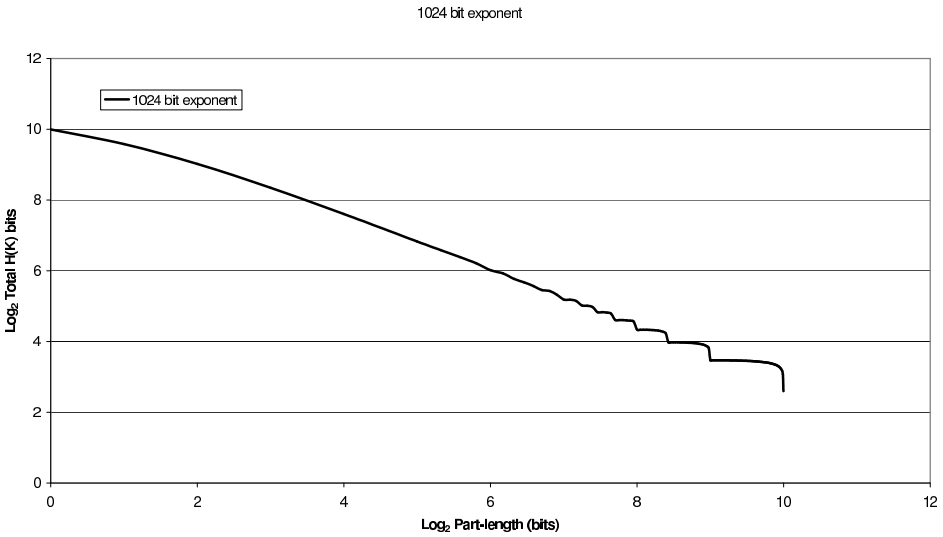


Figure 7.2: Random Exponent Model - Observation in Parts.

Our next step is to consider a more realistic model in which exponents are taken to be random, but co-prime to a random even composite number should be further developed. Also according to the work of Wiener [247] short secret RSA exponents can be successfully attacked, so in practice not all bit-lengths are possible. He showed that when d is less than $N^{0.25}$ the RSA system is insecure. Boneh and Durfee improved this bound to $N^{0.292}$ [35].

7.5 How Relevant is the Model

In this section we will look into the more realistic model in which exponents are taken to be random, but co-prime to a random even composite number.

When using the RSA public key cryptosystem, one encounters the Euler Totient function $\varphi(N)$, where N is the modulus, usually a composite integer, that

is the product of two primes. The public encryption exponent e and the corresponding secret decryption exponent d are related by the fact that they are each others multiplicative inverse in the residue ring where the operation is modular multiplication (mod $\varphi(N)$), *i.e.*: $e \cdot d \equiv 1 \bmod \varphi(N)$. Clearly, e and d must be units in the ring, *i.e.* $\mathrm{GCD}(e, \varphi(N)) = 1$. The number N_e of exponents satisfying this requirement is given by $N_e = \varphi(\varphi(N))$. This number of exponents is of interest in problems where the total diversity of public-secret key-pairs is relevant, e.g. in considering the exponent information leakage due to side-channel information. In the sequel we will look at the minimum and maximum values that this number N_e can take on for various moduli N.

Given an integer N in its general form:

$$N = \prod_i {p_i}^{e_i}, p_i \neq p_j$$

and all $e_i > 0$, where p_i are prime factors of N. Then, by the definition of $\varphi(n)$ [129]:

$$\varphi(N) = \prod_i (p_i - 1){p_i}^{e_i - 1} = N \prod_i (1 - \frac{1}{p_i}) \,. \tag{7.15}$$

From the formulas above it is evident that the highest value that φ can attain is $N - 1$, if and only if N is a prime. Moreover, it follows from (7.15) that the multiplicities e_i do not matter, and hence the least value of φ occurs when N is composite and equal to the product of many different small primes.

In the standard RSA setting the modulus N is the product of two distinct primes p and q, and therefore $\varphi(N) = (p-1)(q-1) = N - (p+q) + 1$, which always contains a factor of 4. If the objective is to have as many exponents as possible, *i.e.* to get the highest value of $\varphi(\varphi(N))$, then clearly the best option is that $(p-1) = 2r$ and $(q-1) = 2s$, with r and s again primes.

In the sequel we will consider the relative diversity of RSA exponents, *i.e.* $\eta = N_e/N$.

7.5.1 Sophie Germain Primes

For $(p-1) = 2r$ and $(q-1) = 2s$, with r and s again primes one gets $\varphi(\varphi(N)) = 2(r-1)(s-1)$. Prime numbers r with the property that $2r+1$ is again prime, are known as the so-called "safe primes" [155] or Sophie Germain primes.

Suppose one uses safe primes, then we have $\varphi(N) = 4rs$ and $\varphi(\varphi(N)) = 2(r-1)(s-1)$, so we have

$$\frac{\varphi(\varphi(N))}{\varphi(N)} = \frac{1}{2}\left(1 - \frac{1}{s} - \frac{1}{r} + \frac{1}{rs}\right), \tag{7.16}$$

showing that the fraction of real RSA exponents is about half that of $\varphi(N)$.

Also, $\varphi(\varphi(N)) = 2(\frac{p-3}{2})(\frac{q-3}{2}) = \frac{1}{2}(N - 3(p+q) + 9)$. As the minimum value of $p+q$ is obtained for $p = q = \sqrt{N}$ we have the upper bound $\varphi(\varphi(N)) = \frac{1}{2}(N - 6\sqrt{N} + 9)$. Now if we assume that indeed $p \simeq q \simeq \sqrt{N}$, we loose no more than approximately 1 bit in the entropy of the secret exponent and this is comparable with our previous results when taking all odd exponents. Under this assumption we arrive at

$$\frac{\varphi(\varphi(N))}{\varphi(N)} = \frac{1}{2}\left(\frac{\sqrt{N}-3}{\sqrt{N}-1}\right)^2 .$$

7.5.2 Strong Primes

Prime numbers p, with $r|(p-1)$, r a large prime, where $p+1$, $p-1$, and $r-1$ all have a large prime factor are known as strong primes [155]. Let us consider strong primes, *i.e.* $p = 2k_p r + 1$ and $q = 2k_q s + 1$, where r and s are large prime factors of $p-1$ and $q-1$ resp. and k_p and k_q are arbitrary small integers. We now have $\varphi(N) = 4k_p k_q rs$ and consequently $\varphi(\varphi(N)) = (r-1)(s-1)\varphi(4k_p k_q)$, so one can write

$$\frac{\varphi(\varphi(N))}{\varphi(N)} = \frac{\varphi(4k_p k_q)}{4k_p k_q}\left(1 - \frac{1}{s} - \frac{1}{r} + \frac{1}{rs}\right) \approx \frac{\varphi(4k_p k_q)}{4k_p k_q} . \tag{7.17}$$

It now depends on the factorization of k_p and k_q with how many bits the entropy of the exponent is decreased. The smallest decrease occurs if k_p and k_q are powers of 2, the reduction being $\frac{1}{2}$, or 1 bit in entropy. The biggest decrease occurs if k_p and k_q are composite, containing many small distinct prime factors. For example, suppose again that $p \simeq q \simeq \sqrt{N}$, and that the product $k_p k_q$ is at most a 32-bit composite integer with value $k_p k_q = 3 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19 \cdot 23 \cdot 29$. In this case $\varphi(4k_p k_q) = 2 \cdot 2 \cdot 4 \cdot 6 \cdot 10 \cdot 12 \cdot 16 \cdot 18 \cdot 22 \cdot 28$ and the reduction factor $\frac{\varphi(4k_p k_q)}{4k_p k_q}$ becomes 0.158, resulting in a decrease of the exponent entropy of at most 2.6 bits. The way strong primes are generated [94], the values of k_p and k_q are usually less than 16 bits for 512 bit primes. Consequently, the decrease in exponent diversity is also less than 2.6 bits.

Strong primes were introduced in the context of so-called "non-weak" RSA keys. A simple method for finding strong, random large primes was given by Gordon already in 1984 [94]. The reasoning behind it was that with strong primes one counters factoring, as well as cycling attacks. There exist arguments for both, using strong or just random primes. Rivest and Silverman [194] argue that it is unnecessary to use strong primes in the RSA cryptosystem. Namely, for the most recent factoring method based on elliptic curves (ECM) due to H. Lenstra [143], strong primes offer no extra protection. On the other hand, Joye et al. [118] state that, since strong primes can only yield a safer solution and it is not much more difficult to generate them, it is better to use strong instead random primes. Some

standards still recommend the use of strong primes for RSA, for example ISO/IEC 9594-8.

7.5.3 Random Primes

Similar reasoning as in the previous subsection shows that a lower bound to $\varphi(\varphi(N))$ is obtained by assuming $\varphi(N)$ to be equal to $2^2{\cdot}3{\cdot}5{\cdot}7{\cdot}11{\cdot}13{\cdot}17{\cdot}19\ldots a_{max}$. In this case the fraction of exponents $\frac{\varphi(\varphi(N))}{\varphi(N)}$ is given by equation (7.15) when taking a_{max} for the largest p_i. This bound is realistic in that it sometimes refers to primes of the form $(a_j \cdot \ldots \cdot a_k + 1)$ that exist but sometimes they are the product of two or more smaller primes. Figure 7.3 shows this bound for the decrease of the exponent entropy as a function of bit-lengths. As can be seen the extra condition imposed on exponents results in a minor decrease of the entropy, viz. at most 3.6 bits for 1024-bit RSA.

Let us consider an example. Take $\varphi(N) = 2^2 \cdot 3 \cdot 5 \cdot 7 \cdot 11 \cdot 13 = 60060$, so $\varphi(\varphi(N)) = 2^2 \cdot 4 \cdot 6 \cdot 10 \cdot 12 = 2^8 \cdot 3^2 \cdot 5 = 11520$. One candidate for N is $(2 \cdot 5 \cdot 13 + 1)(2 \cdot 3 \cdot 7 \cdot 11 + 1) = 131 \cdot 463 = 60653$. The same holds for $N = 23 \cdot 2731 = 62813$. So a 16-bit modulus results in an exponent entropy of somewhat less than 14 bits.

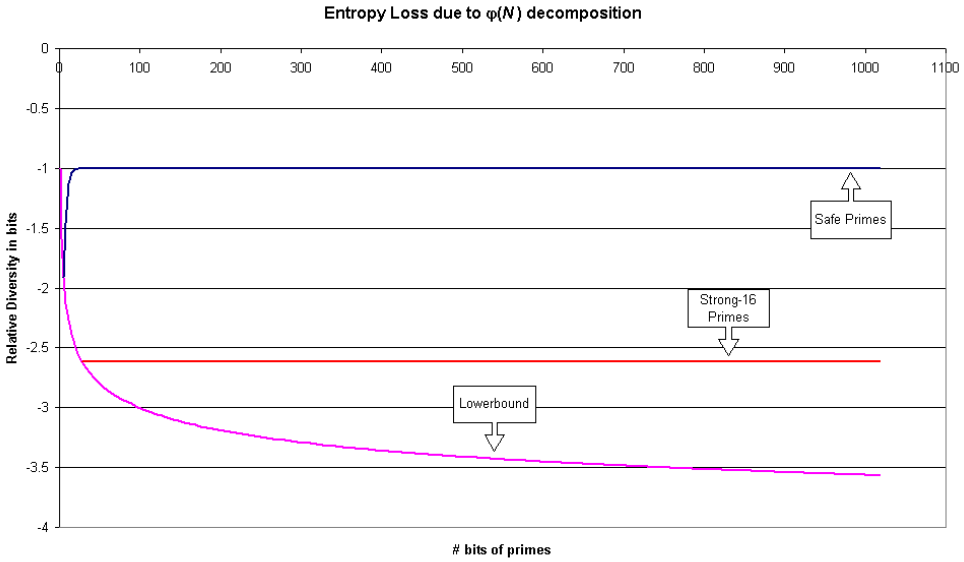


Figure 7.3: Entropy loss due to decomposition of $\varphi(N)$ into primes.

The conclusion from the discussion above is that the diversity (or entropy) of

the secret exponents is a little bit less than assumed in our previous model. One should consider odd numbers d, $3 \leq d \leq \varphi(N) - 1$, that are relatively prime to $\varphi(N)$ and there are exactly $\varphi(\varphi(N)) - 1$ of them. Hence, the lowest entropy is achieved for $2^2 \cdot 3 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19 \ldots a_{max}$. The values of a_{max} can be determined for 512, 1024, 2048-bit moduli and consequently $2^2 \cdot 3 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19 \ldots a_{max}$ computed. From number theory it is known that $a_{max} \sim \log N$ and $\frac{\varphi(\varphi(N))}{\varphi(N)} \sim \frac{1}{\log \log N}$. The first 132 prime numbers were used to calculate the bound depicted in Figure 7.3.

7.6 Related Work

As related previous work we mention that of Waldvogel and Massey [238] in which they considered the entropy of Diffie-Hellman keys. More precisely, they derived the probability distribution and the entropy of the key generated by the Diffie-Hellman key-exchange protocol. As a Diffie-Hellman key, they consider the number $\alpha^{x_A x_B}$ where x_A, x_B are the private keys of the involved parties, say Alice and Bob. It is assumed that $x \in \mathbb{Z}_{p-1}$ and for the public key holds: $y = \alpha^x \in \mathbb{Z}_p^*$. They also assume that each user chooses her/his private key uniformly at random in the set of all units of $\mathbb{Z}_{p-1}$. Then the probability $\mathcal{P}(\alpha^{x_A x_B} = \alpha^t)$ that Alice and Bob generated the Diffie-Hellman key α^t, where $t \in \mathbb{Z}_{p-1}$ equals $\mathcal{P}(\alpha^{x_A x_B}) = \frac{1}{\varphi(p-1)}$. This means that if each user chooses her/his private key independently and uniformly at random then the Diffie-Hellman keys are uniformly distributed over $\mathbb{Z}_{p-1}^*$. For the case where p is a safe prime, the probability distribution of keys is shown to be close to the uniform distribution and the key entropy is the maximum possible. In this context safe primes are not weaker than other primes. Therefore, our conclusions are consistent with those of [238].

Another related work that is considering this problem as a pure number theoretic question is the paper of Banks and Shparlinski [8] in which they estimated the number of sparse exponents e in RSA exponentiation that are co-prime to $\varphi(N)$. They determined the number $N_{n,k}(N)$ of n-bit exponents with k non-zero bits which and they showed that this number is close to its expected value, so more precisely it holds:

$$N_{n,k}(N) \sim 2\binom{n-1}{k-1}\frac{\varphi(\varphi(N))}{\varphi(N)},$$

when N runs through the set of RSA moduli $N = pq$, where p and q are two prime numbers. They also use the fact that the private exponent d is uniformly distributed $\bmod \varphi(N)$.

7.7 Conclusions

We have considered secret exponent information leakage for modular exponentiation algorithms in an information theoretic setting. To this end, expressions for the number of exponents resulting in a given number of square and multiply operations have been derived. These expressions straightforwardly lead to the probability distribution of the number of operations in the random exponent model. This so-called random exponent model (REM) has subsequently been used to obtain data on the exponent leakage, showing that the amount of information is too small to be exploited in practice. A further refinement of the model, considering the observation of multiple parts of the exponent, has also been given.

We have also extended the validity of the Random Exponent Model to the more realistic situation of RSA exponents which are co-prime to $\varphi(N)$. If one considers the total number $\varphi(\varphi(N))$ of such exponents, it was shown that for safe primes the available fraction of exponents is about one half of the total. For strong primes this fraction can be somewhat less, depending on the small factors in the $\varphi(N)$. In this case the equivalent loss in entropy, assuming equally probable and independent exponents, is less than 3 bits for 1024-bit exponents.

For the case of random primes an obvious lower bound to the fraction $\frac{\varphi(\varphi(N))}{\varphi(N)}$ results in a useful upper bound for the entropy loss. This bound indicates a loss of at most 3.6 bits for 1024-bit exponents.

In summary, the results of this research indicate that the REM is a good way to model the information leakage due to side-channel analysis. Our conclusion is that Hamming weight information in this case is not really useful information *i.e.* if the only info is the number of multiplications and squarings (or Hamming weight of the exponent), the adversary cannot do much with it.

Chapter 8

Algebraic Tori Implementations on Reconfigurable Hardware

When introducing the solution for PKC, Diffie and Hellman suggested to use the DLP in a large field of a prime order and ever since there was a lot of work on the DLP. The present situation is as follows: in order to prevent index-calculus attacks one should use a finite field $\mathbb{F}_p$, which has an order of at least 1024 bits, *i.e.* $p \geq 2^{1024}$. However, it could be attractive to work in a subgroup of $\mathbb{F}_p$, $\langle g \rangle \subset \mathbb{F}_p$ for the purpose of compression. For all types of square root attacks it is required that $ord(g) \geq 2^{160}$. Therefore, the question was whether it is possible to represent elements in a subgroup with fewer bits than the elements of the full $\mathbb{F}_p$ field. The first work that considered this problem was done by Brouwer *et al.* [44]. They showed that compression is possible in an extension field and they also conjectured that a factor of compression of $\frac{n}{\varphi(n)}$ could be obtained, where n is the degree of the extension field. The examples that appeared later include the LUC cryptosystem by Lennon and Smith [139] (for $n = 2$) and the XTR cryptosystem of Lenstra and Verheul [142] (for $n = 6$). Rubin and Silverberg revisited the problem of compression for extension fields for the case of algebraic tori. They showed that the factor $\frac{n}{\varphi(n)}$ can be achieved for compression if the algebraic torus T_n is rational. They introduced a new pulic-key cryptsystem CEILIDH [196] that is based on torus T_6. Afterwards, van Dijk *et al.* [234] introduced efficient arithmetic for T_{30}.

In this chapter we consider hardware implementations of CEILIDH. The work of Granger *et al.* [97] was first to introduce efficient arithmetic on T_6 and they compared the performance of CEILIDH and XTR on a PC. They concluded that

the difference in performance is much smaller than one would expect (in favour of XTR). We follow the approach of [97] for efficient T_6 representations, but we make some typically hardware oriented choices. We also outline an FPGA architecture for the hardware implementation of this cryptosystem and we compare it to RSA and ECC.

8.1 The Torus $T_6(\mathbb{F}_p)$

The public-key cryptosystem based on the torus $T_6(\mathbb{F}_p)$ was introduced at CRYPTO 2003 by Rubin and Silverberg under the name CEILIDH. The idea behind it was to obtain the security of $\mathbb{F}_{p^6}^*$, while data to be transmitted are compressed with a factor 3. More precisely, the elements required for cryptographic protocols are from the affine plane $\mathbb{A}^2(\mathbb{F}_p)$. More details are given in Chapter 2.

8.2 Representations and Algorithms for $T_6(\mathbb{F}_p)$

In this section we discuss several field representations and algorithms for a hardware implementation of the CEILIDH cryptosystem. We follow the notation and algorithms as described in [97]. An algebraic torus $T_6(\mathbb{F}_p)$ is defined in such a way that over $\mathbb{F}_{p^6}$, this variety is isomorphic to $\varphi(6) = 2$ copies of $\mathbb{F}_p^*$. This means that one can achieve the security of $\mathbb{F}_{p^6}$, while transmitting two elements of $\mathbb{F}_p$. This compression is possible because there exist birational maps by which the so-called compression from $\mathbb{F}_{p^6}$ to $\mathbb{A}^2(\mathbb{F}_p)$ is performed. In this way, the group operation in CEILIDH is actually performed in $\mathbb{F}_{p^6}$, which can be done efficiently in various representations as shown in [97]. First we consider several representations of the field $\mathbb{F}_{p^6}$ and mappings between different representations.

The first representation denoted as F_1 is an extension field of $\mathbb{F}_p$ of degree 6, *i.e.* $F_1 = \mathbb{F}_{p^6} = \mathbb{F}_p[x]/(f(x))$, where $deg(f) = 6$. The second representation F_2 is a quadratic extension of $\mathbb{F}_{p^3}$, so $F_2 = \mathbb{F}_{p^3}[y]/g(y)$ and $deg(g) = 2$. The authors of [97] referred to F_3 as a semi-compressed fractional form of T_6, which is actually T_2. This construction is as follows [97]:

$$F_1 \xrightarrow{\sigma} F_2 \xrightarrow{\tau} F_3 \xrightarrow{\rho} \mathbb{A}_2(\mathbb{F}_p) \tag{8.1}$$

As these mappings are isomorphisms, there exist inverse mappings. More precisely, it holds:

$$F_1 \xleftarrow{\sigma^{-1}} F_2 \xrightarrow{\tau^{-1}} F_3 \xleftarrow{\psi} \mathbb{A}_2(\mathbb{F}_p) \tag{8.2}$$

So, it is possible to choose a representation in which field arithmetic can be performed more efficiently. This choice is related to the implementation platform.

8.2.1 The Representation F_1

As mentioned above, F_1 is an extension field of degree 6, so $F_1 = \mathbb{F}_{p^6} = \mathbb{F}_p[x]/(f(x))$. Let $p \equiv 2 \bmod 9$ (or $p \equiv 5 \bmod 9$) and $z = \zeta_9$; the cyclotomic polynomial ϕ_9 is the minimal polynomial of ζ_9 of degree $\varphi(9) = 6$. So, $f(x) = \frac{x^9-1}{x^3-1} = x^6 + x^3 + 1$ is an irreducible polynomial with a root z. Then, $z^6 = -z^3 - 1$ and each element from F_1 can be represented in the basis $\{1, z, z^2, z^3, z^4, z^5\}$. Hence, an arbitrary element from this group is denoted as $A(z) = \sum_{i=0}^{i=5} a_i z^i$. In order to perform exponentiation in this group we will first describe the basic operations in this field *i.e.* addition and multiplication. We denote multiplication/squarings and additions/subtractions in $\mathbb{F}_p$ with M and A respectively.

Addition in $\mathbb{F}_{p^6}$

Take two elements from F_1, $A(z) = \sum_{i=0}^{i=5} a_i z^i$ and $B(z) = \sum_{i=0}^{i=5} b_i z^i$. Then the sum is defined as:

$$C(z) = \sum_{i=0}^{i=5} c_i z^i = \sum_{i=0}^{i=5} (a_i + b_i) z^i \tag{8.3}$$

and it requires 6 additions in $\mathbb{F}_p$. In the following section we consider some hardware realizations of this operation.

Multiplication in $\mathbb{F}_{p^6}$

Now we look into the multiplication in this group. The product of two elements is based on the following equation:

$$\begin{aligned} C(z) = & \quad \sum_{i=0}^{i=5} c_i z^i \equiv A(z) \cdot B(z) \bmod f(z) \\ \equiv & \quad b_0 \cdot A + b_1 (Az \bmod f(z)) + \ldots + b_5 (Az^5 \bmod f(z)) \end{aligned}$$

One way to compute this product is by use of Algorithm 8.1 by Bertoni *et al.* [26].

Algorithm 8.1: Modular multiplication in $\mathbb{F}_{p^m}$

Require: $A(z) = \sum_{i=0}^{i=m-1} a_i z^i$ and $B(z) = \sum_{i=0}^{i=m-1} b_i z^i$ where $a_i, b_i \in \mathbb{F}_p$
$f(x) = x^m + \sum_{i=0}^{i=m-1} f_i x^i$ is an irreducible polynomial
Ensure: $C \leftarrow 0$

1: **for** i from 0 to $m-1$ **do**
2: $C \leftarrow b_i A + C$
3: $A \leftarrow Az \bmod f$
4: **end for**
5: Return(C)

In our case $m = 6$ and we use an irreducible trinomial which further simplifies this calculation. We examine the reduction step in more detail. In Step 3 of Algorithm 8.1 the following has to be computed:

$$A(z)z \bmod f \equiv -a_5 - a_5 z^3 + \sum_{i=0}^{i=4} a_i z^{i+1} \equiv a_4 z^5 + a_3 z^4 + (a_2 - a_5) z^3 + a_1 z^2 + a_0 z - a_5$$

To summarize, for Algorithm 8.1 the operation count per loop is:

- 1 multiplication of an element from $\mathbb{F}_p$ by an element $\mathbb{F}_{p^6}$ ($6M$ in $\mathbb{F}_p$)
- 1 addition in $\mathbb{F}_{p^6}$ ($6A$ in $\mathbb{F}_p$)
- 1 multiplication with z (a cyclic shift)
- 2 additions in $\mathbb{F}_p$ *i.e.* $2A$ (reduction step)

This means that the total operation count for $m = 6$ would include $36M$ plus lots of additions in the general case. However, multiplication of two polynomials of degree five can be performed in $18M$ plus many additions [32]. We explain that in more detail as it was also elaborated by Stam and Lenstra in [216]. Let $A(z) = \sum_{i=0}^{i=5} a_i z^i$ and $B(z) = \sum_{i=0}^{i=5} b_i z^i$ be two fifth degree polynomials to be multiplied. Write $A = A_0 + A_1 z^3$ and $B = B_0 + B_1 z^3$ where A_i, B_i for $i = 0, 1$ are second degree polynomials. Then

$$A \cdot B = A_0 B_0 + (A_0 B_1 + A_1 B_0) z^3 + A_1 B_1 z^6 \,,$$

where one can precompute the values $C_0 = A_0 B_0$, $C_1 = A_1 B_1$ and $C_2 = (A_0 - A_1)(B_0 - B_1)$. This results in

$$A \cdot B = C_0 + (C_0 + C_1 - C_2) z^3 + C_1 z^6 \,.$$

Let $A_0 = a_0 + a_1 x + a_2 x^2$ and $B_0 = b_0 + b_1 x + b_2 x^2$, where $a_i, b_i \in \mathbb{F}_p$ for $i = 0, 1, 2$; then for C_0 we get

$$C_0 = a_0 b_0 + (a_0 b_1 + a_1 b_0) x + (a_2 b_0 + a_1 b_1 + a_0 b_2) x^2 + (a_2 b_1 + a_1 b_2) x^3 + a_2 b_2 x^4 \,.$$

If the following values $c_0 = a_0 b_0$, $c_1 = a_1 b_1$, $c_2 = a_2 b_2$, $c_3 = (a_0 - a_1)(b_0 - b_1)$, $c_4 = (a_0 - a_2)(b_0 - b_2)$ and $c_5 = (a_1 - a_2)(b_1 - b_2)$ are precalculated we finally get:

$$C_0 = c_0 + (c_0 + c_1 - c_3) x + (c_0 + c_1 + c_2 - c_4) x^2 + (c_1 + c_2 - c_5) x^3 + c_2 x^4 \,.$$

It follows from above that each C_i requires $6M + 11A$ so the total number of multiplications adds to $18M$. The result $A \cdot B$ still has to be reduced modulo an irreducible polynomial in $\mathbb{F}_{p^6}$, which adds a few more additions to the total number of multiplications. According to [97] this all adds to $18M + 53A$ as the cost for one multiplication in F_1 in basis $\{z, z^2, z^3, z^4, z^5, z^6\}$.

Inversion in F_1

Inversion can be performed by means of Fermat's theorem (cf. Chapter 2), but some improvements are also possible. Namely, as the order of $T_6(\mathbb{F}_p)$ is equal to $\phi_6(p) = (p^2 - p + 1)|(p^3 + 1)$, it follows that for $g \in G_{p,6}$ we have $g^{p^3+1} = 1$. Then the inverse of g can be calculated as $g^{-1} = g^{p^3}$, which means that we have to evaluate the cost of the Frobenius map.

The Frobenius map $\psi : \mathbb{F}_{p^m} \to \mathbb{F}_{p^m}$, which is defined by $\psi(a) = a^p$; is sometimes used to speed-up the inversion and the exponentiation operation. We consider the cost of the Frobenius in the F_1 case. Let p here be a prime such that $p \equiv 2 \bmod 9$. Then p generates $\mathbb{Z}_9^*$ and as mentioned above $\Phi_9(x) = x^6 + x^3 + 1$ is an irreducible polynomial in $\mathbb{F}_p$ for which we denote a root with z. The basis for $\mathbb{F}_{p^6}$ is $\{1, z, z^2, z^3, z^4, z^5\}$. Then $a \in \mathbb{F}_{p^6}$ can be represented as $a = \sum_{i=0}^{i=5} a_i z^i$. We have to calculate the p-th power of a. So, we get

$$\begin{aligned} a^p = & \ (\textstyle\sum_{i=0}^{i=5} a_i z^i)^p = a_0 + a_1 z^2 + a_2 z^4 + a_3 z^6 + a_4 z^8 + a_5 z^{10} \\ = & \ a_0 + a_1 z^2 + a_2 z^4 + a_3(-z^3 - 1) + a_4 z^2(-z^3 - 1) + a_5 z \\ = & \ (a_0 - a_3) + a_5 z + (a_1 - a_4) z^2 - a_3 z^3 + a_2 z^4 - a_4 z^5 \,, \end{aligned}$$

by using the following facts: $\Phi_9(z) = 0$ and $z^n = z^{n \bmod 9}$. This means that the total cost of the Frobenius in F_1 is $4A$ in this case. If one uses $\{z, z^2, z^3, z^4, z^5, z^6\}$ as a basis, then it follows [216]

$$\begin{aligned} a^p = & \ (\textstyle\sum_{i=0}^{i=5} a_i z^{i+1})^p = a_0 z + a_1 z^2 + a_2 z^3 + a_3 z^4 + a_4 z^5 + a_5 z^6 \\ = & \ a_4 z + (a_0 - a_3) z^2 + a_5 z^3 + a_1 z^4 - a_3 z^5 + a_2 z^6 \,, \end{aligned}$$

which has the cost of only $2A$.

8.2.2 The Representation F_2

This representation considers $\mathbb{F}_{p^6}$ as a quadratic extension of the field $\mathbb{F}_{p^3}$ and it is important for arithmetic in the torus. Here we have, $F_2 = \mathbb{F}_{p^6} = \mathbb{F}_{p^3}[y]/(g(y))$, $(deg(g) = 3)$ where $x = \zeta_3$ and $y = \zeta_9 + {\zeta_9}^{-1}$. The bases used are $\{1, y, y^2 - 2\}$ and $\{1, x\}$. Let us consider the basic operations in this case.

Addition and Multiplication in F_2

Addition of two elements from F_2 is similar as in the previous case *i.e.* 1 addition in $\mathbb{F}_{p^3}$ can be computed as 3 additions in $\mathbb{F}_p$. Therefore, for addition of two elements in F_2, a total of $6A$ is required.
Let $c \in F_2$ and $c = (a_0 + a_1 y + a_2(y^2 - 2), a_3 + a_4 y + a_5(y^2 - 2)) = c_0 + c_1 x$, where $c_i \in \mathbb{F}_{p^3}$ for $i = 1, 2$. We want to multiply two elements in F_2, say $c = c_0 + c_1 x$ and $d = d_0 + d_1 x$. Then we have:

$$c \cdot d = (c_0 + c_1 x)(d_0 + d_1 x) = (c_0 d_0 - c_1 d_1) + (c_0 d_1 + c_1 d_0 - c_1 d_1) x \,.$$

If the values $t_{00} = c_0 d_0$, $t_{11} = c_1 d_1$ and $t_{01} = (c_0 + c_1)(d_0 + d_1)$ are precomputed, the previous product equals:

$$c \cdot d = (t_{00} - t_{11}) + (t_{01} - t_{00} - 2t_{11})x\,.$$

It follows that multiplication is calculated by 3 multiplications and 5 additions in $\mathbb{F}_{p^3}$. We will consider this multiplication in more detail.
Multiplication in $\mathbb{F}_{p^3}$: Let $a = a_0 + a_1 y + a_2(y^2 - 2)$ and $b = b_0 + b_1 y + b_2(y^2 - 2)$. Here $\mathbb{F}_{p^3} = \mathbb{F}_p/(g(y))$ and $g(y) = y^3 - 3y + 1$. Then we get:

$$\begin{aligned} a \cdot b \quad &= (a_0 b_0 + 2a_1 b_1 - a_1 b_2 - a_2 b_1 + 2a_2 b_2) + \\ &\quad (a_0 b_1 + a_1 b_0 + a_1 b_2 + a_2 b_1 - a_2 b_2) y + \\ &\quad (a_0 b_2 + a_2 b_0 + a_1 b_1 - a_2 b_2)(y^2 - 2)\,. \end{aligned}$$

If some variables are precomputed: $t_{00} = a_0 b_0$, $t_{11} = a_1 b_1$, $t_{22} = a_2 b_2$, $t_{01} = (a_0 + a_1)(b_0 + b_1)$, $t_{12} = (a_1 - a_2)(b_2 - b_1)$ and $t_{20} = (a_2 - a_0)(b_0 - b_2)$ then the previous multiplication becomes:

$$a \cdot b = (t_{00} + t_{11} + t_{22} - t_{12}) + (t_{01} + t_{12} - t_{00})y + (t_{20} + t_{00} + t_{11})(y^2 - 2).$$

From the previous equation it follows that one multiplication in $\mathbb{F}_{p^3}$ can be calculated by $6M + 12A$. This results in a total of $18M + 51A$ for one multiplication in F_2.

Frobenius in F_2

For an element $c \in F_2$, where $c = c_0 + c_1 x$, the Frobenius map gives: $c^p = (c_0 + c_1 x)^p = ({c_0}^p - {c_1}^p) - {c_1}^p x$. So, the calculation of this map takes 2 maps of Frobenius of an element from $\mathbb{F}_{p^3}$ and two additions in $\mathbb{F}_{p^3}$ $(6A)$. Next we consider the Frobenius map in the subfield. As it holds that $p \equiv 2 \bmod 9$ and ${\zeta_9}^9 = 1$ for our basis we get:

$$y^p = {\zeta_9}^p + {\zeta_9}^{-p} = {\zeta_9}^2 + {\zeta_9}^{-2} = ({\zeta_9} + {\zeta_9}^{-1})^2 - 2 = y^2 - 2.$$

On the other hand, it holds:

$$(y^2 - 2)^p = (y^2)^p - 2 = (y^p)^2 - 2 = (y^2 - 2)^2 - 2 = -y^2 - y + 2.$$

Hence, for an element $a \in \mathbb{F}_{p^3}$ the Frobenius map is:

$$\begin{aligned} a^p &= (a_0 + a_1 y + a_2(y^2 - 2))^p \\ &= {a_0}^p + {a_1}^p y^p + {a_2}^p (y^2 - 2)^p \\ &= a_0 + a_1(y^2 - 2) + a_2(-y - (y^2 - 2)) \\ &= a_0 - a_2 y + (a_1 - a_2)(y^2 - 2). \end{aligned}$$

The operation count is just two additions in $\mathbb{F}_p$ $(2A)$. Summing up all that gives the total of $10A$ to compute the Frobenius map in the representation F_2.

Inversion in F_2

Consider the element $c \in F_2$, where $c = c_0 + c_1 x$, $c_i \in \mathbb{F}_{p^3}$. We want to find an element $d = d_0 + d_1 x \in F_2$, such that $d = c^{-1}$ and $d_i \in \mathbb{F}_{p^3}$. By solving the systems of equations we get:

$$d = (c_0 + c_1 x)^{-1} = \frac{1}{{c_0}^2 - c_0 c_1 + {c_1}^2} \begin{pmatrix} -c_1 + c_0 \\ -c_1 \end{pmatrix},$$

which results in one inversion, five multiplications and four additions in $\mathbb{F}_{p^3}$. We consider now the inversion in $\mathbb{F}_{p^3}$.
Let $a = a_0 + a_1 y + a_2(y^2 - 2) \in \mathbb{F}_{p^3}$. We need to find the inverse of a *i.e.* a^{-1} such that $a \cdot a^{-1} = 1$ and a^{-1} is of the same form. For this purpose we compute the norm function of a:

$$N(a) = a \cdot a^p \cdot a^{p^2} = \Delta\,,$$

and $\Delta = N(a) \in \mathbb{F}_p$. One can also write:

$$a \cdot \frac{a^p \cdot a^{p^2}}{\Delta} = 1\,.$$

From the previous equation it follows: $a^{-1} = \frac{a^p \cdot a^{p^2}}{\Delta}$.
Then, the total cost of inversion in $\mathbb{F}_{p^3}$ is equal to the cost of two computations of the Frobenius map, one multiplication in $\mathbb{F}_{p^3}$, one inversion in $\mathbb{F}_p$ and three multiplications in $\mathbb{F}_p$.

8.2.3 The Representation F_3

The representation F_3 specifies elements of $T_6(\mathbb{F}_p)$ and we have the following definition:

Definition 35 [97] *F_3 is the set of elements*

$$\left\{ \frac{a_0 + a_1 x}{a_0 + a_1 x^2}, a_i \in \mathbb{F}_{p^3} \right\}.$$

If $a_1 = 1$ we say that the element is reduced.

We need to specify functions that map F_2 to F_3 and vice versa. According to Lemma 2 in [97], for an element $a = a_0 + a_1 x \in Ker[N_{\mathbb{F}_{p^6}/\mathbb{F}_{p^3}}]$ it holds

$$\tau(a) = \frac{b + x}{b + x^2},$$

where $b = (1 + a_0)/a_1$ if $a_1 \neq 0$ and $b = a_0/(a_0 - 1)$ otherwise and

$$\tau^{-1}(b) = \frac{b^2 - 1}{b^2 - b + 1} + \frac{2b - 1}{b^2 - b + 1} x.$$

We add here also the descriptions of the inverse birational maps ψ and ρ as they are required to implement public-key protocols.

Definition 36 *Let $V(f)$ be the zero set of $f(\alpha_1, \alpha_2) = 1 - {\alpha_1}^2 - {\alpha_2}^2 + \alpha_1\alpha_2$ in $\mathbb{A}^2(\mathbb{F}_p)$; then the following inverse birational maps are defined $\psi : \mathbb{A}^2(\mathbb{F}_p) \setminus V(f) \xrightarrow{\sim} T_6(\mathbb{F}_p) \setminus \{1, x^2\}$ and $\rho : T_6(\mathbb{F}_p) \setminus \{1, x^2\} \xrightarrow{\sim} \mathbb{A}^2(\mathbb{F}_p) \setminus V(f)$:*

$$\psi(\alpha_1, \alpha_2) = \frac{1 + \alpha_1 y + \alpha_2(y^2 - 2) + (1 - {\alpha_1}^2 - {\alpha_2}^2 + \alpha_1\alpha_2)x}{1 + \alpha_1 y + \alpha_2(y^2 - 2) + (1 - {\alpha_1}^2 - {\alpha_2}^2 + \alpha_1\alpha_2)x^2}$$

and

$$\rho(\beta) = (\frac{u_2}{u_1}, \frac{u_3}{u_1}),$$

where for $\beta = \beta_1 + \beta_2 x$, $\beta_1, \beta_2 \in \mathbb{F}_{p^3}$, let $(1 + \beta_1)/\beta_2 = u_1 + u_2 y + u_3(y^2 - 2)$.

We do not discuss here multiplication, inversion and the Frobenius operations in F_3 as they are performed in the same way as in F_2.

8.3 Hardware Implementations of $T_6(\mathbb{F}_p)$

Here we discuss possible hardware implementations related to the representations above. Basically we need to consider whether to decompress to F_2 or to F_1. In order to do that we evaluate the area complexity and the latency for both cases.

8.3.1 Implementing the Arithmetic in F_1

The area and time complexities for the general case as in Algorithm 8.1 are given in Table 8.1. The first row gives the numbers for the fastest performance of this multiplier and the second row gives a more compact solution. This solution is more flexible than the one that uses precomputation, but the latency is longer. The basic blocks are the same in all cases and those consist of $\mathbb{F}_p$ multipliers and $\mathbb{F}_p$ adders.

Table 8.1: Options for area complexity and critical path delay of the $\mathbb{F}_{p^6}$ multiplier.

Area compl.	Crit. Path Delay
6 $\mathbb{F}_p$ multipliers + 8 $\mathbb{F}_p$ adders	$1M + 1A$
1 $\mathbb{F}_p$ multipliers + 1 $\mathbb{F}_p$ adders	$6M + 8A$

8.3.2 Implementing the Arithmetic in F_2

In order to implement the arithmetic in F_2 the same basic blocks are required *i.e.* the blocks for multiplication and addition in $\mathbb{F}_p$. From the area point of view the necessary hardware resources are similar for both, F_1 and F_2. Therefore, we conclude that F_1 does not offer any substantial advantage when compared with F_2. Even more, the F_2 representation is more convenient as in that case the maps σ and σ^{-1} are not required. We need to implement all components for F_2 and the corresponding maps for a hardware implementation of $T_6(\mathbb{F}_p)$. More details are provided in the next section.

8.3.3 Hardware Architecture for $T_6(\mathbb{F}_p)$

We discuss now basic blocks for our hardware implementation. From the previous discussion it is clear that we need the following arithmetic operations in $\mathbb{F}_p$: addition, subtraction, multiplication and inversion. Exponentiation is performed as repeated multiplications.

8.3.4 Addition/Subtraction in $\mathbb{F}_p$

Modular addition and subtraction are calculated as in Algorithm 8.2 and Algorithm 8.3 respectively [48].

Algorithm 8.2: Modular addition

Require: p, $A, B \in \mathbb{F}_p$
Ensure: $C = A + B \bmod p$
1: $S' \leftarrow A + B$
2: $S'' \leftarrow S' - p$
3: If $S'' < 0$, then $C \leftarrow S'$
4: else $C \leftarrow S''$
5: Return(C)

Algorithm 8.3: Modular subtraction

Require: p, $A, B \in \mathbb{F}_p$
Ensure: $C = A - B \bmod p$
1: $S' \leftarrow A - B$
2: $S'' \leftarrow S' + p$
3: If $S' < 0$, then $C \leftarrow S''$
4: else $C \leftarrow S'$
5: Return(C)

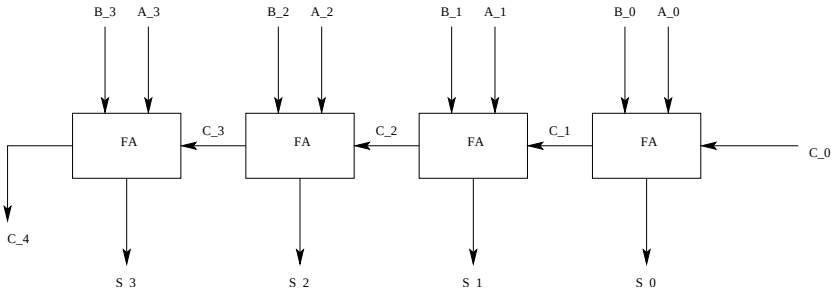


Figure 8.1: Schematic of the ripple carry adder as in [151].

The numbers are represented in two's complement representation and in this way both the addition and subtraction operation can be combined into one circuit as explained by Mano and Kime [151]. We use a serial adder/subtractor that consists of: one full adder, two shift-registers (one for A and C and the other for B), one flip-flop (for a carry bit), a counter and a controller.

We chose to perform a digit-serial addition that calculates a 32-bit addition at the same time by means of Ripple Carry Adder (RCA) [151]. This type of adder for a 4-bit case is depicted in Fig. 8.1 [151]. It consists of four full adders and four additions occur simultaneously to compute the sum. In our case, for parameters that are 160 bits long, 1 modular addition/subtraction takes 10 clock cycles. As an addition in the field $\mathbb{F}_{p^6}$ consists of 6 parallel additions in $\mathbb{F}_p$, the critical path delay of an $\mathbb{F}_{p^6}$ addition is equal to the delay of one $\mathbb{F}_p$ addition. Schematic of this adder is given in Figure 8.2.

8.3.5 Modular Multiplication and Inversion in $\mathbb{F}_p$

As shown in the previous section, modular multiplication in the composite field $\mathbb{F}_{p^n}$ is performed by means of multiplication in $\mathbb{F}_p$, where we used the systolic array for multiplication that is based on Montgomery's algorithms as described in Chapter 3. The multiplier is also used for our ECC and RSA implementations.

The usual way to implement modular inversion is by means of Fermat's theorem (cf. Chapter 2), so by repeated multiplications. However, in this specific composite field the modular inversion operation can be performed more efficiently by use of the Frobenius map. Details were given in Sect. 8.2.

8.3.6 The $T_6(\mathbb{F}_p)$ Hardware Architecture

The complete architecture for hardware implementations of $T_6(\mathbb{F}_p)$ is shown in Fig. 8.3. The operation blocks on each level from top to bottom are as follows:

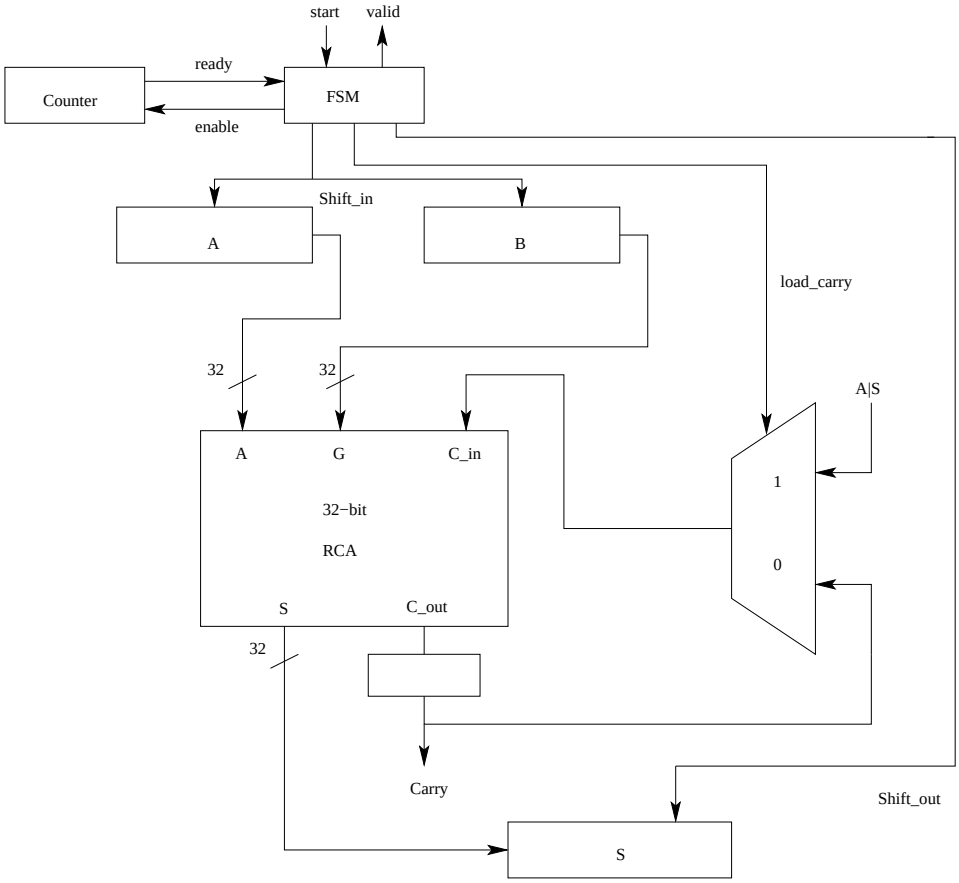


Figure 8.2: Schematic of the adder for addition in $\mathbb{F}_p$.

- Level 1: Main Controller
- Level 2:
  1. The map τ that maps an element from F_2 to $F_3 = T_6$ representation
  2. The map ρ that maps T_6 to $\mathbb{A}_2(\mathbb{F}_p)$
  3. The map τ^{-1} for the decompression from T_6 to F_2
  4. $\mathbb{F}_{p^6}$ inversion
  5. $\mathbb{F}_{p^6}$ multiplication
  6. $\mathbb{F}_{p^6}$ addition
  7. $\mathbb{F}_{p^6}$ Frobenius
  8. The ψ map that maps $\mathbb{A}_2(\mathbb{F}_p)$ to T_6 *i.e.* the inverse of ρ
- Level 3: $\mathbb{F}_{p^3}$ arithmetic
- Level 4: $\mathbb{F}_p$ arithmetic

The main control Finite State Machine (FSM) first commands the block ψ to map the input $g \in \mathbb{A}^2(\mathbb{F}_p)$ to F_3 and then the block τ^{-1} to F_2. The ψ block is using only the lowest level blocks for modular multiplication and addition in $\mathbb{F}_p$. On the other hand, the τ^{-1} block is using only the blocks for operations in $\mathbb{F}_{p^3}$: multiplication, addition and inversion *i.e.* it instantiates $\mathbb{F}_{p^3}$ *inv.*, $\mathbb{F}_{p^3}$ *mul.* and $\mathbb{F}_{p^3}$ *add.* Those three blocks are all using $\mathbb{F}_p$ *add* and $\mathbb{F}_{p^3}$ *mul.* needs also $\mathbb{F}_p$ *mul.* The exponentiation in F_2 is performed by use of $\mathbb{F}_{p^6}$ *mul.*, which invokes $\mathbb{F}_{p^3}$ *mul.* and $\mathbb{F}_{p^3}$ *add* to perform the modular multiplication in $\mathbb{F}_{p^6}$. When the exponentiation operations are finished the Main Controller orchestrates τ and ρ to start the compression back to the element in $\mathbb{A}^2(\mathbb{F}_p)$. It writes the resulting g^a to its output registers.

8.4 Torus-based Cryptosystems

To make a suitable comparison of implementations of a torus-based cryptosystem with the one based on ECDLP we should consider the same protocol for both types of PKC. All standard public-key services such as encryption, signatures, key-exchange *etc.* were developed for CEILIDH as well as T_{30} or any other torus T_n. Even voting schemes and mix-nets were discussed in the context of tori [234]. As an example we give here the key agreement scheme that is based on torus T_n as proposed by Rubin and Silverberg in [196].

Parameter selection: Choose a prime power q and an integer n such that for the torus T_n over $\mathbb{F}_q$ holds $nlog(q) \approx 1024$. This condition prevents the index-calculus attack as it guarantees 1024-bit security as required for nowadays applications. Further, $\phi_n(q)$ should be divisible by a prime l that has at least 160 bits (to

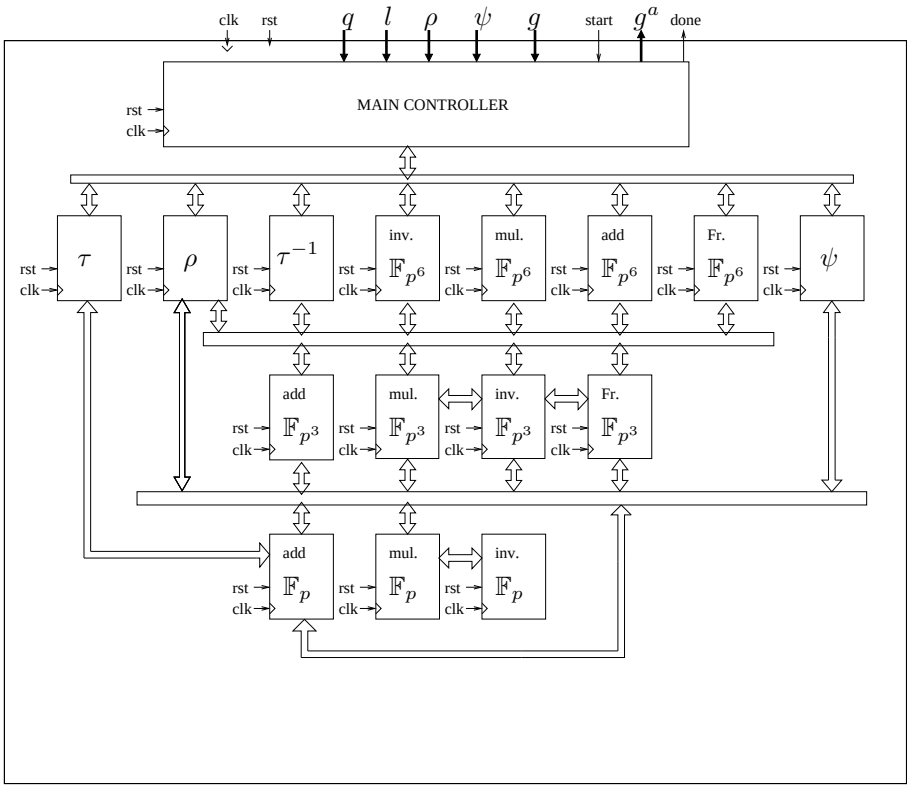


Figure 8.3: Architecture of the $T_6(\mathbb{F}_p)$ processor.

avoid the square root attacks). Let $m = \varphi(n)$ and let the birational maps be $\rho : T_n(\mathbb{F}_q) \to \mathbb{F}_q^m$ and its inverse ψ. Choose $\alpha \in T_n$ of order l and let $g = \rho(\alpha) \in \mathbb{F}_q^m$. For the key-agreement protocol that is given below the public data are n, q, ρ, ψ, l and g or $\alpha = \psi(g)$.

Key agreement scheme:

1. Alice chooses a random integer $a \in [1, l-1]$. She computes $P_A := \rho(\alpha^a) \in \mathbb{F}_q^m$ and sends it to Bob.
2. Bob chooses a random integer $b \in [1, l-1]$. He computes $P_B := \rho(\alpha^b) \in \mathbb{F}_q^m$ and sends it to Alice.
3. Alice computes $\rho(\psi(P_B)^a) \in \mathbb{F}_q^m$.
4. Bob computes $\rho(\psi(P_A)^b) \in \mathbb{F}_q^m$.

As ρ and ψ are the inverse of each other, it holds that $\rho(\psi(P_B)^a) = \rho(\alpha^{ab}) = \rho(\psi(P_A)^b)$ and this is the shared key of Alice and Bob.

8.5 Estimated Results

We need to estimate the performances of the exponentiation operation in different field representations. Also both exponentiations, *i.e.* with a fixed and a random base have to be evaluated as each of those might be required, depending on the specific protocol that has to be implemented.

It is evident from the torus-based Diffie-Hellman protocol for key exchange that each user has to perform one exponentiation with the fixed base α and one exponentiation with a random base. This is similar to key-agreement schemes of other PK cryptosystems *e.g.* XTR, ECC *etc.* Therefore we can estimate the latency in any of those two cases.

For the random base exponentiation in F_2 the required number of multiplications in $\mathbb{F}_p$ is 4320 when the square-and-multiply method for exponentiation is used. On the other hand, for ECC over $\mathbb{F}_p$ the corresponding situation assumes the same prime field, Jacobian projective coordinates and the double-and-add method for point multiplication. In this case the total number of multiplications required is 2560. We do not count here other operations for the torus such as the mappings required to change from one representation to another nor the costs of additions and inversions. The reason is that the mappings are only required at most one time per protocol and the most expensive one in that case is the ρ map that takes $1I + 10M + 17A$. The remaining maps require only a few multiplications and one (τ and τ^{-1}) or no inversion at all (ψ). The inversion operation itself is not performed more than a few times in total and therefore is not taken into account, which can be compared to the case of projective coordinates for ECC. We do not calculate the additions in $\mathbb{F}_p$ for this first estimate as the addition operation does not take much time when compared with the $\mathbb{F}_p$ multiplication. For the details see Table 8.2.

Table 8.2: Performance comparison for ECC, RSA and Torus on our FPGA platform running at 100 MHz.

	RSA	ECC	T_6
bit-lengths	1024	160	170
# FPGA slices	9606	1525	1619
latency for add. $[ns]$	n.a.	100	120
latency for mult. $[\mu s]$	30.77	4.85	5.15
latency for exp. $[ms]$	47	12	22

The chosen FPGA platform for our implementations is Spartan 3 XC355000 and the results are obtained after synthesis by use of Xilinx tool. The preliminary results are given in Table 8.2. The size of the hardware implementations seems to be comparable for both, ECC and T_6 and the data-path is also the same. The reason is that both cyptosystems use the same low-level blocks and although T_6 needs more higher level blocks it is not a big difference as for ECC there are also separate blocks required for the point operations, conversions from affine to projective coordinates and back *etc.* as observed in Chapters 4 and 5. Furthermore, the bandwidth is smaller for torus based cryptosystem as the required parameters that have to be communicated are consisting of only two $\mathbb{F}_p$ elements. For ECC, in projective coordinates each point consists of three $\mathbb{F}_p$ elements unless the points are compressed. However, point compressions and decompressions add to the hardware complexity and to the estimated latency.

8.6 Conclusions and Future Work

We have investigated all operations required for hardware implementations of the algebraic torus T_6. Furthermore, we have proposed the architecture to be implemented on an FPGA platform. Our preliminary results show that torus-based cryptography is a feasible solution for PK protocols. It appears to be not much slower than ECC but faster than RSA on an FPGA platform. Savings in bandwidth are the most attractive feature for torus-based cryptography as one using the field $\mathbb{F}_p$, where p is 170 bits long, could achieve the equivalent of 1024 RSA security. We have also noticed that hardware resources could be efficiently shared with ECC and our next step is to develop one solution for both types of cryptography.

We mention here that although we used the ripple carry adder this solution should not be considered as the most suitable. It is probably a good choice for compactness, but there are other solutions for adder that one should also take into account. For example, so-called logarithmic time adders such as carry look-ahead adder and carry-select adder are preferred for performance [181]. Having in mind a good solution as a compromise between area and power one should investigate other adders as well. However, our intention in this work was merely to compare various cryptosystems that are using the same low-level hardware blocks and for that purpose the choice of adder was not a crucial one.

Chapter 9

Conclusions and Future Research

9.1 Conclusions

In this thesis we deal with algorithms and arithmetic for several PK cryptosystems *i.e.* RSA, ECC, HECC and torus-based PKC. The emphasis was on efficient hardware implementations with awareness of side-channel cryptanalysis and on applications with resource constrained devices.

One example is curve-based cryptography, more precisely ECC and HECC. There exist several ways to accelerate the curve-based arithmetic. Following a bottom-up approach in the hierarchical structure of ECC/HECC (cf. Chapter 2), these are as follows: speeding-up the finite-field arithmetic (especially multiplication and inversion), choosing "good" coordinates for an efficient group operation and accelerating the scalar multiplication operation. Most of these aspects are considered in literature, but mainly targeted towards efficient software implementations. Nevertheless, a hardware co-processor is a necessity in most applications to speed-up the performance and/or to lower the energy requirements. Therefore, careful hardware design consideration implies also new criteria on the underlying arithmetic for a certain PK protocol.

First, we focused on implementations that should provide a simple side-channel protection. Starting from the state of the art of PKC algorithms and going one level down to architectures (cf. the pyramid in Chapter 1), we investigated which modifications could lead to more efficient and secure solutions. Two best-known and most widely used public-key cryptosystems are RSA and Elliptic Curve Cryptosystems (ECC). We have investigated an option for a unique hardware platform that could be used for both types of cryptosystems. Consequently, we have optimized the bound for the Montgomery parameter for both RSA and ECC (over

a prime field) in order to improve the performance. Nevertheless, we have also obtained a more secure solution w.r.t. side-channel security. A fair comparison in the performances of RSA and ECC on the same hardware platform is given. Our new option for parallelism among sequences of operations for the ECC point addition is compared with the so-called fast RSA variants *i.e.* CRT, MultiPrime and MultiPower.

Apart from ECC over a large prime field, we have looked into ECC over a binary field. Our contribution in that part of the thesis is in balancing algorithms for the group operation. More precisely, we have made the point addition and doubling balanced, *i.e.* they were tweaked from the original ones (that are due to Lopez and Dahab) to obtain identical sequences of operations. Our efforts in the line of "side-channel atomicity" were confirmed by power consumption graphs showing improved simple side-channel resistance.

Another goal of this research was to provide suitable embedded arithmetic to support the security in resource constrained devices. Embedded devices have limited energy supply, are battery operated, run on small embedded processors and have a limited amount of storage. What is needed for embedded systems is a systematic computational security-energy-memory trade-off. The most challenging tasks are implementations of Public Key Cryptography. Hardware/software co-design is the new implementation alternative and the most promising platform for the computationally intensive operations such as those required for PK operations. The new emerging applications such as sensor networks have limited memory resources at their disposal and firm constraints on low-power. All these requirements should facilitate also an acceptable speed. We have described such a solution for Curve-based Cryptography (ECC/HECC). Our results prove HECC to be a more suitable choice than ECC on similar low-power and low-footprint architectures.

Radio frequency identification (RFID) is an example of an emerging technology which requires authentication as a cryptographic service. This property can be achieved by symmetric as well as asymmetric primitives. Whether an authentication protocol based on asymmetric cryptography is feasible on a tag is still an open question. We concluded that it might be possible to have "fancy" PK protocols running on a tag, at least in the case of higher-cost tags. Besides the implementation issues, there are many other RFID technology-related questions to be answered. One of them is also counterfeiting. Our solution is to make an RFID-tag unclonable by linking it inseparably to a Physical Unclonable Function (PUF). We have presented the security protocols for both, of-line and on-line case that are needed for the detection of the authenticity of a product when it is equipped with such a system.

9.2 Future Work

PKC implementations and side-channel security are an interesting research area and there are many aspects that should be taken into account. Especially embedded cryptography as a multidisciplinary area, poses lots of challenges to PK applications nowadays.

Two emerging examples of PKC applications are RFIDs and sensor networks. We believe that the arithmetic for PK cryptosystems could be further optimized which would result in certain improvements and innovations in software and especially hardware implementations. Hardware implementations allow for an extra degree of freedom w.r.t. parallelism of various processes which can lead to a more compact and also more efficient solution. Promising candidates for emerging applications of PKC in embedded devices appear to be Elliptic and Hyperelliptic curve cryptosystems.

RFID-tags are becoming very popular tools for the identification of products in various applications like *e.g.* supply-chain management. A recently proposed application is the use of RFID-tags for anti-counterfeiting by embedding them into a product. However, there is a risk related to naively using those tags for several applications. In particular, if no appropriate cryptographic measures are taken, the privacy of a user carrying tagged items can be severely damaged. In order to enable these applications and at the same time minimize the risks, PKC offers attractive solutions. Although some preliminary work exists, whether a public-key cryptosystem can be implemented on an RFID tag remains an open problem due to ever evolving standards and new applications. It is required not merely to examine existing asymmetric protocols, but also to look for alternatives. Therefore, there will be a lot of work on PK crypto applications in these new drastic conditions. Also, low-cost implementations imply light-weight protocols. This means that previously known protocols should be revisited for this purpose. As usual, side-channel resistance is also a concern, but in this case at a very low cost.

The future work related to implementation attacks should also address fault attacks. These attacks and the countermeasures are still not very well understood. The faults could be applied on ROM or RAM to a chosen bit positions. Adding a CRC check to each of sensitive system parameters would be a suitable countermeasure but probably not feasible on a tag and in general very expensive. Therefore, it remains for future work to investigate some other cheap solution in order to circumvent these threats. In general, these issues have to be addressed and cheap solutions for thwarting all types of implementations attacks have to be found. We believe that we made the first step in this direction with our concept of cheap balancing the critical operations, but plenty of work still remains to be done.

The new torus-based cryptography is just one of many proposals for PK crypto. However, as preliminary results in software and hardware implementations are

promising, more work on $\mathbb{F}_{p^m}$ arithmetic is required. In particular, for hardware implementations there are very few papers that looked into this problem. This includes the case when $p = 3$, which is suitable for pairing-based cryptography as another important research area.

From the technology viewpoint, we need to take into account the challenges of deep submicron technologies. These include the effect of process variations and leakage currents on side-channel attacks. We need to investigate how these deep submicron effects affect side-channel attacks: do they provide more or less information? Suitable countermeasures also need to be developed at different abstraction levels.

For a longer-term research topic, we mention again that the leakage current will become more important due to process variations. Therefore, a new line of research dealing with power consumption models will be needed. With respect to side-channel security issues, memories are also very important and there already exists a line of research considering secure memories. As general recommendations sensitive parts of the design should not be scaled too much and area of impact should be minimized. Also, the use of special techniques as countermeasures should be targeted only to those sensitive parts.

Bibliography

[1] G.B. Agnew, R.C. Mullin, and S.A. Vanstone. A Fast Elliptic Curve Cryptosystem. In J.-J. Quisquater and J. Vandewalle, editors, *Advances in Cryptology: Proceedings of EUROCRYPT'89*, number 434 in Lecture Notes in Computer Science, pages 706–708. Springer-Verlag, 1989.

[2] G.B. Agnew, R.C. Mullin, and S.A. Vanstone. On the Development of a Fast Elliptic Curve Cryptosystem. In R. A. Rueppel, editor, *Advances in Cryptology: Proceedings of EUROCRYPT'92*, number 658 in Lecture Notes in Computer Science, pages 482–487. Springer-Verlag, 1992.

[3] M.-L. Akkar and C. Giraud. An Implementation of DES and AES, Secure against Some Attacks. In Ç.K. Koç, D. Naccache, and C. Paar, editors, *Proceedings of 3rd International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, number 2162 in Lecture Notes in Computer Science, pages 309–318. Springer-Verlag, 2001.

[4] R. Anderson and M. Kuhn. Low cost attacks on tamper resistant devices. In B. Christianson, B. Crispo, T. M. A. Lomas, and M. Roe, editors, *Proceedings of 5th International Workshop on Security Protocols*, number 1361 in Lecture Notes in Computer Science, pages 125–136. Springer-Verlag, 1997.

[5] ANSI. ANSI X9.62 The Elliptic Curve Digital Signature Algorithm (ECDSA). `http:www.ansi.org`.

[6] C. Aumüller, P. Bier, W. Fischer, P. Hofreiter, and J.-P. Seifert. Fault attacks on RSA with CRT: Concrete Results and Practical Countermeasures. In B.S. Kaliski Jr., Ç.K. Koç, and C. Paar, editors, *Proceedings of 4th International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, number 2523 in Lecture Notes in Computer Science, pages 260–275. Springer-Verlag, 2002.

[7] S. Baktır, J. Pelzl, T. Wollinger, B. Sunar, and C. Paar. Optimal Tower Fields for Hyperelliptic Curve Cryptosystems. In *Proceedings of 38th Asilo-*

mar Conference on Signals, Systems and Computers, Pacific Grove, USA, November 7-10 2004.

[8] W.D. Banks and I. Shparlinski. On the number of sparse rsa exponents. *J. Number Theory*, 95:340–350, 2002.

[9] L. Batina, G. Bruin-Muurling, and S.B. Örs. Flexible hardware design for RSA and elliptic curve cryptosystems. In T. Okamoto, editor, *In Topics in Cryptology - CT-RSA - The Cryptographers' Track at the RSA Conference*, volume 2964 of *Lecture Notes in Computer Science*, pages 250–263. Springer-Verlag, 2004.

[10] L. Batina, A. Hodjat, D. Hwang, K. Sakiyama, B. Preneel, and I. Verbauwhede. Hardware/Software Co-design for Curve-based Cryptography on the 8051 μP. *IEEE Transactions on Very Large Scale Integration: Special Section on Hardware/Software Codesign and and System Synthesis.* submitted.

[11] L. Batina, D. Hwang, A. Hodjat, B. Preneel, and I. Verbauwhede. Hardware/Software Co-design for Hyperelliptic Curve Cryptography (HECC) on the 8051 μP. In J. R. Rao and B. Sunar, editors, *Proceedings of 7th International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, number 3659 in Lecture Notes in Computer Science, page 106118. Springer-Verlag, 2005.

[12] L. Batina and C. Jansen. Secret Exponent Information Leakage for Timing Analyses. In B. Macq and J.-J. Quisquater, editors, *Proceedings of the 23rd Symposium on Information Theory in the Benelux*, pages 225–232, Louvain-la-Neuve, Belgium, May 29-31 2002. Werkgemeeschap voor Informatie-en-Communicatietheorie, Enschede, The Netherlands.

[13] L. Batina and C. Jansen. Side-Channel Entropy for Modular Exponentiation Algorithms. In L. Tolhuizen, editor, *Proceedings of the 24th Symposium on Information Theory in the Benelux*, pages 37–44, Veldhoven, The Netherlands, May 2003. Werkgemeeschap voor Informatie-en-Communicatietheorie, Enschede, The Netherlands.

[14] L. Batina, C. Jansen, G. Muurling, and S. Xu. Almost Montgomery based Multiplier in in GF(2^n). In B. Macq and J.-J. Quisquater, editors, *Proceedings of the 23rd Symposium on Information Theory in the Benelux*, pages 61–68, Louvain-la-Neuve, Belgium, May 29-31 2002. Werkgemeeschap voor Informatie-en-Communicatietheorie, Enschede, The Netherlands.

[15] L. Batina, N. Mentens, S.B. Örs, and B. Preneel. Serial multiplier architectures over GF(2^n) for Elliptic Curve Cryptosystems. In *Proceedings of The 12th IEEE MELECON 2004*, Dubrovnik, Croatia, May 12-15, 4 pages.

[16] L. Batina, N. Mentens, B. Preneel, and I. Verbauwhede. Balanced point operations for side-channel protection of Elliptic Curve Cryptography. *IEE Proceedings on Information Security: Special Issue on Cryptographic Algorithms and Architectures for System-on-Chip*, 2005. To appear.

[17] L. Batina, N. Mentens, B. Preneel, and I. Verbauwhede. Side-channel aware design: Algorithms and Architectures for Elliptic Curve Cryptography over GF(2^n). In *Proceedings of the IEEE International Conference on Application-Specific Systems, Architectures, and Processors (ASAP)*, pages 350–355, Samos, Greece, July 23-25 2005. IEEE Computer Society Press.

[18] L. Batina, N. Mentens, and I. Verbauwhede. Side-channel issues for designing secure hardware implementations. In *In 11th IEEE International On-Line Testing Symposium*, pages 118–121, San Raphael, France, July 2005.

[19] L. Batina and G. Muurling. Another Way of Doing RSA Cryptography in Hardware. In B. Honary, editor, *Proceedings of 8th IMA International Conference on Cryptography and Coding*, number 2260 in Lecture Notes in Computer Science, pages 364–373, Cirencester, UK, December 17-19 2001. Springer-Verlag.

[20] L. Batina and G. Muurling. Montgomery in Practice: How to Do It More Efficiently in Hardware. In B. Preneel, editor, *In Topics in Cryptology - CT-RSA - The Cryptographers' Track at the RSA Conference*, number 2271 in Lecture Notes in Computer Science, pages 40–52, San Jose, USA, February 18-22 2002. Springer-Verlag.

[21] L. Batina, S.B. Örs, B. Preneel, and J. Vandewalle. Hardware Architectures for Public Key Cryptography. *Elsevier Science Integration the VLSI Journal*, 34(1-2):1–64, 2003.

[22] M. Bednara, M. Daldrup, J. Teich, J. von zur Gathen, and J. Shokrollahi. Tradeoff Analysis of FPGA Based Elliptic Curve Cryptosystems. In *Proceedings of The IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 797–800, Scottsdale, Arizona, USA, May 26-29 2002.

[23] M. Bellare, C. Namprempre, and G. Neven. Security proofs for identity-based identification and signature schemes. In C. Cachin and J. Camenisch, editors, *Proceedings of Eurocrypt 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 268–286. Springer-Verlag, 2004.

[24] G. Bertoni, L. Breveglieri, T. Wollinger, and C. Paar. Finding Optimum Parallel Coprocessor Design for Genus 2 Hyperelliptic Curve Cryptosystems. In *Proceedings of ITCC, April 5-7, 2004*, Las Vegas, Nevada, USA, 2004.

[25] G. Bertoni, L. Breveglieri, T. Wollinger, and C. Paar. *Hyperelliptic Curve Cryptosystem: What is the Best Parallel Hardware Architecture?*, chapter in Embedded Cryptographic Hardware: Design and Security. Nova Science, 2004.

[26] G. Bertoni, J. Guajardo, S. Kumar, G. Orlando, C. Paar, and T. Wollinger. Efficient GF(p^m) Arithmetic Architectures for Cruptographic Applications. In M. Joye, editor, *In Topics in Cryptology - CT-RSA - The Cryptographers' Track at the RSA Conference*, number 2612 in Lecture Notes in Computer Science, San Francisco, April 2003. Springer-Verlag.

[27] T. Beth and D. Gollmann. Algorithm Engineering for Public Key Algorithm. *IEEE Journal on Selected Areas in Communications*, 7(4):458–465, May 1989.

[28] E. Biham and A. Shamir. Differential fault analysis of secret key cryptosystems. In B.S. Kaliski Jr., editor, *Advances in Cryptology: Proceedings of CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 513–525. Springer-Verlag, 1997.

[29] I. Blake, G. Seroussi, and N.P. Smart. *Elliptic Curves in Cryptography.* London Mathematical Society Lecture Note Series. Cambridge University Press, 1999.

[30] I. Blake, G. Seroussi, and N.P. Smart. *Advances in Elliptic Curve Cryptography.* London Mathematical Society Lecture Note Series, 317. Cambridge University Press, 2005.

[31] T. Blum and C. Paar. Montgomery modular exponentiation on reconfigurable hardware. In *Proceedings of 14th IEEE Symposium on Computer Arithmetic (ARITH)*, pages 70–77, Adelaide, Australia, April 14-16 1999. IEEE Computer Society.

[32] T. Blum and C. Paar. High-radix Montgomery modular exponentiation on reconfigurable hardware. *IEEE Transactions on Computers*, 50(7):759–764, July 2001.

[33] D. Boneh. Twenty years of attacks on the RSA cryptosystem. *Notices of the American Mathematical Society*, 46(2):203–213, 1999.

[34] D. Boneh, R.A. DeMillo, and R.J. Lipton. On the importance of checking cryptographic protocols for faults (extended abstract). In W. Fumy, editor, *Advances in Cryptology: Proceedings of EUROCRYPT'97*, number 1233 in Lecture Notes in Computer Science, pages 37–51. Springer-Verlag, 1997.

[35] D. Boneh and G. Durfee. Cryptanalysis of RSA with private key d less than $N^{0.292}$. *IEEE Transactions on Information Theory*, 46(4):1339–1349, 2000.

[36] D. Boneh and H. Shacham. Fast variants of rsa. *CryptoBytes*, Vol. 5(1):1–9, 2002.

[37] J. Borst. *Block Ciphers: Design, Analysis and Side-Channel Analysis*. PhD thesis, K.U.Leuven, September 2001.

[38] J. Borst, B. Preneel, and V. Rijmen. Cryptography on smart cards. *Computer Networks*, 36(4):423–435, 2001.

[39] N. Boston, T. Clancy, Y. Liow, and J. Webster. Genus two hyperelliptic curve coprocessor. In B.S. Kaliski Jr., Ç.K. Koç, and C. Paar, editors, *Proceedings of 4th International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, number 2523 in Lecture Notes in Computer Science, pages 400–414. Springer-Verlag, 2002.

[40] X. Boyen, Y. Dodis, J. Katz, R. Ostrovsky, and A. Smith. Secure remote authentication using biometric data. In R. Cramer, editor, *Proceedings of Eurocrypt 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 147–163. Springer-Verlag, 2005.

[41] E.F. Brickell. A fast modular multiplication algorithm with application to two key cryptography. In D. Chaum, R. L. Rivest, and A. T. Sherman, editors, *Advances in Cryptology: Proceedings of CRYPTO'82*, number 740, pages 51–60, Santa Barbara, CA, USA, August 1982. Springer-Verlag.

[42] E.F. Brickell and K.S. McCurley. An interactive identification scheme based on discrete logarithms and factoring. *Journal of Cryptology*, (5):29–39, 1992.

[43] E. Brier and M. Joye. Weierstrass Elliptic Curves and Side-Channel Attacks. In D. Naccache and P. Paillier, editors, *Proceedings of PKC'02*, volume 2274 of *LNCS*, pages 335–345. Springer-Verlag, 2002.

[44] A.E. Brouwer, R. Pellikaan, and E.R. Verheul. Doing More with Fewer Bits. In K.Y. Lam, E. Okamoto, and C. Xing, editors, *Proceedings of ASIACRYPT 1999*, number 1716 in Lecture Notes in Computer Science, pages 321–332. Springer-Verlag, 1999.

[45] B. Byramjee and S. Duquesne. Classification of genus 2 curves over F_{2^n} and optimization of their arithmetic. Cryptology ePrint Archive: Report 2004/107.

[46] D.G. Cantor. Computing in the jacobian of a hyperelliptic curve. *Mathematics of Computation*, 48:95–101, 1987.

[47] Ç.K. Koç. Analysis of sliding window techniques for exponentiation. *Computers and Mathematics with Applications*, 10(30):17–24, 1995.

[48] Ç.K. Koç. RSA hardware implementation. Technical report, RSA Laboratories, RSA Data Security, Inc., Redwood City, CA, August 1995.

[49] Ç.K. Koç and T. Acar. Montgomery multiplication in GF(2^k). *Designs, Codes and Cryptography*, 14:57–69, 1998.

[50] Ç.K. Koç, T. Acar, and B.S. Kaliski Jr. Analyzing and comparing Montgomery multiplication algorithms. *IEEE Micro*, pages 26–33, June 1996.

[51] B. Chevallier-Mames, M. Ciet, and M. Joye. Low-cost solutions for preventing Simple Side-Channel Analysis: Side-Channel Atomicity. *IEEE Transactions on Computers*, 53(6):760–768, June 2004.

[52] T. Clancy. FPGA-based Hyperelliptic Curve Cryptosystems. 2003. Invited paper at AMS Meeting.

[53] H. Cohen, A. Miyaji, and T. Ono. Efficient elliptic curve exponentiation using mixed coordinates. In K. Ohta and D. Pei, editors, *Proceedings of ASIACRYPT 1998*, number 1514 in Lecture Notes in Computer Science, pages 51–65. Springer-Verlag, 1998.

[54] Compaq. Cryptography using Compaq MultiPrime technology in a parallel processing environment. Electronic Commerce Technical Brief, January 2000. `http://www.compaq.com`.

[55] Many contributors. *Cracking DES: Secrets of Encryption Research, Wiretap Politics & Chip Design.* O'Reilly Newsletters, 2002.

[56] J.-S. Coron and L. Goubin. On boolean and arithmetic masking against differential power analysis. In Ç.K. Koç and C. Paar, editors, *Proceedings of 2nd International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, number 1965 in Lecture Notes in Computer Science, pages 231–237, Worcester, Massachusetts, USA, August 17-18 2000. Springer-Verlag.

[57] T.M. Cover and J.A. Thomas. *Elements of Information Theory.* John Wiley and sons, inc., 1991.

[58] F. Crowe, A. Daly, and W. Marnane. A Scalable Dual Mode Arithmetic Unit for Public Key Cryptosystems. In *Proc. IEEE International Conference Conference on Information Technology - ITCC'05*, pages 568–573, Las Vegas, 2005.

[59] J. Daemen and V. Rijmen. Resistance against implementation attacks: A comparative study of the AES proposals. In *Proceedings of the Second Advanced Encryption Standard (AES) Candidate Conference*, March 1999.

[60] Dallas. Dallas semiconductor ds89c420 ultra-high-speed microcontroller. `http://www.maxim-ic.com/quick_view2.cfm/qv_pk/2963`.

[61] Dalton. Dalton 8051 processor. `http://www.cs.ucr.edu/~dalton/8051/`.

[62] W. Diffie and M.E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22:644–654, 1976.

[63] Y. Dodis, J. Katz, S. Xu, and M. Yung. Strong Key-Insulated Signature Schemes. In Y. Desmedt, editor, *Proceedings of 6th International Workshop on Practice and Theory in Public Key Cryptosystems (PKC 2003)*, number 2567 in LNCS, pages 130–144. Springer-Verlag, 2003.

[64] Y. Dodis, M. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In C. Cachin and J. Camenisch, editors, *Proceedings of Eurocrypt 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 523–540. Springer-Verlag, 2004.

[65] P. Ebinger and E. Teske. Factoring $N = pq^2$ with the Elliptic Curve Method. In C. Fieker and D.R. Kohel, editors, *Proceedings of Algorithmic Number Theory: 5th International Symposium, ANTS-V*, volume 2369 of *LNCS*, pages 475–490, Sidney, Australia, 2002. Springer-Verlag.

[66] ECMNET. The ECMNET project. `http://www.loria.fr/~zimmerma/records/ecmnet.html`, April 2005.

[67] ECRYPT. Hardness of the Main Computational Problems Used in Cryptography. Deliverable 1.1, 2004.

[68] ECRYPT. ECRYPT Yearly Report on Algorithms and Keysizes (2004). Document D.SPA.10, January 2005.

[69] S.E. Eldridge and C.D. Walter. Hardware implementation of Montgomery's modular multiplication algorithm. *IEEE Transactions on Computers*, 42:693–699, 1993.

[70] G. Elias, A. Miri, and T. H. Yeap. High-performance, FPGA based hyperelliptic curve cryptosystem. In *In Proceedings of the 22nd Biennial Symposium on Communications*, May 2004.

[71] M.D. Ercegovac and T. Lang. *Digital Arithmetic*. The Morgan Kaufmann Series in Computer Architecture and Design, 2004.

[72] M. Ernst, S. Klupsch, O. Hauck, and S.A. Huss. Rapid prototyping for hardware accelerated elliptic curve public-key cryptosystems. In *Proceedings of 12th IEEE Workshop on Rapid System Prototyping*, pages 24–31, Monterey, CA, June 2001.

[73] B. Gassend et al. Silicon physical unknown functions. *Proc. 9th ACM Conference on Computer and Communications Security*, 2002.

[74] M. Feldhofer, S. Dominikus, and J. Wolkerstorfer. Strong Authentication for RFID Systems using the AES Algorithm. In M. Joye and J. J. Quisquater, editors, *Proceedings of 6th International Workshop on Cryptographic Hardware in Embedded Systems (CHES)*, 2004.

[75] W. Fischer, C. Giraud, E.W. Knudsen, and J.-P. Seifert. Parallel scalar multiplication on general elliptic curves over $\mathbb{F}_p$ hedged against non-differential side-channel attacks. IACR ePrint archive: Report 2002/007, 2002.

[76] International Organization for Standardization. ISO/IEC 18000-3. Information Technology AIDC Techniques - RFID for Item Management, March 2003.

[77] J.J.A. Fournier, S. Moore, H. Li, R. Mullins, and G. Taylor. Security evaluation of asynchronous circuits. In C. Walter, Ç.K. Koç, and C. Paar, editors, *Proceedings of the 5th International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, volume 2779 of *Lecture Notes in Computer Science*, pages 137–151. Springer-Verlag, 2003.

[78] W. Fulton. *Algebraic Curves.* Benjamin, New York, 1969.

[79] R. Gallant, R. Lambert, and S. Vanstone. Faster point multiplication on elliptic curves with efficient endomorphisms. In J. Kilian, editor, *Advances in Cryptology: Proceedings of CRYPTO'01*, number 2139 in Lecture Notes in Computer Science, pages 190–200. Springer-Verlag, 2001.

[80] K. Gandolfi, C. Mourtel, and F. Olivier. Electromagnetic analysis: Concrete results. In Ç.K. Koç, D. Naccache, and C. Paar, editors, *Proceedings of 3rd International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, number 2162 in Lecture Notes in Computer Science, pages 255–265. Springer-Verlag, 2001.

[81] L. Gao, S. Shrivastava, H. Lee, and G.E. Sobelman. A compact fast variable key size elliptic curve cryptosystem coprocessor. In *Proceedings of the Seventh Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 304–305, 1999.

[82] L. Gao, S. Shrivastava, and G.E. Sobelman. Elliptic curve scalar multiplier design using FPGAs. In Ç.K. Koç and C. Paar, editors, *Proceedings of the 1st International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, number 1717 in Lecture Notes in Computer Science, pages 257–305, Worcester, MA, USA, August 1999. Springer-Verlag.

[83] B. Gassend, D. Clarke, M. van Dijk, , and S. Devadas. Controlled physical random functions. *In Proceedings of the 18th Annual Computer Security Conference*, December 2002.

[84] G. Gaubatz, J.-P. Kaps, E. Öztürk, and B. Sunar. State of the Art in Ultra-Low Power Public Key Cryptography for Wireless Sensor Networks. In *2nd IEEE International Workshop on Pervasive Computing and Communication Security (PerSec 2005)*, Kauai Island, Hawaii, March 2005.

[85] G. Gaubatz, J.-P. Kaps, and B. Sunar. Public Key Cryptography in Sensor Networks - Revisited. In *1st European Workshop on Security in Ad-Hoc and Sensor Networks (ESAS 2004)*, Heidelberg, Germany, August 2004.

[86] P. Gaudry. An algorithm for solving the discrete log problem on hyperelliptic curves. In B. Preneel, editor, *Advances in Cryptology: Proceedings of EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 19–34, 2000.

[87] P. Gaudry, F. Hess, and N.P. Smart. Constructive and destructive facets of Weil descent on elliptic curves. *Journal of Cryptology*, 15:19–46, 2002.

[88] GEZEL. `http://www.ee.ucla.edu/~schaum/gezel`.

[89] H. Gilbert, M. Robshaw, and H. Sibert. An Active Attack Against HB+ - A Provably Secure Lightweight Authentication Protocol. ePrint Archive, 2005. 2005/237.

[90] J.D. Golić and C. Tymen. Multiplicative masking and power anaylsis of AES. In B.S. Kaliski Jr., Ç.K. Koç, and C. Paar, editors, *Proceedings of 4th International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, number 2535 in Lecture Notes in Computer Science, pages 198–212, Redwood Shores, CA, USA, August 13-15 2002. Springer-Verlag.

[91] D. Gollmann, Y. Han, and C. Mitchell. Redundant integer representation and fast exponentiation. *Designs, Codes and Cryptography*, 7:135–151, 1998.

[92] J. Goodman and A.P. Chandrakasan. An energy-efficient reconfigurable public-key cryptography processor. *IEEE Journal of Solid-State Circuits*, 36(11):1808–1820, November 2001.

[93] D. M. Gordon. A survey of fast exponentiation methods. *Journal of Algorithms*, 27:129–146, 1998.

[94] J. Gordon. Strong primes are easy to find. In T. Beth, N. Cot, and I. Ingemarsson, editors, *Advances in Cryptology: Proceedings of EUROCRYPT'84*, number 209 in Lecture Notes in Computer Science, pages 216–223. Springer-Verlag, 1984.

[95] L. Goubin. A sound method for switching between boolean and arithmetic masking. In Ç.K. Koç, D. Naccache, and C. Paar, editors, *Proceedings of 3rd International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, number 2162 in Lecture Notes in Computer Science, pages 3–15, Paris, France, May 2001. Springer-Verlag.

[96] L. Goubin and J. Patarin. DES and Differential Power Analysis The "Duplication" Method. In Ç.K. Koç and C. Paar, editors, *Proceedings of 1st International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, number 1717 in Lecture Notes in Computer Science, pages 158–172, Worcester, Massachusetts, USA, August 1999. Springer-Verlag.

[97] R. Granger, D. Page, and M. Stam. A comparison of CEILIDH and XTR. In D. Buell, editor, *Proceedings of Algorithmic Number Theory - ANTS-VI*, number 3076 in Lecture Notes in Computer Science, pages 235–249, 2004.

[98] N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz. Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs. In M. Joye and J.-J. Quisquater, editors, *Proceedings of 6th International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, number 3156 in Lecture Notes in Computer Science, pages 119–132, 2004.

[99] N. Gura, S.C. Shantz, H. Eberle, D. Finchelstein, S. Gupta, V. Gupta, and D. Stebila. An end-to-end systems approach to elliptic curve cryptography. In B. Kaliski Jr., Ç.K. Koç, and C. Paar, editors, *Proceedings of 4th International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, Lecture Notes in Computer Science 2523, 2002.

[100] G. Hachez, F. Koeune, and J.-J. Quisquater. Timing attack: what can be achieved by a powerful adversary? In A. Barbé, E. C. van der Meulen, and P. Vanroose, editors, *Proceedings of the 20th symposium on Information Theory in the Benelux*, pages 63–70, May 1999.

[101] G. Hachez and J.-J. Quisquater. Montgomery exponentiation with no final subtractions: Improved results. In Ç.K. Koç and C. Paar, editors, *Proceedings of 2nd International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, number 1965 in Lecture Notes in Computer Science, pages 293–301. Springer-Verlag, 2000.

[102] D. Hankerson, J.L. Hernandez, and A. Menezes. Software implementation of elliptic curve cryptography over binary fields. In Ç.K. Koç and C. Paar, editors, *Proceedings of 2nd International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, number 1965 in Lecture Notes in Computer Science, pages 1–24, Worcester, Massachusetts, USA, August 17-18 2000. Springer-Verlag.

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

[104] G.H. Hardy and E.M. Wright. *An Introduction to the Theory of Numbers*. Oxford Science Publications, fifth edition, 1979.

[105] M.J. Hinek, M.K. Low, and E. Teske. On some attacks on Multi-prime RSA. In K. Nyberg and H. Heys, editors, *In Selected Areas in Cryptography: SAC 2002*, volume 2595 of *LNCS*, pages 385–404. Springer-Verlag, 2002.

[106] A. Hodjat, L. Batina, D. Hwang, and I. Verbauwhede. HW/SW Co-design of a Hyperelliptic Curve Cryptosystem using a μCode Instruction Set Coprocessor. *Elsevier Science Integration the VLSI Journal*. To appear.

[107] A. Hodjat, L. Batina, D. Hwang, and I. Verbauwhede. A Hyperelliptic Curve Crypto Coprocessor for an 8051 Microcontroller. In *Proceedings of The IEEE 2005 Workshop on Signal Processing Systems (SIPS'05)*, Athens, Greece, 2005. To appear.

[108] F. Hoornaert, M. Decroos, J. Vandewalle, and R. Govaerts. Fast RSA-hardware: dream or reality. In C.G. Günther, editor, *Advances in Cryptology: Proceedings of EUROCRYPT'88*, number 330 in Lecture Notes in Computer Science, pages 257–264. Springer-Verlag, 1988.

[109] D.M. Hopkins, L.T. Kontnik, and M.T. Turnage. *Counterfeiting Exposed: Protecting your Brand and Customers*. Business Strategy. Wiley, 2003.

[110] T.W. Hungerford. *Algebra*. Number 73 in Graduate Texts in Mathematics. Springer-Verlag, 1974.

[111] IEEE P1363. Standard specifications for public key cryptography, 1999.

[112] T. Itoh and S. Tsujii. Effective recursive algorithm for computing multiplicative inverses in GF(2^m). *Electronics Letters*, 24(6):334–335, 1988.

[113] K. Iwamura, T. Matsumoto, and H. Imai. Montgomery modular multiplication method and systolic arrays suitable for modular exponentiation. *Electronics and Communications in Japan*, 77(3):40–50, 1994.

[114] T. Izu and T. Takagi. A fast parallel elliptic curve multiplication resistant against side channel attacks. In D. Naccache and P. Paillier, editors, *Proceedings of 5th International Workshop on Practice and Theory in Public Key Cryptosystems (PKC 2002)*, number 3027 in LNCS, pages 280–296. Springer-Verlag, 2002.

[115] C.J.A. Jansen. Key signature schemes. In *Proceedings of the Seventh Symposium on Information Theory in the Benelux*, pages 197–205, Noordwijkerhout, The Netherlands, 1986.

[116] D. Johnson and A. Menezes. The elliptic curve digital signature algorithm (ECDSA). Technical Report CORR 99-34, Department of Combinatorics & Optimization, University of Waterloo, Canada, February 24 2000. `http://www.cacr.math.uwaterloo.ca`.

[117] M. Joye, A.K. Lenstra, and J.-J. Quisquater. Chinese remaindering based cryptosystem in the presence of faults. *Journal of Cryptology*, 4(12):241–245, 1999.

[118] M. Joye, J.-J. Quisquater, and T. Takagi. How to choose secret parameters for RSA and its extensions to elliptic curves. *Designs, Codes and Cryptography*, 3(23):297–316, 2001.

[119] M. Joye and S.-M. Yen. The montgomery powering ladder. In B.S. Kaliski Jr., Ç.K. Koç, and C. Paar, editors, *Proceedings of 4th International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, number 2523 in Lecture Notes in Computer Science, pages 291–302. Springer-Verlag, 2002.

[120] A. Juels. RFID Security and Privacy: A Research Survey. Manuscript. Available at: `http://lasecwww.epfl.ch/~gavoine/rfid/#webpages`, September 2005.

[121] A. Juels. Strengthening EPC tags against cloning. *http://www.rsasecurity.com/rsalabs/*, 2005.

[122] A. Juels and S.A. Weis. Authenticating pervasive devices with human protocols. In V. Shoup, editor, *Advances in Cryptology: Proceedings of CRYPTO'05*, number 3621 in Lecture Notes in Computer Science, pages 293–308. Springer-Verlag, 2005.

[123] H. Kim, T. Wollinger, Y. Choi, K. Chung, and C. Paar. Hyperelliptic curve coprocessors on a FPGA. In *Workshop on Information Security Applications - WISA*, Jeju Island, Korea, August 23-25 2004.

[124] P. Kitsos, G. Theodoridos, and O. Koufopavlou. An efficient reconfigurable multiplier architecture for Galois Field GF(2^m). *Elsevier Science Microelectronics Journal*, 34:975–980, 2003.

[125] D.E. Knuth. *The Art of Computer Programming – Vol. 2 – Seminumerical Algorithms*. Addison-Wesley, third edition, 1998.

[126] N. Koblitz. Elliptic curve cryptosystem. *Math. Comp.*, 48:203–209, 1987.

[127] N. Koblitz. A family of Jacobians suitable for Discrete Log Cryptosystems. In S. Goldwasser, editor, *Advances in Cryptology: Proceedings of CRYPTO'88*,

number 403 in Lecture Notes in Computer Science, pages 94–99. Springer-Verlag, 1988.

[128] N. Koblitz. Hyperelliptic cryptosystems. *Journal of Cryptology*, 1(3):129–150, 1989.

[129] N. Koblitz. *A Course in Number Theory and Cryptography*, volume 114 of *Graduate text in mathematics*. Springer-Verlag, Berlin, Germany, second edition, 1994.

[130] N. Koblitz. *Algebraic Aspects of Cryptography*. Springer-Verlag, first edition, 1998.

[131] N. Koblitz, A. Menezes, and S. Vanstone. The state of elliptic curve cryptography. *Designs, Codes and Cryptography*, 19:173–193, 2000.

[132] M. Kochanski. Developing an RSA chip. In H. C. Williams, editor, *Advances in Cryptology: Proceedings of CRYPTO'85*, number 218 in Lecture Notes in Computer Science, pages 350–357. Springer-Verlag, 1985.

[133] P. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS and other systems. In N. Koblitz, editor, *Advances in Cryptology: Proceedings of CRYPTO'96*, number 1109 in Lecture Notes in Computer Science, pages 104–113. Springer-Verlag, 1996.

[134] P. Kocher, J. Jaffe, and B. Jun. Introduction to differential power analysis and related attacks. `http://www.cryptography.com/dpa/technical`, 1998.

[135] P. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In M. Wiener, editor, *Advances in Cryptology: Proceedings of CRYPTO'99*, number 1666 in Lecture Notes in Computer Science, pages 388–397. Springer-Verlag, 1999.

[136] S. Kumar and C. Paar. Reconfigurable instruction set extension for enabling ECC on an 8-bit processor. In *Proceedings of International Conference on Field-Programmable Logic and Applications (FPL) 2004*, Antwerp, Belgium, August 30-September 1, 2004.

[137] T. Lange. *Efficient Arithmetic on Hyperelliptic Cyrves*. PhD thesis, Universität Gesamthochschule Essen, 2001.

[138] T. Lange. Formulae for arithmetic on genus 2 hyperelliptic curves. *Applicable Algebra in Engineering, Communication and Computing*, 15(5):295–328, February 2005.

[139] M.J.J. Lennon and P.J. Smith. LUC: A New Public Key System. In *In IFIP TC11, 9th International Conference on Information Security IFIP/Sec*, pages 103–117, 1993.

[140] A. Lenstra. Unbelievable Security: Matching AES Security Using Public Key Systems. In C. Boyd, editor, *Advances in Cryptology - ASIACRYPT 2001*, volume 2248 of *LNCS*, 2001. Invited talk. Available at: `http://www.win.tue.nl/~klenstra/aes_match.pdf`.

[141] A. Lenstra and E. Verheul. Selecting cryptographic key sizes. In H. Imai and Y. Zheng, editors, *Proceedings of Third International Workshop on Practice and Theory in Public Key Cryptography (PKC 2000)*, number 1751 in Lecture Notes in Computer Science, pages 446–465. Springer-Verlag, 2000.

[142] A. Lenstra and E. Verheul. The XTR Public Key System. In M. Bellare, editor, *Advances in Cryptology: Proceedings of CRYPTO'00*, number 1880 in Lecture Notes in Computer Science, pages 1–19. Springer-Verlag, 2000.

[143] H.W. Lenstra Jr. Factoring integers with elliptic curves. *Ann. of Mathematics*, 126:649–673, 1987.

[144] K.H. Leung, K.W. Ma, W.K. Wong, and P.H.W. Leong. FPGA implementation of a microcoded elliptic curve cryptographic processor. In *Proceedings of Field-Programmable Custom Computing Machines (FCCM)*, pages 68–76, 2000.

[145] R. Lidl and H. Niederreiter. *Finite fields*, volume 20 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, second edition, 2000.

[146] C.H. Lim and P.J. Lee. More flexible exponentiations with precomputation. In Y. Desmedt, editor, *Advances in Cryptology: Proceedings of CRYPTO'94*, number 839 in Lecture Notes in Computer Science, pages 95–107. Springer-Verlag, 1994.

[147] J.P. Linnartz and P. Tuyls. New shielding functions to enhance privacy and prevent misuse of biometric templates. In J. Kittler and M. Nixon, editors, *Proc. of the 3rd Conference on Audio and Video Based Person Authentication*, volume 2688 of *Lecture Notes in Computer Science*, pages 238–250. Springer-Verlag, 2003.

[148] J. López and R. Dahab. Fast multiplication on elliptic curves over GF(2^m). In Ç.K. Koç and C. Paar, editors, *Proceedings of 1st International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, volume 1717 of *Lecture Notes in Computer Science*, pages 316–327. Springer-Verlag, 1999.

[149] E. Macii. Design Techniques and Tools for Low-Power Digital Systems. course notes, 2003. IMEC course.

[150] S. Mangard. *Securing Implementations of Block Ciphers Against Side-Channel Attacks*. PhD thesis, Institute for Applied Information Processing and Communications (IAIK), TU Graz, 2004.

[151] M.M. Mano and C.R. Kime. *Logic and Computer Design Fundamentals*. Prentice Hall, Upper Saddle River, New Jersey 07458, second edition, 2001.

[152] J.L. Massey. Applied Digital Information Theory II. Lecture notes, ETH, Zurich, 1997.

[153] C. McIvor, M. McLoone, J. McCanny, A. Daly, and W. Marnane. Fast Montgomery Modular Multiplication and RSA Cryptographic Processor Architectures. In *Proceedings of 37th Annual Asilomar Conference on Signals, Systems and Computers*, pages 379–384, November 2003.

[154] A. Menezes, E. Teske, and A. Weng. Weak fields for ECC. In Springer-Verlag, editor, *In Topics in Cryptology - CT-RSA - The Cryptographers' Track at the RSA Conference*, number 2964 in LNCS, pages 366–386, 2004.

[155] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

[156] A. Menezes and S. Vanstone. Elliptic curve cryptosystems and their implementations. *Journal of Cryptology*, 6(4):209–224, 1993.

[157] A. Menezes, Y.-H. Wu, and R. Zuccherato. *An Elementary Introduction to Hyperelliptic Curves - Appendix*, pages 155–178. Springer-Verlag, 1998. N. Koblitz: Algebraic Aspects of Cryptography.

[158] A.J. Menezes. *Elliptic Curve Public Key Cryptosystems*. Kluwer Academic Publishers, 1993.

[159] N. Mentens, S.B. Örs, B. Preneel, and J. Vandewalle. An FPGA implementation of an elliptic curve processor over GF(2^m). In *In Proceedings of the 2004 Great Lakes Symposium on VLSI (GLSVLSI 2004)*, Boston, MA, USA, April.

[160] T.S. Messerges. *Power Analysis Attacks and Countermeasures on Cryptographic Algorithms*. PhD thesis, 2000.

[161] T.S. Messerges. Using second-order power analysis to attack DPA resistant software. In Ç.K. Koç and C. Paar, editors, *Proceedings of the 2nd International Workshop on Cryptograpic Hardware and Embedded Systems (CHES)*, volume 1965 of *Lecture Notes in Computer Science*, pages 238–251. Springer-Verlag, 2000.

[162] T.S. Messerges, E. A. Dabbish, and R. H. Sloan. Examining smart-card security under the threat of power analysis attacks. *IEEE Transactions on Computers*, 51(5):541–552, May 2002.

[163] V. Miller. Uses of elliptic curves in cryptography. In H. C. Williams, editor, *Advances in Cryptology: Proceedings of CRYPTO'85*, number 218 in Lecture Notes in Computer Science, pages 417–426. Springer-Verlag, 1985.

[164] P. Montgomery. Modular multiplication without trial division. *Mathematics of Computation*, 44(170):519–521, 1985.

[165] P. Montgomery. Speeding the pollard and elliptic curve methods of factorization. *Mathematics of Computation*, 48(177):243–264, 1987.

[166] Michael Neve, Eric Peeters, David Samyde, and Jean-Jacques Quisquater. Memories: A Survey of their Secure Uses in Smart Cards. In *2nd International IEEE Security In Storage Workshop (IEEE SISW 2003)*, pages 62–72, Washington DC, USA, 2003.

[167] I. M. Niven, H. S. Zuckerman, and H. L. Montgomery. *An Introduction to the Theory of Numbers*. Wiley, 5th edition, 1991.

[168] R. Novak. SPA-based adaptive chosen-ciphertext attack on RSA implementation. In D. Naccache and P. Paillier, editors, *Proceedings of the 5th International Workshop on Practice and Theory in Public Key Cryptosystems (PKC)*, volume 2274 of *Lecture Notes in Computer Science*, pages 252–262. Springer-Verlag, 2002.

[169] C.W. O'Donnel, G.E. Suh, and S. Devadas. PUF-based random number generation. Technical Report 481, MIT CSAIL.

[170] K. Okeya, H. Kurumatani, and K. Sakurai. Elliptic curves with the Montgomery-form and their cryptographic applications. In H. Imai and Y. Zheng, editors, *Proceedings of the 3rd International Workshop on Practice and Theory in Public Key Cryptography (PKC)*, volume 1751 of *Lecture Notes in Computer Science*, pages 238–257. Springer-Verlag, 2000.

[171] G. Orlando and C. Paar. A high-performance reconfigurable elliptic curve processor for GF(2^m). In Ç.K. Koç and C. Paar, editors, *Proceedings of 2nd International Workshop on Cryptograpic Hardware and Embedded Systems (CHES)*, number 1965 in Lecture Notes in Computer Science, pages 41–56. Springer-Verlag, 2000.

[172] G. Orlando and C. Paar. A scalable GF(p) elliptic curve processor architecture for programmable hardware. In Ç.K. Koç, D. Naccache, and C. Paar, editors, *Proceedings of 3rd International Workshop on Cryptograpic Hardware*

and Embedded Systems (CHES), number 2162 in Lecture Notes in Computer Science, pages 356–371. Springer-Verlag, 2001.

[173] S.B. Örs, L. Batina, and B. Preneel. Hardware implementation of elliptic curve processor over GF(p). New European Schemes for Signatures, Integrity, and Encryption (NESSIE) working paper NES/-DOC/KUL/WP5/023, Department of Electrical Engineering, ESAT/-COSIC, Katholieke Universiteit Leuven, 10 October 2002.

[174] S.B. Örs, L. Batina, B. Preneel, and J. Vandewalle. Hardware implementation of a Montgomery modular multiplier in a systolic array. In *The The 10th Reconfigurable Architectures Workshop (RAW)*, Nice, France, April 22 2003.

[175] S.B. Örs, L. Batina, B. Preneel, and J. Vandewalle. Hardware implementation of an elliptic curve processor over GF(p). In *IEEE 14th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, The Hague, The Netherlands, June 24-26 2003.

[176] S.B. Örs, L. Batina, B. Preneel, and J. Vandewalle. *IJES*, 2005. To appear.

[177] S.B. Örs, E. Oswald, and B. Preneel. Power-analysis attacks on an FPGA – first experimental results. In C. Walter, Ç.K. Koç, and C. Paar, editors, *Proceedings of 5th International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, number 2779 in Lecture Notes in Computer Science, pages 35–50. Springer-Verlag, 2003.

[178] Sıddıka Berna Örs. *Hardware Design of Elliptic Curve Cryptosystems and Side-channel Attacks*. PhD thesis, Katholieke Universiteit Leuven, Belgium, February 2005.

[179] H. Orup. Simplifying quotient determination in high-radix modular multiplication. In *Proceedings of the 12th Symposium on Computer Arithmetic*, pages 193–199. IEEE, 1995.

[180] R. Pappu. Physical one-way functions. *Science*, 297(6):2026, 2002.

[181] B. Parhami. *Computer Arithmetic: Algorithms and Hardware Designs*. Oxford University Press, Inc., New York, 2000.

[182] J. Pelzl, T. Wollinger, J. Guajardo, and C. Paar. Hyperelliptic curve cryptosystems: Closing the performance gap to elliptic curves. In C. Walter, Ç.K. Koç, and C. Paar, editors, *Proceedings of 5th International Workshop on Cryptograpic Hardware and Embedded Systems (CHES)*, number 2779 in Lecture Notes in Computer Science, pages 351–365. Springer-Verlag, 2003.

[183] J. Pelzl, T. Wollinger, and C. Paar. High performance arithmetic for hyperelliptic curve cryptosystems of genus two. In *Proceedings of ITCC, April 5-7, 2004*, Las Vegas, Nevada, USA, 2004.

[184] J. Pelzl, T. Wollinger, and C. Paar. *Special Hyperelliptic Curve Cryptosystems of Genus Two: Efficient Arithmetic and Fast Implementation*, chapter in Embedded Cryptographic Hardware: Design and Security. Nova Science Publishers, 2004.

[185] T. Popp and S. Mangard. Masked Dual-Rail Pre-Charge Logic: DPA-Resistance without Routing Constraints. In J. R. Rao and B. Sunar, editors, *Proceedings of 7th International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, number 3659 in Lecture Notes in Computer Science, page 172186. Springer-Verlag, 2005.

[186] M.J. Potgieter. A hardware implementation of the group operations necessary for implementing an elliptic curve cryptosystem over a characteristic two finite field. Final report of project EPR400, Technical University Eindhoven, 2002.

[187] O. Pretzel. *Error-Correcting Codes and Finite Fields*. Clarendon Press – Oxford, 1992.

[188] J.-J. Quisquater and C. Couvreur. Fast decipherment algorithm for RSA public-key cryptosystem. *Electronics Letters*, 18:905–907, October 1982.

[189] J.-J. Quisquater and D. Samyde. ElectroMagnetic Analysis (EMA): Measures and Couter-Measures for Smard Cards. In I. Attali and T. P. Jensen, editors, *Smart Card Programming and Security (E-smart 2001)*, volume 2140 of *Lecture Notes in Computer Science*, pages 200–210. Springer-Verlag, 2001.

[190] J. Rabaey. *Digital Integrated Circuits: A Design Perspective*. Prentice Hall, 1996.

[191] G. Rankine. Thomas – a complete single chip RSA device. In A. M. Odlyzko, editor, *Advances in Cryptology: Proceedings of CRYPTO'86*, number 263 in Lecture Notes in Computer Science, pages 481–487. Springer-Verlag, 1986.

[192] R.L. Rivest. RSA Chips: Past/Present/Future. In T. Beth, N. Cot, and I. Ingemarsson, editors, *Advances in Cryptology: Proceedings of EUROCRYPT'84*, number 209 in Lecture Notes in Computer Science, pages 159–168. Springer-Verlag, 1984.

[193] R.L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[194] R.L. Rivest and R.D. Silverman. Are 'strong' primes needed for RSA. 1999.

[195] K.H. Rosen. *Handbook of Discrete and Combinatorial Mathematics*. CRC Press, 2000.

[196] K. Rubin and A. Silverberg. Torus-based cryptography. In D. Boneh, editor, *Advances in Cryptology: Proceedings of CRYPTO'03*, number 2729 in Lecture Notes in Computer Science, pages 349–365. Springer-Verlag, 2003.

[197] K. Rubin and A. Silverberg. Using primitive subgroups to do more with fewer bits. In D. Buell, editor, *Proceedings of Algorithmic Number Theory - ANTS-VI*, number 3076 in Lecture Notes in Computer Science, pages 18–41, 2004.

[198] D. Samyde and J.-J. Quisquater. Eddy Current for Magnetic Analysis with Active Sensors. In *Smart Card Programming and Security (E-smart 2002)*, pages 185–194, 2002.

[199] D. Samyde, S. Skorobogatov, R. Anderson, and J.-J. Quisquater. On a new way to read data from memory. In *First International IEEE Security in Storage Workshop, Greenbelt Marriott, Maryland, USA*, 2002.

[200] A. Satoh and K. Takano. A scalable dual-field elliptic curve cryptographic processor. In C. Paar and Ç.K. Koç, editors, *IEEE Transactions on Computers, special issue on cryptographic hardware and embedded systems*, volume 52, pages 449–460, April 2003.

[201] E. Savaş, A. F. Tenca, and Ç.K. Koç. A scalable and unified multiplier architecture for finite fields GF(p) and GF(2^m). In C. Paar and Ç.K. Koç, editors, *Proceedings of 2nd International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, number 1965 in Lecture Notes in Computer Science, pages 281–296, Worcester, Massachusetts, USA, August 17-18 2000. Springer-Verlag.

[202] P. Schaumont and I. Verbauwhede. A reconfiguration hierarchy for elliptic curve cryptography. In *Proceedings of 35th Asilomar Conference on Signals, Systems, and Computers*, November 4-7 2001.

[203] P. Schaumont and I. Verbauwhede. Domain specific codesign for embedded security. *IEEE Computer Magazine*, 36(4):68–74, April 2003.

[204] P. Schaumont, I. Verbauwhede, K. Keutzer, and M. Sarrafzadeh. A quick safari through the reconfiguration jungle. In *Proceedings of 38th Design Automation Conference (DAC'01)*, pages 172–177, Las Vegas, NV, USA, June 18-22 2001.

[205] W. Schindler, F. Koeune, and J.-J. Quisquater. Unleashing the full power of timing attack. Technical Report 2002-3, UCL Crypto Group, 2002.

[206] A. Shamir. Method and apparatus for protecting public key schemes from timing and fault attacks. US patent number 5,991,415, November 1999.

[207] M. Shand and J. Vuillemin. Fast implementations of RSA cryptography. In *Proceedings of the 11th IEEE Symposium on Computer Arithmetic*, Windsor, Ontario, Canada, 1993.

[208] C.E. Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal*, 27:379–423, July, October 1948. Reprinted with corrections.

[209] J.H. Silverman. *The Arithmetic of Elliptic Curves*. Number 106 in Graduate Texts in Mathematics. Springer-Verlag, 2nd edition, 1992.

[210] G.J. Simmons. Identification of data, devices, documents and individuals. In *Proc. 25th Ann. Intern. Carnahan Conference on Security Technology*, pages 197–218, Taipei, Taiwan, ROC, October 1–3, 1991. IEEE.

[211] S. Singh. *The Code Book*. 1999.

[212] B. Skoric, P. Tuyls, and W. Ophey. Robust key extraction from physical unclonable functions. In J. Ionnidis, A.D. Keromytis, and M. Yung, editors, *Proceedings of the Applied Cryptography and Network Security Conference 2005*, volume 3531 of *Lecture Notes in Computer Science*, pages 407–422. Springer-Verlag, 2005.

[213] S.P. Skorobogatov and R.J. Anderson. Optical fault induction attacks. In B.S. Kaliski Jr., Ç.K. Koç, and C. Paar, editors, *Proceedings of the 4th International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, volume 2523 of *Lecture Notes in Computer Science*, pages 2–12. Springer-Verlag, 2002.

[214] L. Song and K.K. Parhi. Low-Energy Digit-Serial/Parallel Finite Field Multipliers. *Journal of* VLSI *Signal Processing*, 19:149–166, 1998.

[215] M. Stam. *Speeding up Subgroup Cryptosystems*. PhD thesis, Technische Universiteit Eindhoven, The Netherlands, 2003.

[216] M. Stam and A.K. Lenstra. Efficient Subgroup Exponentiation. In B.S. Kaliski Jr., Ç.K. Koç, and C. Paar, editors, *Proceedings of 4th International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, number 2535 in Lecture Notes in Computer Science, page 318332. Springer-Verlag, 2002.

[217] S. Sutikno, R. Effendi, and A. Surya. Design and implementation of arithmetic processor GF(2^{155}) for elliptic curve cryptosystems. In *Proceedings of the 1998 IEEE Asia-Pacific Conference on Circuits and Systems (APCCAS'98)*, pages 647–650, 1998.

[218] S. Sutikno, A. Surya, and R. Effendi. An implementation of ElGamal elliptic curve cryptosystem. In *Proceedings of the IEEE Asia Pacific Conference on Circuits and Systems*, pages 483–486, Chiangmai, Thailand, 1998.

[219] T. Takagi. Fast RSA-type cryptosystem using *n-adic expansion.* In B.S. Kaliski Jr., editor, *Advances in Cryptology: Proceedings of CRYPTO'97*, number 1294 in Lecture Notes in Computer Science, pages 372–384. Springer-Verlag, 1997.

[220] T. Takagi. Fast RSA-type cryptosystem modulo $p^k q$. In H. Krawczyk, editor, *Advances in Cryptology: Proceedings of CRYPTO'98*, number 1462 in Lecture Notes in Computer Science, pages 318–326. Springer-Verlag, 1998.

[221] A.F. Tenca and Ç.K. Koç. A scalable architecture for Montgomery multiplication. In Ç.K. Koç and C. Paar, editors, *Proceedings of 1st International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, number 1717 in Lecture Notes in Computer Science, pages 94–108, Worcester, Massachusetts, USA, August 12-13 1999. Springer-Verlag.

[222] A.F. Tenca, G. Todorov, and Ç.K. Koç. High-radix design of a scalable modular multiplier. In Ç.K. Koç, D. Naccache, and C. Paar, editors, *Proceedings of 3rd International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, number 2162 in Lecture Notes in Computer Science, pages 189–205. Springer-Verlag, 2001.

[223] N. Thériault. Index calculus attack for hyperelliptic curves of small genus. In C. S. Laih, editor, *Proceedings of Advances in Cryptology - ASIACRYPT: 9th International Conference on the Theory and Application of Cryptology and Information Security*, number 2894 in Lecture Notes in Computer Science, pages 75–92. Springer-Verlag, 2003.

[224] K. Tiri and I. Verbauwhede. Securing encryption algorithms against DPA at the logic level: Next generation smart card technology. In C. Walter, Ç.K. Koç, and C. Paar, editors, *Proceedings of 5th International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, number 2779 in Lecture Notes in Computer Science, page 125136. Springer-Verlag, 2003.

[225] K. Tiri and I. Verbauwhede. A logic level design methodology for a secure dpa resistant asic or fpga implementation. In *Proceedings of Design, Automation and Test in Europe Conference (DATE)*, pages 246–251, February 2004.

[226] E. Trichina and A. Bellezza. Implementation of elliptic curve cryptography with built-in counter measures against side channel attacks. In B.S. Kaliski Jr., Ç.K. Koç, and C. Paar, editors, *Proceedings of 4th International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, number 2535 in Lecture Notes in Computer Science, page 98113. Springer-Verlag, 2002.

[227] E. Trichina and A. Tiountchik. Scalable algorithm for Montgomery multiplication and its implementation on the coarse-grain reconfigurable chip. In D. Naccache, editor, *In Topics in Cryptology - CT-RSA - The Cryptographers' Track at the RSA Conference*, number 2020 in Lecture Notes in Computer Science, pages 235–249. Springer-Verlag, 2001.

[228] W.-C. Tsai and S.-J. Wang. A systolic architecture for elliptic curve cryptosystems. In *Proceedings of the ICSP2000*, pages 591–597, 2000.

[229] P. Tuyls and L. Batina. RFID-tags for Anti-Counterfeiting. In D. Pointcheval, editor, *In Topics in Cryptology - CT-RSA - The Cryptographers' Track at the RSA Conference*, Lecture Notes in Computer Science, San Jose, USA, February 13-17 2006. Springer Verlag.

[230] P. Tuyls and J. Goseling. Capacity and examples of template protecting biometric authentication systems. In D. Maltoni and A.K. Jain, editors, *Proceedings of Biometric Authentication Workshop*, volume 3087 of *Lecture Notes in Computer Science*, pages 158–170. Springer-Verlag, 2004.

[231] P. Tuyls and B. Skoric. Secret key generation from classical physics. *Philips Research Book Series*, September 2005.

[232] P. Tuyls, B. Skoric, S. Stallinga, A.H.M. Akkermans, and W. Ophey. Information Theoretical Security Analysis of Physical Unclonable Functions. In A.S. Patrick and M. Yung, editors, *Proceedings Conf on Financial Cryptography and Data Security 2005*, volume 3570 of *Lecture Notes in Computer Science*, pages 141–155. Springer-Verlag, 2005.

[233] SHARCS 2005 - Workshop on Special-Purpose Hardware for Attacking Cryptographic Systems. ENSTA, Paris, February 2005.

[234] M. van Dijk, R. Granger, D. Page, K. Rubin, A. Silverberg, M. Stam, and D. Woodruff. Practical Cryptography in High Dimensional Tori. In R. Cramer, editor, *Advances in Cryptology: Proceedings of EUROCRYPT'05*, number 3494 in Lecture Notes in Computer Science, pages 234–250. Springer-Verlag, 2005.

[235] M van Dijk and P. Tuyls. Robustness, reliability and security of biometric key distillation in the information theoretic setting. In N. Cerf and J. Cardinal,

editors, *Proceedings of the 26th Benelux Symposium on Information Theory*, volume 26 of *Proceedings of the WIC*, 2005.

[236] H.C.A. van Tilborg. *Error-correcting codes - a first course*. Studentlitteratur, Lund, 1993.

[237] H.C.A. van Tilborg. *Fundamentals of Cryptology*. Kluwer Academic Publishers, 2000.

[238] C.P. Waldvogel and J.L. Massey. The probability distribution of the Diffie-Hellman key. In *Advances in Cryptology: Proceedings of AUSCRYPT'92*, number 718 in Lecture Notes in Computer Science, pages 492–504. Springer-Verlag, 1993.

[239] C.D. Walter. Systolic modular multiplication. *IEEE Transactions on Computers*, 42:376–378, 1993.

[240] C.D. Walter. An improved linear systolic array for fast modular exponentiation. *IEE Computers and Digital Techniques*, 147(5):323–328, September 2000.

[241] C.D. Walter. MIST: An efficient, randomized exponentiation algorithm for resisting power analysis. In B. Preneel, editor, *Proceedings of RSA 2002 Cryptographers' Track*, number 2271 in Lecture Notes in Computer Science, pages 53–66, San Jose, USA, February 18-2 2002. Springer Verlag.

[242] C.D. Walter. Precise bounds for Montgomery modular multiplication and some potentially insecure RSA moduli. In B. Preneel, editor, *In Topics in Cryptology - CT-RSA - The Cryptographers' Track at the RSA Conference*, number 2271 in Lecture Notes in Computer Science, pages 30–39, 2002.

[243] C.D. Walter. Some security aspects of the MIST randomized exponentiation algorithm. In B.S. Kaliski Jr., Ç.K. Koç, and C. Paar, editors, *Proceedings of 4th International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, number 2523 in Lecture Notes in Computer Science. Springer-Verlag, 2002.

[244] C.D. Walter and S. Thompson. Distinguishing exponent digits by observing modular subtractions. In D. Naccache, editor, *In Topics in Cryptology - CT-RSA - The Cryptographers' Track at the RSA Conference*, number 2020 in Lecture Notes in Computer Science, pages 192–207, San Francisco, 8-12 April 2001. Springer-Verlag.

[245] S.-W. Wei. VLSI architectures for computing exponentiations, multiplicative inverses, and divisions in GF(2^m). *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 44(10):847–855, October 1997.

[246] S.H. Weingart. Physical security devices for computer subsystems: A survey of attacks and defenses. In Ç.K. Koç and C. Paar, editors, *Proceedings of 2nd International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, number 1965 in Lecture Notes in Computer Science, pages 302–317. Springer-Verlag, 2000.

[247] M.J. Wiener. Cryptanalysis of short RSA secret exponents. *IEEE Transactions on Information Theory*, 36(3):553–558, May 1990.

[248] M.J. Wiener. Efficient DES Key Search. presented at the Rump session of CRYPTO '93, 1993. Reprinted in "Practical Cryptography for Data Internetworks", W. Stallings, editor, IEEE Computer Society Press, pp. 31-79 (1996).

[249] J. Wolkerstorfer. Dual-field arithmetic unit for $GF(p)$ and $GF(2^m)$. In B.S. Kaliski Jr., Ç.Koç, and C. Paar, editors, *Proceedings of 4th International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, number 2523 in Lecture Notes in Computer Science. Springer-Verlag, 2002.

[250] J. Wolkerstorfer. Is Elliptic-Curve Cryptography Suitable to Secure RFID Tags?, 2005. Workshop on RFID and Lightweight Crypto, Graz, Austria.

[251] J. Wolkerstorfer. Scaling ECC Hardware to a Minimum. In ECRYPT workshop - Cryptographic Advances in Secure Hardware - CRASH 2005, September 6-7 2005. Invited talk.

[252] T. Wollinger. *Software and Hardware Implementation of Hyperelliptic Curve Cryptosystems*. PhD thesis, Ruhr-University Bochum, Germany, 2004.

[253] T. Wollinger, G. Bertoni, L. Breveglieri, and C. Paar. Performance of HECC coprocessors using inversionfree formulae. In *International Workshop on Information Security & Hiding, Singapore (ISH '05)*.

[254] T. Wollinger, J. Pelzl, V. Wittelsberger, C. Paar, G. Saldamli, and Ç.Koç. Elliptic and hyperelliptic curves on embedded μP. *ACM Transactions on Embedded Computing Systems*, 3(3):509–533, 2004.

[255] A.D. Woodbury, D.V. Bailey, and C. Paar. Elliptic curve cryptography on smartcards without coprocessors. In *Proceedings of Fourth Smart Card Research and Advanced Applications (CARDIS 2000) Conference*, Bristol, UK, September 20-22 2000.

[256] H. Wu. Montgomery multiplier and squarer for a class of finite fields. *IEEE Transactions on Computers*, 51(5):521–529, May 2002.

List of Publications

Journals

1. L. Batina, S. B. Örs, B. Preneel, J. Vandewalle, "Hardware Architectures for Public Key Cryptography", Elsevier Science Integration, the VLSI Journal, 2003, Vol. 34, Nr. 1-2, pp. 1-64.

2. S. B. Örs, L. Batina, B. Preneel, J. Vandewalle, "Hardware Implementation of an Elliptic Curve Processor over GF(p) with Montgomery Modular Multiplier" accepted for publication in International Journal of Embedded Systems (IJES), 2005, 15 pages.

3. L. Batina, N. Mentens, B. Preneel, I. Verbauwhede, "Balanced algorithms for side-channel aware design of Elliptic Curve Cryptography", Special Issue on Cryptographic Algorithms and Architectures for System-on-Chip in the new IEE Proceedings on Information Security, 152(1), 2005, pp. 57-65.

4. A. Hodjat, L. Batina, D. Hwang, I. Verbauwhede, "HW/SW Co-design of a Hyperelliptic Curve Cryptosystem using a μ-Code Instruction Set Coprocessor", accepted for publication in Elsevier, VLSI Journal of Integration.

5. L. Batina, A. Hodjat, D. Hwang, K. Sakiyama, B. Preneel, I. Verbauwhede "Hardware/Software Co-design for Curve-based Cryptography on the 8051 Microprocessor", submitted for publication in IEEE Transactions on Very Large Scale Integration Special Section on Hardware/Software Codesign and and System Synthesis.

Journals (national level)

6. L. Batina, P. Buysschaert, E. De Mulder, N. Mentens, S. B. Örs, B. Preneel, G. Vandenbosch, I. Verbauwhede, "Side channel attacks and fault attacks on cryptographic algorithms", Revue HF Tijdschrift 2004(4), pp. 36-45.

LNCS publications

7. S. Xu, L. Batina, "Efficient Implementation of Elliptic Curve Cryptosystems on an ARM7 with Hardware Accelerator", Proceedings of Information Security Conference, ISC'01, Malaga, Spain, October 2001, G. I. Davida and Y. Frankel (eds.) Lecture Notes in Computer Science LNCS 2200, pp. 266-279, Springer-Verlag.

8. L. Batina, G. Muurling, "Another Way of Doing RSA Cryptography", Cryptography and Coding 8th IMA International Conference, Cirencester, UK, December, 2001, B. Honary (ed.), Lecture Notes in Computer Science LNCS 2260, pp. 364-373, Springer-Verlag.

9. L. Batina, G. Muurling, "Montgomery in practice: How to do it more efficiently in hardware", In Topics in Cryptology - CT-RSA - The Cryptographers' Track at the RSA Conference 2002, San Jose, CA, USA, February, 2002, B. Preneel (ed.), Lecture Notes in Computer Science LNCS 2271, pp.40-52, Springer-Verlag.

10. L. Batina, G. Bruin-Muurling, S. B. Örs, "Flexible Hardware Design for RSA and Elliptic Curve Cryptosystems", In Topics in Cryptology - CT-RSA - The Cryptographers' Track at the RSA Conference 2004, San Francisco, February 23-27, 2004, T. Okamoto (ed.), Lecture Notes in Computer Science LNCS 2964, pp. 250-263, Springer-Verlag.

11. N. Mentens, L. Batina, B. Preneel, I. Verbauwhede, "A Systematic Evaluation of Compact Hardware Implementations for the Rijndael S-box", In Topics in Cryptology - CT-RSA - The Cryptographers' Track at the RSA Conference, 2005, San Francisco, February, 2005, A. Menezes (ed.), Lecture Notes in Computer Science 3376, pp. 323-333, Springer Verlag.

12. L. Batina, D. Hwang, A. Hodjat, B. Preneel, I. Verbauwhede", "Hardware/-Software Co-design for Hyperelliptic Curve Cryptography (HECC) on the 8051 μP", Proceedings of 7th International Workshop on Cryptographic Hardware and Embedded Systems (CHES), 2005, J. R. Rao and B. Sunar (eds.), Lecture Notes in Computer Science, LNCS 3659, pp. 106-118, Springer-Verlag.

13. P. Tuyls and L. Batina, "RFID-tags for Anti-Counterfeiting", In Topics in Cryptology - CT-RSA - The Cryptographers' Track at the RSA Conference, 2006, San Jose, February, 2006, D. Pointcheval (ed.), Lecture Notes in Computer Science, LNCS 3860, pp. 115-131, Springer Verlag, to appear.

International conference articles

14. S. B. Örs, L. Batina, B. Preneel, J. Vandewalle, "Hardware Implementation of a Montgomery Modular Multiplier in a Systolic Array", Proceedings of 10th Reconfigurable Architectures Workshop (RAW 2003), Nice, France, April 2003, 8 pages, ACM.
15. S. B. Örs, L. Batina, B. Preneel, J. Vandewalle, "Hardware Implementation of an Elliptic Curve Processor over GF(p)", Proceedings of the IEEE 14th International Conference on Application-specific Systems, Architectures and Processors, The Hague, The Netherlands, June 24-26, 2003, pp. 433-443.
16. L. Batina, N. Mentens, S. B. Örs, B. Preneel, "Serial multiplier architectures over GF(2^n) for Elliptic Curve Cryptosystems", in the Proceedings of the 12th IEEE MELECON 2004, Dubrovnik, Croatia, May, 2004, 4 pages.
17. L. Batina, N. Mentens, B. Preneel, I. Verbauwhede, "A New Systolic Architecture for Multiplication in GF(2^n)", In IFAC Workshop - PDS 2004, Programmable Devices and Systems, 6 pages.
18. N. Mentens, L. Batina, B. Preneel, I. Verbauwhede, "An FPGA Implementation of Rijndael: Trade-offs for side-channel security", In IFAC Workshop - PDS 2004, Programmable Devices and Systems, 6 pages, 2004.
19. L. Batina, N. Mentens, I. Verbauwhede, "Side-channel Issues for Designing Secure Hardware Implementations", In 11th IEEE International On-Line Testing Symposium, San Raphael, France, July, 2005, eds. C. Metra, K. Roy, L. Anghel and M. Nicolaidis, pp.118-121, IEEE Computer Society.
20. L. Batina, N. Mentens, B. Preneel, I. Verbauwhede, "Side-channel aware design: Algorithms and Architectures for Elliptic Curve Cryptography over GF(2^n)", Proceedings of the IEEE International Conference on Application-Specific Systems, Architectures, and Processors (ASAP)", Samos, Greece, July, 2005, eds. S. Vassiliadis, N. Dimopoulos and S. Rajopadhye, pp. 350-355, IEEE Computer Society Press.
21. A. Hodjat, L. Batina, D. Hwang, I. Verbauwhede, "A Hyperelliptic Curve Crypto Coprocessor for an 8051 Microcontroller", Proceedings of The IEEE 2005 Workshop on Signal Processing Systems (SIPS'05), Athens, Greece, November, 2005, 6 pages.

Other articles

22. L. Batina, C. Jansen, G. Muurling, S. Xu, "Almost Montgomery based multiplier in GF(2^n)", Proceedings of the 23rd Symposium on Information Theory

in The Benelux, Louvain-la-Neuve, Belgium, May, 2002, B. Macq and J.-J. Quisquater (eds.) pp. 61-68.

23. L. Batina, C. Jansen "Secret exponent information leakage for timing analysis", Proceedings of the 23rd Symposium on Information Theory in The Benelux, Louvain-la-Neuve, Belgium, May, 2002, B. Macq and J.-J. Quisquater (eds.), pp. 225-232 – *Best Poster Award*.

24. L. Batina, C. Jansen "Side-channel entropy for modular exponentiation algorithms", Proceedings of the 24th Symposium on Information Theory in The Benelux, Veldhoven, The Netherlands, May, 2003, L. Tolhuizen (ed.), pp. 37-44.

25. L. Batina, J. Lano, N. Mentens, S. B. Örs, B. Preneel, and I. Verbauwhede, "Energy, Performance, Area versus Security Trade-offs for Stream Ciphers", In State of the Art of Stream Ciphers, Special Workshop hosted by the ECRYPT Network of Excellence, 2004, 9 pages.

26. N. Mentens, L. Batina, B. Preneel, I. Verbauwhede, "Cracking Unix Passwords using FPGA Platforms", In ECRYPT Workshop, SHARCS - Special Purpose Hardware for Attacking Cryptographic Systems, 2005, pp. 83-91.

27. K. Sakiyama, L. Batina, P. Schaumont and I. Verbauwhede, "HW/SW Co-design of TA/SPA-resistant Public-key Cryptosystems", In ECRYPT Workshop, CRASH - Cryptographic Advances in Secure Hardware, 2005, 8 pages.

28. P. Tuyls, L. Batina, "Uncloneable RFID-tags", In ECRYPT Workshop, CRASH - Cryptographic Advances in Secure Hardware, 12 pages.

Lejla Batina received her M.Sc. degree in Mathematics from the University of Zagreb, Croatia in July 1995. She was a teaching assistant and a lecturer at the University of Split, Croatia until 1999. Afterwards she studied and worked as a research assistant at the Technical University of Eindhoven, The Netherlands until 2001. She obtained her MTD degree (Master of Technological Design) in Mathematics for Industry - Postmasters study. Her MTD thesis dealt with power analysis attacks on cryptographic algorithms. In addition, she spent 9 months of her studies as a visiting researcher at the Institute for Experimental Mathematics in Essen, Germany. After graduation she worked as a cryptographer for Pijnenburg - Securealink, in Vught, The Netherlands (later SafeNet, BV) until December, 2003. In March 2002, she started her Ph.D. studies in the COSIC group (Computer Security and Industrial Cryptography) in the Department of Electrical Engineering (ESAT) of the K.U.Leuven and she is a research assistant at the same group since December, 2003.