UCL
Université
catholique
de Louvain

ÉCOLE POLYTECHNIQUE DE LOUVAIN

INSTITUTE OF INFORMATION AND COMMUNICATION TECHNOLOGIES, ELECTRONICS AND APPLIED MATHEMATICS (ICTEAM)

UCL CRYPTO GROUP

Understanding Power in Cryptography

Giacomo de Meulenaer

Thèse présentée en vue de l'obtention du grade de Docteur en Sciences de l'Ingénieur

Jury:

Pr. Jean-Jacques Quisquater	UCL - ICTEAM	*Promoteur*
Pr. Olivier Pereira	UCL - ICTEAM	*Promoteur*
Pr. François-Xavier Standaert	UCL - ICTEAM	*Promoteur*
Pr. Sébastien Canard	Orange Labs	
Pr. Jean-Sébastien Coron	Université du Luxembourg	
Pr. Ingrid Verbauwhede	Katholieke Universiteit Leuven	
Pr. Michel Verleysen	UCL - ICTEAM	*Président*

Novembre 2010

To Catherine.

Acknowledgements

A Ph. D. aims to provide an original contribution to knowledge in one or several research areas. In this noble and challenging quest, cooperative strategies among researchers allow us to overcome the obstacles, while the support of friends and family is necessary to face the unavoidable setbacks. Time has come for me to thank all the persons who contributed in some way to the achievement of my Ph. D.

I am first grateful to Jean-Jacques Quisquater, my first adviser, for welcoming me in the UCL Crypto Group, a research team with a long tradition of excellence in applied cryptography. I also thank François-Xavier Standaert and Olivier Pereira, my other two advisers, for supporting me in the (sometimes painful) writing process of my papers and encouraging me to go to conferences and research meetings, where I met very interesting researchers.

Then my acknowledgments naturally go to my (sometimes numerous) coauthors and co-workers, who shared their knowledge and wisdom with me. Several persons took a particular importance in some of my works: Guerric, François *Le* Gosset *de Soignies*, Christophe, Nidal, Francesco, Sébastien and Iwen. It has been a real pleasure to work with you.

My colleagues of the Crypto Group also deserve my gratitude for creating a stimulating and pleasant atmosphere in the lab. Olivier *Olmar*, François *FK*, Nicolas and Philippe were especially good at this. I retain very good memories of our various team-building activities, especially the ones aiming to "test the network" with Philippe, Nicolas, Mathieu and Julien. My office mates must be acknowledged for supporting me, especially Philippe, with whom it was very rewarding to share an office. Special thanks also go to Sylvie, Christophe and Iwen, for carefully reading and improving the quality of this text.

Finally, I would like to thank my friends and my family for the trust and support they gave me. Lastly, and most importantly, I wish to thank Catherine, my beloved fiancée, for her love, patience and confidence during these four years.

Abstract

The concept of power may be of fundamental importance in many application fields of cryptography. In the context of small embedded devices, of growing interest with the advent of the "information era", it is crucial to make the best use of the device weak resources, especially electrical power and energy. Another example is cryptanalysis which requires a huge amount of computational effort to break cryptographic algorithms. To improve the success probability of the cryptanalytic attempts, it is essential to make the most out of the available computing power. This is even more relevant when the attacks are conducted from a constructive point of view, in order to assess the security level of cryptographic algorithms.

This thesis deals with two contexts of cryptography in which power is of central concern: cryptanalysis with special-purpose hardware and cryptography for low-power embedded devices. The common approach we follow is to carefully understand the available power resources in the studied contexts.

In the first part, we exploit the computational power of Field Programmable Gate Arrays (FPGAs) and Application-Specific Integrated Circuit (ASICs) for practical attacks on public-key cryptosystems. We first propose an improved architecture to implement the Elliptic Curve Method of factoring integers on FPGA. This method is very useful when aiming to break RSA. We then study how to tackle the Elliptic Curve Discrete Logarithm Problem, relying on ASICs, in the special case of Koblitz curves. Our results underline the vulnerability of the 131-bit key size for Elliptic Curve Cryptography.

In the second part, our aim is to decrease the overhead of cryptography for small embedded devices. We first analyze the energy cost of communication and cryptography in wireless sensor networks, allowing the selection of the less consuming protocol to achieve a given cryptographic task. We then study how to adapt group signatures to low-power devices, these signatures being very appealing for privacy-preserving applications. Our cooperative solution makes group signatures tractable for small devices like contactless smart cards.

Contents

Introduction

What could have in common a huge specialized code-breaking machine and a tiny embedded device like a contactless smart card or a wireless sensor node?

Cryptanalysis is the science of recovering encrypted or hidden information. Also known as the practice of breaking codes, it has the reputation of being associated with malicious purposes, as it may provide the mean to break into secure systems or to crack encrypted passwords. It has fortunately another constructive role in the field of cryptography: it is widely used by researchers to evaluate the security of cryptographic algorithms, both from a practical and a theoretical point of view. In particular, practical attacks provide concrete estimates concerning the effort needed to break cryptographic algorithms with the technology available today. They thus refine the existing theoretical security evaluations which rely on many hypotheses, in particular concerning the expected evolution of the computing power.

Typically the complexity of cryptanalytic attempts is proportional to the size of the key. For large key sizes, the huge effort they involve requires gathering the highest possible computational power. In this context, special-purpose hardware is very attractive as it largely surpasses conventional computers in terms of cost-performance ratio. However, its potential for cryptanalysis has not yet been fully explored. For security reasons, it is very important to understand the possibilities brought by these powerful machines: it allows to provide sound and up-to-date security recommendations concerning the cryptographic key sizes.

Small embedded devices, as opposed to code-breaking machines, are much more familiar: they have invaded our everyday life, with laptops, mobile phones, Personal Digital Assistants, etc. In fact, the technological evolution increasingly encourages the use of smaller embedded devices, as with the development of advanced radio-frequency identification (RFID) tags or wireless sensor nodes, of interest in many monitoring applications. This pervasive technology

allows connecting the physical world to our information networks, bringing to reality concepts like the "Internet of things", but in the same time raising privacy issues. Small embedded devices being also intended for sensitive applications, like access control, health surveillance, infrastructure monitoring or micro-payments, they may have to meet a high level of security. However, these devices are by nature very constrained: they have very limited resources, especially energy or electrical power, to dedicate to the implementation of robust defense mechanisms. As a result, reaching a sufficient security level may be a particularly challenging task with these low-power devices.

Having described the contexts surrounding code-breaking machines and small embedded devices, we can attempt to answer the initial question regarding their common characteristics. In fact, both kinds of device try to implement cryptography in the most optimized possible way: they both attempt to make the most out of the available power – computational or electrical – to achieve their respective tasks. Additionally, embedded devices raise the question of security margins. As the use of cryptography is demanding on such platforms, it is very valuable to understand as precisely as possible what security parameters have to be used in which context.

This thesis covers two application fields of cryptography where power is a central concern: cryptanalysis with special-purpose hardware and cryptography for low-power embedded devices. The common approach we follow throughout this thesis is to carefully understand the available power resources in the studied contexts. The thesis is divided into two parts.

First part. The first part of the thesis deals with cryptanalysis using dedicated hardware. The hardware platforms we consider are the Field Programmable Gate Arrays (FPGAs) and Application-Specific Integrated Circuits (ASICs), which are the most powerful available platforms.

The cryptographic algorithms we target are the two most popular public-key cryptosystems, RSA (named after the initials of its inventors) and Elliptic Curve Cryptography (ECC). These fundamental algorithms are used in a wide range of applications, like secure electronic transactions, digital signatures, virtual private networks on the Internet, etc. Our attacks aim at solving the mathematical hard problems on which they rely, i.e., integer factorization and Elliptic Curve Discrete Logarithm Problem (ECDLP) for RSA and ECC respectively.

First, concerning factorization, we study how to improve the performances on reconfigurable hardware (i.e., FPGA) of the Elliptic Curve Method (ECM) for factoring integers. This method indeed tackles most of the computations

of the Number Field Sieve, the best known algorithm for factoring the large composite numbers used in RSA.

Second, for ECDLP, we consider the most preferred option for solving it: the Pollard's rho algorithm. We restrict ourselves to the special case of the Koblitz curves which are particularly attractive in cryptography because of the fast ECC implementations they permit. We study the complexity of an attack relying on ASICs against a Koblitz curve DLP instance with a 131-bit key size. This problem corresponds to the least difficult unsolved challenge of a series with increasing complexity. These challenges were published to stimulate research on ECDLP.

Second part. The second part of the thesis focuses on cryptography for small embedded devices. Our general goal is to decrease the overhead of cryptography for these devices, so that they can handle more advanced cryptographic functionalities currently out of reach, or simply devote their resources to other purposes.

Wireless sensor networks are the technology we are especially interested in. Sensor nodes are small autonomous devices with sensing and processing abilities. As introduced earlier, they are particularly suited for distributed monitoring applications, such as pollution detection or perimeter protection for instance. With respect to high-end RFID tags like contactless smart cards, they have a comparable computing power, so that the conclusions drawn from this point of view can also be extended to this class of devices.

Willing to help constrained devices to implement cryptography, a natural question is whether they can be assisted by other entities in this view. We observe that in many contexts, small embedded devices are surrounded with powerful platforms: a base station for sensor nodes, an RFID reader for the tags, etc. This raises two additional interrogations. On the one hand, to which extent do we need to trust the helping party? On the other hand, is the overall gain offered by the cooperation significant when taking into account the extra communication cost? These questions are at the root of the works presented in the second part of the thesis.

In this part, we first study the energy cost of communication and cryptography in wireless sensor networks. Our analysis makes it possible to precisely assess the relative contributions of computation and communication in the energy consumption of a protocol. It is very helpful to select the less consuming alternative to implement a cryptographic functionality in a given context and network topology. We illustrate this by comparing symmetric and asymmetric techniques for key agreement in sensor networks from an energy point of view.

The second topic we investigate in this part of the thesis addresses the privacy concern regarding low-power devices and more generally the applications where the user's anonymity is strongly desired. We study how to adapt group signatures to small embedded devices like a sensor node or an RFID tag. These signatures have the particularity of allowing a member of a group to produce signatures on behalf of the group; the resulting signatures are anonymous and unlinkable for any verifier, except for a given authority. They are very appealing for specialized applications like anonymous credentials, electronic cash and voting systems. They are unfortunately too complex for weak constrained devices. Therefore, we explore a cooperative approach, in which a signer securely outsources a large part of the computations to a non trusted powerful intermediary to produce a signature. This allows a small embedded device to implement group signatures, thereby generating two main advantages. First, the mobility of users can be supported in privacy-preserving applications relying on group signatures. Second, the security of real-world group signatures can be enhanced by storing the signer's cryptographic secrets into a well-protected device like a smart card, rather than leaving them in a potentially compromised platform like a desktop computer.

Contributions. The main contributions of this thesis can be summarized as follows.

- First, we improve the implementation of the Elliptic Curve Method for factorization in reconfigurable hardware by a factor of 15 in terms of throughput-hardware cost ratio with respect to previous results. The proposed architecture, based on a fully unrolled and pipelined modular multiplier, exploits the power of the embedded multipliers available in modern FPGAs.

- Second, we provide estimates of the complexity of an ASIC-based attack on the ECDLP for the case of Koblitz curves. For the design of the arithmetic operators, we underline the interest of the normal-basis representation, as opposed to previous studies. Most importantly, we show that the "infeasible" ECDLP challenge over $\mathbf{F}_{2^{131}}$ can in fact be solved in a year with about 200 ASICs, costing altogether about 60 000 €, contributing therefore to review the perceived security level for this key size.

- Third, we propose a methodology to assess the real energy cost of cryptography in wireless sensor networks. Our case study on key agreement protocols highlights the importance of the communication cost, especially when the nodes need to stay in listen mode, e.g., between the

protocol rounds. It also compares protocols from both secret-key and public-key cryptography and allows selecting the less consuming option in function of the network topology.

- Finally, we present a cooperative version of a very efficient group signature protocol, XSGS, proposed by Delerablée and Pointcheval at Vietcrypt 2006. The protocol is proved secure in the security model we introduce for cooperative group signatures. Moreover, our implementations on sensor nodes demonstrate its suitability for small embedded devices.

Outline. This thesis is organized in two main parts. The first part is dedicated to cryptanalysis of public-key cryptosystems with special-purpose hardware. Chapter 1 is a preliminary chapter in which we provide some background information concerning the RSA and ECC cryptosystems and the hardware featured in our works, the FPGAs and ASICs. Then, in Chapter 2, we study how to improve the Elliptic Curve Method in reconfigurable hardware. Chapter 3 ends the first part; its subject is the solution of the discrete logarithm problem on Koblitz curves with cryptanalytic machines based on ASICs.

The second part deals with cryptography for low-power devices. We begin with a description of the devices under focus and a brief review of the existing methods to secure them in Chapter 4. Then, in Chapter 5, we examine the energy cost of communication and cryptography in wireless sensor networks. Finally, in Chapter 6, we study how to adapt group signatures, of interest in many privacy-preserving applications, to small embedded devices.

Part I

Cryptanalysis Powered by Dedicated Hardware

CHAPTER 1

Mathematical and Hardware Background

The first part of the thesis deals with cryptanalysis of public-key cryptosystems using dedicated hardware. In this preliminary chapter, we provide the necessary mathematical and hardware background on the topic. We first introduce public-key algorithms and the mathematical hard problems on which their security relies. We then describe some of the various hardware platforms that are available when performing real-world cryptanalytic attempts, focusing on the Field Programmable Gate Arrays (FPGAs) and Application-Specific Integrated Circuits (ASICs), which are the devices considered in the attacks presented in the next two chapters.

1.1 Introduction

Automated Teller Machine, online banking and electronic commerce are some helpful services that most of us regularly employ in our everyday life. They have the common feature of requiring a high level of security in a rather insecure environment. They are however implemented in a simple, flexible and quick manner, which looks like magic to the ones who have known the ancient ways of performing these tasks. This magic has been made possible by the apparition of public-key cryptography.

Introduced in the seventies by Diffie, Hellman and Merkle, public-key cryptography has the remarkable property of allowing the establishment of a common secret key between two entities which have not previously exchanged any secret. It offers other interesting advantages, such as providing very practical digital signatures, and is therefore widely used in practice.

The security of the most common public-key cryptosystems rests on the

intractability of well-studied mathematical hard problems. The size of the cryptographic keys is the parameter which controls the complexity of such problems in practice: the larger the key, the harder it is for an adversary to recover it with generic attacks. However, for performance reasons, it is generally advised to use the shortest key length ensuring a sufficient security level.

To guide the choice of implementors, several recommendations regarding the key lengths exist [GB], either based on a theoretical analysis [LV99], or issued by famous authorities in the field of cryptography: e.g., the US National Institute of Standards and Technology (NIST) or the European Network of Excellence in Cryptology (ECRYPT [ECR10]). These recommendations attempt to predict how long a given key length will remain secure against the most efficient attacks. To capture the effect of both the technological evolution and the improvements in the cryptanalytic algorithms, they need to be regularly updated, in particular on the basis of practical studies on the effort required to break a given security level with state-of-the-art techniques. This is what we do in the first part of the thesis, focusing on the attacks of public-key cryptosystems with special-purpose hardware.

In this preliminary chapter, we begin with a short presentation of the two most used public-key cryptosystems, RSA and Elliptic Curve Cryptography (ECC), and their underlying hard mathematical problems (Section 1.2). Our aim is to introduce these algorithms, as they will be the target of our hardware-based attacks in Chapter 2 and Chapter 3. We then briefly review the hardware arsenal available to the adversary to implement a powerful cryptanalytic effort (Section 1.3). We give more details for the Field-Programmable Gate Array (FPGA) and Application-Specific Integrated Circuit (ASIC), as they are the platforms considered in the first part of the thesis.

1.2 Public-Key Cryptosystems

Proposed in the mid-seventies, public-key cryptography aims to palliate the problem of key distribution and management in secret-key cryptography. It relies on information which is public – but authenticated– to set up secure and confidential communications. Each user possesses a public and a private key. Based on the public key, any user can encrypt a message which can only be decrypted by the corresponding secret key. Deducing the secret key from the public information is assumed to be computationally infeasible: this problem is related to the solution of a well-studied mathematical hard problem. With digital signatures and certificates, public-key cryptography also provides the mean to ensure the authenticity of the public keys. In this section, we briefly

describe the RSA and ECC public-key cryptosystems, and their underlying mathematical problems.

1.2.1 RSA

RSA was invented by Rivest, Shamir and Adleman in 1977 and has been the most successful public-key cryptosystem for the past 30 years. To implement the public-key main functionalities, it relies on the following basic principle.

Principle. The RSA basic property is described with the example of encryption. Let m be an integer $0 \leq m < n$ representing a message to encrypt and n being an RSA modulus, i.e., a product of two large secret primes p and q. To encrypt the message, the following equation is applied:

$$m^e \mod n = c$$

where e is the encryption exponent and c is the ciphertext. The pair (e, n) represents the RSA public key. The decryption of c is done with

$$c^d \mod n = m$$

where d is the RSA secret key. To enable the RSA encryption scheme, the triple (e, d, n) must be chosen as follows. Let $\phi(n)$ be equal to $(p-1)(q-1)$. Then, e and d must be taken in $1 < e, d < \phi(n)$ with $gcd(e, \phi(n)) = 1$ and must satisfy $ed = 1 \mod \phi(n)$. With these relations, the Fermat–Euler theorem (i.e., $a^{\phi(n)} = 1 \mod n$) ensures

$$c^d \equiv m^{ed} \equiv m^{1 \bmod \phi(n)} \equiv m \pmod{n}.$$

The RSA functionalities, such as the RSA signing scheme, are built from this property. We will not detail them here (for this, consult [MVO96]); instead, we focus on the factorization problem on which RSA security is based.

Factorization problem. In the above RSA encryption, if one is able to factorize n into p and q, he can obtain $\phi(n)$ and thus recover the secret key $d = e^{-1} \mod \phi(n)$. The RSA cryptosystem is thus protected by the factorization problem, which is known –at least believed– to be intractable for sufficiently large RSA moduli. The size of n is typically 1024 or 2048 bits, although the security of 1024-bit RSA keys is currently being questioned. The larger generic RSA modulus ever factored was announced by Kleinjung et al. [KAF+10] in December 2009 for a 768-bit modulus. This factorization

record was obtained using clusters of computers from academic partners running for two and a half years. Note that Kleinjung et al. assess the effort of breaking a 1024-bit RSA key as high as 1000 time the task for the 768-bit length.

The Number Field Sieve (NFS) is the most efficient algorithm to factorize an RSA modulus. We briefly introduce it, as we will study in the next chapter a hardware architecture to speed up the NFS computations. The NFS is an improvement of the simpler rational sieve algorithm that we describe. Suppose that we know how to express the integers z and $z + n$ as products of small prime powers of the form $\prod p_i^{e_i}$. This gives us an equality $\bmod n$ between the two products. By combining such equalities, we can obtain a relation with even exponents of the form $a^2 \equiv b^2 \pmod{n}$, leading to the factorization of $n = gcd(a - b, n) \cdot gcd(a + b, n)$.

Most of the NFS computation (perhaps 90 %) is spent while finding the relations. In this sieving step, a very large amount of mid-sized numbers (100 – 200 bits for a 1024-bit n) need to be tested whether they are *smooth*, i.e., decomposable in powers of primes below a given bound. A very efficient algorithm to perform this task is the Elliptic Curve Method, which will be presented in Section 2.3. The interested reader is referred to [Pom96] for a broader introduction to the NFS and to [LHWL93] for the details.

1.2.2 Elliptic Curve Cryptography

The use of elliptic curves in cryptography was proposed independently by Miller and Koblitz in 1985. With respect to RSA, Elliptic Curve Cryptography (ECC) has the advantages of allowing faster operations and the use of smaller key sizes for an equivalent security level. It is thus increasingly popular and becoming the de facto standard public-key cryptosystem in place of RSA. It is famous for its efficient way of achieving key exchange and signature algorithms, with Elliptic Curve Diffie–Hellman (explained in Chapter 5) and the Elliptic Curve Digital Signature Algorithm. Our intention here is not to detail the schemes allowed by ECC (for this, consult [HMV04]), but rather to introduce the mathematical background of elliptic curves and the hard problem on which the security of ECC is based.

Elliptic curves. Elliptic curves provide the group structure on which ECC is constructed. The group elements are the points on the curve, i.e., the solutions of the elliptic curve equation, which is defined below.

Let $\mathbf{F}_p$ be a finite field with characteristic p represented by the set of integers $\{0, 1, \ldots, p - 1\}$ with addition and multiplication (mod p). An elliptic

curve E over $\mathbf{F}_p$ is defined as a set of points $(X, Y) \in \mathbf{F}_p \times \mathbf{F}_p$ satisfying the following equation, together with an additional point, the point at infinity $\mathcal{O}$:

$$y^2 = x^3 + ax + b$$

where $a, b \in \mathbf{F}_p$ and $4a^3 + 27b^2 \neq 0$. The elliptic curve E together with an addition law admits a group structure where the point at infinity $\mathcal{O}$ is the neutral element. The addition of two points $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$ (assuming that $P, Q \neq \mathcal{O}$) gives $R = (x_R, y_R)$ through the following computations:

$$\begin{cases} x_R &= \lambda^2 - x_P - x_Q \\ y_R &= \lambda(x_P - x_R) - y_P \end{cases} \text{ with } \lambda = \begin{cases} (y_Q - y_P)/(x_Q - x_P) & \text{if } P \neq \pm Q \\ (3x_P^2 + a)/2y_P & \text{if } P = Q \end{cases}$$

The definition of $\mathcal{O}$ solves the problem in the computation of λ when P is added to its inverse $-P = (x_P, -y_P \mod p)$. The result is defined as $P + (-P) = \mathcal{O}$. Adding k times the same point P is an operation known as the scalar (or point) multiplication and denoted kP. It is the ECC basic operation, roughly corresponding to an exponentiation in RSA.

Apart from the affine coordinates (x, y), other coordinate systems can be used to represent the points, the simplest being the projective coordinates. In this system, the triple $(x : y : z)$ represents $(\frac{x}{z}, \frac{y}{z})$ in affine coordinates and $(0 : 1 : 0)$ is the point at infinity $\mathcal{O}$. The main advantage of projective coordinates consists in avoiding the costly modular inversion at each point addition or doubling.

Binary fields. Besides the prime fields, elliptic curves can be defined over binary fields $\mathbf{F}_{2^n}$. These fields contain 2^n elements which can be represented by polynomials of degree $n-1$ having binary coefficients. The field operations are performed modulo an irreducible binary polynomial of degree n. The curve equation is now of the form:

$$y^2 + xy = x^3 + ax^2 + b$$

with $a, b \in \mathbf{F}_{2^n}$. We do not detail here the formulae for point addition and doubling. For this, the reader is referred to [HMV04], which provides a thorough description of elliptic curves.

There exists a special kind of binary elliptic curves which allows especially efficient implementations: Koblitz curves. These anomalous curves have a curve equation of the form:

$$y^2 + xy = x^3 + ax^2 + 1$$

with this time $a \in \mathbf{F}_2$ while the points (x, y) are defined in $\mathbf{F}_{2^n} \times \mathbf{F}_{2^n}$. The importance of these curves in cryptography comes from a very fast point multiplication algorithm exploiting the Frobenius automorphism of the field extension $\mathbf{F}_{2^n}/\mathbf{F}_2$, which maps a field element to its square (more details will be provided in Section 3.2.3). Because of this advantage, there are five Koblitz curves among the fifteen standardized elliptic curves [Qu99], the rest being equally divided into curves over prime fields and random binary curves.

Elliptic Curve Discrete Logarithm Problem. Elliptic curve cryptographic schemes are secure provided that the Elliptic Curve Discrete Logarithm Problem is intractable. This problem consists in finding the scalar k, given the points P and $Q = kP$. It is believed to be especially hard to solve (for well-chosen parameters), although there is no mathematical proof showing the non-existence of efficient algorithms for solving this problem. The confidence in the hardness of ECDLP comes from the fact that it has been extensively examined by the research community since the proposition of ECC.

Currently, the best algorithms for solving an ECDLP are the generic Pollard's rho [Pol78] and Shanks' Baby-step Giant-step [Sha71] algorithms. They have a time complexity in $O(\sqrt{l})$, where l is the elliptic curve group size and the complexity basic operation is roughly a single group operation like a point addition. The Pollard's Rho method is generally the most preferred option for attacking the problem, because of its lower memory costs. It will be described in Chapter 3, where an attack of ECDLP relying on ASICs will be presented.

As for RSA, sufficiently large secret keys ensure the intractability of the ECC underlying hard problem. Currently, the largest ECDLP instance solved was for a 112-bit key for a curve over a finite field [BKK+09]. For practical applications, a key length around 160 bits appears sufficiently large according to the recommendations of [GB], at least until end 2010. Until end 2009, key lengths around 130 bits were still believed to be unreachable for the computing power available today; however, our work presented in Chapter 3 contributed to review this statement [Cer97].

1.3 Hardware Platforms for Cryptanalysis

Besides studying how to hide information, cryptography also covers the techniques for recovering the meaning of encrypted information, i.e., *cryptanalysis*.[1] Also known as the art of breaking codes, cryptanalysis is famous for

[1] Although the term *cryptology* is sometimes employed to refer to both cryptography and cryptanalysis, thus limiting cryptography to the information hiding part.

Figure 1.1: The cluster of 214 PlayStation 3 consoles of the Laboratory for Cryptologic Algorithms of EPFL.

its crucial role in World War II, as it was successfully employed by the allies to decrypt the German communications, by breaking the Enigma cipher machine. Today, cryptanalysis is largely employed by researchers to investigate the security of cryptographic algorithms and highlight any potential weakness.

Although clever engineering and pen and paper techniques could be sufficient in the ancient and glorious times of cryptanalysis, a considerable amount of computing power is definitely needed when aiming to break modern cryptographic methods. In fact, any increase in the computing abilities can be in general translated into a somewhat larger success probability for the cryptanalytic attempt.

In this section, we are interested in the hardware platforms which can be used to power cryptanalysis. After a quick overview of various possible devices, we focus on the Field Programmable Gate Array (FPGA) and Application Specific Integrated Circuit (ASIC), which are featured in the next two chapters.

Performance vs flexibility. Traditionally, large cryptanalytic efforts used to resort to a pure software-based computing power with the use of numerous clusters of conventional computers, as for instance in [Har00]. However, standard computers give up some efficiency for being highly flexible: more specialized platforms outperform them for very specific tasks like cryptanalytic efforts.

The most efficient hardware platform is the ASIC, which is especially designed to achieve a given functionality: all its resources are dedicated to the task. There are many other intermediate platforms between the two extreme software and ASIC solutions, achieving different tradeoffs between performance and flexibility: FPGAs, Digital Signal Processors (DSPs), Graphics Processing Units (GPUs) or improved processors like the IBM Cell Broadband Engine [IBM10] ...

The potential for cryptanalysis of many of these platforms was underlined in various papers, e.g., [GKN+08, BBB+09, BCC+09]. Several research groups have even constructed hardware engines dedicated to cryptanalysis, like the COPACOBANA (Cost-Optimized PArallel COde Breaker [GKN+08]) based on FPGAs or the Playstation 3 cluster[2] of the Swiss EPFL [LAC] (see Figure 1.1), employed in many successful cryptanalytic efforts.

1.3.1 FPGA

Invented in 1984, the Field Programmable Gate Array is an integrated circuit which can be (re)configured by the user for any specific application. It represents an interesting compromise between the performances reached by static integrated circuits and the flexibility offered by configurable processors. With respect to ASIC, it provides a shorter time to market and lower development costs, together with a long-term performance as an FPGA design can be migrated to a newer FPGA without too much effort. This platform is also employed to quickly prototype a hardware circuit.

An FPGA is made of programmable logic blocks, which can together implement any functionality depending on how they are configured and interconnected. The interconnections are managed through programmable switches. The configuration of the whole FPGA (logic blocks, interconnections, inputs and outputs...) is stored in a memory (e.g., a SRAM). Besides clusters of logic gates, an FPGA is usually equipped with some specialized features, such as RAM blocks, distributed RAM, embedded multipliers or even microprocessors.

Virtex-4 FPGA. FPGAs can be divided in several families, depending on the available resources and overall performances. For instance, the FPGAs of the Xilinx provider are separated into two families: the high-performance Virtex series and the low-cost Spartan series. In this thesis, we consider the use of Virtex-4 FPGAs; we therefore detail here their main features.

[2] This game console is the major application of the Cell processor.

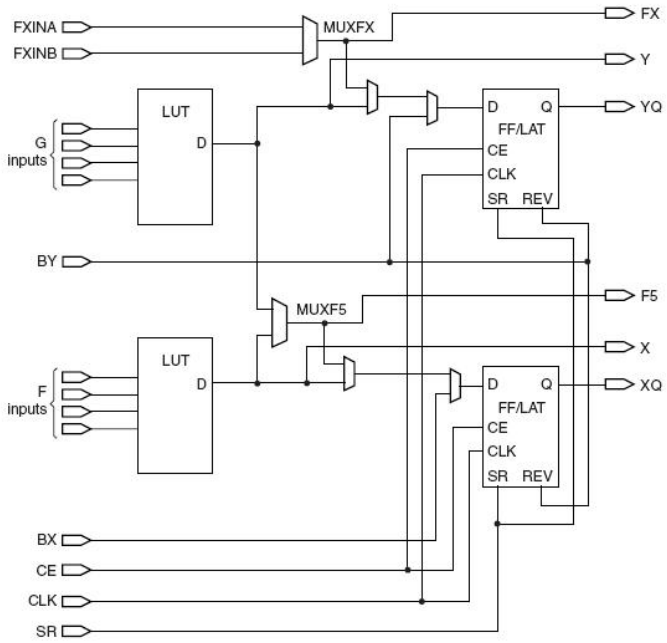


Figure 1.2: Virtex-4 slice (simplified).

The Virtex-4 slice, i.e., the basic elements of its configurable logic blocks, is shown in Figure 1.2. It mainly contains two 4-input Look-Up Table (LUT) to implement arbitrary boolean functions, two storage elements, some multiplexers, a carry chain and a few additional logic gates. A Virtex-4 embeds between 10 000 and 100 000 such slices to implement arbitrary designs.

Besides the slices, the Virtex-4 is also equipped with up to 512 dedicated Digital Signal Processing (DSP) blocks in the form of DSP48 slices (see Figure 1.3). With respect to the previous Virtex family, this is a novelty as these blocks not only achieve two's complement 18-bit x 18-bit multiplications, but they can be configured in many modes, like multiply-and-accumulate or multiply-and-add. The DSP48 are cascadable in order to implement high-speed DSP applications. Memory is also available in the Virtex-4 with up to 552 dual-port 18 kb configurable RAM blocks. The slice storage elements can also be used as distributed RAM. For more information about the Virtex-4 FPGA and the DSP48, we refer to the device datasheet and user guide.

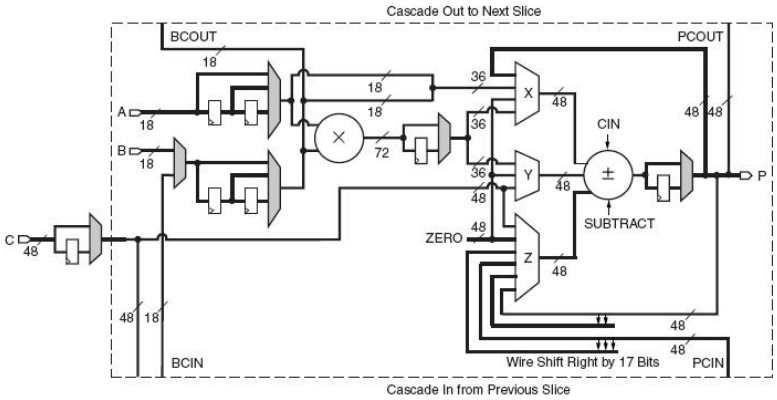


Figure 1.3: Virtex-4 DSP48 slice.

FPGA cost. When assessing the security of cryptographic algorithms with practical cryptanalytic attacks, an interesting complexity measure is the amount of money necessary to break the algorithm within a given period, like in a year. For this, the hardware cost must be well established. For FPGAs, the prices are very dependent on the volume and customer size. Although price lists can be found in the website of FPGA distributors, those figures represent upper limits: in practice, most customers are likely to pay considerably less, by negociating the prices with the distributors. In this work, the FPGA prices correspond to a pricing proposition made to the UCL in 2007. Each time an FPGA price is given, we indicate the volume to which it refers.

To further reduce the costs, Xilinx has proposed the EasyPath solution [Xil10], which consists in limiting the FPGA to a single design, i.e., sacrificing its reconfigurability. It is a mean for Xilinx to sell defective FPGAs which cannot be fully reconfigurable but still work in certain configurations. The price reduction per FPGA is significant, but a fixed cost must be taken into account to support the Non Recurring Engineering (NRE).

For real attacks gathering a large number of FPGAs, the cost overhead of assembling the FPGAs into clusters must be taken into account. Therefore, a more realistic cost estimate can be achieved on the basis of the purchase cost of clusters of FPGAs, such as the COPACOBANA engine (see Figure 1.4). This low-cost alternative to commercial supercomputers has been developed at the Ruhr University of Bochum to support cryptanalytic efforts [GKN+08]. Gathering the computing power of 120 FPGAs, its cost is amounted to about US$ 10 000, in the case of low-cost Spartan-3 XC3S1000 FPGAs costing each around US$ 35 (per 1000 devices).

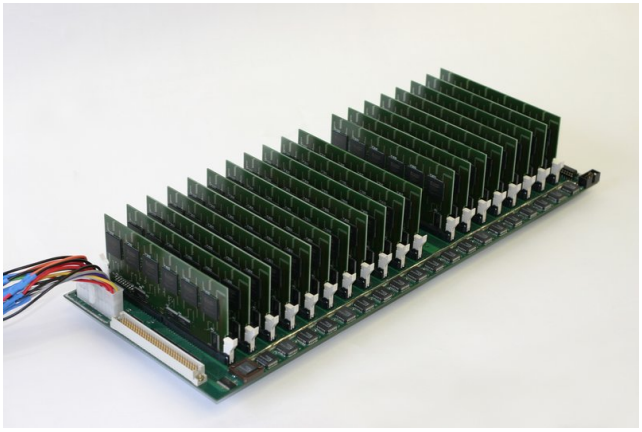


Figure 1.4: The first prototype of COPACOBANA, hosting 120 Spartan-3 XC3S1000 FPGAs.

1.3.2 ASIC

Being static circuits, ASICs are fully customized and thus feature the best possible performance and lowest area cost for a given functionality. Their main drawback is their long and costly development; because of this, their use is in general limited to applications involving high-volume productions. ASICs can be of three main kinds, depending on the level of customization. The first is the gate arrays: the gates type and position are predefined and the functionality is implemented by defining the interconnections. The second is the standard cell design: the circuit can be fully customized, but using standard gates (e.g., *nand*, *nor* and *xor* gates) from a given library. The third removes this abstraction level by allowing the ASIC designer to optimize the circuit at the transistor level: the full custom designs, requiring highly skilled designers.

Standard cell design. In this thesis, we consider ASICs designed with the popular standard cell methodology. The resulting ASICs can contain up to several million gates. Their conception flow comprises very roughly the following steps, which appear with some variations in the design flow of other platforms, like FPGAs:

- Synthesis: the hardware description of the circuit, written in VHDL for instance, is translated into a netlist, which depicts the circuit with gates from the standard-cell library and interconnections.
- Placement: this step decides the position of the gates within the circuit

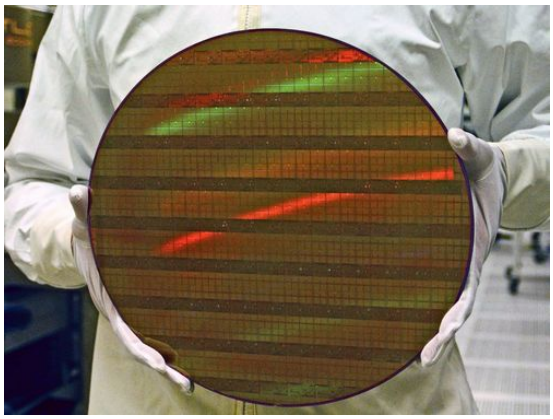


Figure 1.5: Intel wafer of 300 mm diameter for 45 nm technology.

core area.

- Routing: finally, the wires connecting the circuit elements are added in such a way that the user constraints and the rules for a safe design are met. This last design step can be computationally intensive and might fail if the constraints are too severe.

IC fabrication. Before detailing the cost of an ASIC, we have to introduce the manufacturing process of integrated circuits. The physical layout resulting of the place and route design steps is used to build a set of photomasks, denoted by the *mask*, which define the circuit layers. The circuit is then constructed on a slice of silicon, the *wafer* (cf. Figure 1.5), from the mask by applying various complex techniques, like photolithography, etching and ion implantation. A single wafer can produce many identical circuits by applying several times the same mask. It is then cut into small pieces, the *dies*, which contain each a copy of the circuit. The dies must then be packaged before the integrated circuits can be used.

The number of gates per area unit is an important factor in the production of IC; it is determined by the semiconductor process technology, usually designated by the transistor channel length. Typical technologies are the 90 nm, 65 nm and 45 nm processes.

ASIC cost. As for FPGAs, the cost of the ASIC plays an important role in the cost estimate of an attack exploiting this platform. We have already reported the very high Non Recurring Engineering cost of the ASIC; it might

exceed US$ 500 000 when using the most recent technologies. By contrast, the variable cost is relatively low, such that ASICs become cheaper than other integrated circuits for large quantities: the break-even point between ASICs and FPGAs is roughly between 10 000 and 100 000 units. This is why the latter kind of platforms is generally considered for volumes inferior to this point.

Nevertheless, there exists a mean to obtain a few ASIC prototypes at a moderate cost: the Multi-Project Wafer (MPW), in which several designs from private firms, students and researchers are manufactured together using the same mask and wafer. These projects allow to share the NRE among independent customers. Since most of this cost is associated with the production of the mask, MPWs reduce the mask cost per chip. In a MPW, the manufacturer typically produces two wafers (for process reliability reasons), from which 60-120 dies can be obtained, for a total cost of € 60 000.[3]

[3]Frank K. Gurkaynak (ETH Zurich Microelectronics Design Center) is acknowledged for this estimate.

Bibliography

[BBB+09] Daniel V. Bailey, Brian Baldwin, Lejla Batina, Daniel J. Bernstein, Peter Birkner, Joppe W. Bos, Gauthier van Damme, Giacomo de Meulenaer, Junfeng Fan, Frank Gurkaynak, Tim Gneysu, Thorsten Kleinjung, Tanja Lange, Nele Mentens, Christof Paar, Francesco Regazzoni, Peter Schwabe, and Leif Uhsadel. The Certicom Challenges ECC2-X. In *Workshop on Special Purpose Hardware for Attacking Cryptographic Systems (SHARCS'09)*, 9 2009.

[BCC+09] Daniel J. Bernstein, Tien Chen, Chen Cheng, Tanja Lange, and Bo yin Yang. ECM on Graphics Cards. Cryptology ePrint Archive, Report 2008/480, January 2009. `http://eprint.iacr.org/`.

[BKK+09] Joppe W. Bos, Marcelo E. Kaihara, Thorsten Kleinjung, Arjen K. Lenstra, and Peter L. Montgomery. On the security of 1024-bit rsa and 160-bit elliptic curve cryptography. Cryptology ePrint Archive, Report 2009/389, 2009. `http://eprint.iacr.org/`.

[Cer97] Certicom. Certicom Elliptic Curve Cryptosystem Challenge. `http://www.certicom.com/images/pdfs/cert_ecc_challenge.pdf`, 1997.

[ECR10] ECRYPT II. Yearly Report on Algorithms and Keysizes (2009-2010). Available at `http://www.ecrypt.eu.org/documents.html`, 2010.

[GB] Damien Giry and Philippe Bulens. Cryptographic Key Length Recommendation. `http://www.keylength.com`.

[GKN+08] Tim Güneysu, Timo Kasper, Martin Novotný, Christof Paar, and Andy Rupp. Cryptanalysis with COPACOBANA. *IEEE Trans. Comput.*, 57(11):1498–1513, 2008.

[Har00] Robert Harley. Elliptic Curve Discrete Logarithms Project. `http://pauillac.inria.fr/~harley/ecdl/`, 2000.

[HMV04] D. Hankerson, A. Menezes, and S. Vanstone. *Guide to Elliptic Curve Cryptography*. Springer, 2004.

[IBM10] IBM. Cell Broadband Engine. `https://www-01.ibm.com/chips/techlib/techlib.nsf/products/Cell_Broadband_Engine`, September 2010.

[KAF+10] Thorsten Kleinjung, Kazumaro Aoki, Jens Franke, Arjen Lenstra, Emmanuel Thom, Joppe Bos, Pierrick Gaudry, Alexander Kruppa, Peter Montgomery, Dag Arne Osvik, Herman te Riele, Andrey Timofeev, and Paul Zimmermann. Factorization of a 768-bit RSA modulus. Cryptology ePrint Archive, Report 2010/006, 2010. `http://eprint.iacr.org/`.

[LAC] LACAL. Laboratory for Cryptologic Algorithms, within the Ecole Polytechnique Fédérale de Lausanne (EPFL). http://lacal.epfl.ch/.

[LHWL93] Arjen K. Lenstra and Jr. Hendrik W. Lenstra. *The development of the number field sieve*, volume 1554 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 1993.

[LV99] Arjen K. Lenstra and Eric R. Verheul. Selecting Cryptographic Key Sizes. *Journal of Cryptology*, 14:255–293, 1999.

[MVO96] Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. *Handbook of Applied Cryptography*. CRC Press, Inc., Boca Raton, FL, USA, 1996.

[Pol78] John M. Pollard. Monte Carlo methods for index computation (mod p). *Mathematics of Computation*, 32:918–924, 1978.

[Pom96] Carl Pomerance. A tale of two sieves. *Notices of the American Mathematical Society*, 43:1473–1485, 1996.

[Qu99] Minghua Qu. Standards for efficient cryptography SEC 2: Recommended Elliptic Curve Domain Parameters. Available online at `http://www.secg.org/collateral/sec2.pdf`, 1999.

[Sha71] Daniel Shanks. Class Number, a Theory of Factorization, and Genera. In *1969 Number Theory Institute (Proc. Sympos. Pure Math., Vol. XX, State Univ. New York, Stony Brook, N.Y., 1969)*, pages 415–440. Providence, R.I., 1971.

[Xil10] Xilinx. EasyPath FPGAs. http://www.xilinx.com/products/easypath/, September 2010.

CHAPTER 2

Improving the Elliptic Curve Method on Reconfigurable Hardware

Currently, the best known algorithm for factorizing a modulus of the RSA public key cryptosystem is the Number Field Sieve. One of its important phases usually combines a sieving technique and a method for checking smoothness of mid-size numbers. For this factorization, the Elliptic Curve Method (ECM) is an attractive solution. As ECM is highly regular and many parallel computations are required, hardware-based platforms were shown to be more cost-effective than software solutions.
The few previous works dealing with implementation of ECM on FPGA were all based on bit-serial architectures. They used only general-purpose logic and low-cost FPGAs, which appeared as the best performance/cost solution. In this chapter, we explore another approach, based on the exploitation of embedded multipliers available in modern FPGAs and the use of high-performance FPGAs.
The proposed architecture – based on a fully parallel and pipelined modular multiplier circuit – exhibits a 15-fold improvement in terms of throughput/hardware cost ratio with respect to the previously published results.

2.1 Introduction

With the advent of public key cryptography, many useful functionalities appeared such as digital signature, public key encryption, key agreement, ... For those needs, the most deployed scheme remains RSA, co-invented in 1977 by

R. Rivest, A. Shamir and L. Adleman [RSA78]. The security of this cryptosystem relies on the intractability of the factorization of big composite integers. This mathematical hard problem experiences therefore a renewed interest. It is believed that 1024-bit RSA keys are sufficiently big to sustain today's attacks, although they are not recommended for applications requiring a long-term protection [GB].

Currently, the best known algorithm for factorizing an RSA modulus is the Number Field Sieve (NFS), introduced by Pollard in 1991. As sketched in Section 1.2.1, it is composed of a sieving and a matrix step, the former being the most expensive part for 1024-bit keys [GSST05].

The work presented in this chapter focuses on the sieving step and more precisely on the relation collection step, which represents about 90 % of the overall NFS computations. This task is usually performed by a combination of a sieving technique and a method for factorizing mid-size numbers [Kle06]. While those numbers are easily factorizable, the challenge lies in the amount of computation: the factorization of 10^{14} 125-bit numbers is required in the case of a 1024-bit modulus (with the parametrization proposed in [FKP$^+$05]). For this task, the Elliptic Curve Method (ECM) appears as an attractive solution.

Up to now, the best successful factorization attempt was for RSA-768, achieved in October 2009 by Kleinjung et al. [KAF$^+$10]. Before that, RSA-200 (663-bit) and RSA-640, were solved in 2005 by Bahr, Boehm, Franke and Kleinjung. These successful efforts were obtained by gathering the power of hundreds of standard computers. Besides those software-based solutions, propositions related to special purpose hardware came out to lower the cost of both machine and power consumption. Such proposals in the context of the NFS sieving step are the TWINKEL device, mesh-based sieving, TWIRL and SHARK (see [Sha05] for an overview and references). ECM is one of the building block of SHARK [FKP$^+$05] and it was proposed to combine it with TWIRL in [GJK$^+$06].

For the design of a hardware computing engine, a platform has first to be chosen. The two main possibilities are ASICs (Application-Specific Integrated Circuits) or FPGAs (Field Programmable Gate Arrays), which were both introduced in the preceding chapter. While ASICs have potentially the highest performance per transistor and the lowest power consumption, the achievement of a real and working device can be a long and very expensive process. FPGAs have the advantage of adding an extra layer of abstraction during the design. This results in low non-recurring engineering (NRE) costs and a low development-production time. Together with their increasing area & speed and falling prices & power consumption, FPGAs are highly prized in the context of cryptanalysis. Even if it appears that taking all the costs into account, ASICs

provide a less expensive solution than FPGAs, these are at least an interesting development platform. Additionally, FPGAs make also more sense in the context of a commercially-based adversary as they can be purchased without specifying their real purpose.

In this chapter, we propose a novel architecture to implement ECM in FPGA. Our motivation is to assess the cost of an ECM processor as a support for the factorization of a 1024-bit RSA modulus with the NFS algorithm. With respect to previous works, a different approach is chosen. In this work, we rely on high-performance FPGAs and especially take advantage of their powerful embedded multipliers. The work presented in this chapter was done in collaboration with François Gosset and Guerric Meurice de Dormale and was published in 2007 in [dMGMdDQ07a, dMGMdDQ07b].

The chapter is structured as follows: Section 2.2 presents the previous works on hardware implementation of ECM and explains our contribution. Then, Section 2.3 reminds the mathematical background of ECM. The choices about the arithmetic circuits necessary for implementing ECM are discussed in Section 2.4. The proposed ECM architecture stands in Section 2.5 and implementation results together with a quick cost assessment for factorizing 1024-bit modulus are provided in Section 2.6 and 2.7. Finally, a comparison with related works is given in Section 2.8.

2.2 Previous Works and Contribution

The first published hardware implementation of ECM was proposed by Pelzl, Šimka et al. in 2005 [PŠK+05]. The aim of this circuit was to check the smoothness of sieving reports of the SHARK device [FKP+05]. It is formed by a collection of parallel ECM units in a Xilinx FPGA together with an external ARM microcontroller. Each unit embeds a memory, a controller and an ALU able to perform addition/subtraction and radix-2 Montgomery multiplication [Mon85]. Carry propagate adders were chosen. An improved pipelined version of this design was later used in [GJK+06].

This proof of concept was deeply improved by Gaj et al. in 2006 [GKB+06]. They removed the external microcontroller and improved the operating frequency and the Montgomery multiplier. They used two multipliers and one adder/subtracter in parallel in each ECM unit. Carry save adders were chosen. Their conclusion was that, compared with high-performance FPGAs and general purpose processors, low-cost FPGAs were the best choice for implementing ECM.

Contribution. The common feature of all the previous works is the use of a bit-serial by parallel architecture. All the arithmetic circuits are built with the general purpose logic of FPGAs.

Modern FPGAs have interesting features including embedded multipliers (cf. Section 1.3.1). It makes therefore sense to try to exploit them. For instance, Virtex-4SX devices embed 128, 192 or 512 DSP48 blocks (depicted in Figure 1.3) able to perform 17×17 unsigned multiply and accumulate/shift-and-add. Their low-cost counterparts, the Spartan-3 devices, are equipped with up to 104 17×17 unsigned multipliers. Hardware cost plays an important role here and fortunately, high-performance devices like Virtex-4 are no longer much more expensive than low-cost FPGAs such as Spartan-3. Together with their increased speed, enhanced functionalities of DSP blocks and their number of multipliers per US$, Virtex-4SX are therefore potentially the sweet spot for the ECM application.

Our contribution is to show that building the core component – the modular multiplier – with fully pipelined embedded multipliers of high-performance FPGAs leads to a highly efficient ECM processor. Our resulting implementation shows an improvement of an order of magnitude with respect to the low-cost solution presented in [GKB$^+$06] in terms of cost/performance ratio. To exhibit the full power of this approach, a parallel architecture with a throughput of 1 multiplication per clock cycle was chosen.

2.3 Elliptic Curve Method

The Elliptic Curve Method is a probabilistic method for integer factorization which uses elliptic curves. It is the best known method to factorize mid-sized numbers (up to a few hundred bits) together with the Multiple Polynomial Quadratic Sieve (MPQS) [Bre00]. ECM seems to be the best choice for a hardware implementation since it is highly regular, not too I/O intensive and requires many parallel computations [PŠK$^+$05].

Principle. The elliptic curve method for integer factorization was invented by Lenstra [Len87]. It is an improvement of the $p-1$ method of Pollard. Its description follows [Mon87].

The $p-1$ method tries to find a prime factor p of n. The first phase of the algorithm selects an integer a and computes $b = a^k \mod n$ with $k > 0$ divisible by all powers of primes below a bound B_1. If $p-1$ divides k, then p divides $b-1$ by Fermat's little theorem ($a^{p-1} = 1 \mod p$ if $\gcd(a,p) = 1$). Let $d = \gcd(b-1, n)$. If $1 < d < n$, then d is a factor of n. If $d = n$, k must

be reduced. The first phase fails if $d = 1$. In this case, the computation can be continued with the second phase. For this purpose, a second bound $B_2 \gg B_1$ is selected and it is assumed that $p - 1 = qs$, where q divides k and s is a prime between B_1 and B_2. If $t \equiv b^s - 1 \equiv a^{ks} - 1 \pmod{n}$ then p divides t by Fermat's little theorem, since $p - 1$ divides ks. The problem is to find s, and hence p.

The $p - 1$ method can be seen as an attempt to find the neutral element (i.e., 1) of the multiplicative group of the nonzero elements of $\mathbf{F}_p$. It fails if $p-1$, i.e., the number of elements (group order), cannot be written as a product of prime powers smaller than B_1 and at most one prime between B_1 and B_2. ECM avoids this by considering the group of an elliptic curve over $\mathbf{F}_p$, where the number of elements randomly varies. As a result, if the method fails with one curve, another curve can be chosen with hopefully a group order satisfying the required property.

Algorithm. In the ECM algorithm (Algorithm 1), the computations are done modulo n as if $\mathbf{F}_n$ was a field. If the point computed at the end of Phase 1 is the point at infinity $\mathcal{O}$ of E over $\mathbf{F}_d$ (d being a prime factor of n), then, in projective coordinate (cf. Section 1.2.2), its coordinate z equals $0 \mod d$ and $\gcd(z, n)$ can lead to d.

If the point multiplication $Q = kP \neq \mathcal{O}$, k might miss just one large prime factor to divide the group order (as in Pollard $p - 1$ method). Phase 2, also called continuation, tries every prime p in the range $[B_1, B_2]$. There are several ways to perform this task [Mon87]. The standard continuation was chosen here and seems to be the most suitable continuation for a hardware implementation. Its description follows [GKB+06].

The standard continuation (Algorithm 2) avoids computing every pQ by using a parameter $0 < D < B_2$ in order to express each prime $B_1 < p < B_2$ in the form $p = mD \pm j$ with $j \in \left[0, \lfloor \frac{D}{2} \rfloor\right]$ and m varying to cover all the primes in the interval $[B_1, B_2]$. Since p is prime, $\gcd(j, D) = 1$.

To speed up the computations, a usual way to proceed is to precompute a table T of the multiples jQ of Q. Then, the sequence of the points mDQ is computed and the product of every $x_{mDQ} \cdot z_{jQ} - x_{jQ} \cdot z_{mDQ}$ for which $mD \pm j$ is prime is accumulated. It is indeed possible to show (see [PŠK+05]) that $pQ = \mathcal{O}$ if and only if $x_{mDQ} \cdot z_{jQ} - x_{jQ} \cdot z_{mDQ} = 0 \mod d$, with d being an unknown prime factor of n.

Parametrization. The different values and parameters of this work were chosen as in the work of Šimka et al. [PŠK+05] and Gaj et al. [GKB+06].

Algorithm 1 ECM factorization.

Input: n to be factored, arbitrary curve E, point $P \in E$ over $\mathbf{F}_n$, bounds $B_1 < B_2 \in \mathbb{N}$

Output: d: factor of n, $1 < d \leq n$, or Fail

Phase 1:

$k \leftarrow \prod_{p \in \mathbb{P}, p \leqslant B_1} p^{e_p},\ e_p \leftarrow \max\{q \in \mathbb{N} : p^q \leqslant B_2\}$
$Q = (x_q, y_q, z_q) \leftarrow kP$
$d \leftarrow \gcd(z_q, n)$
if $d > 1$ **then**
  return d
else
  goto Phase 2.

Phase 2:

$t \leftarrow 1$
for each $p \in \mathbb{P}$ such that $B_1 < p < B_2$ **do**
  $(x_{pQ}, y_{pQ}, z_{pQ}) \leftarrow pQ$
  $t \leftarrow t \cdot z_{pQ} \mod n$
$d \leftarrow \gcd(t, n)$
if $d > 1$ **then**
  return d
else
  Fail (Retry with another curve).

More precisely, B_1 and B_2 are set to the following values: $B_1 = 960$ and $B_2 = 57000$. They were chosen to find factors of up to around 40 bits. The parametrization of Suyama [ZD06] is also employed for the computation of initial points and curve coefficients (a, b of Equation 2.2).

In the second phase, D should be ideally taken near $\sqrt{B_2}$ in order to minimize the computations for the table T and the sequence of points Q_m. D must also be small enough and with many prime factors in order to reduce the number of all possible j (since $\gcd(j, D) = 1$) hence the size of the table T. We chose $D = 2 \cdot 3 \cdot 5 \cdot 7 = 210$ which is close to $\sqrt{B_2} = 239$, leading to 24 possible values for j.

As opposed to [GKB$^+$06], we propose to normalize the coordinates of the $jQ = (x_{jQ}, z_{jQ})$, i.e., divide these by $z_{jQ} \mod n$. In this way, only the new coordinates x_{jQ} need to be stored in the table T. This also decreases the number of modular multiplications performed in Phase 2 since

Algorithm 2 Phase 2 optimized.

Input: n, E, Q (result of Phase 1), $B_1 < B_2 \in \mathbb{N}$ and D
Output: d: factor of n, $1 < d \leq n$, or Fail

Precomputations:
$M_{min} \leftarrow \left\lfloor \left(B_1 + \frac{D}{2}\right)/D \right\rfloor$, $M_{max} \leftarrow \left\lceil \left(B_2 - \frac{D}{2}\right)/D \right\rceil$
Table J: $j \in \{1, \ldots, \frac{D}{2}\}$ s.t. $\gcd(j, D) = 1$
Table MJ: $MJ[m, j] \leftarrow 1$ **if** $mD \pm j$ is prime, **else** 0,
with $j \in J$ and $M_{in} \leq m \leq M_{max}$
Table T: $\forall j \in J$, coordinates $(x_{jQ} :: z_{jQ})$ of points jQ

Main Computations:
$t \leftarrow 1$, $Q_D \leftarrow DQ$, $Q_m \leftarrow M_{min}Q$
for $m = M_{min}$ **to** M_{max} **do**
  for $j \in J$ **do**
    if $MJ[m, j] = 1$ **then**
      retrieve jQ from table T
      $t \leftarrow t \cdot (x_{Q_m} \cdot z_{jQ} - x_{jQ} \cdot z_{Q_m})$
  $Q_m \leftarrow Q_m + Q_D$
return $d \leftarrow \gcd(t, n)$

$t = t \cdot (x_{Q_m} \cdot z_{jQ} - x_{jQ} \cdot z_{Q_m})$ becomes $t = t \cdot (x_{Q_m} - x_{jQ} \cdot z_{Q_m})$. This technique spares a little less than one third of the total of the modular multiplications performed in the main computations of Phase 2. For the chosen parameters, about 4350 multiplications can be avoided at the expense of 24 divisions. If those divisions appear to be too expensive, the Montgomery trick can be applied (cf. Section 3.4.4): they can be traded with 1 inversion, $24 + 3 \cdot 23$ multiplications and some additional memory.

Curves in the Montgomery form. The elliptic curves considered in ECM are not exactly the same as the ones we presented in Section 1.2.2. To speed up the computations, it is more efficient to use elliptic curves in the Montgomery form, which were specially designed for ECM [Mon87]. These curves as defined as follows:

$$bY^2 = X^3 + aX^2 + X \tag{2.1}$$

where $a, b \in F_p$, $a^2 \neq 4$ and $b \neq 0$.

When using homogeneous (projective) coordinates instead of affine coordinates (cf. Section 1.2.2), Equation 2.1 becomes:

$$by^2z = x^3 + ax^2z + xz^2. \tag{2.2}$$

With elliptic curves in homogeneous coordinates, the addition and doubling of points can be done without the costly modular division. Curves in Montgomery form have the additional advantage that the intermediate computations of these operations do not use the y coordinate (cf. Algorithm 3), which can be recovered from the other coordinates. Nevertheless, this is not necessary as ECM does not use y coordinates.

Algorithm 3 Point add and double for Montgomery curves.

$$
\begin{aligned}
x_{P+Q} &\leftarrow z_{P-Q}[(x_P - z_P)(x_Q + z_Q) + (x_P + z_P)(x_Q - z_Q)]^2 \\
z_{P+Q} &\leftarrow x_{P-Q}[(x_P - z_P)(x_Q + z_Q) - (x_P + z_P)(x_Q - z_Q)]^2 \\
4x_P z_P &\leftarrow (x_P + z_P)^2 - (x_P - z_P)^2 \\
x_{2P} &\leftarrow (x_P + z_P)^2 (x_P - z_P)^2 \\
z_{2P} &\leftarrow (4x_P z_P)[(x_P - z_P)^2 + a_{24}(4x_P z_P)]
\end{aligned}
$$

with $a_{24} = \frac{a+2}{4}$, a being the parameter of the curve (cf. Equation 2.2).

In Algorithm 3, the addition $P + Q$ uses the coordinates of the point difference $P - Q$. This additional information compensates for the absence of the y coordinate in the formulae.

The scalar multiplication can be efficiently computed with the Montgomery ladder (Algorithm 4). The additions in this algorithm can be computed with Algorithm 3 since the point difference always equals P_0. If $z_{P_0} = 1$ (e.g., after normalization), the number of multiplications of each step of the algorithm decreases from 11 to 10 (squarings are performed with multiplications).

Algorithm 4 Montgomery ladder.

Input: Point $P_0 \in E$, $k = (k_{s-1}k_{s-2}\ldots k_1 k_0)_2 > 0$
Output: $Q = kP_0$.

$Q \leftarrow P_0$, $P \leftarrow 2P_0$
for $i = s - 2$ **downto** 0 **do**
if k_i = 1 **then**
$Q \leftarrow P + Q$, $P \leftarrow 2P$
else
$P \leftarrow P + Q$, $Q \leftarrow 2Q$
return Q

In Phase 2, the sequence $Q_m = Q_m + Q_D$ can also be computed with Algorithm 3 because the difference $Q_m - Q_D$ equals the previous result Q_{m-1} (for the first addition, $Q_{M_{min}-1}$ has to be precomputed).

In this work, elliptic curves with Montgomery form in homogeneous coordinates are used, as in the previous related works.

2.4 Arithmetic Circuits

The effectiveness of the ECM processor lies in an efficient Area-Time (AT) arithmetic unit. In particular, the modular multiplication plays a crucial role.

This work considers the use of a Xilinx's Virtex-4SX FPGA, as it offers a large number of DSP48 blocks. To ensure a high operating frequency and scalability, each combinatorial operation is pipelined as much as necessary. A digit size of 17 bits is chosen since it fits with the 17×17 unsigned multipliers and the dedicated shift-and-add functionality of the DSP48. Pipelined carry propagate adders are also used to take advantage of the available fast carry chain. It is also consistent with multipliers' input. The design is therefore both pipelined *horizontally* (e.g., the carry chain) and *vertically* (e.g., between two adders).

2.4.1 Parallel versus Serial

Having expressed our intention to rely on the FPGA powerful embedded multipliers, we still need to choose the type of architecture in which they will be exploited. As the modular multiplication algorithms are typically iterative and regular, the two main choices are an iterative (looped) or a parallel (fully unrolled) architecture.

Iterative architecture. A good choice for an iterative architecture is a digit-serial by parallel multiplier. It can be built with a pipelined row of $\lceil \log_{2^{17}} n \rceil$ multipliers (e.g., [TTL03]). For this work, it means a 17×136 multiplier is required. The computation loops over this multiplier and alternately processes a multiplication by a digit and the modular reduction. The advantage of this methodology is that resources grow linearly with the operand width. The disadvantage is its *worst-case* behavior: area (or time, caused by extra iterations) has to be spent to compute any irregular operation.

Parallel architecture. Another choice is a parallel multiplier. Both operands enter in parallel and one modular multiplication is computed each clock cycle. The high latency causes no data dependency problems as there are many factorizations to perform at the same time. Such a circuit can be built with roughly $2\lceil \log_{2^{17}}(n) \rceil^2$ multipliers (the exact complexity is explained in Section 2.4.2). For this work, it means 16 cascaded 17×136 multipliers, or in other words 2 interleaved 136×136 multipliers, are required. Economy of scale in the number of multipliers can be achieved by the use of sub-quadratic algorithms

for the multiplication[1]. However, our preliminary implementation results show that the improvement is not worth the effort for the considered bit-sizes. A parallel circuit has several advantages: the circuit can be data-driven – removing the need for control logic – and fully specialized for each operation of the algorithm. Moreover, data loading and unloading are simpler compared to a set of iterative architectures. As a result this solution exhibits the best AT product. The disadvantage is an inferior flexibility with respect to the iterative architecture: the maximum bit-size that the circuit can handle depends on the number of available multipliers in the FPGA. The unused remaining multipliers cannot be directly exploited as well.

To take the best advantage of the multipliers, a parallel architecture was chosen in this work. As shown below, it has many advantages from an AT point of view. With the aforementioned complexity, the maximum bit-sizes for the Virtex-4 FPGA are 135-bit (SX25), 169-bit (SX35, with 8 multipliers based on Look-Up Tables or LUTs) and 271-bit (SX55). In the context of SHARK, a Virtex-4 SX25 with 128 DSP multipliers is sufficient. Indeed, a number of digits $d = 8$ provides a 135-bit modular multiplier and requires $2d^2 + 1 = 129$ multipliers (as detailed in the next section). The last multiplier has to be implemented with LUTs. If flexibility is an issue, it is still possible to move to 17×136 iterative modular multipliers. The improvement will still be substantial with respect to a pure LUT-based architecture.

2.4.2 Modular Multiplier

Montgomery multiplication [Mon85] is an efficient way to perform the modular multiplication, the most important operation in ECM. Indeed, divisions by non power of two are avoided and it mainly involves computations depending on the Least Significant Bits (LSBs). This is consistent with the carry propagation direction of adders and enables pipelining of the carry chain.

Classical Montgomery multiplication. The classical Montgomery multiplication algorithm (Algorithm 5) works mod n and needs a conditional final subtraction as the result is otherwise bounded by $2n$. This comparison causes not only extra computations, it forces the complete propagation of the carry pipeline. To avoid this problem, a convenient solution is to work mod $2n$. Provided that $4n < R$, the *without final subtraction* version of the algo-

[1]This approach supposes to not interleave the multiplication and the modular reduction. Here, this technique is only useful for the multiplication (without modular reduction) as truncated multiplications are used in the modular reduction (cf. Section 2.4.2).

rithm [Wal02] ensures a bounded output ($< 2n$) if bounded inputs $x, y < 2n$ are applied. This technique is used in this work.

The Montgomery multiplication works in the Montgomery domain: it computes $xyR^{-1} \mod n$ instead of $xy \mod n$. If the inputs x, y are in the Montgomery domain, $\tilde{x} = xR \mod n, \tilde{y} = yR \mod n$, the output is also in the Montgomery domain: $\tilde{x}\tilde{y}R^{-1} \mod n = xyR \mod n$. All the phases of ECM can therefore be computed in the Montgomery domain, leaving the removal of the R factor at the very end of the computation.

Algorithm 5 Radix-b Montgomery modular multiplication.

Input: n, x, y with $0 \leq x, y < n, R = b^d$ with $\gcd(n, b) = 1$ and $n' = -n_0^{-1} \mod b$.
Output: $xyR^{-1} \mod n$.

```
A ← 0
for i = 0 to (d − 1) do
    u_i ← (a_0 + x_i · y_0) · n' mod b
    A ← (A + x_i · y + u_i · n)/b        (A < 2n)
if A ≥ n then
    A ← A − n                            (A < n)
return(A)
```

Modified version. In addition to the removal of the final subtraction, a modified version of the Montgomery multiplication introduced by Orup in [Oru95] and called "Montgomery Tail Tailoring" in [Har04] is used. It supposes a radix b (here equal to 2^{17}) and inputs represented by their digits: i.e., $n = (n_{d-1} \cdots n_1 n_0)_b$. Compared to the original algorithm of Orup, we choose to compute the last iteration as in the classical algorithm to avoid the increase of the dynamic by one digit.

The resulting algorithm is Algorithm 6. It performs the $d-1$ first iterations with a scaled module $ns = n \cdot n'$ such that its constant $ns' = -ns_0^{-1} \mod b$ is equal to 1. This is always the case since $n' = -n_0^{-1} \mod b$. This simplification saves the $d - 1$ digit multiplications by n' previously needed to update u_i in Algorithm 5. ns has one extra digit compared to n but this does not increase the number of digit multiplications since $ns_0 = -1 \mod b = b - 1$ and the least significant digit of $(A + x_i \cdot y + u_i \cdot ns)$ equals 0 (by construction). The update of A in the for loop

$$A \leftarrow (A + x_i \cdot y + u_i \cdot ns)/b$$

is therefore computed in practice by:

$$A \leftarrow (A + x_i \cdot y)/b + u_i \cdot (ns_{d..1} + 1),$$

where $ns_{d..1} = ns/b$, meaning that only the digits 1 to d of ns are kept. This input $\widetilde{ns} = ns_{d..1} + 1$ can be precomputed. The total number of digit multiplications is $(d-1) \cdot 2d + 2d + 1 = 2d^2 + 1$ instead of $2d^2 + d$ using the classical Montgomery algorithm.

Algorithm 6 Montgomery Tail Tailoring multiplication.

Input: $n,\ x, y$ with $0 \leq x, y < 2n, R = 2b^d$ with $\gcd(n, b) = 1$ and $n' = -n_0^{-1} \mod b$.
Output: $xyR^{-1} \mod 2n$.
Precomputation: $ns = (ns_d \cdots ns_1 ns_0)_b = n \cdot n'$.

$A \leftarrow 0$
for $i = 0$ **to** $(d-2)$ **do**
  $u_i \leftarrow (a_0 + x_i \cdot y_0) \mod b$
  $A \leftarrow (A + x_i \cdot y)/b + u_i \cdot (ns_{d..1} + 1)$ $\quad (A < bn + n)$
$u_{d-1} \leftarrow (a_0 + x_{d-1} \cdot y_0) \cdot n' \mod b$
$A \leftarrow (A + x_{d-1} \cdot y + u_{d-1} \cdot n)/b$ $\quad (A < 3n)$
if $A \mod 2 = 1$ **then**
  $A \leftarrow A + n$ $\quad (A < 4n)$
return$(A/2)$ $\quad (A < 2n)$

Our analysis on the bound of the Tail Tailoring Algorithm showed that an extra correction step ($A = A/2 \mod n$) is required to ensure the $x, y < 2n$ precondition. To obtain the bound, it is important to remember the condition $x < 2n$, meaning that $x_{d-1} < b/2 - 1$.

Modular multiplier architecture. The circuit of the modular multiplication is presented in Figure 2.1. With our fully unrolled architecture, the d (i.e., 8) iterations are implemented with d similar circuits. However, they are slightly different for the first and last iterations: the first is simpler since $A = 0$ and the last one uses n' and n in place of $\widetilde{ns}$. In Figure 2.1, only one of the 6 (i.e., $d-2$) identical circuits for the other iterations is represented.

In Figure 2.1, each 17×136-bit multiplier (i.e., $1 \times d$ digit) is implemented with cascaded DSP48s, as shown in Figure 2.2. In this multiplier, the 8 digits of the 136-bit operand are temporally shifted, i.e., the next digit is sent one clock cycle after the current one (starting with the least significant digit). This is necessary since each DSP48 adder takes as input the most significant half of the previous adder output. The temporal shift in the operand digits is kept

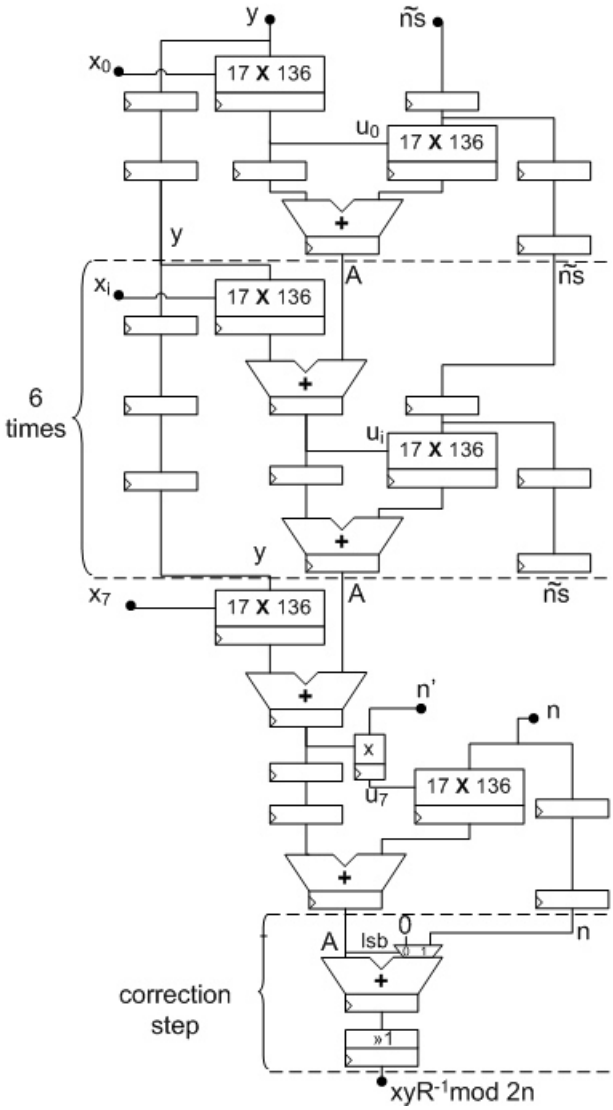


Figure 2.1: Circuit of the 136×136 pipelined modular multiplier. The fully unrolled architecture allows to reach a throughput of 1 modular multiplication per clock cycle.

throughout the whole circuit as it also allows to break the carry chain of the 136-bit adders (implemented as pipelined ripple-carry adders) and therefore keep a high clock frequency.

The multiplier $\times n'$ in the last iteration in Figure 2.1 can be implemented with general purpose logic, especially as it is a truncated multiplication: only

the least significant half of the product is needed for the calculation of u_{d-1} in Algorithm 6.

Because of the parallel behavior of the modular multiplier, shift registers are needed to retime and feed the operands throughout the circuit. According to Figure 2.1, the parallel operand y must be provided in the d circuits implementing the iterations. The same stands for $\widetilde{ns}$ except for the last iteration. For the operand x, each iteration consumes a digit. As a result, the width of the shift register decreases with the iteration number (not represented in Figure 2.1). Inputs n and n' can be inserted after a delay corresponding to the latency of the circuits of the first $d-1$ iterations. The correction step $A = A/2 \mod n$ is implemented by a shifter and a conditional adder depending on the LSB of A.

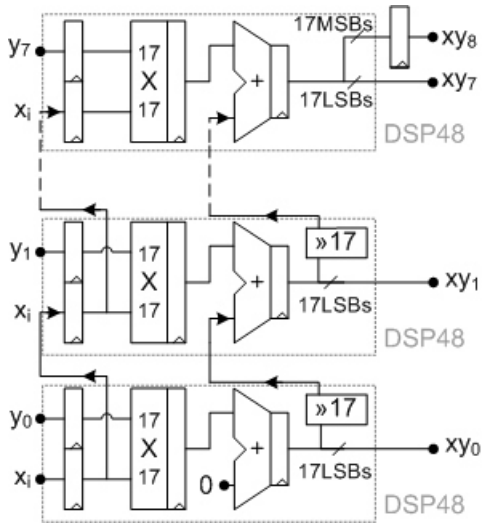


Figure 2.2: Pipelined multiplier achieving the 17×136-bit product $x_i \times y$, based on almost exclusively 8 DSP48 blocks. Each 17-bit digit y_i must be fed with a delay of i cycles. The output digits xy_i are also temporally shifted.

Using the parallel approach, the circuits for the last iteration and for the correction step maximize the AT product as they are used at each clock cycle. With the serial approach, those irregular operations would have been tackled either with a worst case circuit (like the classical algorithm, losing the savings of $d-1$ digit multiplications) or with more time (using extra cycles to lift the extra factors).

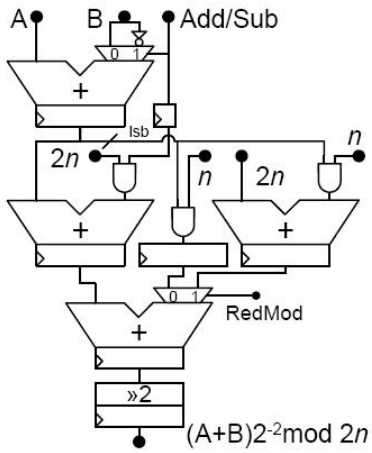


Figure 2.3: Adder/Subtractor $\mod 2n$.

2.4.3 Modular Adder/Subtracter

When using a parallel architecture, the design of a modular adder/subtracter is not straightforward. Usually, several comparisons are required to ensure the result of an add/sub $\in [0, n[$. Performing those comparisons would require breaking the temporal shifting behavior of the inputs. A solution to this problem is to make the result always positive and divide it by $4 \mod n$.

The add/sub circuit is represented in Figure 2.3. It uses two's complement representation and assumes $A, B \in [0, 2n[$. The worst case for the addition $(A + B)/4 \mod n$ is $(A + B + 3n)/4$. The result is therefore bounded by $2n$ as $(4n + 3n)/4 < 2n$. For the subtraction the result is bounded as $-2n < (A - B) < 2n$. If $2n$ is added to this result, it also $\in [0, 4n[$. The output is therefore also bounded by $2n$.

The RedMod control signal is in charge of clearing the penultimate bit (LSB_1) of the intermediary result before the division by $4 \mod n$. It is computed as follows:

if $\text{LSB}_1(n) = 1$ **then**
  $\text{LSB}_1(A \pm B)$ xor Add/Sub
else
  $\text{LSB}_1(A \pm B)$ xor Add/Sub xor $\text{LSB}_0(A \pm B)$.

This circuit adds of course an extra 2^{-2} factor. To avoid this problem, all the operands are pre-multiplied by a factor $2^4 \mod n$. This is consistent with

the Montgomery multiplication since:

$$\begin{aligned}(A2^4R \pm B2^4R)/4 \times (A'2^4R \pm B'2^4R)/4 &= C2^2R \times C'2^2R \\ &= C2^2R \cdot C'2^2R \cdot R^{-1} \\ &= C \cdot C'2^4R.\end{aligned}$$

2.5 Proposed ECM Architecture

Based on the arithmetic core developed above, this section presents the whole ECM architecture, starting with a global overview, then explaining the software part and finally focusing on the hardware part.

Architecture Overview. The global ECM architecture is made of a server (e.g., a PC) and an arbitrary number of clients (FPGAs for instance). The hardware clients (illustrated in Figure 2.4) take care of the computationally-intensive part of the work: the scalar multiplication. The software server handles the low-throughput operations like pre-computations and communications. It must compute and dispatch the starting points, recover the results of the scalar multiplication and handle the final $\gcd()$ calculations in order to find the factor.

Here, the hardware clients are not supposed to perform both Phase 1 and 2. As many FPGAs are expected for a 1024-bit factorization, it is not a problem to specialize an FPGA for a given task. Moreover, as Phase 2 is not always necessary, this approach is more convenient from a computational load balancing point of view. The drawback is the increased bandwidth requirements.

Software Server. Currently, the server computes for each curve the parameter a_{24}, points P_0 and $2P_0$ (for Algorithm 4) and Montgomery constants ns, $\widetilde{ns}$ and n'. P_0 is normalized (dividing x_{P_0} by z_{P_0}) as it is also the point $P - Q$ of Algorithm 3. Nevertheless, as coordinates of points must be pre-multiplied by the Montgomery constant, z_{P_0} is set to R instead of 1. Those data are then transmitted to the clients by any compliant mean. After the computation of Phase 1, the coordinate z_Q of the result is retrieved and a $\gcd()$ is computed. The factor R must also be lifted from z_Q by a division.

Those computations are achieved by a C program. It relies on the easy-to-use and highly efficient GNU Multiple Precision Arithmetic Library (GMP [GMP10]).

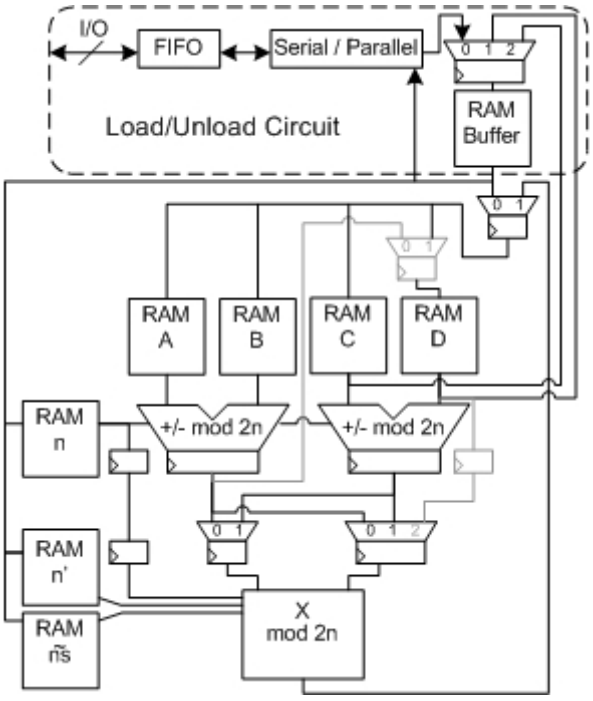


Figure 2.4: Architecture of the hardware client of ECM, Phase 1.

If the server task appears to generate a bottleneck in the overall ECM performance, a small arithmetic processor can still be included in the hardware to take care of those computations.

Hardware Client. Concerning Phase 1, the hardware client computes a scalar multiplication for each curve. According to Algorithm 3 and 4, each step implies the computation of both $2P$ and $P + Q$. The previous points P and Q or Q and P are then overwritten by those new points, according to the current bit of the constant scalar k (pre-stored in memory).

The circuit implementing Phase 1 is presented in Figure 2.4. Besides the two add/sub circuits and the modular multiplier, memory and a simple load/unload architecture to communicate with the server are needed.

In each of the 4 RAM banks A-B-C-D, 2 memory locations reachable with the same addresses are foreseen for each curve. These 8 possible locations per curve are in charge of storing the coordinates of the points P and Q, a_{24}, x_{P-Q} and some intermediate results. There are more intermediate results than the 2 remaining locations but some can be overwritten. The data is placed in

the memory in such a way that each input of both add/sub circuits is directly fed by a RAM bank. This avoids the use of costly multiplexers.

A particular case occurs in the computation of M_{10} (see below) where $(M_2 - M_1)$ cannot be re-computed later. This subtraction is stored in the D RAM bank and bypasses the add/sub to not add an extra 2^{-2} factor. The registers and connexions implementing this special case are represented with shaded elements in Figure 2.4. The purpose of the two multiplexers after the two add/sub circuits is to allow the modular multiplication between all the combinations of operands foreseen in the point multiplication algorithm, and also to enable the modular squaring operation.

Each of the A, B, C, D, n, $\widetilde{ns}$ RAM banks is composed of $\lceil d/2 \rceil$ (i.e., 4) parallel Virtex-4 RAM blocks (bRAMs) of 34-bit width. Two extra 17-bit registers are used to retime the read and write of the two digits held in each 34-bit bRAM. The global temporal shift of the digits is taken into account for the read/write operations of each RAM bank. The n' RAM bank is different since it stores only one digit.

Due to the latency of the arithmetic core, data-dependency problems occur during the computation of Algorithm 3. Three independent steps can be extracted:

I	$M_1 \leftarrow (x_P - z_P)^2$
	$M_2 \leftarrow (x_P + z_P)^2$
	$M_3 \leftarrow (x_P - z_P) \cdot (x_Q + z_Q)$
	$M_4 \leftarrow (x_P + z_P) \cdot (x_Q - z_Q)$
II	$M_5 \leftarrow M_1 \cdot M_2$
	$M_6 \leftarrow (M_2 - M_1) \cdot a_{24}$
	$M_7 \leftarrow (M_3 + M_4)^2$
	$M_8 \leftarrow (M_3 - M_4)^2$
III	$M_9 \leftarrow x_{P-Q} \cdot M_8$
	$M_{10} \leftarrow (M_2 - M_1) \cdot (M_1 + M_6)$

To cope with this dependency problem, each step is repeated with other sets of data (i.e., other curves). This ensures that the results for the first set are available before starting the following step. With operands up to 135 bits, 32 curves running simultaneously suffice to fill the pipeline.

In order to feed the FPGA and transmit the results back to the server, a load/unload circuit is used (see Figure 2.4). A FIFO (made of 2 bRAMs) buffers the communications while a simple shift register deserializes/serializes the inputs/outputs. In order to reduce the time overheads, a buffer (made of a RAM bank) stores the results and the new data during the computation of a

Phase 1. This renders negligible the delay between two Phase 1 computations. For each curve, the new data is a set of 8 values of $17d$ bits (i.e., 136), while the result consists in the coordinate z_Q, located in the D RAM bank.

Phase 2. In this work, we focused on the achievement of Phase 1, while only a preliminary design of Phase 2 was studied, due to time constraints. Nevertheless, the results concerning Phase 1 already give an overview of the performances that could be reached for an implementation of Phase 2, the arithmetic circuits being the same for both phases. With respect to Phase 1, the circuit for Phase 2 can be implemented with a similar structure, provided that the results are sufficient to show the effectiveness of the proposed method. The main difference is the need of extra RAM banks for the precomputed values x_{jQ} and extra intermediate results. As the computations are less regular, more multiplexers are also required to connect the different RAM banks to the add/sub circuits.

The high-level control of Phase 2 lies in the iterative lookup of table MJ (Algorithm 2). As the number of tries before finding the condition $MJ[m, j] = 1$ is smaller than the latency of the arithmetic core, no pipeline stalls occur.

2.6 Implementation Results

Implementation results were achieved for the smallest Xilinx Virtex-4 SX FPGA with the lowest speedgrade. ISE 8.1 was used for synthesis and place & route while test/debug was performed with Modelsim SE 6.1. For the real tests, the setup was a Pentium4 3.2 GHz desktop computer running WindowsXP for the server and an Avnet *Virtex-4 SX35 Evaluation Kit* for the client. For the communication between the PC and the FPGA, we used a slow RS232 connection which is sufficient for testing purposes. A more appropriate fast connection like USB2 should be foreseen for a practical attack.

2.6.1 Software

In order to have a consistent Server-Client model, the software server must be able to feed many hardware clients. This puts the cost of the hardware in foreground, as expected. While in [GKB+06] the server overhead is very small, it appears that our hardware is so fast that pre-computations and communication are not negligible at all.

The bandwidth requirements between the server and a client are assessed as follows. For the communication, a burst of 32 data sets has to be sent

per execution of a Phase 1. For the 125-bit inputs of SHARK, each data set requires $7\times125+17$ bits for the a_{24}, P_0, $2P_0$, $\widehat{ns}$, n and n' values. Considering the performance of our hardware client (16 000 Phase 1 per second per FPGA, as later shown in Table 2.2), a bandwidth of 15 Mb/s is required, assuming a 32-bit data bus. If all the pre-computation is performed in hardware, this requirement falls to 2Mb/s. For the transmission of the results, only z_Q has to be sent back. This also corresponds to a bandwidth of 2Mb/s.

In the pre-computations, the slowest operations are the $\gcd()$ and the 4 modular inversions required for the computation of a_{24}, n', the normalization of x_P and lifting of R in z_Q. Other significant operations are the 25 multiplications. Multiplication is roughly 10 times faster than the inversion on the selected software platform. With our setup, the PC has the computational power to only deal with 4 FPGAs. A low-throughput arithmetic processor with support for modular division and multiplication should therefore be included in the FPGA. It should not pose any particular problem as many slices remain available (roughly 4000 from Table 2.2). Phase 2 will definitively need hardware support for the pre-computations and in particular for the generation of table T (the set of jQ).

2.6.2 Hardware Arithmetic Operators

The arithmetic operators were implemented for numbers up to 135 bits, a little more than the SHARK requirements. As all the circuits are pipelined, moving to other lengths does not raise any particular issue. With this length, the operands are represented by 8 17-bit digits and the Montgomery constant R equals $2^{1+17\cdot8} \mod n$. This means that operands must be pre-multiplied by $2^{4+1+17\cdot8} \mod n$.

The implementation results are given in Table 2.1. For the modular multiplier, the maximum frequency reaches 220 MHz. Although the 100% utilization of the DSP multipliers, the routing step is not so complicated: the DSP blocks are cascaded in groups of 8 which exactly divides the number of DSPs per columns in the Virtex4-SX25 (containing 4 columns of 32 DSPs).

The area requirements are satisfying since almost 66% of the slices are available for the rest of the ECM processor. The arithmetic operators produce an output per clock cycle.

The maximum frequency of the circuit is limited by long routing delays. It is probably the price to pay for automatic place and route tools. While the theoretical maximum frequency of the chosen FPGA is 400 MHz, 300 MHz could be reached with semi-automatic place and route. This is left for further works. As there are no dedicated routing wires between DSP columns in the

135-bit Designs	Frequ. [MHz]	Area [Slices]	Through. [Gbit/s]	DSP48
Mult	220	3405 (33%)	30	128 (100%)
Add/Sub	245	446 (4%)	33	0

Table 2.1: Implementation results on a XC4VSX25-10 FPGA.

targeted device, long routing delays can occur if a multiplier (e.g., 17 x 136) is split over different DSP columns. Fortunately, four 17 x 136 multipliers exactly fit within one (of the four) DSP column in the chosen FPGA. This problem is therefore avoided in this work. If such routing problems happen, it is still possible to use additional null pipelining stages (two should be enough) to cut this routing delay.

2.6.3 ECM Architecture

We now examine the implementation results for the whole hardware client and compare them with the two main previous works [PŠK+05, GKB+06].

Comparing different designs. In this work, Phase 1 was implemented for 135-bit numbers, i.e., a different size with respect to the 198-bit numbers used by Gaj et al. [GKB+06] and Pelzl, Šimka et al. [PŠK+05]. Nevertheless, a comparison of implementation results can be done on the basis of extrapolations, since the size of their design varies linearly with the size of the numbers. For 135-bit numbers, the size of their design is approximately decreased by a factor $\frac{198}{135}$ and more ECM units can therefore fit in an FPGA. The running time is also shortened because of the serial-by-parallel architecture. For instance, a modular multiplication is then performed in $135 + 16$ clock cycles in place of $198 + 16$ (for [GKB+06]).

The impact of this bit-size scale down on the operating frequency is more difficult to evaluate. However, as the design of Gaj et al. is scalable and uses carry save adders, it is assumed that the operating frequency will not vary.

Compared to the design by Pelzl, Šimka et al., results of Gaj et al. exhibit an improvement factor of 3.4 in terms of throughput/hardware cost ratio for Phase 1 [GKB+06]. Only this factor will be used for further comparison with the design of Šimka et al.

FPGA choice. In order to achieve the best throughput/hardware cost ratio, the cheapest FPGA able to hold the modular multiplier was chosen: the Virtex-

4SX25-10. For other bit-sizes, 2007/2008 Xilinx's prices for FPGAs of the same family are US$ 183 for the SX35 and US$ 454 for the SX55 (for a quantity of 2500 devices). Using the EasyPath solution (cf. Section 1.3.1), those prices can be lowered even further: US$ 73 for the SX35 (per 10 000 devices) and US$ 230 (per 5 000 devices) for the SX55 with both a Non-Recurring Engineering (NRE) cost of $US 73 000. Prices for low-cost Spartan-3 FPGAs range from US$ 20 to US$ 130 (per 10 000 devices). The biggest of this family was chosen by Gaj et al.

Brand-new Spartan-3A devices could also be an attractive solution. Indeed, some of them have a high DSP block density. For instance, the largest device (XC3SD3400A-4) embeds 126 DSP slices and 20k slices (twice the number of the Virtex-4SX25). Provided that two multipliers are implemented with general purpose logic, the ECM processor could be fitted in this device. However, because prices are essentially the same (for 1000 devices), we are convinced that it will not result in a superior performance-cost ratio. Indeed, such a device is expected to perform slower than the high-performance Virtex-4SX. On the other hand, processors based on general purpose logic could be implemented with the remaining 10k slices, but they would probably not compensate for the lower speed.

Results. The comparison of performances for Phase 1 is given in Table 2.2. The size of the whole ECM processor is dominated by the modular multiplier (cf. Table 2.1). In terms of throughput/hardware cost ratio, our design outperforms the architecture of Gaj et al. by a factor $\frac{13793}{883} = \mathbf{15.6}$ and the design of Pelzl, Šimka et al. by a factor $15.6 \cdot 3.4 = \mathbf{53}$. This big improvement factor really suggests using high-performance FPGAs and embedded multipliers instead of general purpose logic.

2.7 Cost Estimate for a Sieving Device

A significant improvement in the design of the ECM processor has a great impact when considering the implementation of the entire NFS sieving step. Based on the results of Section 2.6.3, a quick cost assessment can be provided. Even if Phase 2 of the ECM algorithm was not explicitly implemented, it is claimed that a similar structure as for Phase 1 can be used. The running time of Phase 2 should be about $13k$ clock cycles since it is roughly the number of modular multiplications. Assuming a frequency of 220 MHz can be reached as for Phase 1, Phase 2 should be performed ca. $\frac{2.2\ 10^8}{13\ 10^3} \approxeq 16\,900$ times per second per FPGA (i.e., per US$ 116).

Phase 1, 135-bit	Gaj et al. [GKB+06]	Our design
FPGA	XC3S5000-5	XC4VSX25-10
Slices / ECM unit	2300 (7%)	6006 (58%)
DSP	0	128 (100%)
bRAMS / ECM unit	2 (2%)	31 (24%)
Max frequency	100 MHz	220 MHz
# T_{clk} / Mod. Mult.	151	1
# T_{clk} / Phase1	$1.22\ 10^6$	13 750
# Phases1/sec. / ECM unit	82	16 000
# ECM units / FPGA	14	1
# of Phases1/sec. / FPGA	1148	16 000
FPGA price (quantity)	US$ 130 (10^4)	US$ 116 (2500)
# of Phases1/sec. / $100	883	13 793
Improvement factor		**15.6**

Table 2.2: Comparison of implementation results.

From [FKP+05], 10^{14} up to 125-bit numbers must be factored in the sieving step of SHARK. If this task is distributed over a year, approximately $5.5\ 10^6$ of these numbers must be factored per second. According to [PŠK+05], 20 curves per number must be used to have a probability of success $> 80\%$. The success rate of Phase 1 only is roughly one order of magnitude smaller than both phases[2]. The worst case – meaning Phase 2 is always performed – is assumed in order to obtain a higher bound for the cost estimate. In a second, $5.5\ 10^6$ factorizations require $20 \cdot 5.5\ 10^6 = 1.1\ 10^8$ Phases 1 and Phases 2, which are respectively achieved by $\frac{1.1\ 10^8}{16\ 10^3} \approxeq 6900$ and $\frac{1.1\ 10^8}{16.9\ 10^3} \approxeq 6500$ FPGAs.

The overall purchase cost of FPGAs for the ECM factorizations of the sieving step is approximatively $(6900 + 6500) \cdot 116 \approxeq$ **US$ 1.6 M**. Roughly speaking, this price could even be halved using SX55 FPGAs (containing 4 ECM units each) and the EasyPath solution. Using the design of Gaj et al., the same computations (135-bit inputs) – with about 1148 Phases 1 and 1075 Phases 2 per second – gives an overall cost of roughly US$ 25.8 M.

To provide a more realistic cost estimate, the COPACOBANA engine [KPP+06] could be used. It embeds 120 low-cost Spartan-3 XC3S1000 FPGAs and costs US$ 10k. It was estimated that such an engine could be fitted with 60 Virtex-4SX25FF668-10 for US$ 9.5k, including a power supply with

[2]Thanks to Paul Zimmermann (LORIA lab.) for this clarification.

higher capabilities and excluding the expenses for the FPGAs[3]. With a price of US$ 116 per FPGA, it means such an engine could be build for US$ 16.5k. Consequently, 224 COPACOBANA engines for a total price of US$ 3.14M should therefore suffice for the whole computation. Nevertheless, it is important to specify that the bandwidth required by the current architecture is not adapted to the capabilities of COPACOBANA. To use such a device efficiently, the 20 attempts on the different curves (including both Phases 1 and 2) should be performed without any external communications. Curve parametrization should therefore be supported in hardware as well.

For a one-year computation with clusters of FPGAs, the cost of the power supply is not negligible. We note that in a similar context (the solution of ECDLP with COPACOBANA engines), Meurice et al. [MdDBQ07] assessed that the expenses for the power consumption could be as high as a 10^{th} of the purchase cost of the device. The estimate of this cost in our context is left to a future work.

Our cost estimate for a sieving device suggests that the NFS sieving step of an RSA 1024 modulus is not unreachable for a powerful organization. Since the sieving step is about 90% of the entire NFS complexity, our results confirm that it is time to move to a larger key length, as suggested by the recommandations of [GB].

2.8 Comparison with Related Works

This work presented a novel hardware architecture for implementing ECM on FPGA. We already compared our design with the best previously published results [PŠK+05, GKB+06], underlining the big improvement achieved using embedded multipliers and high-performance FPGAs.

Since the year of publication of this work (2007), several works have investigated the implementation of ECM in hardware. They are briefly discussed below.

ECM on FPGA. In [GKB+10], Gaj et al. provide an extended version of their previous results [GKB+06]. The presented architecture is still based on an iterative radix-2 modular multiplier and does not feature any major improvement in terms of performance. Interestingly, the authors compare the performances of their design to the state-of-the-art software ECM implementation, the GMP-ECM [JFZ05]. They remark that their implementation relying

[3]Thanks to Tim Güneysu (Univ. Bochum) for this estimate.

on low-cost FPGAs (Spartan-3 and Spartan-3E) outperform the same generation of microprocessors in terms of performance to cost ratio by about an order of magnitude. This comparison suggests that our design, based on the FPGA embedded multipliers, has an advantage of more than 2 orders of magnitude in performance/cost with respect to a software solution for ECM.

In his thesis [GÖ9], Tim Güneysu describes an FPGA implementation of ECM also relying on DSP blocks of modern FPGAs. The presented design is based on a serial-by-parallel modular multiplier powered with DSP multipliers. This unpublished result turns out to outperform the performance/cost ratio of the design by Gaj et al. by a factor of 3.85, while it is still inferior to our design based on a fully unrolled architecture. Another contribution of this work is the exploration of alternative elliptic curve representations for ECM. Especially, the author shows that the promising Edwards curves [Edw07] are less interesting than Montgomery curves for the considered set of parameters, although more efficient in software [BBLP08]. These curves necessitate more memory than available in the FPGA to show their full potential.

ECM on Graphics Cards. Another kind of hardware for ECM was studied by Bernstein et al. [BCC+09]: they report an implementation of ECM on graphics cards capable of achieving 41.88 million modular multiplications per second for a 280-bit modulus (for a NVIDIA GTX 295 costing about US$ 500). This remarkable performance is obtained thanks to the use of Edwards curves, which can be supported on the more flexible graphic cards. The performance/cost ratio on this type of hardware is superior to the previously reported FPGA implementations, while it appears very roughly 5 times inferior to ours (considering that a design for 271-bit moduli could fit into the Virtex-4SX55, priced US$ 454). Graphics cards seem to be another very competitive platform to run ECM in hardware. They somehow compensate an inferior computational power with a higher flexibility, which can be apparently exploited for ECM with the Edwards curves.

Bibliography

[BBLP08] Daniel J. Bernstein, Peter Birkner, Tanja Lange, and Christiane Peters. ECM using Edwards curves. Cryptology ePrint Archive, Report 2008/016, January 2008.

[BCC+09] Daniel J. Bernstein, Tien Chen, Chen Cheng, Tanja Lange, and Bo yin Yang. ECM on Graphics Cards. Cryptology ePrint Archive, Report 2008/480, January 2009.

[Bre00] Richard P. Brent. Recent progress and prospects for integer factorisation algorithms. In Ding-Zhu Du, Peter Eades, Vladimir Estivill-Castro, Xuemin Lin, and Arun Sharma, editors, *COCOON*, volume 1858 of *Lecture Notes in Computer Science*, pages 3–22. Springer, 2000.

[dMGMdDQ07a] Giacomo de Meulenaer, Franois Gosset, Guerric Meurice de Dormale, and Jean-Jacques Quisquater. Elliptic Curve Factorization Method : Towards Better Exploitation of Reconfigurable Hardware. In *Workshop on Special Purpose Hardware for Attacking Cryptographic Systems (SHARCS'07)*, 9 2007.

[dMGMdDQ07b] Giacomo de Meulenaer, Franois Gosset, Guerric Meurice de Dormale, and Jean-Jacques Quisquater. Integer Factorization Based on Elliptic Curve Method: Towards Better Exploitation of Reconfigurable Hardware. In *IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM07)*, pages 197–207. IEEE Computer Society Press, 4 2007.

[Edw07] Harold M. Edwards. A normal form for elliptic curves. In *Bulletin of the American Mathematical Society*, pages 393–422, 2007.

[FKP+05] Jens Franke, Thorsten Kleinjung, Christof Paar, Jan Pelzl, Christine Priplata, and Colin Stahlke. SHARK: A Realizable Special Hardware Sieving Device for Factoring 1024-Bit Integers. In Rao and Sunar [RS05], pages 119–130.

[GÖ9] Tim Güneysu. *Cryptography and Cryptanalysis on Reconfigurable Devices*. PhD thesis, Ruhr-University Bochum, 2009.

[GB] Damien Giry and Philippe Bulens. Cryptographic Key Length Recommendation. `http://www.keylength.com`.

[GJK+06] Willi Geiselmann, Fabian Januszewski, Hubert Köpfer, Jan Pelzl, and Rainer Steinwandt. A Simpler Sieving Device: Combining ECM and TWIRL. In Min Surp Rhee and Byoungcheon Lee, editors, *ICISC*, volume 4296 of *Lecture Notes in Computer Science*, pages 118–135. Springer, 2006.

[GKB+06] Kris Gaj, Soonhak Kwon, Patrick Baier, Paul Kohlbrenner, Hoang Le, Mohammed Khaleeluddin, and Ramakrishna Bachimanchi. Implementing the elliptic curve method of factoring in reconfigurable hardware. In Goubin and Matsui [GM06], pages 119–133.

[GKB+10] Kris Gaj, Soonhak Kwon, Patrick Baier, Paul Kohlbrenner, Hoang Le, Mohammed Khaleeluddin, Ramakrishna Bachimanchi, and Marcin Rogawski. Area-time efficient implementation of the elliptic curve method of factoring in reconfigurable hardware for application in the number field sieve. *IEEE Transactions on Computers*, 59:1264–1280, 2010.

[GM06] Louis Goubin and Mitsuru Matsui, editors. *Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings*, volume 4249 of *Lecture Notes in Computer Science*. Springer, 2006.

[GMP10] GMP. The GNU Multiple Precision Arithmetic Library. Available online at http://gmplib.org/, April 2010.

[GSST05] Willi Geiselmann, Adi Shamir, Rainer Steinwandt, and Eran Tromer. Scalable hardware for sparse systems of linear equations, with applications to integer factorization. In Rao and Sunar [RS05], pages 131–146.

[Har04] Laszlo Hars. Long modular multiplication for cryptographic applications. In Marc Joye and Jean-Jacques Quisquater, editors, *CHES*, volume 3156 of *Lecture Notes in Computer Science*, pages 45–61. Springer, 2004.

[JFZ05] A. Kruppa D. Newman J. Fougeron, L. Fousse and P. Zimmermann. GMP-ECM. `http://www.komite.net/laurent/soft/ecm/ecm-6.0.1.html`, 2005.

[KAF+10] Thorsten Kleinjung, Kazumaro Aoki, Jens Franke, Arjen K. Lenstra, Emmanuel Thom, Pierrick Gaudry, Peter L. Montgomery, Dag Arne Osvik, Herman Te Riele, Andrey Timofeev, and Paul Zimmermann. Factorization of a 768-bit RSA modulus, 2010.

[Kle06] Thorsten Kleinjung. Cofactorisation strategies for the number field sieve and an estimate for the sieving step for factoring 1024-bit integers. In *Proceedings of Special-purpose Hardware for Attacking Cryptographic Systems 2006 (SHARCS'06)*, April 2006.

[KPP+06] Sandeep Kumar, Christof Paar, Jan Pelzl, Gerd Pfeiffer, and Manfred Schimmler. Breaking Ciphers with COPACOBANA - A Cost-Optimized Parallel Code Breaker. In Goubin and Matsui [GM06], pages 101–118.

[Len87] Hendrik W. Lenstra. Factoring integers with elliptic curves. *The Annals of Mathematics*, 126(3):649–673, November 1987.

[MdDBQ07] Guerric Meurice de Dormale, Philippe Bulens, and Jean-Jacques Quisquater. Collision Search for Elliptic Curve Discrete Logarithm over GF(2^m) with FPGA. In *Workshop on Cryptographic Hardware and Embedded Systems (CHES 2007)*, pages 378–393. Springer, 2007.

[Mon85] Peter L. Montgomery. Modular multiplication without trial division. *Mathematics of Computation*, 44:519–521, 1985.

[Mon87] Peter L. Montgomery. Speeding the Pollard and elliptic curve methods of factorization. *Mathematics of Computation*, 48:243–264, 1987.

[Oru95] Holger Orup. Simplifying quotient determination in high-radix modular multiplication. In *IEEE Symposium on Computer Arithmetic*, pages 193–. IEEE Computer Society, 1995.

[PŠK+05] Jan Pelzl, Martin Šimka, Thorsten Kleinjung, Jens Franke, Christine Priplata, Colin Stahlke, Miloš Drutarovský, Viktor Fischer, and Christof Paar. Area-Time Efficient Hardware Architecture for Factoring Integers with the Elliptic Curve Method. *IEE Proceedings Information Security*, 152(1):67–78, October 2005.

[RS05] Josyula R. Rao and Berk Sunar, editors. *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, volume 3659 of *Lecture Notes in Computer Science*. Springer, 2005.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21:120–126, 1978.

[Sha05] SHARCS'05, Workshop on Special Purpose Hardware for Attacking Cryptographic Systems, 2005.

[TTL03] S. H. Tang, K. S. Tsui, and P. H. W. Leong. Modular exponentiation using parallel multipliers. In *Proceedings of the 2003 IEEE International Conference on Field Programmable Technology (FPT*, pages 52–59, 2003.

[Wal02] Colin D. Walter. Precise Bounds for Montgomery Modular Multiplication and Some Potentially Insecure RSA Moduli. In *CT-RSA '02: Proceedings of the The Cryptographer's Track at the RSA Conference on Topics in Cryptology*, pages 30–39, London, UK, 2002. Springer.

[ZD06] Paul Zimmermann and Bruce Dodson. 20 years of ECM. In F. Hess, S. Pauli, and M. Pohst, editors, *7th Algorith-*

mic Number Theory Symposium (ANTS VII), volume 4076 of *Lecture Notes in Computer Science*, pages 525–542, Berlin/Germany Allemagne, 2006. Springer Verlag.

CHAPTER 3

Solving the Discrete Logarithm Problem on Koblitz Curves with ASICs

Elliptic Curve Cryptography is becoming the most preferred algorithm to achieve public-key cryptography. Its security is related to the intractability of the Elliptic Curve Discrete Logarithm Problem (ECDLP). To encourage research on this mathematical hard problem, many challenges were made available, in the form of ECDLP instances to solve with increasing complexity. In this chapter, we focus on ECC2K-130, the yet unsolved challenge defined on Koblitz curves over $\mathbf{F}_{2^{131}}$. We consider an attack relying on Application-Specific Integrated Circuits (ASICs), as they potentially offer the best price-performance ratio. For the design of the arithmetic circuits, we explore both options of the polynomial and normal-basis representations. As opposed to previous studies, our implementation results underline the interest of the latter representation for the arithmetic core. Moreover, our estimates show that the "infeasible" ECC2K-130 challenge can in fact be broken in one year with around 200 ASICs, costing altogether 60 000 €.

3.1 Introduction

Elliptic Curve Cryptography (ECC) is becoming the standard public-key primitive not only for mobile devices but also for high-security applications. Its advantages are the higher cryptographic strength per key bit in comparison with RSA and the higher speed in implementations. The security of ECC relies on the hardness of a mathematical problem, the Elliptic Curve Discrete Logarithm

Problem (ECDLP, introduced in Section 1.2.2). To improve the understanding of the exact strength of this problem, the well-known cryptographic company Certicom has published a series of challenges [Cer97], i.e., ECDLP instances to solve of growing difficulty. These challenges cover curves over prime fields and binary fields at several different sizes.

For the binary curves, each field size has two challenges: a Koblitz curve and a random curve defined over the full field (see Section 1.2.2). For small bit sizes, the challenges were broken quickly – the 79-bit challenges fell already in 1997, those with 89 bits in 1998 and those with 97 bits in 1998 and 1999; Certicom described these parameter sizes as training exercises. In April 2000, the first Level I challenge (the Koblitz curve ECC2K-108) was solved by Harley's team in a distributed effort on a multitude of PCs on the Internet [Har00]. After that, it took some time until the remaining challenges with 109 bit fields were tackled. The ECCp-109 (elliptic curve over a prime field of 109 bits) was solved on November 2002 and the ECC2-109 (random elliptic curve over a binary field with 109 bits) was solved in April 2004; both efforts were organized by Chris Monico.

The gap of more than one year between these results is mostly due to the Koblitz curves offering less security per bit than curves defined over the extension field or over prime fields. As explained in Section 1.2.2, the Frobenius automorphism can be used to speed up the protocols using such curves. It is the main reason why Koblitz curves are attractive in practice, but it also gives an advantage to the attacker. In particular, over $\mathbf{F}_{2^n}$ the attack is sped up by a factor of approximately $\sqrt{n}$.

Since 2004, not much was heard about attempts to break the larger challenges. When this work was undertaken, Certicom's documentation stated "The 109-bit Level I challenges are feasible using a very large network of computers. The 131-bit Level I challenges are expected to be infeasible against realistic software and hardware attacks, unless of course, a new algorithm for the ECDLP is discovered. The Level II challenges are infeasible given today's computer technology and knowledge." The few studies of the complexity of the Level I challenges for an attack with dedicated hardware tend to confirm this statement: according to [BMdDQ06, MdDBQ07], US$ 20 000 000 are needed to solve the challenge over $\mathbf{F}_{2^{131}}$ in one year. However, these estimates concern general binary curves and does not investigate the special case of the Koblitz curves, however more reachable to an attacker.

In this chapter, we analyze the cost of breaking ECC2K-130, the binary Certicom challenge on Koblitz curve over $\mathbf{F}_{2^{131}}$. We focus on an attack relying on special-purpose hardware, as we believe ASICs should provide the best price-performance ratio for the huge computing task required for breaking this

yet unsolved Certicom challenge. For the design of the arithmetic circuits, we explore the solutions based on both the polynomial and the normal bases, since the latter representation could be more beneficial when considering Koblitz curves. Based on our implementation results for the arithmetic operators, we assess the cost of resolving the full Certicom challenge ECC2K-130: we show how much effort is needed to break this "infeasible" challenge with dedicated ASIC engines.

The work presented in this chapter was achieved in close collaboration with the Working Group 1 of the Virtual Applications and Implementations Research Lab (VAMPIRE [VAM10]), within the framework of the European Network of Excellence in Cryptology (ECRYPT II) [ECR10]. The overall goal was to study the feasibility of attacking the Certicom challenge ECC2K-130 with a large variety of platforms. The preliminary results appeared in [BBB+09a], while [BBB+09b] describes a more advanced status of the project. This chapter specializes in the attack relying on ASICs. It provides the initial estimates of a work in progress, which already feature some interesting contributions.

Contributions. Our first contribution is to demonstrate the interest of the normal basis as underlying representation for a hardware implementation aiming to solve the challenges based on Koblitz curves. Our ASIC implementations show that the design in normal basis, although less optimized, can already compete with the one in polynomial basis, and should even outperform it once fully improved.

Our second contribution is to show that the "infeasible" ECC2K-130 challenge is in fact feasible. From our implementations, we assess that about 200 ASICs costing altogether approximately 60 000 € suffice to break it in an expected time of a year. For comparison, [BMdDQ06] and [MdDBQ07] estimated a cost of nearly US$ 20 000 000 to break ECC2K-130 in a year with FPGA-based engines. The improvement is obtained with an iteration step optimized for Koblitz curves and considering a Multi-Project Wafer (MPW cf. Section 1.3.2) for the production and packaging of the ASICs, leading to a very cheap cost for the hardware.

Outline. We start by giving an overview of Pollard's rho method for solving the discrete logarithm problem and describe some of its main improvements (Section 3.2). We then detail the iteration function defining the pseudo-random walks in Pollard's rho (Section 3.3). This is followed by a description of how to perform the underlying arithmetic operations with both the polyno-

mial and normal-basis representations. Afterwards, our implementation results concerning ASICs are presented, together with a cost assessment of an attack based on ASICs to solve the ECC2K-130 challenge (Section 3.5). Finally, we end the chapter by comparing our results with the related works and reporting the current status of the distributed attack undertaken by the ECRYPT Vampire working group to break the same challenge (Section 3.6).

3.2 Overview of Pollard's Rho and its Improvements

The two best methods for solving the Elliptic Curve Discrete Logarithm Problem (ECDLP) are the Pollard's rho [Pol78] and Shanks' Baby-step Giant-step [Sha71] algorithms. While sharing the same time complexity, the former method requires a very little amount of memory and is therefore in its parallelized form the preferred way of solving a generic ECDLP.

In this section, we explain the parallelized version of Pollard's rho method. First, we describe the single-instance version of the method and then show how to parallelize it with the distinguished-point method as done by van Oorschot and Wiener in [vOW99]. We end the section by presenting another important improvement of the method, the exploitation of fast group automorphisms.

3.2.1 Single-instance Pollard rho

Pollard's rho method is an algorithm to compute the discrete logarithm in generic cyclic groups. It was originally designed for finding discrete logarithms in $\mathbf{F}_p^*$ [Pol78] and is based on Floyd's cycle-finding algorithm [Flo67] and the birthday paradox. In the following, let G be a cyclic group, using additive notation, of order ℓ with a generator P. Given $Q \in G$, our goal is to find an integer k such that $[k]P = Q$.

The idea of Pollard's rho method is to construct a sequence of group elements with the help of an iteration function $f : G \to G$. This function generates a sequence

$$P_{i+1} = f(P_i),$$

for $i \geq 0$ and some initial point P_0. We compute elements of this sequence until a collision of two elements occurs. A collision is an equality $P_q = P_m$ with $q \neq m$. Let us assume we know how to write the elements P_i of the sequence as $P_i = [a_i]P \oplus [b_i]Q$, then we can compute the discrete logarithm $k = \log_P(Q) = \frac{a_q - a_m}{b_m - b_q}$ if a collision $P_q = P_m$ with $b_m \neq b_q$ has occurred. We show later how to obtain such sequences.

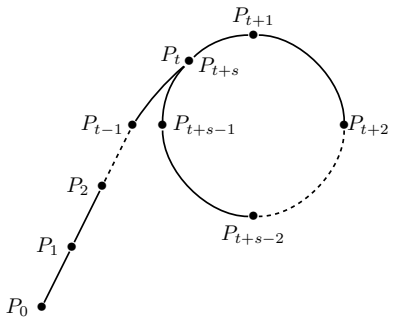


Figure 3.1: Abstract diagram of the rho method.

Assuming the iteration function is a random mapping of size ℓ, i.e., f is equally probable among all functions $G \to G$, Harris showed in [Har60] that the expected number of steps before a collision occurs is approximately $\sqrt{\pi\ell/2}$. The sequence $(P_i)_{i\geq 0}$ is called a *random walk* in G.

A pictorial description of the rho method can be given by drawing the Greek letter ρ representing the random walk and starting at the tail at P_0. "Walking" along the line means going from P_i to P_{i+1}. If a collision occurs at P_t, then $P_t = P_{t+s}$ for some integer s, and the elements $P_t, P_{t+1}, \ldots, P_{t+s-1}$ form a loop (see Figure 3.1).

In the original paper by Pollard it is proposed to find a collision with Floyd's cycle-finding algorithm. The idea of this algorithm is to walk along the sequence at two different speeds and wait for a collision. This is usually realized by using the two sequences P_i and P_{2i}. Doing a step means to increase i by 1. If $P_i = P_{2i}$ for some i, then we have found a collision.

To benefit from this method it is necessary to construct walks on G that behave like random mappings and for which for each element a representation as $P_i = [a_i]P \oplus [b_i]Q$ is known. An example of such a class of walks is the so-called r-adding walk as studied by Teske [Tes01]. The group G is divided in r partitions using a partition function $\psi : G \to [0, r-1]$. An element $R_j = [c_j]P \oplus [d_j]Q$, for random c_j and d_j, is associated to each partition and the iteration function is defined as

$$P_{i+1} = f(P_i) = P_i \oplus R_{\psi(P_i)},$$

and the values of a_i and b_i are updated as $a_{i+1} = a_i + c_j,\ b_{i+1} = b_i + d_j$.

When a collision $P_q = P_m$ for $q \neq m$ is found, then we obtain

$$[a_q]P \oplus [b_q]Q = [a_m]P \oplus [b_m]Q,$$

which implies $(a_q - a_m)P = (b_m - b_q)Q$ and hence $k = \log_P(Q) = \frac{a_q - a_m}{b_m - b_q}$. This solves the discrete logarithm problem. With negligible probability the difference $b_m - b_q = 0$; in this case the computation has to be restarted with a different starting point P_0. Teske showed that choosing $r = 20$ and random values for the c_j and d_j approximates a random walk sufficiently well for the purpose of analyzing the function. For implementations, a power of 2 such as $r = 8$ or $r = 16$ or $r = 32$ is more practical.

3.2.2 Parallelized Version and Distinguished Points

When running N instances of Pollard's rho method concurrently, a speed-up of $\sqrt{N}$ is obtained. To get a linear speedup, i.e., by a factor N, and thus to parallelize Pollard's rho method efficiently, van Oorschot and Wiener [vOW99] proposed the distinguished-points method. It works as follows: one defines a subset D of G such that D consists of all elements that satisfy a particular condition. For instance, we can choose D to contain all group elements of which the s least significant bits are zero, for some positive integer s. This allows to easily check if a group element is in D or not.

Each instance starts at a new linear combination $[a_0]P \oplus [b_0]Q$ and performs the random walk until a distinguished point is found. The distinguished point, together with the coefficients a_i, b_i that lead to it, is then stored on a central server. The server checks for collisions within the set of distinguished points and can solve the DLP once one collision is found. As before, the difference $b_m - b_q$ can be zero. In this case the distinguished point P_q is discarded and the search continues; note that the other stored distinguished points can lead to collisions since all processes are started independently at random.

3.2.3 Automorphisms of Small Order

Pollard's rho can be further sped up by taking advantage of the automorphisms of small order, i.e., the very efficient ways of computing certain classes of points in elliptic curve groups (significantly faster than point addition). In our context, the two following automorphisms are exploitable.

Negation. Elliptic curve groups allow very fast negation. On binary curves such as the Certicom challenge curves, the negative of $P = (x, y)$ is $-P = (x, y+x)$. One can speed up the rho method by a factor of $\sqrt{2}$ (cf. [WZ98]) by

choosing an iteration function defined on sets $\{P, -P\}$. For example, one can take any iteration function g, and define $f(x, y)$ as $g(\min\{(x, y), (x, y + x)\})$, ensuring that $f(-P) = f(P)$. Here $\min$ means lexicographic minimum. Special care has to be taken to avoid fruitless, short cycles; see [WZ98] for details.

Frobenius. Another amelioration is available in the case of Koblitz curves: they allow a very fast Frobenius automorphism [HMV04]. Specifically, if (x, y) is on the ECC2K-130 curve then $\sigma(x, y) = (x^2, y^2), \sigma^4(x, y) = (x^{2^4}, y^{2^4})$ and so on through $(x^{2^{130}}, y^{2^{130}})$ are on the ECC2K-130 curve; remark that $(x^{2^{131}}, y^{2^{131}}) = (x, y)$. One can speed up the rho method by an additional factor of $\sqrt{131}$ by defining the iteration function on sets $\left\{\pm\sigma^i(x, y) | 0 \leq i < 131\right\}$; there are only about $l/262 \approx 2^{121}$ of these sets. In this case, the expected number of rho iterations is approximately $\sqrt{\pi l/524} \approx 2^{60.8}$. On the other hand, automorphisms somewhat reduce the randomness of the walks and thus increase the number of iterations by a few percents, as explained below.

3.3 Iteration Function

We now explain the iteration function we consider in this work. The way used to define (pseudo-)random walks in Pollard's rho is a key point as it impacts the expected running time as well as the memory needs and the operations to implement in hardware.

The main contributions regarding efficient iteration functions for elliptic curves with automorphisms were provided by Wiener and Zuccherato [WZ98], Gallant, Lambert, and Vanstone [GLV00], and Harley [Har00]. The method we selected was described by Gallant, Lambert, and Vanstone [GLV00]. Harley used a similar method in his code to attack the earlier Certicom challenges ECC2K-95 and ECC2K-108 [Har00].

Used iteration function. In the iteration function we use, the walks take place on the classes of points under the Frobenius automorphism: the class of a point P also contains $-P$, $\pm\sigma(P)$, $\pm\sigma^2(P)$, ..., $\pm\sigma^{130}(P)$. The method does not need any precomputed random points R_i on the curve. Instead, we define the walk and the iteration function f as

$$P_{i+1} = f(P_i) = P_i \oplus \sigma^j(P_i), \tag{3.1}$$

where σ is the Frobenius automorphism and j the Hamming weight of the binary representation of $x(P_i)$ in normal-basis representation. We discuss the

field representation in the next section, but note that in normal-basis representation, the Hamming weight of $x(P_i)$ is equal to the Hamming weight of $x(\pm\sigma^k(P_i))$ for all $k \in \{0, \ldots, 130\}$. If the computations are not carried out in normal-basis representation, it is necessary to change the representation of $x(P)$ from polynomial basis to normal basis at every step.

The iteration function is well defined on the classes of points since

$$\pm\sigma(P_{i+1}) = \pm\sigma(P_i) \pm \sigma^j(\sigma(P)) = f(\pm\sigma(P)).$$

The proposed version by Gallant, Lambert, and Vanstone [GLV00] does not use the Hamming weight to define j. Instead, they use a unique representative per class, e.g. the point with lexicographically smallest x-coordinate, and put $j = \mathrm{hash}(x(P))$ for hash a hash function from $\mathbf{F}_{2^n}$ to $\{0, 1, \ldots, n-1\}$. Using the Hamming weight of $x(P)$ instead avoids the computation of the unique representative at the expense of a somewhat less random walk. There are many more points with a Hamming weight around $n/2$ than there are around the extremal values 0 and $n-1$. Brent and Pollard [BP81] give a theoretical way of analyzing the loss of randomness.

Harley included an extra tweak to the method by using the Hamming weight for updating the points but restricting the maximal exponent of σ in Equation 3.1 to the remainder of j modulo 7. He observed that for $n = 109$ the equality $(1 + \sigma^3) = -(1 + \sigma)$ holds. He settled for scalars $(1 + \sigma^1), (1 + \sigma^2), (1+\sigma^4), (1+\sigma^5), (1+\sigma^6), (1+\sigma^7)$, and $(1+\sigma^8)$. This limits the number of times σ has to be applied per step. For sizes larger than 109 somewhat larger exponents should be used. In our iteration function, we take $j = ((HW(x_i) \mod 8) + 3$, i.e., j is limited to the set $\{3, 4, ..., 10\}$. The walks resulting from this method are even less resembling random walks but the computation is sped up by requiring fewer squarings. As explained in [BBB+09b], the use of the Hamming weight together with our restriction on j increase the running time of the attack to $\sqrt{\pi l/(2 \cdot 2 \cdot 131)} \cdot 1.069993 \approx 2^{60.9}$ iterations.

Distinguished points. For parallel Pollard rho, we also need to define distinguished points (or classes in our case). A class is a distinguished one if the Hamming weight of $x(P)$ is less than or equal to a bound w. Note that the Hamming weight of each x coordinate is even in the subgroup of order ℓ [Ser98]. This means that $(\binom{n}{w} + \binom{n}{w-2} + \cdots)/2^{n-1}$ approximates the probability of distinguished points as only about 2^{n-1} different values can occur as x-coordinates. In our case, we take $w = 34$, leading to a probability of $2^{-25.27}$ that a point is a distinguished one (this choice is justified in [BBB+09b].

Benefits of recomputation. Standard practice in ECDLP computations is to report each distinguished point as a linear combination of P and Q. Some memory and computation resources have to be foreseen to store and update these coefficients. In this work, we propose to avoid this by recomputing the coefficients once a distinguished point is found and involved in a collision. This is possible since the sequence of the successive values of j is deterministic and only depends on the starting random point. As a result, a distinguished point report only has to mention the walk starting point (or its seed from which it is computed) and the distinguished point (or its hash value). This technique save bandwidth and reduces the complexity of the main loop at the cost of episodic recomputations on the central server (see [BBB+09b] for more details).

3.4 Finite Field Arithmetic

In Section 3.5, we will consider an implementation of the main loop computations on special-purpose hardware in different representations — in particular looking at polynomial and normal basis representations. Beforehand, in this section, we describe these field representations and explain how to perform the arithmetic operations in both bases.

For the Koblitz curves when using the iteration function discussed above, each step mainly takes 1 elliptic curve addition and 2 squarings to a variable power: $x(P)^{2^m}$ and $y(P)^{2^m}$, with $m \in \{3, 4, ..., 10\}$. As the intermediate powers are not needed, special routines can be implemented: we denote these operations as m-squarings.

Concerning point addition, we note that we cannot use inversion-free coordinate systems, i.e., a redundant representation $P = (X_P : Y_P : Z_P)$ to denote the affine point $(X_P/Z_P, Y_P/Z_P)$ (for $Z_P \neq 0$). In Pollard's rho, it is crucial that P_{i+1} is uniquely determined by P_i. Thus we have to work with affine points. For ordinary binary curves $y^2 + xy = x^3 + ax^2 + b$, the addition of $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$ with $x_P \neq x_Q$ is given by

$$(x_R, y_R) = (\lambda^2 + \lambda + a + x_P + x_Q, \lambda(x_R + x_P) + y_P + x_R), \text{ where } \lambda = \frac{y_P + y_Q}{x_P + x_Q}.$$

Each addition thus needs 1 inversion, 2 multiplications and 1 squaring. As we will see below, the expense of the inversion, the by far most expensive field operation, can be reduced using an optimization.

3.4.1 Field Representation

The most common way to express the elements of $\mathbf{F}_{2^n}$ is to do it in terms of a polynomial basis. Making use of another representation, the normal basis, might however be more advantageous. We here present both possibilities, which will be explored in our implementation of next section.

Polynomial basis. Typically fields are represented in a polynomial basis using the isomorphism $\mathbf{F}_{2^n} \cong \mathbf{F}_2[z]/F(z)$, where $F(z)$ is an irreducible polynomial of degree n. A basis is given by $\{1, z, z^2, \ldots, z^{n-1}\}$. In this representation addition is done component wise (i.e., it is bitwise *xor*) and multiplication is done as polynomial multiplication modulo F. The Certicom challenges [Cer97] are given in polynomial-basis representation with F an irreducible trinomial or pentanomial. A low-weight irreducible polynomial renders the modular reduction relatively inexpensive: each bit of the result is obtained by xoring a few bits.

Normal basis. An alternative representation of finite fields is to use normal bases [ACD+06]. A normal basis is a basis of the form $\{\theta, \theta^2, \theta^{2^2}, \ldots, \theta^{2^{n-1}}\}$ for some $\theta \in \mathbf{F}_{2^n}$. Also in this representation addition is done component wise. Squaring is very easy as it corresponds to a cyclic shift of the coefficient vector: if $c = \sum_{i=0}^{n} c_i \theta^{2^i}$ then $c^2 = \sum_{i=0}^{n} c_i \theta^{2^{i+1}} = \sum_{i=0}^{n} c_{i-1} \theta^{2^i}$, where $c_{-1} = c_n$. On the other hand, multiplications are significantly more complicated: expressing $\theta^{2^i+2^j}$ in the basis usually requires a multiplication matrix.

Converting from one basis to the other might be necessary with our iteration function relying on normal basis. This can be done with the help of a transition matrix, an $n \times n$ matrix over $\mathbf{F}_2$.

3.4.2 Multiplication

We now detail how the main arithmetic operations are performed in polynomial and normal bases, starting with multiplication.

Polynomial basis. In polynomial representation, multiplication is done as polynomial multiplication modulo F, so the classical strategies discussed in the preceding chapter can be employed, such as a digit-serial multiplication for instance. However, with respect to the techniques of the preceding chapter, there are two notable differences. First, the addition is very easy and especially avoids the carry and the increased bit-length of the result. Second,

reduction is simple with the special-form modulus. This makes the use of sub-quadratic multiplication algorithms such as the divide-and-conquer method of Karatsuba-Ofman [MvOV96] very attractive. This method recursively splits the product into smaller products, minimizing the overall number of digit multiplication at the cost of an increased number of additions. Considering the low complexity of additions, this tradeoff is very valuable in our case.

Normal basis. In normal basis, as sketched earlier, the standard way of multiplying necessitates a multiplication matrix to express the product terms $\theta^{2^i+2^j}$ in the basis. Using such a matrix m (cf. Section 11.2.2 of [ACD+06]), the general term c_h of the product between a and b is:

$$c_h = \sum_{0<i,j\leq n} a_i \, b_j \, m_{i-j,h-j}.$$

If this matrix has particularly few entries then multiplications are faster. The minimal number is $2n-1$; bases achieving this number are called *optimal normal bases*. They are not available for every extension field size, but for $n = 131$ such a basis exists.

To perform the multiplication efficiently, Massey and Omura proposed a multiplier which has the property that the same boolean function can be used to compute each bit of the result; a cyclic shift of the operands only has to be done for each bit [HWB93]. Their technique suggests an interesting iterative multiplier, which can also parallelized. Their approach was refined by Sunar and Koç [ScK01] and Reyhani-Masoleh and Hasan [RMH02], leading to a space complexity of n^2 *and* and $3n(n-1)/2$ *xor* gates for the multiplier. By comparison, the polynomial multiplication requires only n^2 *and* and $(n-1)^2$ *xor* gates. This observation was exploited by von zur Gathen and Shokrollahi [vzGSS07], who proposed an hybrid multiplier of which the core works in polynomial basis by using very efficient basis conversions. When implementing a multiplier solely based on normal basis, the best approach appears to be the highly regular multiplier by Novotnỳ and Schmidt [NS06], which is an adaptation of the Massey-Omura multiplier that can be scaled to any digit width.

3.4.3 Squaring

Since squaring is a simple cyclic shift in normal basis, as shown above, we focus here on the operation in polynomial basis only. In this representation, squaring can be implemented as a special case of the multiplication: as all the

mixed terms disappear in characteristic 2, the squared polynomial is obtained by simply inserting a 0 between two consecutive bits of its binary representation. The cost of a squaring is basically that of the reduction modulo F. As the Certicom challenges are given in polynomial-basis representation with F an irreducible trinomial or pentanomial, the reduction is relatively easy: each bit of the result is computed with mainly a 3-input *xor* gate for a trinomial or a 5-input *xor* gate for a pentanomial.

3.4.4 Inversion

Inversion is by far the most costly of the basic arithmetic operations. Most implementations use one of two algorithms: the Extended Euclidean Algorithm (EEA), published in many papers and books [MvOV96], [ACD+06] and Itoh-Tsujii's method [IT88] which essentially is using Fermat's little theorem.

Extended Euclidean Algorithm. This approach is commonly seen when the elements of $\mathbf{F}_{2^n}$ are represented in polynomial basis. In this basis, each field element f can be regarded as a polynomial over $\mathbf{F}_2$ of degree less than n. The EEA finds elements u and v such that $fu + Fv = 1$. Then $uf \equiv 1 \bmod F$, $\deg u < n$ and thus u represents the multiplicative inverse of f. The classical EEA performs a full division at each step. In practice for $\mathbf{F}_{2^n} \cong \mathbf{F}_2[X]/F(X)$, implementers often choose a version of the EEA that replaces the divisions with division by X (a right shift of the operand). Although this approach requires more iterations, it is generally faster in practice. This method is preferred to the Extended Euclidean Algorithm, according to the comparison of both methods performed in the work by Bulens et al. [BMdDQ06].

According to the work by Bulens et al. [BMdDQ06], the inversion method by Itoh and Tsujii outperform the EEA even in the case of a design in polynomial basis. This is why we consider in this work Itoh-Tsujii method for inversion.

Itoh-Tsujii method. This algorithm proceeds from the observation that in $\mathbf{F}_{2^n}$, $f^{2^n} = f$, $f^{2^n-1} = 1$, and therefore $f^{2^n-2} = f^{-1}$. Instead of performing divisions as in EEA, this algorithm computes the $2(2^{n-1} - 1)$th power. If f is represented in normal basis, squarings are simply a left shift of the operand. Although variants of EEA have been developed for normal basis representation [Sun06], the most efficient approach is generally based on Itoh-Tsujii.

The exponent is fixed throughout the computation, so addition chains can be used to minimize the number of multiplications needed in the calculation of the $2(2^{n-1} - 1)$th power. For $n = 131$, a minimum-length addition chain

to reach 130 is $1, 2, 4, 8, 16, 32, 64, 128, 130$ and the corresponding addition chain on the exponents is $2^1 - 1 = 1, 2^2 - 1, 2^4 - 1, 2^8 - 1, 2^{16} - 1, 2^{32} - 1, 2^{64} - 1, 2^{128} - 1, 2^{130} - 1$. As a result, only 8 multiplications are needed in an inversion in $\mathbf{F}_{2^{131}}$.

Simultaneous inversion. We assume that the arithmetic processor will run multiple walks in parallel. This makes it possible to reduce the cost of the inversion by computing several inversions simultaneously, using a trick due to Montgomery [Mon87]. Montgomery's trick is easiest explained by his approach to simultaneously inverting two elements a and b.

One first computes the product $c = a \cdot b$, then inverts c to $d = c^{-1}$ and obtains $a^{-1} = b \cdot d$ and $b^{-1} = a \cdot d$. The total cost is 1 inversion and 3 multiplications instead of 2 inversions. By extending this trick to N inputs one can obtain inverses of N elements at the cost of 1 inversions and $3(N - 1)$ multiplications.

If N walks are running in parallel on one arithmetic processor and the implementation uses Montgomery's trick to simultaneously invert all N denominators appearing in the λ's above, the cost per point addition decreases to $(1/N)$ inversion, $(2 + 3(N - 1)/N)$ multiplications, and 1 squaring.

3.5 Implementation Results and Cost Assessment

In this section, we describe the performances and area cost we assessed for an ASIC implementation of the iteration step function. As we present the initial stage of a work in progress, we base our estimates on individual implementation results for the finite field arithmetic operators. Our preliminary estimates, while approximate, already provide some intuition about the choice of the most appropriate representation among polynomial and normal bases. They also give an idea about the effort needed to break the whole Certicom challenge ECC2K-130 with ASICs, pointing out the feasibility of the attack. Our results were confirmed and refined in an extended version of this work [BBB+09b]. Note that throughout this section, we give the results for $\mathbf{F}_{2^{163}}$, the size of the next Certicom challenge, after ECC2K-130. They could be exploited in a future work assessing the effort to solve ECDLP for this field size.

Synthesis and Placement. To obtain the estimates presented in this work, we used the UMC L90 CMOS technology, using the Free Standard Cell library from Faraday Technology Corporation characterized for HS-RVT (High

	$\mathbf{F}_{2^{131}}$ (Pre-layout)	$\mathbf{F}_{2^{163}}$ (Pre-layout)
Delay (ps) $\sim$	250	250
Frequency (GHz) $\sim$	4.0	4.0
Flip-Flop Number	131	163
Area (mm^2) $\sim$	0.003	0.004
GE $\sim$	960	1200

Table 3.1: Results for ASIC implementation of squaring in polynomial basis.

Speed Regular Vt) process option with 8 Metal layers. This specific technology was selected because of its wide availability in academic environments. For synthesis results, we employed design compiler 2008.09 from Synopsys, and for placement and routing we used SoC Encounter 7.1 from Cadence Design Systems.

Both synthesis and post-layout analysis use typical corner values, and reasonable assumptions for constraints. The designs are for performance estimate and are not refined for actual production (i.e., they do not contain test structures, a practical I/O subsystem). During synthesis, multiple runs were made with different constraints, and the results with the best area time product have been selected.

3.5.1 Polynomial Basis

We here provide implementation results of the principal arithmetic operations involved in the iteration main loop when using a polynomial-basis representation. In [BMdDQ06, MdDBQ07], Bulens and Meurice de Dormale proposed a hardware architecture for implementing the parallelized Pollard's rho on FPGA. Their core component features the best performances in terms of throughput per consumed area in the literature for polynomial basis. It relies on a fully-pipelined parallel multiplier based on Karatsuba-Ofman. For this work, the authors kindly accepted to put their architecture at our disposal. We thus reuse their arithmetic operators in our estimates. This section can be seen as an extension of their work for the ASIC platform, although our iteration function is significantly different (e.g., we do not store nor update any coefficient at each step).

Addition The addition of n-bits requires n 2-input *xor* gates. No separate synthesis has been made for this circuit. In a practical circuit, the inter-

	$\mathbf{F}_{2^{131}}$		$\mathbf{F}_{2^{163}}$	
	(Pre-layout)	(Post-layout)	(Pre-layout)	(Post-layout)
Delay (ps) $\sim$	470	585	500	575
Frequency (GHz) $\sim$	2.1	1.7	2.0	1.7
Flip-Flop Number	4608	4608	5768	5768
Area (mm^2) $\sim$	0.175	0.185	0.227	0.250
GE $\sim$	55 000	59 000	72 500	80 000

Table 3.2: Results for ASIC implementation of multiplication in polynomial basis.

connection between the adder the rest of the circuit would have a significant impact upon the delay and the area of the circuit. The delay for the addition is approximately 75 ps, while the required area is approximately $n \cdot 2.5$ Gate Equivalents (GEs). No post-layout results are given for this circuit as a standalone implementation is not representative.

Squaring Squaring is performed with a parallel squarer. This operator is relatively inexpensive thanks to the use of a sparse irreducible polynomial, a pentanomial, which is hardwired. Each bit of the result is therefore computed using few *xor* gates. The detailed results for ASIC implementation are reported in Table 3.1, where, due to the small size of this component, no post-layout analysis is provided. Concerning the m-Squaring, this operation is implemented using a single squarer that iteratively computes the m squarings in m clock cycles.

Multiplication The multiplication is achieved by a sub-quadratic Karatsuba-Ofman parallel multiplier, as explained earlier. The operands are recursively divided into two parts, and the small multiplications are performed with simple array multipliers. As for squaring, the modular reduction step is easily performed with a few *xor* gates per bit. The modular multiplier has a pipeline depth of 8 clock cycles in both $\mathbf{F}_{2^{131}}$ and $\mathbf{F}_{2^{163}}$ and produces a result at each clock cycle. An extended description of the multiplier details is given in [BMdDQ06]. The results for the modular multiplication are reported in Table 3.2

Inversion The inversion is based on Fermat's little theorem with the multiplication chain technique by Itoh and Tsujii. An inversion requires 130 squarings and 8 multiplications in $\mathbf{F}_{2^{131}}$ and 162 squarings and 9 multiplications in $\mathbf{F}_{2^{163}}$. The inverter reuses the parallel multiplier described

	$\mathbf{F}_{2^{131}}$ (Pre-layout)	$\mathbf{F}_{2^{163}}$ (Pre-layout)
Delay $(ps) \sim$	410	460
Frequency $(GHz) \sim$	2.4	2.15
Flip-Flop Number	0	0
Area $(mm^2) \sim$	0.044	0.061
GE $\sim$	13900	19400

Table 3.3: Results for ASIC implementation of polynomial to normal basis converter.

above, together with a block of several pipelined consecutive squarers (see Figure 2 in [MdDBQ07]). This block allows to speed up the consecutive squarings required in the inversion. It is done by putting several squarers in a serial way, so that every bit of the result is now computed with a larger number of *xor* gates, depending on the number of squarings to be performed. The block of consecutive squarers implements the different numbers of squarings specified in the technique of Itoh and Tsujii. For $\mathbf{F}_{2^{131}}$, it contains the following squarers put in a serial way: $x^{2^1}, x^{2^1}, x^{2^2}, x^{2^4}, x^{2^{25}}$, such that the various powers of Itoh and Tsujii are reachable by looping at most twice over this block.

The inversion is done by looping over the multiplier and the block of squarers according to the technique of Itoh and Tsujii. For this purpose, extra shift registers are required. The inverter has a pipeline depth of 16 and achieves an inversion with a mean delay of 10 and 11 clock cycles in $\mathbf{F}_{2^{131}}$ and $\mathbf{F}_{2^{163}}$ respectively. For the parallel pipelined inverter described above, a lower bound on the area requirement can be obtained by gathering the results for the squarers and the multiplier. The area of the block of squarers is approximated to 30 000 GE for $\mathbf{F}_{2^{131}}$ and 50 000 for $\mathbf{F}_{2^{163}}$, based on the single squarer cost. This leads to a lower bound of around 90 000 GE and 130 000 GE for the full parallel inverter on $\mathbf{F}_{2^{131}}$ and $\mathbf{F}_{2^{163}}$ respectively.

The inverter has a much higher area cost and lower throughput with respect to the multiplier. From an area-time point of view, it is interesting to combine several inversions using the Montgomery trick instead of replicating several inverters when trying to increase the throughput of inversions. As each combined inversion requires 3 multiplications to be performed, it seems to be always interesting to use the Mont-

	addition	squaring	m-squaring	multiplication	inversion
$\mathbf{F}_{2^{131}}$	1	1	m	1	10
$\mathbf{F}_{2^{163}}$	1	1	m	1	11

Table 3.4: Cycle counts for field operations in polynomial basis.

gomery trick instead of replicating several inverters. In practice, the gain of using Montgomery's trick can be smaller than expected, as stated in [BMdDQ06]. Simultaneous inversion does not require a specific operator as it can be built upon the multiplier and the inverter.

Conversion A conversion from polynomial to normal basis is required with our iteration function as it is defined in normal basis to benefit from the invariance of the Hamming weight for the whole class of point in this representation. This operation requires a conversion matrix which express the normal basis in terms of the polynomial one. The bits of the element in normal basis can be computed in parallel. As a result, one clock cycle is sufficient to perform a conversion. The detailed results for this operation are reported in Table 3.3. The cycle counts for the field operations in polynomial basis are indicated in Table 3.4.

3.5.2 Normal Basis

We now present the implementations results of the arithmetic operations for the normal-basis representation. As opposed to polynomial basis, the implementation is in its initial stage. For this reason, the chosen architecture is iterative, and it is foreseen to modify it toward a parallel behaviour, in order to explore the various options to minimize the area-time product.

In our description of the various arithmetic circuits, we directly move to squaring, as addition is performed in the same way as in the polynomial basis.

Squaring In normal basis, the squaring is equivalent to a rotation, thus the impact on the area and delay of this component will be mainly due to the interconnections.

For m-Squaring, the operation can be achieved by an iterative use of a single squarer as for the polynomial basis. Alternatively, since a squaring is equivalent to a rotation, one can anticipate the m rotations and perform this operation in a single clock cycle. However, the various

	$\mathbf{F}_{2^{131}}$		$\mathbf{F}_{2^{163}}$	
	(Pre-layout)	(Post-layout)	(Pre-layout)	(Post-layout)
Delay $(ps) \sim$	510	550	485	576
Frequency $(GHz) \sim$	2.0	1.8	2.0	1.75
Flip-Flop Number	402	402	661	611
Area $(mm^2) \sim$	0.015	0.016	0.025	0.027
GE $\sim$	5000	5250	7900	8800

Table 3.5: Results for ASIC implementation of the field element multiplier in normal basis.

choices for m require to include a m-input multiplexer to select the appropriate result. As a result, we prefer to resort to the iterative option.

Multiplication For multiplication, we implemented the Massey-Omura multiplier described previously. This multiplier computes each bit of the result based on the same boolean circuit; the inputs must be rotated at each result bit. The design computes a single bit of the product in each clock cycle. Because of this relatively slow performance, it consumes very little amount of area. The detailed results for the modular multiplication are reported in Table 3.5.

Inversion Also in normal basis, the inversion is implemented using Fermat's little theorem, thus the considerations held for polynomial basis are relevant in this case as well. The 8 multiplications needed in an inversion are performed with the Massey-Omura multiplier, while the multiple squarings come now almost for free: as the number of repeated squarings is fixed, they can be anticipated with interconnections only and achieved in a single clock cycle. The cycle counts of the inversion and the other operations in normal basis are shown in Table 3.6. For the inversion, the 8 multiplications mainly compose the latency.

The area requirement of the inverter should not be significantly higher than the cost of the multiplier, since the multiple squarings can be performed with interconnections only. Therefore, it should be quite close to the cost of the multiplier, i.e., 5250 GE for $\mathbf{F}_{2^{131}}$.

As for the polynomial inverter, it is appealing to perform simultaneous inversions to decrease the average duration of an inversion. For instance, combining 5 inversions theoretically lowers the average cycle count for the inversion by about 50% (including the extra multiplications overhead). However, as noted in [BMdDQ06], the gain of this method may

	addition	squaring	m-squaring	multiplication	inversion
$\mathbf{F}_{2^{131}}$	1	1	m	131	1065
$\mathbf{F}_{2^{163}}$	1	1	m	163	1629

Table 3.6: Cycle counts for field operations in normal basis.

be inferior than expected and should be assessed once the whole processor is assembled.

3.5.3 Comparing Both Representations

With the implementation results of the individual arithmetic blocks given above for polynomial and normal-basis representations, we can now attempt to weigh these options for a practical ASIC realization of Pollard's rho.

Cost of one step. A step in Pollard's rho on ECC2K-130 mainly requires 1 inversion, 2 multiplications, 1 squaring, 2 m-squarings and the additional conversion to normal basis if the computations are done in polynomial basis.

In polynomial basis, this should take around 30 cycles in average, based on the cycle counts of Table 3.4 and 7 cycles in average for the m-squaring (as m is uniformly taken in $\{3, 4, ..., 10\}$). The same estimate for normal basis gives 1335 cycles, considering this time Table 3.6 and the fact that the 2 m-squarings are computed in parallel (as squaring is trivial in normal basis).

A larger cycle count for the normal basis is due to the iterative approach of the architecture in this basis as opposed to the parallel design used for the polynomial basis. These cycles counts do not consider the use of simultaneous inversion as we prefer to study the gain of this technique once the whole step is implemented, following the recommendations of [BMdDQ06]. Nevertheless, this technique is likely to be significantly beneficial.

Concerning the area, a lower bound for the circuit achieving the step can be determined by gathering the costs of the operators. In polynomial basis, this gives a cost of 105 000 GE, which is mostly due to the inverter, as its multiplier can also be shared to perform the two multiplications needed in each step, and to the circuit for conversion. In normal basis, the same estimate leads to 6000 GE (mainly the cost of the multiplier).

Area-Time comparison. From an area-time point of view, the processor relying on the polynomial basis appears to be about twice more efficient: it is around 40 times faster and only 20 times larger. However, this comparison

must be moderated for the following reasons. First, the designs for both representations are not equally optimized: the area-time product of the normal-basis implemention can surely be improved by parallelizing the computations and introducing pipeline, as for the design in polynomial basis. Second, as described in Section 3.4.2, there are several known options to improve the normal-basis multiplier, such as the hybrid technique by von zur Gathen and Shokrollahi [vzGSS07]. As the multiplier is the core component of the step circuit, any advance concerning it is expected to be significantly advantageous for the whole step.

This comparison indicates that the polynomial-basis design is about twice more efficient in terms of Area-Time product with respect to the normal-basis one. In the literature, the normal-basis option was considered to be much less attractive than the polynomial-basis one. We show here that the gap between the two representations is not that wide. The circuit in normal basis, when fully optimized, is expected to be very competitive; it might even turn out to be the best option for Pollard's rho in hardware.

The results presented here are approximations of what the circuit for the entire step should behave: our estimates do not capture the various overheads, like memory. The uncertainty around these preliminary results is intended to be lifted in a forthcoming work; this is partially done in [BBB$^+$09b].

3.5.4 Cost of the Full Attack on ECC2K-130

Although the implementation of the whole iteration step has not fully been completed, it is still possible to obtain a first order estimate of the cost and performance of an ASIC that could be used for the ECC2K-130 challenge.

The performance of the components described in the preceding sections were all post-layout figures that include interconnection delays, clock distribution overhead and all the required registers. Let us now assess what the clock frequency could be for a real chip. Supplying a clock externally to the chip would limit the frequency to around 500-600 MHz. To obtain a higher frequency, we consider that the chip integrates a Phase-Locked Loop (PLL) to distribute the internal clock. We therefore estimate that the overall clock rate could reach around 1 GHz when all the components are combined into one system.

For the ASIC, we will consider using a standard 16 mm^2 die in the 90 nm CMOS technology using a Multi-Project Wafer (MPW) production in prototype quantities. The cost of producing and packaging 50-250 of such ASICs is estimated to be less than 60 000 €. When using the specific standard cell library, the 16 mm^2, 90 nm CMOS die has a net core area of 12 mm^2, which

could support approximately 3 000 000 gates with generous overhead for power routing and placement. Reserving 1 000 000 gates for PLL, I/O interface (including memory for storing results), we will consider that only 2 000 000 gates will be available for computation units.

The design in polynomial basis described above is considered for the attack, as it is currently our most efficient architecture. Its area occupation, 105 000 GE for the arithmetic, should be largely bounded by 200 000 GE once the overheads like memory are included. Therefore, a single ASIC should thus be able to support at least 10 cores in parallel. Considering that each core requires 30 clock cycles per iteration, such an ASIC would be able to calculate approximately 330 Miterations/s.

Breaking ECC2K-130 in one year would require to achieve the $2^{60.9}$ expected iterations at a rate of 69 000 Miterations/s. Even our overly pessimistic estimation shows that this performance can be achieved by a modest collection of around 200 dedicated ASICs, thus allowing the attack for a hardware cost of about 60 000 €.[1]

3.6 Related Works

Several preceding works attempted to assess the cost of resolving ECDLP on binary fields using hardware. In this section, we first describe these works and then present the ECRYPT distributed attack in the framework of which this work has been achieved.

Previous works. In 1999, van Oorschot and Wiener [vOW99] made a rough estimation of the cost of an attack relying on dedicated hardware to solve ECDLP over $\mathbf{F}_{2^{155}}$. Based on the ASIC implementation of [AMV93], they assessed that the problem could be solved for US$ 10 million in 32 days. However, the underlying ASIC implementation uses projective coordinates and therefore cannot directly allow the collision detection needed in the attack as this representation is redundant. Affine coordinates should be employed, largely decreasing the overall design performances as an inversion is necessary.

The first real implementation on special-purpose of Pollard's rho for curves over characteristic-two finite fields was achieved by Bulens and Meurice de Dormale [BMdDQ06, MdDBQ07]. As discussed in Section 3.5, their design

[1] Frank K. Gurkaynak (ETH Zürich Microelectronics Design Center) is acknowledged for this estimation.

was parallel and fully pipelined; we reused in this work several of their building blocks for the arithmetic. They did not specifically target the Koblitz curves and employed a less optimized iteration function (avoiding the benefits of the negation map, and updating coefficients at each step). They concluded that solving the Certicom challenge over $\mathbf{F}_{2^{155}}$ would require about US$ 20 million using COPACABANA engines (containing each 120 Spartan3-1000 FPGAs). A straightforward comparison with our results is difficult considering the different hardware and iteration function, and our specialization to Koblitz curves. Nevertheless, our analysis concerning ASICs completes the work by Bulens and Meurice de Dormale and goes even further by considering the choice of normal basis for the field arithmetic and an iteration step optimized for Koblitz curves.

ECRYPT distributed attack. The work presented in this chapter has been undertaken within the framework of a more general study initiated by the ECRYPT Vampire working group [VAM10] to establish the overall effort required to break the ECC2K-130 challenge. The global study, covering many platforms such as GPU, FPGAs and ASICs, was first presented at SHARCS 2009 [BBB$^+$09a] and later improved in [BBB$^+$09b]. This work zooms into the ASIC part of this study. As it is currently in its initial stage, our results concerning the interest of the normal-basis representation and the cost of an attack based on ASICs are expected to be refined in future works. [FBB$^+$10] is one of them: in this paper to appear, Fan et al. provide a deep comparison of the multiplication options to solve ECC2K-130 with FPGAs. Especially, they compare three architectures for the iteration step: one based on polynomial basis, one in type-II optimal normal basis and the third exploiting the hybrid multiplier of Shokrollahi (cf. Section 3.4). They highlight the supremacy of this last approach and confirm the interest of the normal-basis representation for solving ECDLP on Koblitz curves. The Shokrollahi's method for multiplication has also received the attention of Berstein and Lange within the ECRYPT framework: in [BL10], the authors present several improvements to the method, reducing the multiplication overhead by 1.4 in a software implementation.

Besides studying the effort of breaking ECC2K-130 with state-of-the art hardware and software implementations, the partners of the ECRYPT Vampire working group are also carrying out a real attack based on distributed computing resources: clusters of conventional computers, PlayStation 3 clusters, computers with powerful graphics cards and FPGAs.

The progress of the attack is reported online at `www.ecc-challenge.`

`info/`. With the considered iteration function, a distinguished point is found with on average $2^{25.27}$ iterations; the total computation time of $2^{60.9}$ iterations corresponds to finding about $2^{35.63}$ distinguished points. With the compressed representation described in [BBB+09b], 850 GB of storage are needed. To this date (August 2010), 65 GB have already been received by the servers. The expected end of the attack largely depends on the computing power available to the project partners. In the near future, the attack is going to benefit from the power of an FPGA cluster (128 Spartan-3 XC3S5000 FPGAs). Five of these could resolve the challenge in one year [FBB+10]. Remark that a powerful organization with much more means compared to this academic effort should have already resolved the ECC2K-130 challenge.

This work has shown that the key-length of 131 bits is insecure in the case of Koblitz curves. The larger key sizes are protected by the fast-growing complexity of ECDLP. For instance, for the next challenge with key-lengths of $n = 163$ bits, the expected number of iterations $\sqrt{\pi n/4}$ grows with a factor 65536 with respect to $n = 131$ (the iterations being also more expensive). Concerning the non-Koblitz binary curves, the absence of the Frobenius map increases the time complexity of an attack by a factor of $\sqrt{n}$; they thus offer more security per bit.

Bibliography

[ACD+06] Roberto Avanzi, Henri Cohen, Christophe Doche, Gerhard Frey, Tanja Lange, Kim Nguyen, and Frederik Vercauteren. *Handbook of Elliptic and Hyperelliptic Curve Cryptography*. CRC Press, 2006.

[AMV93] Gordon B. Agnew, Ronald C. Mullin, and Scott A. Vanstone. An Implementation of Elliptic Curve Cryptosystems Over $F_{2^{155}}$. *IEEE Journal on Selected Areas in Communications*, 11(5):804–813, 1993.

[BBB+09a] Daniel V. Bailey, Brian Baldwin, Lejla Batina, Daniel J. Bernstein, Peter Birkner, Joppe W. Bos, Gauthier van Damme, Giacomo de Meulenaer, Junfeng Fan, Frank Gurkaynak, Tim Gneysu, Thorsten Kleinjung, Tanja Lange, Nele Mentens, Christof Paar, Francesco Regazzoni, Peter Schwabe, and Leif Uhsadel. The Certicom Challenges ECC2-X. In *Workshop on Special Purpose Hardware for Attacking Cryptographic Systems (SHARCS'09)*, 9 2009.

[BBB+09b] Daniel V. Bailey, Lejla Batina, Daniel J. Bernstein, Peter Birkner, Joppe W. Bos, Hsieh-Chung Chen, Chen-Mou Cheng, Gauthier Van Damme, Giacomo de Meulenaer, Luis Julian Dominguez Perez, Junfeng Fan, Tim Güneysu, Frank Gürkaynak, Thorsten Kleinjung, Tanja Lange, Nele Mentens, Ruben Niederhagen, Christof Paar, Francesco Regazzoni, Peter Schwabe, Leif Uhsadel, Anthony Van Herrewege, and Bo-Yin Yang. Breaking ECC2K-130. Cryptology ePrint Archive, Report 2009/514, 2009. `http://eprint.iacr.org/2009/541/`.

[BL10] Daniel Bernstein and Tanja Lange. Type-II Optimal Polynomial Bases. In *International Workshop on the Arithmetic of Finite Fields (WAIFI 2010)*. Springer, 2010.

[BMdDQ06] Philippe Bulens, Guerric Meurice de Dormale, and Jean-Jacques Quisquater. Hardware for Collision Search on Elliptic Curve over GF(2^m). In *Proceedings of SHARCS'06*, 2006. `http://www.hyperelliptic.org/tanja/SHARCS/talks06/bulens.pdf`.

[BP81] Richard P. Brent and John M. Pollard. Factorization of the eighth Fermat number. *Mathematics of Computation*, 36:627–630, 1981.

[Cer97] Certicom. Certicom Elliptic Curve Cryptosystem Challenge. `http://www.certicom.com/images/pdfs/cert_ecc_challenge.pdf`, 1997.

[ECR10] ECRYPT II. European Network of Excellence in Cryptology II . Network of excellence funded within the Information & Communication Technologies (ICT) programme of the European Commission, under contract ICT-2007-216676. `http://www.ecrypt.eu.org/index.html`, 2010.

[FBB+10] Junfeng Fan, Daniel V. Bailey, Lejla Batina, Tim Güneysu, Christof Paar, and Ingrid Verbauwhede. Breaking Elliptic Curves Cryptosystems using Reconfigurable Hardware. In *2010th International Conference on Field Programmable Logic and Applications (FPL 2010)*. IEEE, 2010.

[Flo67] Robert W. Floyd. Nondeterministic algorithms. *Journal of the ACM (JACM)*, 14(4):636–644, 1967.

[GLV00] Robert P. Gallant, Robert J. Lambert, and Scott A. Vanstone. Improving the parallelized Pollard lambda search on anomalous binary curves. *Mathematics of Computation*, 69(232):1699–1705, 2000.

[Har60] Bernard Harris. Probability distributions related to random mappings. *The Annals of Mathematical Statistics*, 31:1045–1062, 1960.

[Har00] Robert Harley. Elliptic Curve Discrete Logarithms Project. `http://pauillac.inria.fr/~harley/ecdl/`, 2000.

[HMV04] D. Hankerson, A. Menezes, and S. Vanstone. *Guide to Elliptic Curve Cryptography*. Springer, 2004.

[HWB93] M. A. Hasan, M. Z. Wang, and V. K. Bhargava. A Modified Massey-Omura Parallel Multiplier for a Class of Finite Fields. *IEEE Trans. Comput.*, 42(10):1278–1280, 1993.

[IT88] Toshiya Itoh and Shigeo Tsujii. A Fast Algorithm for Computing Multiplicative Inverses in $GF(2^m)$ Using Normal Bases. *Inf. Comput.*, 78(3):171–177, 1988.

[MdDBQ07] Guerric Meurice de Dormale, Philippe Bulens, and Jean-Jacques Quisquater. Collision Search for Elliptic Curve Discrete Logarithm over GF(2^m) with FPGA. In *Workshop on Cryptographic Hardware and Embedded Systems (CHES 2007)*, pages 378–393. Springer, 2007.

[Mon87] Peter L. Montgomery. Speeding the Pollard and elliptic curve methods of factorization. *Mathematics of Computation*, 48:243–264, 1987.

[MvOV96] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[NS06] M. Novotnỳ and J. Schmidt. Two Architectures of a General Digit-Serial Normal Basis Multiplier. In *Proceedings of 9th Euromicro Conference on Digital System Design*, pages 550–553, 2006.

[Pol78] John M. Pollard. Monte Carlo methods for index computation (mod p). *Mathematics of Computation*, 32:918–924, 1978.

[RMH02] A. Reyhani-Masoleh and M.A. Hasan. A New Construction of Massey-Omura Parallel Multiplier over GF(2^m). *IEEE Transactions on Computers*, 51:511–520, 2002.

[ScK01] B. Sunar and Ç. Koç. An Efficient Optimal Normal Basis Type II Multiplier. *IEEE Transactions on Computers*, 50:83–87, 2001.

[Ser98] Gadiel Seroussi. Compact representation of elliptic curve points over GF(2^n). Technical report, Research Contribution to IEEE P1363, 1998.

[Sha71] Daniel Shanks. Class Number, a Theory of Factorization, and Genera. In *1969 Number Theory Institute (Proc. Sympos. Pure Math., Vol. XX, State Univ. New York, Stony Brook, N.Y., 1969)*, pages 415–440. Providence, R.I., 1971.

[Sun06] Berk Sunar. A Euclidean algorithm for normal bases. *Acta Applicandae Mathematicae*, 93:57–74, 2006.

[Tes01] Edlyn Teske. On random walks for Pollard's rho method. *Mathematics of Computation*, 70(234):809–825, 2001.

[VAM10] VAMPIRE. Virtual Applications and Implementations Research Lab. `http://hyperelliptic.org/ECRYPT/vampire/`, 2010.

[vOW99] Paul C. van Oorschot and Michael J. Wiener. Parallel Collision Search with Cryptanalytic Applications. *Journal of Cryptology*, 12(1):1–28, 1999.

[vzGSS07] Joachim von zur Gathen, Amin Shokrollahi, and Jamshid Shokrollahi. Efficient multiplication using type 2 optimal normal bases. In *WAIFI*, pages 55–68, 2007.

[WZ98] Michael J. Wiener and Robert J. Zuccherato. Faster attacks on elliptic curve cryptosystems. In *Selected Areas in Cryptography*, volume 1556 of *LNCS*, pages 190–200, 1998.

Part II

Cryptography for Low-Power Devices

CHAPTER 4

Overview of Constrained Devices and Lightweight Cryptography

In the second part of the thesis, we focus on the emerging technology of small embedded devices. Our general goal is to provide robust cryptography to these weak devices, by taking advantage of the power and energy resources that are available in their context. In this preliminary chapter, we first describe the small hardware considered throughout this part of the thesis, explaining their constraints and security needs. We then review the existing means to secure them, with an overview of lightweight cryptography. After a brief presentation of the performances and resource requirements of lightweight implementations, we recall what can be done at the protocol level to reduce the cost of cryptography for weak devices, with a special attention to cooperative techniques, of which an example will be extensively discussed in Chapter 6.

4.1 Introduction

The second part of the thesis studies the security of on an emerging technology, the small embedded devices. The miniaturization of electronic components and their dramatic cost reduction have allowed the development of low-cost tiny devices with very limited processing capabilities, which are of interest in a huge number of applications. The growing use of these devices brings closer to reality concepts like the “Internet of things” or “smart dust”, with all the benefits it provides, such as a better control by the human being of its environment or an easier way to accomplish many everyday tasks. However,

the advent of this technology reveals in the same way its drawbacks, which are becoming more and more visible. Especially, the presence of small pervasive devices is often considered as a threat to privacy. Such considerations are likely to slow down the adoption of this promising technology.

Small embedded devices operate in an increasing number of sensitive applications, such as access control or infrastructure monitoring, where a high level of security is required. However, their limited resources prevent the straightforward use of well-known cryptographic techniques employed in many other contexts. As a result, a sound level of security may be hard to achieve when using constrained devices.

These security concerns are at the root of the work presented in this part of the thesis. In the next two chapters, we will attempt to facilitate the use of robust cryptographic techniques for small devices, starting from an understanding of the resources in terms of energy or computational power that are available in their context. But first, there is this preliminary chapter, of which the outline is given below.

In this chapter, we begin with the description of small embedded devices, detailing their constraints and security needs (Section 4.2). We dedicate a special attention to wireless sensor nodes and RFID tags, which are the platforms illustrated in the next two chapters. Afterward, we provide a short survey of the existing means to enable cryptography on small devices, i.e., lightweight cryptography (Section 4.3). We start by giving the performances of cryptographic algorithms when implemented in small devices. The aim is to give some intuition about what is currently feasible on small devices and at which cost. As we will see, the cost of the so-called lightweight cryptographic implementations is not always tractable for very constrained hardware. We then go beyond the implementations by examining what can be done at the protocol level to obtain lightweight cryptography. Especially, we briefly focus on the cooperative cryptographic techniques (Section 4.3.2), since we apply their basic idea to enable group signatures on small devices, as presented in Chapter 6.

4.2 Constrained Devices

The expression “constrained devices” encompasses a large variety of embedded devices, such as laptops, phones, personal digital assistants or smart cards. All these platforms are constrained in one or several ways: they may have strict limitations in terms of computing power, memory, energy, size, bandwidth, etc. In this thesis, we restrict ourselves to the very small embedded devices with wireless abilities like RFID tags and wireless sensor nodes. In the subse-

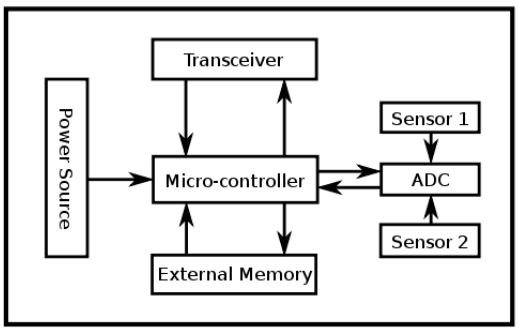


Figure 4.1: Wireless sensor node architecture.

quent descriptions, we dedicate a special attention on the latter kind of device, as sensor nodes were our testing platforms in the methodology introduced in Chapter 5 and in the implementations described in Chapter 6.

4.2.1 Wireless Sensor Nodes

Wireless sensor nodes are tiny devices, which are deployed in an environment where their task is to acquire and process physical data, such as classically temperature, pressure or humidity. Their applications are numerous, such as infrastructure monitoring (building, bridge, pipeline), user surveillance (road traffic, elderly people at home, perimeter protection in military applications), machine monitoring (power plants, plane structure, ship engines), phenomena detection (pollution, fire) and many others. Sensor nodes can also easily detect movements by exploiting the variation in the strength of the received signal. It is thus possible to localize a person inside a house by surrounding the building with sensor nodes [WP10].

Sensor nodes have few resources –especially energy– to meet high security requirements. Here, we first depict their constituting elements and describe exemplary sensor node platforms. This is followed by a discussion of the energy constraints and the security needs of these nodes.

Platform Description

Wireless sensor nodes are built following the typical architecture shown in Figure 4.1. Their components are intended to be cheap: Both the microcontroller and the transceiver can be acquired for a few euros for instance. The relatively high price of the available commercial platforms, around 100 €, is expected to collapse once this technology reaches mass production. Below, we detail the

sensor main components and give a description of two well-known platforms, the MICAz [Croa] and the TelosB [Crob], extensively used in our experiments in the next chapters.

Power Source. The power source is usually a battery, although it could be an energy scavenger system exploiting e.g., sunlight or vibration. In the latter case, the level of power may be as low as 10–100 μW [CDV+05]. When using a battery, the energy must be carefully managed to maximize the node lifetime, as battery replacements involve manpower, which is costly compared to cheap sensor nodes. Typically, a node powered by a pair of AA batteries is likely to spend the available 5000 mAh within several months, possibly more than a year if its duty cycle is very low.

Microcontroller. The microcontroller is the central part of the node. Its role is to control the peripheral components and to preprocess the raw acquired data to limit the bandwidth needs. Wireless sensor node microcontrollers have usually a small operand width (8-bit or 16-bit) and a low frequency (typically less than 10 MHz) even if some more powerful nodes may feature a 32-bit processor such as the ARM7 [Sun09] and even an FPGA [AKR+07]. Sensor nodes usually lacking task-specific hardware, their processing power is rather weak.

The microcontroller memory provides in general a comfortable amount of space to support wireless sensor networks applications, although an external memory is often used for data logging. However, large cryptographic implementations may necessitate a large fraction of it. Moreover, as storing precomputations is very valuable to save energy, the memory constraint may be important when securing sensor nodes.

Sensor nodes usually operate in a periodic manner with a small duty cycle. As a result, their power consumption must not only be very low in active mode, but also in the sleep mode. If this is not the case, it makes sense to add a second microcontroller with a very low sleep consumption. The main controller can be totally switched off and turned on only when needed by the second processor. The Sun SPOT node [Sun09] implements this strategy.

Transceiver. The transceiver is designed to fit the sensor node behaviour, i.e., to periodically exchange small amounts of data within a rather limited range (up to 50-100 m). Therefore, packet frames are small (usually 128 bytes), data rates are low (typically 250 kbps) and the transmit power is moderate (around 1 mW).

Figure 4.2: The MICAz node.

Figure 4.3: The TelosB node.

Most sensor node transceivers are compliant with the IEEE 802.15.4 standard [IEE10], which is dedicated to networks like sensor networks. This standard organizes the physical layer and the media access control. It makes use of Direct Sequence Spread Spectrum (DSSS) to minimize the risk of interferences and uses Carrier Sense Multiple Access (CSMA) to get connected to the physical medium. The upper layers of the protocol stack are specified by several networking protocol suites, among which the most famous is the proprietary ZigBee [Zig10] specification, which has become the name designating the technology. ZigBee and IEEE 802.15.4 rely on the Advanced Encryption Standard with 128-bit keys (AES-128 [DR02]) to provide encryption and data authentication. Therefore, many transceivers include an AES hardware engine.

Besides the microcontroller, the transceiver is the other main component in the node energy consumption. Its power consumption is rather high, usually superior to the microcontroller one. This happens in the three main operating modes: transmit, receive, but also in the listen mode. The time spent while listening must thus be kept as short as possible. In ZigBee, this is achieved by synchronizing senders and receivers using a coordinating powerful node transmitting periodic beacons. The synchronization allows nodes to go to the idle mode while waiting for their time slot. However, this mode only applies for networks having a star topology. For the more general mesh topology, no strategy is proposed in the standard. Research papers have proposed many alternative networking protocols to achieve low-power listening, in either a synchronous (e.g. [YHE02]) or asynchronous way (e.g., [PHC04, MHK10]).

Two Examples of Sensor Nodes. We end the sensor node platform description by detailing two representative sensor node platforms, the MICAz and the TelosB (Figures 4.2 and 4.3). Their microprocessors come from different families and have different word sizes : 8-bit for the MICAz and 16-bit for the TelosB. Both nodes support the popular TinyOS [Tin10] operating system

	MICAz [Croa]	TelosB [Crob]
Microcontroller	ATmega128L	MSP430
Op. width	8-bit	16-bit
Freq.	7.37 MHz	4 MHz
RAM	4 kB	10 kB
ROM	128 kB	48 kB
Software	TinyOS	TinyOS
Transceiver	CC2420	CC2420
Thr.	250 kbps	250 kbps
Battery	2 AA	2 AA
Sensors	Board to plug	Temp., Hum., Light

Table 4.1: Features of the MICAz and TelosB sensor nodes.

and share the same CC2420 [Tex10] transceiver. Their main design features are displayed in Table 4.1. In the next chapters, these two nodes are used as testing platforms.

Energy Constraint

The description of the sensor node hardware has highlighted many constraints inherent to it such as a low cost, a small memory, a modest bandwidth, a low computing power and a limited amount of energy. Among these constraints, energy is the most important one. Indeed, the energy budget must be carefully managed to ensure the node autonomy during the expected duration of the application. Minimizing the energy can extend the sensor node lifetime. Alternatively, the saved energy can be used to supply hardware components with better or more advanced functionalities, enhancing the sensor node range of applications.

From the security point of view, more available energy can be translated in a higher level of security through the use of more secure (and consuming) implementations or techniques.

The technological trend confirms the importance of the energy constraint for sensor devices. On the one hand, no dramatic progress in the battery technology is foreseen in the near future. On the other hand, energy scavengers are increasingly considered as an alternative to batteries and their costly replacement. Minimizing energy brings thus closer the spread use of energy scavengers in sensor nodes.

Security Concerns

When securing wireless sensor networks, the basic goals are the confidentiality and the authentication of the data being exchanged. For this, the IEEE 802.15.4 standard has foreseen the use of encryption with AES-128, which is implemented in a cryptoprocessor present in the transceivers adhering to the standard.

While sufficient for many applications, the straightforward application of symmetric-key based key exchange may not be easily transposable to very large networks (given the considerable number of keys to store on each node) or to applications requiring advanced security functionalities, such as secure data aggregation. As a result, other cryptographic primitives may be of interest, such as public-key cryptography. Of course, the choice of the cryptographic methods must be done in accordance to the tight energy constraint of the sensor nodes, which requires an appropriate method for obtaining the energy cost of cryptographic methods, as discussed in the next chapter.

The physical security of the sensor platforms is probably the biggest issue in the security of wireless sensor networks. Being deployed in a potentially hostile environment, the nodes are easily reachable for an attacker, who can try to tamper with them to obtain the cryptographic secrets. Once a node is compromised, various subsequent insider attacks can be performed. Having few resources to dedicate to tamper-proof mechanisms, these devices can be easily captured. This kind of attack is very efficient and can be done in a stealthy way, as described in [dMS10]. Cryptographic implementations for sensor networks should ideally provide some protection against physical attacks, as it makes little sense to use strong cryptographic algorithms if the implementations can be easily attacked. However, this may be too costly and defenses at the protocol level may be an alternative, such as the protocol tolerant to node capture presented in [PPG05].

Besides physical security, there are many security issues that are being investigated in the research community, such as secure routing, denial of service, privacy, secure group management, ... More details concerning the security of wireless sensor networks and its challenges can be found in [PSW04].

4.2.2 Radio-Frequency Identification Tags

This section describes another type of common constrained device, the Radio-Frequency IDentification transponders, best known as RFID tags. Like many other inventions, the first uses of RFID tags were military. In the forties, tags were employed as espionage tool by the Russians. Radio-identification was

Figure 4.4: Example of high-end RFID transponder: the MOBIB contactless smart card used in Brussels public transportations [STI10].

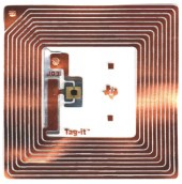


Figure 4.5: Standard low-cost RFID tag [Ste10].

also employed by Great Britain to distinguish friendly aircraft from enemy aircraft during the Second World War. Since the 1980's, the number of RFID tags involved in commercial applications has grown exponentially and has reached the impressive number of several hundred million tags sold per year.

Besides tracking goods, there is a huge number of applications for RFID systems. Examples of applications are: public transportation, toll road systems, automotive ignition keys, electronic documents and passport, building access control, ... Often, advanced RFID tags take the form of smart cards equipped with RFID capabilities. The smart card technology has thus rejoined RFID systems with the contactless smart cards, which are increasingly used in everyday life.

After a quick description of the RFID tag, we discuss its main constraint, power, and its security concerns.

Device Description

An RFID tag is a small device containing a micro-circuit and an antenna to enable it to receive and wirelessly answer queries from an RFID reader. The distance between the reader and the tag is usually small, typically several centimeters. These devices exist in many configurations, depending on their application (Figures 4.5 and 4.4). While the cheapest tags only contain a small memory storing a unique identifier, more advanced tags can be equipped with a microcontroller and computing capabilities allowing the use of symmetric and even asymmetric cryptography. For instance, the tags contained in certain passports embed a cryptoprocessor supporting AES, triple DES, RSA and ECC operations [Ins10]. Of course, the cost of cryptography-enabled tags is more expensive, on the order of several euros, while the simplest tags may be only a few cents worth.

Power Constraint

Being usually batteryless, RFID tags scavenge energy out of the incoming electromagnetic waves from the reader. Energy is not therefore a direct constraint, as the tag may remain in the reader field for an arbitrary long period, although the time of interaction should remain practical.

On the other hand, the strength of the signal from the reader rapidly decreases with the communication range, leaving a very limited power for the tag. The available power typically cannot exceed 10-50 μW [CR07] for cheap tags and a few mW for high-end RFID transponders, the latter devices having to be brought much closer to the reader (less than 10 cm).

As a result, the computing power of the tag must be very limited to meet the hard power constraint. This is why we consider power as the most important constraint for RFID tags, although it is its consequence – a weak computing power – which is perceived in practice. Note that the cost pressure is high on these devices, which contributes to the use of very small computational resources, which are avoided whenever it is possible.

Security Concerns

Many of the RFID applications are sensitive and therefore require a sound security level. We identify three main issues in the security of RFID systems:

- **Malicious traceability**. RFID tags threaten privacy, as they enable to easily collect informations about the moves of a person carrying them. For example, every person might be traced by the infrastructure owner in public transportations, corporate buildings, ...

- **Information leakage**. RFID tags always respond to the queries of readers, even when the read is not authorized. As a result, sensitive information may be revealed. This also constitutes a threat to privacy if the information is personal. For instance, the MOBIB card for public transportation in Brussels (displayed in Figure 4.4) discloses the identity, the birthdate, the postal code and the three latest rides of the card owner at each read of the card, whether authorized or not [Avo10].

- **Impersonation**. Using an information leakage from a tag, an adversary may be able to simulate it in front of a legitimate reader. This may work if no tag authentication is performed during the interactions with the reader. As a result, the adversary may obtain the right attached to the tag, e.g., the access to a restricted perimeter. Authentication techniques

must be carefully designed, as underlined by the recent break of the Keeloq cipher [CBW08], which is widely used in remote keyless entry applications or car locks. And even if it is the case, relay attacks [KW05] may still provide a way to impersonate a tag.

Additionally, like other constrained devices, RFID tags have a weak tamper resistance [HMF07].

In this thesis, we are mainly interested in the privacy concern of RFID systems. In Chapter 6, we will consider the adaptation of group signatures to small devices like RFID transponders. Group signatures allow any member of a group to anonymously sign on behalf of the group (they are formally defined in Section 6.2). They therefore represent an interesting method to protect the user privacy in some applications of RFID systems.

4.3 Lightweight Cryptography

Having just described sensor node platforms and RFID tags, we now introduce the cryptography for small embedded devices, i.e., *lightweight cryptography*. The aim of lightweight cryptography is to yield both low-cost and strong cryptographic implementations to small constrained devices. Here, we first examine the cost of currently available lightweight implementations.

We then go a bit beyond the implementation level by presenting the cooperative cryptographic techniques, in which the computational load of a protocol can be shared with potentially untrusted entities. While usually not included in lightweight cryptography, these techniques appear promising to achieve its ultimate goal, i.e., to enable weak devices to achieve computationally expensive cryptographic tasks.

4.3.1 Lightweight Implementations

Here we briefly review currently available implementations dedicated to small embedded devices. The aim is not to exhaustively describe the state of the art, but to give insights about the costs and performances of lightweight cryptographic implementations and their suitability for small hardware.

First, for each of the main cryptographic primitives, we list the most suitable algorithms. Then, we compare the costs and performances of the implementations. We finally discuss the implications for sensor nodes and RFID tags.

Software vs Hardware. Implementations for lightweight cryptography may rely on hardware, software, or a mix of them. Software implementations are in general preferable, mainly because of their lower development cost and absence of hardware manufacturing. They are therefore extensively employed to secure sensor nodes, which always embed a small general purpose microprocessor. On the other hand, hardware is needed in presence of severe energy or power constraints. This is why hardware cryptographic implementations are often proposed in the context of RFID tags. Hybrid solutions between software and hardware exist, such as instruction set extension [Koc08] or software-hardware codesign [HBHV07], where only a part of the cryptographic algorithm is accelerated with hardware.

Encryption. For encryption, block ciphers are usually more employed than stream ciphers as their are better understood and require a smaller internal state. The most popular choice is of course the AES [DR02], while ciphers specially designed for constrained environments appear more convenient. These are for instance the Scalable Encryption Algorithm (SEA [SPGQ06]) or PRESENT [BKL+07].

Stream ciphers can be very compact, as illustrated with TRIVIUM [CP08], but they require an internal state of at least twice the key size. Block ciphers based on the structure of stream ciphers may combine the advantages of both kinds of ciphers, as shown with the recently proposed KATAN and KTANTAN block ciphers [CDK09].

Hash Function. Hash functions are very helpful to ensure the integrity of sent messages thanks to Message Authentication Codes (MAC) for instance. Although dedicated algorithms like the standardized functions from the SHA family [NIS02] can be used, block cipher based solutions are often preferred, as it allows to reuse an existing implementation. CBC-MAC [BKR00] is a popular construction for this. Note that SHA must still be used when implementing digital signature schemes such as ECDSA [NIS09]. As for ciphers, hash functions specifically targeting a low footprint outperform the standard functions. This was highlighted by Bogdanov et al. in [BLP+08] with the proposition of lightweight hash functions relying on the cipher PRESENT. The recent proposition of QUARK at CHES 2010 [AHMNP10] also follows this direction: this hash function intended for RFID tags is inspired by the lightweight KATAN and GRAIN stream ciphers.

Elliptic Curve Cryptography. Public-key cryptography (PKC) can offer many advantages in the context of small embedded devices, despite involving much more computations than secret-key cryptography. For instance, it is massively used in privacy-preserving authentication protocols for RFID systems [LBSV10]. It constitutes also an interesting alternative for key agreement protocols in large sensor networks thanks to small memory and communication overheads and an excellent scalability [LMK+09].

Elliptic Curve Cryptography was demonstrated to be more efficient than RSA on small devices by Gura et al. [GPW+04]. As a result, ECC has been extensively studied for both software and hardware lightweight implementations. We report below the results of ECC lightweight implementations. The given timings refer to the execution time of the basic operation of ECC, the point multiplication.

Pairing-Based Cryptography. The interest for Pairing-Based Cryptography (PBC [SOK00]) in networks composed of weak devices is relatively recent. It comes from the observation that the use of PKC causes an overhead as public keys must be authenticated, e.g., using certificates. This can be avoided when employing Identity-Based Cryptography [Sha85], which relies on bilinear pairings, or pairings for short. We will formally introduce them in Section 6.4.1.

Implementors have therefore been thoroughly interested in this primitive in the context of lightweight cryptography. Although PBC is more complex than ECC, they have come up with very optimized implementations showing very decent execution times. As a result, the primitive is slowly rejoining the cryptographic arsenal of security designers for networks like wireless sensor networks.

Performances Comparison. The costs and performances of a representative subset of lightweight implementations of the main cryptographic primitives are compared in Table 4.2 for software and Table 4.3 for hardware.

In software, the implementation results refer to a 8-bit AVR microcontroller. The complexity of the implementations is measured through the time of computation and the storage needed in Read Only Memory (ROM) and Random Access Memory (RAM). The timing performance is expressed in both number of cycles and time for a typical frequency of 8 MHz. Concerning speed, the ciphers are roughly three orders of magnitude faster than an elliptic curve point multiplication and about one order of magnitude faster than SHA-1. The computation time for a pairing is quite long but still reasonable. For

memory, the costs grow much more moderately when comparing ciphers with ECC or with the pairing implementation.

	ROM [Byte]	RAM [Byte]	Cycle Count	Time at 8MHz
PRESENT-80 [TE07]	936	0 [1]	10.7 k	1.34 ms
SEA-96 [TE07]	2132	0 [1]	9.7 k	1.21 ms
AES-128 [TE07]	2606	224 [1]	6.6 k	0.83 ms
SHA-1 [KHYM08]	4048	128	84 k	10.5 ms
ECC-163 GF(2^n) [AOLD10]	16218	2267	6.5 M	0.81 s
ECC-192 GF(p) [LMK+09]	n/a	1080	12.3 M	1.54 s
η_T pairing [OSLD08]	47948	2867	40.2 M	5.02 s

[1] The assembly implementations of [TE07] exploit the controller general-purpose registers to save RAM.

Table 4.2: Comparison of state-of-the-art implementations of the main cryptographic primitives in an 8-bit AVR microcontroller.

In hardware, the complexity of the implementations is measured with the silicon area needed to build the circuit. As the areas cannot be directly compared when using different logic processes, they are translated in number of *Gate Equivalents* (GE), i.e., divided by the area of a reference gate – normally a 2-input NAND – in their manufacturing technology.

As for software, ciphers are more compact than standardized hash functions, while ECC has the largest area cost (four times the AES). Interestingly, algorithms specially designed for compact implementations clearly outperform their standardized counterparts. For instance, the area requirements of KTANTAN-32 and DM-PRESENT-80 are about one order of magnitude inferior to the AES and SHA-256 respectively.

The throughput describes the implementations timing performance. It is given in Table 4.3 for a low clock frequency of 100 kHz, which is a typical choice to meet the low power requirements. For the different implementations, the throughputs appear sufficient to ensure reasonable latencies for cryptographic tasks. The slowest implementation is for ECC, in which a point multiplication is performed in about 600 ms.[1] Concerning pairings, the author is not aware of any compact ASIC implementations, and refers to [BDF+10] for the description of a parallel hardware implementation of the η_T pairing.

[1] For a typical frequency of 100 kHz, while the implementation is able to run at 700 kHz.

	Through. at 100kHz [kbps]	Logic process	Area [GE]
KTANTAN-32 [CDK09]	12.5	0.13μm	462
TRIVIUM [MGPV08]	100	0.35μm	749
KATAN-64 [CDK09]	25.1	0.13μm	1054
PRESENT-80 [BKL$^+$07]	200	0.18μm	1570
AES-128 [FDW04]	12.4	0.35μm	3400
DM-PRESENT-80 [BLP$^+$08]	14.6	0.18μm	1600
SHA-1 [FR06]	40.2	0.35μm	8120
SHA-256 [FR06]	45.4	0.35μm	10868
ECC-163 GF(2^n) [LBSV10]	0.55	0.13μm	14556

Table 4.3: Comparison of state-of-the-art lightweight hardware implementations of the main cryptographic primitives.

Implications for Sensor Nodes. The various cryptographic primitives are in general tractable by the microcontrollers equipping most sensor nodes. Even the largest implementations – ECC and the pairing– do not exceed the microcontroller memory limits (see in Table 4.1 for typical RAM and ROM sizes).

On the other hand, the use of cryptography generates an extra energy consumption, which was shown to be tolerable for encryption [GVP$^+$03] and for a moderate use of public-key cryptography [PLP06]. However, when targeting a really low energy consumption, the various possibilities to perform cryptographic tasks must be compared to select the less consuming ones. The comparison must not be based on the computation cost only, given the importance of the energy spent by communications in sensor networks. For instance, the timing performances of AES and ECC differ by three orders of magnitude, as shown in Table 4.1. But for a key exchange based on AES or ECC, this difference is clearly reduced when taking into account the communication energy costs, as we will detail in the next chapter.

Therefore, an appropriate methodology must be employed to compare the energy consumption of cryptographic protocols, which carefully assesses the communication cost. We will present such a methodology in Chapter 5.

Implications for RFID Tags. According to the specifications of low-cost RFID tags (EPC Class 1 Generation 2), only a few thousands gates – up to 10-15k gates – can be dedicated to cryptography on these devices [CR07]. As a result, the implementations with a high gate count such as ECC and SHA in Table 4.3 may not suit these low-cost tags. By contrast, ciphers are perfectly affordable, especially the optimized algorithms as PRESENT or KTANTAN.

Contactless smart cards contain a small microprocessor and are sometimes also equipped with a cryptoprocessor. As they have thus a comparable computing power with respect to sensor nodes, the conclusions drawn above for sensor nodes can be applied for them. The energy concern must however be replaced with an acceptable time of interaction with the reader, which is somehow also an energy limitation for the card.

Physical Security We end this section about lightweight cryptographic implementations by recalling the importance of physical security in the context of small embedded devices. Most of the implementations described above lack tamper-proof mechanisms and may therefore be compromised with high success rate, as explained in [dMS10] for sensor nodes or [HMF07] for RFID tags.

Dedicated protections should be added, with hopefully a security level matching the one of the implemented algorithm. However, the resulting overhead may exceed the device capacity.

4.3.2 Cooperative Cryptographic Techniques

Enabling cryptography in small embedded devices goes beyond the study of lightweight implementations. It can be done at the level of the protocol, for instance by storing in the small devices the results of pre-computations obtained on powerful machines. Another solution is to adapt the cryptographic protocols to take into account the device limited resources, as performed in [GL04] for the GPS identification scheme. We focus here on a third promising approach, the cooperative cryptographic techniques.

In many contexts, low power devices are in the vicinity of more powerful ones, such as a reader for the RFID tags or a base station for sensor nodes. They may therefore achieve strong cryptography at a moderate cost with the help of these powerful helping entities. Of course, the security should not be threatened in presence of malicious helpers. Considering the high computational gain that cooperative cryptographic protocols may bring to small devices, it is highly desirable to develop such techniques.

This idea of cooperative cryptography has already been investigated, in the form of outsourcing computation to an insecure device [GL05, BQ95, MKI88] or to a semi-trusted party, with proxy cryptography [BBS98, AFGH06]. However, finding an efficient cooperative variant of a cryptographic protocol and proving its security in the same time remains in general a tough challenge.

In this thesis, we are interested in the cooperative version of a group signature scheme. In Chapter 6, we first present a cooperative version of the group

signature scheme introduced by Delerablée and Pointcheval in [DP06]. Then, we describe an implementation of it in small sensor nodes, proving the feasibility of the computationally expensive group signatures for constrained devices as contactless smart cards. Beforehand, in Chapter 5, we detail a methodology to analyze the energy cost of computations and communications of cryptographic protocols, which is of interest when assessing the effectiveness of cooperative techniques in wireless sensor networks.

Bibliography

[AFGH06] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Trans. Inf. Syst. Secur.*, 9(1):1–30, 2006.

[AHMNP10] Jean-Philippe Aumasson, Luca Henzen, Willi Meier, and María Naya-Plasencia. QUARK: a lightweight hash. In *CHES*, LNCS. Springer, 2010.

[AKR+07] T. Ahola, P. Korpinen, J. Rakkola, T. Ramo, J. Salminen, and J. Savolainen. Wearable FPGA Based Wireless Sensor Platform. *Conf Proc IEEE Eng Med Biol Soc*, 1, 2007.

[AOLD10] D. F. Aranha, L. B. Oliveira, J. López, and R. Dahab. Efficient implementation of elliptic curve cryptography in wireless sensors. *Advances in Mathematics of Communications*, 4(2):169–187, 2010.

[Avo10] Gildas Avoine. Information extractor of MOBIB, the STIB electronic ticket for public transportation. Information Security Group, Université catholique de Louvain, `http://sites.uclouvain.be/security/mobib.html`, June 2010.

[BBS98] Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In *EUROCRYPT*, pages 127–144, 1998.

[BDF+10] Jean-Luc Beuchat, Hiroshi Doi, Kaoru Fujita, Atsuo Inomata, Piseth Ith, Akira Kanaoka, Masayoshi Katouno, Masahiro Mambo, Eiji Okamoto, Takeshi Okamoto, Takaaki Shiga, Masaaki Shirase, Ryuji Soga, Tsuyoshi Takagi, Ananda

Vithanage, and Hiroyasu Yamamoto. FPGA and ASIC implementations of the η_T pairing in characteristic three. *Comput. Electr. Eng.*, 36(1):73–87, 2010.

[BKL+07] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. PRESENT: An Ultra-Lightweight Block Cipher. In Pascal Paillier and Ingrid Verbauwhede, editors, *CHES*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer, 2007.

[BKR00] Mihir Bellare, Joe Kilian, and Phillip Rogaway. The security of the cipher block chaining message authentication code. *Journal of Computer and System Sciences*, 61(3):362–399, 2000.

[BLP+08] Andrey Bogdanov, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, and Yannick Seurin. Hash functions and RFID tags: Mind the gap. In Elisabeth Oswald and Pankaj Rohatgi, editors, *CHES*, volume 5154 of *Lecture Notes in Computer Science*, pages 283–299. Springer, 2008.

[BQ95] P. Béguin and J.-J. Quisquater. Fast Server-Aided RSA Signatures Secure Against Active Attacks. In *CRYPTO '95*, volume 963 of *LNCS*, pages 57–69. Springer, 1995.

[CBW08] Nicolas T. Courtois, Gregory V. Bard, and David Wagner. Algebraic and Slide Attacks on KeeLoq. In *Fast Software Encryption: 15th International Workshop*, pages 97–115, Berlin, Heidelberg, 2008. Springer-Verlag.

[CDK09] Christophe De Cannière, Orr Dunkelman, and Miroslav Knezevic. KATAN and KTANTAN - A Family of Small and Efficient Hardware-Oriented Block Ciphers. In Christophe Clavier and Kris Gaj, editors, *CHES*, volume 5747 of *Lecture Notes in Computer Science*, pages 272–288. Springer, 2009.

[CDV+05] Benton H. Calhoun, Denis C. Daly, Naveen Verma, Daniel F. Finchelstein, David D. Wentzloff, Alice Wang, Seong-Hwan Cho, and Anantha P. Chandrakasan. Design considerations for ultra-low energy wireless microsensor nodes. *IEEE Transactions on Computers*, 54:727–740, 2005.

[CP08] Christophe De Cannière and Bart Preneel. Trivium. In Matthew J. B. Robshaw and Olivier Billet, editors, *The eSTREAM Finalists*, volume 4986 of *Lecture Notes in Computer Science*, pages 244–266. Springer, 2008.

[CR07] Peter H. Cole and Damith C. Ranasinghe. *Networked RFID Systems and Lightweight Cryptography: Raising Barriers to Product Counterfeiting*. Springer, 2007.

[Croa] CrossBow Technology Inc. MICAz sensor node Datasheet. Available at `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAz_Datasheet.pdf`.

[Crob] CrossBow Technology Inc. TelosB sensor node datasheet. Available at `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/TELOSB_Datasheet.pdf`.

[dMS10] Giacomo de Meulenaer and Franois-Xavier Standaert. Stealthy Compromise of Wireless Sensor Nodes with Power Analysis Attacks. In *2nd International Conference on Mobile Lightweight Wireless Systems (MOBILIGHT 2010)*. Springer's Lecture Notes of ICST - LNICST, 5 2010.

[DP06] Cécile Delerablée and David Pointcheval. Dynamic fully anonymous short group signatures. In *VIETCRYPT 2006*, volume 4341 of *Lecture Notes in Computer Science*, pages 193–210. Springer, 2006.

[DR02] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer Verlag, Berlin, Heidelberg, New York, 2002.

[FDW04] Martin Feldhofer, Sandra Dominikus, and Johannes Wolkerstorfer. Strong authentication for RFID systems using the AES algorithm. pages 357–370. 2004.

[FR06] Martin Feldhofer and Christian Rechberger. A case against currently used hash functions in RFID protocols. *On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops*, pages 372–381, 2006.

[GL04] M. Girault and D. Lefranc. Public key authentication with one (online) single addition. In *CHES 2004*, volume 3156 of *LNCS*, pages 413–427. Springer, 2004.

[GL05] M. Girault and D. Lefranc. Server-aided verification: Theory and practice. In *ASIACRYPT 2005*, volume 3788 of *LNCS*, pages 605–623. Springer, 2005.

[GPW+04] Nils Gura, Arun Patel, Arvinderpal Wander, Hans Eberle, and Sheueling Chang Shantz. Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs. In Marc Joye and Jean-Jacques Quisquater, editors, *CHES*, volume 3156 of *Lecture Notes in Computer Science*, pages 119–132. Springer, 2004.

[GVP+03] Prasanth Ganesan, Ramnath Venugopalan, Pushkin Peddabachagari, Alexander Dean, Frank Mueller, and Mihail Sichitiu. Analyzing and modeling encryption overhead for sensor network nodes. In *WSNA '03: Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*, pages 151–159, New York, NY, USA, 2003. ACM.

[HBHV07] Alireza Hodjat, Lejla Batina, David Hwang, and Ingrid Verbauwhede. HW/SW co-design of a hyperelliptic curve cryptosystem using a microcode instruction set coprocessor. *Integr. VLSI J.*, 40(1):45–51, 2007.

[HMF07] Michael Hutter, Stefan Mangard, and Martin Feldhofer. Power and EM attacks on passive 13.56 MHz RFID devices. In *CHES '07: 9th international workshop on Cryptographic Hardware and Embedded Systems*, pages 320–333. Springer, 2007.

[IEE10] IEEE 802.15 working group. IEEE 802.15.4 standard for physical layer and media acccess control in low-rate wireless personal area networks. `http://www.ieee802.org/15/pub/TG4.html`, July 2010.

[Ins10] Texas Instruments. Government Electronic Identification. Brochure available at `http://www.ti.com/rfid/docs/manuals/brochures/govid_trifold.pdf`, April 2010.

[KHYM08] Firdous Kausar, Sajid Hussain, Laurence T. Yang, and Ashraf Masood. Scalable and efficient key management for heterogeneous sensor networks. *J. Supercomput.*, 45(1):44–65, 2008.

[Koc08] Cetin Kaya Koc. *Cryptographic Engineering*. Springer Publishing Company, Incorporated, 2008.

[KW05] Ziv Kfir and Avishai Wool. Picking Virtual Pockets Using Relay Attacks on Contactless Smartcard Systems. In *Conference on Security and Privacy for Emerging Areas in Communication Networks – SecureComm 2005*, pages 47–58, Athens, Greece, September 2005. IEEE, IEEE Computer Society.

[LBSV10] Yong Ki Lee, Lejla Batina, Dave Singelée, and Ingrid Verbauwhede. Low-cost untraceable authentication protocols for RFID. In Susanne Wetzel, Cristina Nita-Rotaru, and Frank Stajano, editors, *WISEC*, pages 55–64. ACM, 2010.

[LMK+09] Christian Lederer, Roland Mader, Manuel Koschuch, Johann Großschdl, Alexander Szekely, and Stefan Tillich. Energy-Efficient Implementation of ECDH Key Exchange for Wireless Sensor Networks. In *Information Security Theory and Practices — WISTP 2009*, pages 112–127. Springer Verlag, LNCS 5746, September 2009.

[MGPV08] N. Mentens, J. Genoe, B. Preneel, and I. Verbauwhede. A Low Cost Implementation of Trivium. In *ECRYPT Workshop, SASC - The State of the Art of Stream Ciphers*. C. De Cannière, and O. Dunkelman, 2008.

[MHK10] David Moss, Jonathan Hui, and Kevin Klues. Low power listening. TinyOS Enhancement Proposal 105, `http://www.tinyos.net/tinyos-2.x/doc/html/tep105.html`, July 2010.

[MKI88] T. Matsumoto, K. Kato, and H. Imai. Speeding up secret computations with insecure auxiliary devices. In *CRYPTO '88*, volume 403 of *LNCS*, pages 497–506. Springer, 1988.

[NIS02] NIST. Secure Hash Standard. Federal Information Processing Standards Publication 180-2. `http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf`, 2002.

[NIS09] NIST. Digital Signature Standard (DSS). Federal Information Processing Standards Publication 186-3. `http://csrc.nist.gov/publications/fips/fips186-3/fips_186-3.pdf`, 2009.

[OSLD08] L.B. Oliveira, M. Scott, J. Lopez, and R. Dahab. TinyPBC: Pairings for authenticated identity-based non-interactive key distribution in sensor networks. In *Networked Sensing Systems, 2008. INSS 2008. 5th International Conference on*, pages 173–180, June 2008.

[PHC04] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 95–107, New York, NY, USA, 2004. ACM Press.

[PLP06] Krzysztof Piotrowski, Peter Langendoerfer, and Steffen Peter. How public key cryptography influences wireless sensor node lifetime. In *SASN '06: Proceedings of the fourth ACM workshop on Security of ad hoc and sensor networks*, pages 169–176, New York, NY, USA, 2006. ACM.

[PPG05] Bryan Parno, Adrian Perrig, and Virgil Gligor. Distributed detection of node replication attacks in sensor networks. In *SP '05: Proceedings of the 2005 IEEE Symposium on Security and Privacy*, pages 49–63, Washington, DC, USA, 2005.

[PSW04] Adrian Perrig, John Stankovic, and David Wagner. Security in wireless sensor networks. *Commun. ACM*, 47(6):53–57, 2004.

[Sha85] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 47–53, New York, NY, USA, 1985. Springer-Verlag New York, Inc.

[SOK00] R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing. In *Symposium on Cryptography and Information Security – SCIS'2000*, Okinawa, Japan, 2000.

[SPGQ06] François-Xavier Standaert, Gilles Piret, Neil Gershenfeld, and Jean-Jacques Quisquater. SEA: A Scalable Encryption Algorithm for Small Embedded Applications. In Josep Domingo-

Ferrer, Joachim Posegga, and Daniel Schreckling, editors, *CARDIS*, volume 3928 of *Lecture Notes in Computer Science*, pages 222–236. Springer, 2006.

[Ste10] Anton Steeman. Developments in RFID. `http://bestinpackaging.wordpress.com/2010/01/13/developments-in-rfid-an-overview-part-01/`, January 2010.

[STI10] STIB. MOBIB: a world of advantages. `http://www.stib.be/mobib.html?l=en`, August 2010.

[Sun09] Sun Microsystems Inc. Sun SPOT (Sun Small Programmable Object Technology) . `http://www.sunspotworld.com/`, September 2009.

[TE07] L. Uhsadel C. Paar A. Poschmann T. Eisenbarth, S. Kumar. A Survey of Lightweight-Cryptography Implementations. In *Design & Test of Computers*. IEEE, 2007.

[Tex10] Texas Instruments. Single-Chip 2.4 GHz IEEE 802.15.4 Compliant and ZigBee Ready RF Transceiver. `http://focus.ti.com/docs/prod/folders/print/cc2420.html`, July 2010.

[Tin10] TinyOS. TinyOS: an open-source operating system designed for wireless embedded sensor networks. `http://www.tinyos.net/`, July 2010.

[WP10] Joey Wilson and Neal Patwari. See Through Walls: Motion Tracking Using Variance-Based Radio Tomography Networks. *IEEE Transactions on Mobile Computing*, Oct 2010.

[YHE02] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient MAC protocol for wireless sensor networks. In *Proceedings 21st International Annual Joint Conference of the IEEE Computer and Communications Societies*, New York, New York, USA, 2002.

[Zig10] ZigBee Alliance. ZigBee, Specifiaction for low-cost, low-power wireless communications solutions. `http://www.zigbee.org/`, July 2010.

CHAPTER 5

On the Energy Cost of Communication and Cryptography in Wireless Sensor Networks

Energy is a central concern in the deployment of wireless sensor networks. Among the various existing ways to achieve cryptographic tasks, it is crucial to select the less consuming ones. In this chapter, we present a methodology to estimate the energy cost of cryptographic protocols, both from a communication and a computation point of view, based on practical measurements on the MICAz and TelosB sensors. We illustrate its use by assessing the cost of two key agreement protocols: Kerberos, relying on secret-key cryptography, and the Elliptic Curve Diffie-Hellman key exchange with authentication provided by the Elliptic Curve Digital Signature Algorithm (ECDH-ECDSA). We confirm the superiority of Kerberos in contexts where a trusted third party is available. We also underline the importance of the communication cost when the nodes need to stay in listen mode, e.g., between the protocol rounds, even when reduced using a Low Power Listening (LPL) protocol. Therefore, listening should be considered when assessing the cost of cryptographic protocols in wireless sensor networks.

5.1 Introduction

Wireless Sensor Networks (WSN) are composed of small autonomous devices that process and communicate data acquired from the environment in which they are deployed. As explained in Section 4.2, their low cost and rapidity of deployment make them particularly attractive for many applications where there is a need of strong security, such as wearable health monitoring systems, building protection, pollution detection, battlefield management, ... However, sensor nodes having tight energy constraints, the energy cost of security techniques can be prohibitive and must therefore be minimized.

Various techniques can be adopted to perform the cryptographic tasks in WSN. As an example, key exchange can be carried out by relying on methods from symmetric-key cryptography, e.g., through Kerberos [NT94], or from public-key cryptography, e.g., through various modes of SSL/TLS [FKK96]. Besides, in order to provide better security features while preserving low communication and memory costs, different techniques have been proposed, that allow trading between security, communication and computation ([LHKV04] and [GL05] for instance). We ourselves describe in Chapter 6 a cooperative variant of a group signature scheme, which allows to outsource a large part of the computations at the cost of extra communications. In order to appreciate the practical effectiveness of these trading techniques in a specific WSN, the cost of communication and computation must be well understood.

Contradictions appear in previous works concerning the importance of the communication energy cost. For instance, two preceding works ([PLP06] and [WGE+05]) assessing the cost of public-key cryptography on similar hardware draw opposed conclusions concerning the importance of the communication energy cost when comparing cryptographic algorithms in WSN. Moreover, previous related works analyze the communication cost in a very simple way and do not take into account practical elements such as the consumption while nodes remain in the listening mode.

Our goal in this chapter is to assess and analyze the real cost of cryptography on WSN nodes. This will help choosing directions to optimize the cost of cryptography in low power WSN. For this purpose, we investigate the cost of cryptography through a case study based on measurements on the MICAz [Crob] and TelosB [Croc] sensor nodes. We propose a methodology to assess the energy cost of cryptographic protocols and illustrate its use by focusing on two key agreement protocols, Kerberos and ECDH-ECDSA, the Elliptic Curve Diffie-Hellman key exchange with authentication provided by the Elliptic Curve Digital Signature Algorithm (i.e., the ECC-based SSL/TLS handshake, see [BWDH01]). The results presented in this chapter were pub-

lished in [dMGSP08].

Contributions. Our main contributions are :

1. A methodology to assess the real cost of cryptography on WSN nodes, which makes it possible to establish the relative costs of computation and communication.
2. The estimates of the key agreement protocols obtained for the MICAz and TelosB nodes. They allow us to compare symmetric and asymmetric techniques. They point out the importance of the idle listening consumption.

Outline. The chapter is structured as follows. Section 5.2 presents the previous related works. Then, Section 5.3 explains our methodology, which relies on the energy models of the sensors MICAz and TelosB determined by measurements. Next, Section 5.4 provides an assessment and analysis of the energy cost of Kerberos and ECDH-ECDSA, followed by a comparison with related results in Section 5.5. Finally, future perspectives regarding the evolution of the relative costs of computation and communication in sensor networks are stated in Section 5.6.

5.2 Previous Works

Many recent works investigate the usability of cryptographic algorithms in the context of wireless sensor networks, as described in Section 4.3 where we presented lightweight software implementations. The first important library providing symmetric-key cryptography to sensor networks was TinySec [KSW04] in 2004. It did not support AES, which was later demonstrated to be suitable to sensor nodes in several works, for instance in [HNL07] and [LDH06]. For public-key cryptography, the first implementation of Elliptic Curve Cryptography (ECC [HMV03]) on a sensor node was done by Malan et al. [MWS04] in 2004, with poor performances: a point multiplication on $GF(2^{163})$ took 34 seconds on an 8-bit microcontroller. The same year, Gura et al [GPW$^+$04] presented at CHES an ECC implementation performing a 160-bit point multiplication in 0.81 s on a similar controller, i.e., about 40 times faster! Several subsequent works provided reasonably efficient ECC libraries for sensor networks, such as Sizzle [GMF$^+$05, GW08] and TinyECC [LN07].

Several previous works focused on the energy cost of key agreement protocols for WSN. Based on the first implementations of ECC and RSA on 8-bit

microprocessors by Gura et al. [GPW+04], Wander et al. [WGE+05] quantified the energy costs of ECC and RSA based digital signature and key exchange with mutual authentication for networks composed of MICA2DOT sensors [Croa]. They concluded that these operations were affordable for such sensors. In [PLP06], Piotrowski et al. extended those results to other sensor nodes. They based their assessments on the implementation results of [GMF+05] and on the datasheets of the sensors. They found that the energy consumed by transmissions was at least one order of magnitude less than the one required for the computation of the cryptographic operations. Therefore, they concluded that it was not an important factor. Hodjat and Verbauwhede [HV02] compared the cost of the protocols Kerberos and ECDH on 32-bit WINS sensor nodes. The cost of Diffie-Hellman was found between one to two orders of magnitude larger than AES-based Kerberos. Later, Großschädl et al. [GST07] performed the same comparison but with another version of Diffie-Hellman, ECMQV, on WINS nodes. They found that the cost of ECMQV was only up to twice the cost of Kerberos.

To quantify the communication energy costs, the previous works merely used transmission and reception per-bit costs based on datasheets or measurements. We believe that this simple method is not sufficient, as it does not include the energy consumption of practical elements. In particular, it does not consider listening, which occurs when nodes are waiting for incoming packets, of which the exact time of arrival is uncertain: for instance, a node might be busy with another task before participating to the protocol, the communication delay might be longer than expected due to a packet loss or the presence of many hops between the protocol parties, etc. This could result in significantly underestimated communication costs. Therefore, compared to the previous works, we attach more importance to the practical aspects of the energy consumption for communication.

5.3 Our Methodology

In this section, we present our methodology to assess and analyze the energy cost of cryptographic protocols in the context of wireless sensor networks. It is based on the energy models of sensor node platforms. Therefore, we start by determining such models for the sensors MICAz and TelosB. Then, we describe how we assess the energy cost of computations and communications.

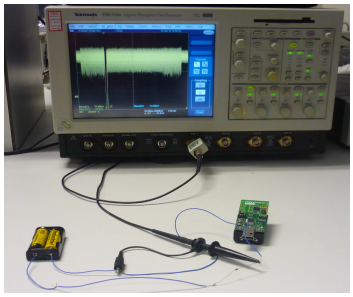


Figure 5.1: Setup of our measurements for the TelosB node, on which our energy models are based.

5.3.1 Energy Models

While the energy consumption of protocols could be obtained with high accuracy through direct measurements of the performance of implementations, an estimation using an energy model presents other advantages. Besides its rapidity of use, an energy model allows to analyze the energy consumption. It is therefore not only useful to select energy-aware protocols, but also to optimize or even design such protocols targeting energy-constrained devices.

Used sensor devices. The nodes we consider in this case study are the MICAz [Crob] of the well-known MICA family and the TelosB [Croc]. These nodes were already presented in Section 4.2.1, but we recall here their main design features. The MICAz is based on the low-power 8-bit microcontroller ATmega128L with a clock frequency of 7.37 MHz. The TelosB features the 16-bit MSP430 microcontroller running at 4 MHz. Both nodes run TinyOS and embed a IEEE 802.15.4 compliant CC2420 transceiver with a claimed data rate of 250 kbps.

Measurements. Willing to obtain estimates reflecting the true consumption of sensor nodes, we base our models on measurements. We thus measured the power consumption of the two sensor devices in their main operating modes. This was done by inserting a sense resistor in their power line and recording the voltage drop on an TDS7104 oscilloscope, as depicted in Figure 5.1 for the TelosB. The results of these measurements are synthesized in Table 5.1 for both platforms. Remark already the relative high consumptions of operating modes associated to communications.

Suspecting that the transceiver data rate may be in practice significantly inferior to the value announced in the datasheet, we also measured it for a

large number of transmitted packets. The measured rates, 121 kbps and 94 kbps for the MICAz and TelosB respectively, are far below the claimed rates (250 kbps). The important difference has also been reported in [PCG+05]. It is due to the Carrier Sense Multiple Access (CSMA) method to access the media: to prevent potential collisions, each transmission is preceded by a small listening period to ensure that the channel is free. This decreases the average data rate. The presence of footers and headers in the frame format and the use of acknowledgments further decrease the rates available for application data to respectively 108 kbps and 75 kbps. Concerning the consumption during transmission, we observed that it did not vary so much with the transmit power: we measured consumptions of 50 mW and 75 mW for the two extreme transmit powers (respectively -25 and 0 dBm) for the MICAz. This is due to the fixed consumption of the controller which is active during the transmissions.

	MICAz		TelosB	
	power	(rel.)	power	(rel.)
Compute	26 mW	(1)	4.8 mW	(1)
Sleep	25 μW	(10^{-3})	35 μW	(10^{-2})
Transmit	65 mW	(2.5)	54 mW	(11.3)
Listen	68 mW	(2.6)	60 mW	(12.5)
Receive	72 mW	(2.8)	61 mW	(12.7)

Table 5.1: Measured power consumption of the MICAz running at 7.37 MHz and TelosB at 4 MHz in their main operating modes. For communications, we use a typical transmit power of -5 dBm.

Energy models. Based on our measurements, we establish the energy models of the sensor nodes, i.e., the set of the energy costs associated to their most common operations. This is done in the following way. For the cost of computation, we make the approximation that the overall power consumption of the node while computing remains constant with the type of microcode operation performed. Therefore, the cost of a particular computation can be assessed knowing the per-cycle mean energy consumption and the total number of cycles of the computation. This simplifying assumption was verified by Law et al. in [LDH06] for the sensor node used in the EYES project [EYE], which is quite similar to the TelosB. This assumption is also used in the power estimator PowerTOSSIM [SHCW04] for the MICA2 sensor node (similar to the MICAz) with a mean error of 5%.

For the communication cost, we also rely on the measurement results of Table 5.1 and consider the practical data rates found earlier together with a

	MICAz		TelosB	
	energy	(rel.)	energy	(rel.)
Compute for 1 cycle	3.5 nJ	(1)	1.2 nJ	(1)
Listen for 1 cycle	9.2 nJ	(3)	15.0 nJ	(13)
Sleep for 1 cycle	3 pJ	(10^{-3})	9 pJ	(10^{-2})
Transmit 1 bit	0.60 μJ	(170)	0.72 μJ	(600)
Receive 1 bit	0.67 μJ	(190)	0.81 μJ	(680)

Table 5.2: Energy costs of common operations on the MICAz running at 7.37 MHz and TelosB at 4 MHz for application data rates of respectively 108 kbps and 75 kbps. The relative costs of these operations are indicated in parentheses.

typical transmit power of -5 dBm. The resulting collections of energy costs are displayed in Table 5.2. The costs for computing, listening and sleeping are given per clock cycle, to allow a quick comparison. For transmission and reception, the basic unit is the payload bit communicated. We will directly use the costs of Table 5.2 to assess the energy cost of cryptographic protocols, at the exception of the listening cost, which must be further detailed.

5.3.2 Low Power Listening

The energy costs of Table 5.1 indicate that the consumption in the listening mode is several times higher than the cost of computation. In fact, listening is quite comparable to packet reception, as the transceiver is also active in this mode: it is trying to acquire incoming packets. A high consumption for the listening mode is problematic since nodes may spend a lot of time in this mode. Therefore, a well-known strategy is to alternate listening periods with sleep periods in order to decrease the average consumption while waiting for packets to arrive. It is implemented under the name of Low Power Listening (LPL) protocols, which save energy at the expense of introducing greater latencies in the communications. These protocols make the time spent in listen mode less important from an energy point of view. They can be synchronous or asynchronous. In synchronous LPL protocols (e.g., S-MAC [YHE02]), the sleep patterns of the nodes are known in advance. Communications can take place exclusively when nodes are listening with high accuracy, avoiding a waste of energy for the transmitters. However, this system requires the synchronization of the internal clock of all nodes, which generates a cost.

By contrast, asynchronous protocols are easier to implement and also efficient, whereas they cannot prevent punctual energy losses when transmitting nodes attempt to reach nodes that are sleeping. We next describe the asyn-

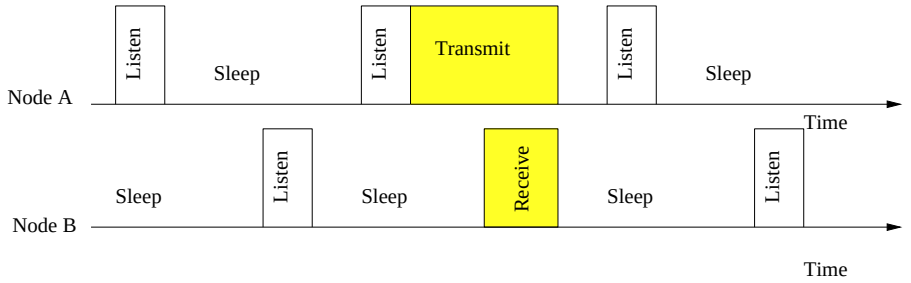


Figure 5.2: Overview of the Low Power Listening protocol employed in TinyOS. Nodes only listen periodically. Therefore, Node A has to retransmit its packet until Node B is listening.

chronous LPL protocol that we consider in this work.

LPL in TinyOS. In TinyOS, the LPL protocol available for nodes equipped with a CC2420 radio [MHLC07] is asynchronous and inspired from B-MAC [PHC04]. In this protocol, the receiving radio modules are periodically turned on to check for activity on the channel and remain active only if a packet is being transmitted. Sending nodes must be kept retransmitting the same packets until the checks of the receivers. This is illustrated in Figure 5.2 for a node A transmitting a packet to a node B.

Thanks to this LPL protocol, the average consumption of a listening node can arbitrarily be reduced by increasing the sleep interval, i.e., the delay between two checks. Denoting the check duration and the sleep interval with t_{check} and t_{sleep} respectively, the power consumption of a node listening with LPL is reduced to:

$$P_{LPL} = P_{listen} \cdot \frac{t_{check}}{t_{check} + t_{sleep}}.$$

However, decreasing P_{LPL} is done at the expense of a longer retransmission for the sending node, since the relative importance of the sleep pattern of the receiving node is increased. The energy cost associated to this retransmission is called the synchronization cost and is a function of the mean delay $\bar{t}_{synchro}$ before the check of the receiver. Of course, if the receiver is already listening, the delay is null. On the other hand, if the receiver is in the sleep interval, the average delay before the receiver next check is $\frac{t_{sleep}}{2}$. As this happens with a probability $\frac{t_{sleep}}{t_{check}+t_{sleep}}$, the mean synchronization delay is:

$$\bar{t}_{synchro} = \frac{t_{sleep}}{2} \cdot \frac{t_{sleep}}{t_{check} + t_{sleep}}.$$

	MICAz		TelosB	
	energy	(rel.)	energy	(rel.)
Compute for 1 cycle	3.5 nJ	(1)	1.2 nJ	(1)
Listen for 1 cycle	0.4 nJ	(0.1)	0.7 nJ	(0.6)
$\bar{E}_{synchro}$	3.1 mJ	(1.1 M)	2.6 mJ	(2.6 M)
$E_{post-transmit}$	0.68 mJ	(0.2 M)	0.60 mJ	(0.5 M)

Table 5.3: Energy costs of the TinyOS LPL protocol for the MICAz and TelosB for a typical choice of parameters. The computation cost is given as a reference.

The associated synchronization cost is thus:

$$\bar{E}_{synchro} = P_{transmit} \cdot \frac{t_{sleep}}{2} \cdot \frac{t_{sleep}}{t_{check} + t_{sleep}}.$$

After a successful transmission, the TinyOS protocol specifies that both sender and receiver remain in the listening state for a small delay, the delay after transmission $t_{post-transmit}$, in case of a consecutive packet transmission. This generates a post-transmission cost $E_{post-transmit}$ for both the sender and the receiver, which similarly amounts to the product of P_{listen} by $t_{post-transmit}$.

In this work, to provide some intuition about the energy costs that Low Power Listening could induce, we chose to set the various delays in the TinyOS LPL to typical values. Of course, optimal values might be derived for given WSN situations, but this is application-specific. We selected the values of respectively 10 ms and 100 ms for the delay after transmission $t_{post-transmit}$ and the sleep interval t_{sleep}, while the check duration t_{check} is set to 5 ms. For these values, we have $\bar{t}_{synchro} \approx 48$ ms. The choice for these parameters in fact depends on the dynamic of the phenomenon being monitored by the sensor nodes. Our choice is thus quite arbitrary, but it still captures many situations in practice. The corresponding energy costs of the TinyOS LPL, obtained using the above formulae, are reported in Table 5.3. We verified experimentally the costs induced by the TinyOS LPL protocol.

5.3.3 Summary

To assess and analyze the energy cost of cryptographic protocols, our methodology relies on energy models of the sensor devices (Table 5.2), obtained through measurements, and the energy costs associated to the use of a Low-Power Listening protocol (Table 5.3).

To obtain the cost of computations, we employ the per-cycle mean energy cost and the average number of cycles needed in the computations. This simple method already exhibits an interesting accuracy, as discussed earlier, and is much faster than a direct measurement.

For communications, our aim is to consider all the elements that are significant in practice. Therefore, we not only use the per-bit transmit and receive costs, as done in many previous works. Our cost estimate is in fact the sum of the following contributions:

- Transmit and receive. Each payload bit of the protocol generates transmit and receive costs, given in Table 5.2. These costs reflect the transceiver power consumption and its data rate in real TinyOS applications, much lower than the promised 250 kbps, as explained above. Concerning the transmit power, we restrict ourselves to a fixed value of -5 dBm. In a real WSN, the lowest transmit power able to ensure the communication on each network link should be selected. As this is dependent on the exact network topology, we decide to not model this point for the sake of simplicity and use a typical value of -5 dBm for the transmit power.

- Low-Power listen. Listening may cause an important energy consumption if the nodes involved in the protocol must remain in this state for a long period while waiting for a message. To reduce it, we assume the use of a LPL protocol, the TinyOS LPL, of which the power consumption is indicated in Table 5.3. The listen duration to consider corresponds to the other party computations and the communication latency.

- Synchronization. With the TinyOS LPL protocol, a transmitted packet induces a synchronization cost for the sender, and a post-transmission cost for both sender and receiver. These costs are given in Table 5.3 for a typical set of parameters. Note that a quick reply or a subsequent packet transmission does not imply a synchronization cost, as the synchronization is kept during the delay after reception $t_{post-transmit}$.

With a more detailed modeling of the communication cost, we hope to better understand how cryptographic protocols consume energy and therefore provide a practical tool to select or even design less consuming alternatives.

5.4 Energy Consumption of Key Agreement Protocols

In this section, we illustrate the use of our methodology to analyze the energy cost of cryptographic protocols by focusing on key agreement protocols. The

establishment of shared secret keys between nodes is a first step to provide other security services such as encryption in sensor networks. This could be achieved by means of pre-deployed shared keys but it raises problems of storage of the keys in large networks and of resiliency to node compromise. Therefore, a solution is to use key distribution or key agreement protocols after the deployment of the nodes. In this section, we first describe the two protocols under investigation and then assess the cost of their cryptographic operations and communications. Finally, since our estimates reveal the importance of the listening cost, we discuss some strategies than can be used to diminish it.

5.4.1 Protocols Description

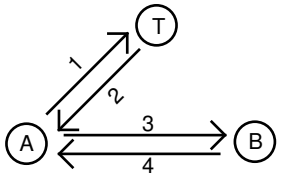
Protocols for key exchange or distribution can be based either on symmetric-key or asymmetric-key cryptography. Here, we study one protocol of each kind, allowing the comparison of the two approaches.

Simplified Kerberos. The first protocol is Kerberos [NT94], a key distribution scheme built on secret-key cryptography, which authenticates the participants. In its original version, Kerberos is intended for computer networks and thus not directly suitable for sensor networks, as it involves rather long messages. Therefore, we use its lightweight version introduced in [GST07], which is described as a "simplified Kerberos without ticket granting service".

Based on symmetric-key cryptography, Kerberos requires a trusted third party, or trusted authority. Two entities A and B wishing to establish a shared secret key k_{AB} are supposed to already share a long-term secret key (k_{AT} and k_{BT} respectively) with the trusted third party T. In the context of sensor networks, T may not be a simple node, as tampering with it would compromise a significant part of the network. Therefore, we assume that T is a secure base station or a well-protected powerful node.

The protocol takes place as featured in Figure 5.3. There is first an exchange of messages between A and T. The request of A contains the identities of A and B, the nodes willing to share a key. In the reply, the key k_{AB} generated by T is encrypted with the long-term keys k_{AT} and k_{BT}.

Then, A recovers the key k_{AB} and forwards to B the piece of the message encrypted with k_{BT} together with its identity encrypted with k_{AB}. Finally, B recovers k_{AB} and sends back to A a timestamp encrypted with k_{AB}. Replay attacks are avoided thanks to a timestamp t_A, a nonce n_A and expiration times t_S, t_E.



1: A,B, n_A
2: $\{k_{AB}$, B, t_S, t_E, $n_A\}$ k_{AT}, $\{k_{AB}$, A, t_S, $t_E\}k_{BT}$
3: $\{k_{AB}$, A, t_S, $t_E\}k_{BT}$, {A, $t_A\}k_{AB}$
4: $\{t_A\}k_{AB}$

Figure 5.3: The simplified Kerberos protocol.

Simplified ECDH-ECDSA. The second studied protocol is ECDH-ECDSA [BWDH01], which is an extension of the basic Diffie-Hellman key agreement (ECDH [HMV03]). Since ECDH does not guaranty authentication in its standard form, ECDH-ECDSA uses certificates verified using the Elliptic Curve Digital Signature Algorithm (ECDSA [HMV03]). This protocol is in fact extensively employed in SSL/TLS handshakes [FKK96] to secure Internet transactions. We here describe its simplest version, as we aim small sensor devices as computing entities.

As opposed to Kerberos, ECDH-ECDSA does not resort to any trusted third party during the protocol. On the other hand, the two participating entities A and B must possess a certificate generated by a trusted authority. They also must agree in advance to use the same curve parameters and generate their private keys, k_A and k_B and corresponding public keys $Q_A = k_A \cdot G$ and $Q_B = k_B \cdot G$, where G is the generator of the group defined by the elliptic curve.

The rounds of the protocol are represented in Figure 5.4. First, A and B exchange random nonces. In the reply to A, B adds its certificate, i.e., essentially its public key signed by the authority using ECDSA. After the certificate verification, A uses his private key and B's public key to perform a point multiplication and arrive to a common secret $k_A \cdot k_B \cdot G$, which is used with the exchanged nonces to derive a shared secret key. Then, A sends its certificate to B who performs the same operations to obtain the shared secret ($k_A \cdot k_B \cdot G$ = $k_B \cdot k_A \cdot G$) and derive the shared secret key.

The possession of the shared secret key is proved in the ability of both parties to encrypt the hash of the exchanged nonces and their identities with the shared secret key (i.e., $\{hash(n_A, n_B, A)\}k_{AB}$ and $\{hash(n_A, n_B, B)\}k_{AB}$ for A and B respectively). These results, forming the content of $Finished$

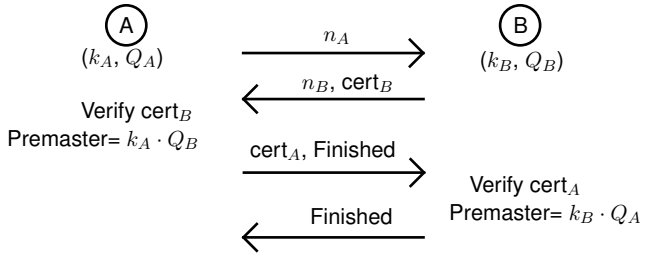


Figure 5.4: The ECDH-ECDSA protocol.

messages, are exchanged at the end of the protocol.

Remark that three rounds only would suffice to perform the protocol. This is possible if A joins its certificate in the first message. In this case, B can generate the $Finished$ message at this point and add it in its reply. Therefore, the last message of the protocol becomes unnecessary. In the following, we refer in general to the 4-round version of the protocol, unless otherwise specified, to remain consistent with the analyses of previous works.

5.4.2 Computation Cost

We now assess the energy costs of the cryptographic computations of the simplified Kerberos and ECDH-ECDSA. For this, we use the energy model of the sensors, especially the mean per-cycle energy cost (from Table 5.2), and the number of cycles of computation of available implementations of cryptographic primitives.

Cost of Cryptographic Primitives. The only cryptographic primitive required for Kerberos is symmetric encryption, whereas ECDH-ECDSA makes use of ECC point multiplication and ECDSA verification.

For the symmetric encryption in Kerberos, we select the popular AES block cipher. Cycle counts for this cipher are reported in the implementation results of Healy et al. [HNL07]. They implemented AES with 128-bit keys on the microcontrollers of both MICAz and TelosB nodes.

For the ECC point multiplications and ECDSA verifications involved in ECDH-ECDSA, we rely on the results of Liu et al. [LN07]. They implemented ECC and ECDSA in TinyOS for many platforms, including MICAz and TelosB, and made them available in the TinyECC library. We use their results for the secp160r1 elliptic curve domain recommended parameters (160-bit keys). While having performances inferior to the state of the art presented

	MICAz		TelosB	
	energy	cycles	energy	cycles
AES-128 encrypt 16 B	38 μJ	10742	9 μJ	7483
ECC-160 point mult	54.6 mJ	15.6 M	16.8 mJ	14.0 M
ECDSA-160 sign	51.5 mJ	14.7 M	15.2 mJ	12.7 M
ECDSA-160 verify	63.0 mJ	18.0 M	19.4 mJ	16.2 M

Table 5.4: Estimated energy costs of cryptographic operations for the MICAz and TelosB, with the corresponding numbers of cycles of computation.

in Section 4.3.1, these implementations are still reasonably efficient. As their code is freely available for both MICAz and TelosB, they constitute an interesting choice for implementers in wireless sensor networks.

Table 5.4 shows the estimated energy costs of these cryptographic computations. The costs were obtained using the operation average number of cycles and the per-cycle mean costs, i.e., 3.5 nJ for the MICAz and 1.2 nJ for the TelosB, as indicated in Table 5.2. The cost of symmetric encryption is negligible compared to elliptic curve operations, as previously highlighted in Section 4.3.1. ECDSA signature, involving one point multiplication, is however less costly than a full point multiplication thanks to precomputations [LN07]. Note that ECDSA operations include the cost of hashing 512 bits using SHA-1.

When comparing both platforms, we see that the number of cycles for elliptic curve computations does not diminish much on the TelosB, however based on a 16-bit microcontroller. This is partly explained by the absence of several optimizations for this platform w.r.t. the MICAz [LN07].

Computation Cost of the Protocols. Having at our disposal the costs of the primitives in Table 5.4, we are now able to estimate the cost of the computations for both protocols.

For Kerberos, the computations consist in the encryption and decryption of 8 128-bit blocks, assuming 64-bit timestamps and node IDs and a 32-bit nonce as in [GST07]. However, the trusted authority T is assumed in this work to be powerful. We thus not consider its computations in the cost of the protocol, as we want to obtain the energy consumed by constrained nodes only. This leads to an encryption of 2 blocks and decryption of 6 blocks for A and B altogether.

In many modes of operation of the block cipher, the decryption is done based on the block cipher encryption, e.g., in counter mode or in cipher feedback mode [NIS01]. Therefore, the computation cost of Kerberos can be computed solely based on the cost of encryption. It is thus equal to 8 block encryptions, leading to respectively 0.3 mJ and 0.1 mJ on the MICAz and TelosB.

For ECDH-ECDSA, each party mainly achieves an ECDSA verification and a point multiplication. The key derivation and symmetric encryption of the nonces and nodes IDs can actually be neglected considering the relative small cost of symmetric primitives with respect to ECC operations (Table 5.4). It leads to a computation energy cost for ECDH-ECDSA of respectively 235.2 mJ and 72.4 mJ on the MICAz and TelosB. These values include both parties computations.

Compared to Kerberos, ECDH-ECDSA is more than 2 orders of magnitude more costly on both platforms. This was expected as elliptic curve operations are much more costly than AES-based encryption. Both protocol are around 4 times less costly on the more energy-efficient TelosB.

5.4.3 Communication Cost

After the computation costs, we now assess the communication energy costs of the protocols. Following our methodology of Section 5.3, the communication costs are divided in transmission, reception and listening costs. As we assume the use of the TinyOS Low-Power Listening protocol, the listening cost is further split up into synchronization cost, post-transmission cost and low-power listening cost. The cost of these various elements is given in Table 5.5 and Table 5.6 for both protocols and both devices. We explain below how they were obtained.

Transmission and reception. For transmission and reception, we make use of the per-bit costs presented in Table 5.2. The total number of bits communicated is 1568 in Kerberos, assuming that the encryption works on a 16-B block basis. Leaving aside the part supported by the powerful trusted part T, 800 bits are sent and 1376 bits are received by the nodes A and B. In ECDH-ECDSA, this number is 2208, considering 86-B certificates, 32-B nonces and 20-B *Finished* messages as in [WGE+05]).

Synchronization. A synchronization cost appears for each transmission, except when a quick reply is performed within the delay after transmission, which is 10 ms for our choice of parameters. In that case, the synchronization is kept between the two communicating nodes.

In Kerberos, there is a single synchronization cost to take into account: for the messages 3 of Figure 5.3. Messages 2 and 4 are indeed quick replies, given the short AES computation timings. Finally, message 1 does not induce any synchronization since it is assumed that T is powerful and is always listening.

In ECDH-ECDSA, three synchronizations occur, as the long delays of ECC computations prevent sending quick replies. Even in the 3-round version of the protocol described at the end of Section 5.4.1, these three synchronizations would remain, since B would no more be able to quickly answer after the first message of the protocol: at this point, B should already perform the computations.

Our assessments of the synchronization costs in the two protocols (Tables 5.5 and 5.6) are done using the average synchronization cost $\bar{E}_{synchro}$ from Table 5.3.

Post-transmission. The TinyOS LPL protocol specifies that the 2 nodes involved in a communication remains in the listen mode for a quick delay, the delay after reception. This generates a cost $E_{post-transmit}$ for each node, of which the value is indicated in Table 5.3.

Both protocols have 4 messages, leading to a total cost of 8 $E_{post-transmit}$ each. However, in Kerberos, 2 of them are chargeable to the powerful entity T and are for this reason not considered.

Low-power listening. During the run of a protocol, a party is listening when waiting for a message, typically while the other party is processing the preceding message. We assess here the resulting energy cost for both protocols using the listening power consumption and the listening total duration. The listening power consumption we consider are given in Table 5.3, i.e., 3.1 mW for the MICAz and 2.7 mW for the TelosB.

Concerning the listening durations in Kerberos and ECDH-ECDSA, we assess them as follows. In ECDH-ECDSA, the listening duration is mainly given by the total computation time, since there is always one party waiting during the other party computations. The synchronization delays do not cause any significant listening energy loss at the receiver side since the receiving node is then sleeping, just before checking the channel for activity. Thus, for the MICAz and TelosB, the listening durations are respectively 9.1 s and 15.1 s for ECDH-ECDSA (i.e., the duration of 2 ECC point multiplications and 2 ECDSA verifications, see Table 5.4).

In Kerberos, the entity A is never in low-power listening as it is always either active or in the post-transmission listening state, given the short AES computation delays. Only entity B is listening before receiving message 3 from A. Its listening duration since the start of the protocol includes the following elements in chronological order:

1. Transmission delay of message 1 (24 B).

2. T computations: encryption of 6 blocks.
3. Transmission delay of message 2 (96 B).
4. A computations: decryption of 3 blocks and 1-block encryption.

Accounting practical data rates of 108/75 kbps and AES block encryption timings of 1.5/1.9 ms for the MICAz/TelosB, we obtain a total listening duration of 24 ms and 32 ms on the MICAz and TelosB respectively. Combining these durations and the nodes low-power listening consumptions leads to the values displayed in Tables 5.5 and 5.6.

Comm. cost Kerberos	MICAz [mJ]	MICAz (perc.)	TelosB [mJ]	TelosB (perc.)
Send	0.5	(6%)	0.6	(8%)
Receive	0.9	(10%)	1.1	(13%)
LP listen	0.1	(1%)	0.1	(1%)
Synchro.	3.1	(36%)	2.6	(33%)
Post-trans.	4.1	(47%)	3.6	(45%)
Total	8.7		8.0	

Table 5.5: Estimated communication energy costs of Kerberos for the MICAz and TelosB.

Comm. cost ECDH-ECDSA	MICAz [mJ]	MICAz (perc.)	TelosB [mJ]	TelosB (perc.)
Send	1.3	(3%)	1.6	(3%)
Receive	1.5	(3%)	1.8	(3%)
LP listen	28.2	(62%)	40.8	(72%)
Synchro.	9.3	(20%)	7.8	(14%)
Post-trans.	5.4	(12%)	4.8	(8%)
Total	45.7		56.8	

Table 5.6: Estimated communication energy costs of ECDH-ECDSA for the MICAz and TelosB.

Total communication costs. The estimated communication costs for Kerberos and ECDH-ECDSA are shown in Tables 5.5 and 5.6. They clearly underline the importance of the elements related to listening, which are much higher than the simple send and receive costs. It is thus important to include them in the assessment of the energy spent by communications.

When comparing both protocols, the total communication cost of ECDH-ECDSA is superior mainly because of the long computation delays of this protocol: they cause high listening costs (even with LPL) and important synchronization costs as they prevent sending quick replies. Strategies to further reduce the listening costs are later discussed in Section 5.4.5.

Remark that besides the energy aspect, the rather long durations of the computations in ECDH-ECDSA can be seen as a disadvantage for the protocol. For instance, in situations where a node has to perform a key exchange with several neighbours in a row, its resources cannot be fully dedicated to the application it is involved in.

5.4.4 Protocols Total Cost

Gathering the computation and communication costs found above provides the total costs for the protocols, shown in Tables 5.7 and 5.8. ECDH-ECDSA is close to respectively 30 times and 15 times more costly than Kerberos on the MICAz and TelosB. Communications compose almost exclusively the cost of Kerberos, as opposed to ECDH-ECDSA. For both protocols, the relative importance of communications grows for the TelosB, which embeds a more energy-efficient computing unit. However, for Kerberos, this effect is negligible due to the very small importance of the computations, resulting in almost identical total costs for both platforms.

Kerberos cost	MICAz [mJ]	MICAz (perc.)	TelosB [mJ]	TelosB (perc.)
Comp.	0.3	(3%)	0.1	(1%)
Comm.	8.7	(97%)	8.0	(99%)
Total	9.0		8.1	

Table 5.7: Estimated total energy costs of Kerberos for the MICAz and TelosB.

ECDH-ECDSA cost	MICAz [mJ]	MICAz (perc.)	TelosB [mJ]	TelosB (perc.)
Comp.	235.2	(84%)	72.4	(61%)
Comm.	45.7	(16%)	46.8	(39%)
Total	280.9		119.2	

Table 5.8: Estimated total energy costs of ECDH-ECDSA for the MICAz and TelosB.

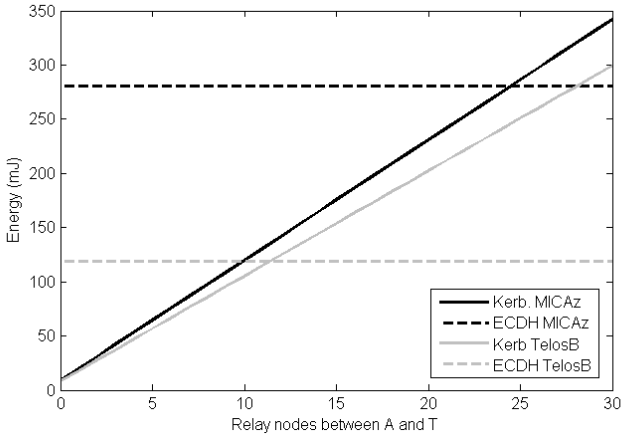


Figure 5.5: Energy cost of Kerberos in function of the number of relay nodes between A and the trusted third party T. After 11 and 25 relays, it becomes more costly than ECDH-ECDSA in networks composed of TelosB and MICAz respectively.

Influence of the number of hops. So far, we have assumed the simplest network topologies, i.e., all the nodes are within communication range, or 1-hop distant. This may not be the case in real situations. Particularly, the powerful trusted entity T in Kerberos is likely to be quite distant from the nodes A and B, if it is a secure base station as assumed previously. It makes therefore sense to study the impact of the number of hops between A and T on the total energy cost of the protocol. Intermediary nodes between these entities generate communication costs. Per relay node, the following elements have to be taken into account:

- Reception and transmission of messages 1 and 2 (120 B in total).
- 2 synchronizations: one for each of the messages 1 and 2.
- 4 post-transmission costs: 1 per send/receive.

Computing the total cost of these elements gives 10.4 mJ and 9.1 mJ per relay for the MICAz and TelosB respectively. Additionally, the presence of relays lengthens the listening duration of node B and creates a listening delay for A. Per relay, the additional duration corresponds to the delay of the 2 synchronizations and the communication of the 120 B. Recall that we consider a mean synchronization duration $\bar{t}_{synchro}$ of 48 ms, as explained in Sec-

tion 5.3.2. Based on these elements, we evaluate the communication costs per relay to about 11.1 mJ and 9.7 mJ for the MICAz and TelosB respectively.

Figure 5.5 illustrates the influence of the number of relay nodes on the total cost of Kerberos. It shows that the protocol becomes more costly than ECDH-ECDSA when the distance to the trusted party T becomes large: starting from 11 relay nodes for the TelosB and 25 for the MICAz. While much less costly in terms of computations, Kerberos becomes less interesting than ECDH-ECDSA in situations where the importance of the communication cost becomes overwhelming. Our methodology interestingly allows to locate the tradeoff point in our setting.

5.4.5 Reducing the Listening Cost

The energy assessments of Kerberos and ECDH-ECDSA underlined the importance of the cost of listening, although already minimized using a LPL protocol. Strategies can be developed to further reduce the cost of listening.

One basic way is to accelerate the computations to reduce the listening time of the entity waiting for the protocol next message. This could be done by employing dedicated hardware, even if it increases the price of the sensors.

A second possibility is to adapt the sleep and listen patterns of the LPL to the protocol being carried out. For instance, one can temporarily increase the sleep interval or even put the nodes in sleep mode while the other party is computing. Indeed, the protocol next message cannot precede the end of the computation. This system is feasible if computation timings are known in advance, which is the case most of the time. There is of course an uncertainty concerning the exact duration of a computation, but the minimum duration – the worst case – can be considered.

This approach is especially interesting for computation-intensive protocols as ECDH-ECDSA. For this protocol, it could save most of the LP listen cost of Table 5.6 which is the main component of the communication cost of the protocol. On the other hand, the reduction of the listening cost is done at the expense of a loss of connectivity during the run of the protocol. This may not be a desired feature, for instance when the ability to quickly react in case of emergency has to be preserved.

Finally, employing another LPL protocol could be beneficial. In this case study, we investigated a popular asynchronous LPL protocol, the one proposed within the TinyOS operating system. It would be interesting to also examine the cost of a synchronous LPL protocol, such as e.g., S-MAC [YHE02]. In this protocol, all the nodes are synchronized on sleep schedules and do not require therefore synchronization costs. However, this requires the periodic exchange

of synchronization information, which also consumes energy.

5.5 Comparison with Related Results

The energy cost of key agreement protocols was also investigated in several previous works. Let us now examine how are our estimates compare with these related results.

Kerberos vs Diffie-Hellman. As described in Section 5.2, two preceding works already compared the energy cost of key exchange protocols based on symmetric and asymmetric cryptography in the context of sensor networks. They both selected protocols based on Kerberos and on the Diffie-Hellman key exchange on elliptic curves.

First, there is the work by Hodjat and Verbauwhede [HV02]. They considered a version of the simplified Kerberos quite similar to ours. For the protocol based on public key, they used the standard version of ECDH, which does not provide any authentication and is therefore lighter than ECDH-ECDSA. They obtained the energy cost of computations by measuring the timing performances of their implementations of both protocols. For the communications, they limited their analysis to the use of per-bit transmit and receive costs obtained through measurements in a previous work. Their testing platforms were high-end sensor nodes, the WINS nodes, which contain a powerful microprocessor (32-bit, 133 MHz). The energy cost of the cryptographic primitives was therefore quite different with respect to our estimates concerning sensor nodes with tighter constraints. For instance, for a comparable amount of energy (140 mJ), WINS nodes are able to run Kerberos while TelosB nodes can perform a full ECDH-ECDSA key exchange. Although employing noticeably different sensor node hardware, the authors found that ECDH was between one to two orders of magnitude more costly than Kerberos, which is consistent with our results for the MICAz and TelosB. However, they did not include the cost of listening in their estimates. As a result, they obtained much inferior relative communication costs for both protocols: only 2% for ECDH and 75% for Kerberos.

The second was presented by Großschädl et al. in [GST07]. The authors also compared AES-based Kerberos with ECMQV, a variant of ECDH which provides authentication, on WINS nodes. Their goal was to update the results of Hodjat and Verbauwhede as progress had been made in efficient implementation of ECC since then. They estimated the cost of computations and communications as Hodjat and Verbauwhede. Similarly, they did not take the

listening cost into account. They found that ECMQV was only up to twice as costly as Kerberos on the WINS node. This significant difference with the results of Hodjat and Verbauwhede is mainly explained by a much faster ECC implementation. The difference with our results comes from the fact that the computation cost is relatively less important for the WINS than for the MICAz or TelosB. Also, ECMQV assumes that both participants have already exchanged their long-term public keys. For large networks, this means a large number of stored keys per node, which may not be desirable. Therefore, the exchange and verification of the long-term public keys could be included in the energy assessment, significantly increasing the energy cost of the protocol. Großschädl et al. assessed that the communication represented almost exclusively the cost of Kerberos while it was about one third in the cost of ECMQV. Including the cost of listening in their estimates may significantly increase the already important communication cost.

Public-key cryptography. Two other works assessed and analyzed the energy cost of public-key cryptography in wireless sensor networks. First, Wander et al. [WGE+05] quantified the energy costs of ECDSA operations and an ECC-based certified key exchange similar to ECDH-ECDSA on a platform close to the MICAz, the MICA2DOT [Croa]. They based their estimations on performance measurement of the ECC point multiplication presented in [GPW+04] for the computations. For the cost of communication, they evaluated it thanks to measurements on the MICA2DOT. They obtained a cost of 188 mJ for the whole key exchange, which is lower than our estimate on the MICAz, as they relied on the faster implementation described in [GPW+04]. Interestingly, the authors examined the relative importance of the costs of computation and communication, pointing out that computation is cheap compared to data transmission. They mentioned the importance of listening. However, they did not include it in the communication cost, which still amounts to 22% of the total cost.

In the second work on the topic, Piotrowski et al. [PLP06] extended the results of Wander et al. on other sensor nodes including the MICAz and TelosB. They deduced the expected processing times and energy consumptions for ECDSA operations and the computational part of the uncertified key exchange from the results of Wander et al. For this purpose, they made use of performance and power consumption ratios. They assessed the communication energy costs based on the datasheets of the transceivers. They found a cost of respectively 53 mJ and 12 mJ for the computations of ECDH on the MICAz and the TelosB. This is much lower than our results for two reasons.

First, the uncertified key exchange is computationally much less costly than its certified counterpart. Second, these results refer to the more efficient ECC implementations of [GPW+04]. Concerning the communication cost, the authors concluded that it was not an important factor when comparing cryptographic algorithms in wireless sensor networks. We illustrated that this conclusion is not valid in general, since practical elements have to be taken into account, especially the consumption of the nodes while listening.

5.6 Future Perspectives

The relative importance of communication in the energy cost of cryptographic protocols significantly varies with the type of sensor node hardware used, as underlined in our case study and in related works. We here examine how it should be modified for the next generation of sensor platforms.

On the one hand, new techniques have been propose to reduce the cost of communication and may soon be integrated in sensor nodes. For instance, ultra low-power radio modules are currently being developed, such as the ultra-wide band transceiver proposed at IMEC, reaching a data rate of 1 Mbps for a low consumption of about 1 mW [IME09]. Another promising technique aims to eliminate the cost of listening with the help of an external low-cost wake-up component, which is radio-triggered [LHR10]. This allows to switch on the transceiver only when necessary. These techniques suggest that protocols trading computation for communication, such as cooperative protocols, are expected to be particularly interesting for the future sensor nodes.

On the other hand, the cost of cryptographic computations is expected to considerably diminish with the increasing use of hardware accelaration. Indeed, low-power microcontroller are now beginning to be equipped with cryptoprocessor supporting encryption, such as the AVR XMEGA [Atm]. This should be followed by a hardware support of ECC operations, at least partially (with instruction set extension).

As a result, both the communication and computation costs may largely diminish in the next generation of sensor nodes. We leave to a further work the task of precisely analyzing the relative importance of these costs for the emerging sensor node technology.

Bibliography

[Atm] Atmel Corporation. AVR XMEGA: 8-bit low power, high performance microcontroller. http://www.atmel.com/products/AVR/default_xmega.asp.

[BWDH01] S. Blake-Wilson, T. Dierks, and C. Hawk. ECC cipher suites for TLS. Transport Layer Security Working Group, Internet draft available from http://tools.ietf.org/id/draft-ietf-tls-ecc-01.txt, 2001.

[Croa] CrossBow Technology Inc. MICA2DOT sensor node datasheet. Available at http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICA2DOT_Datasheet.pdf.

[Crob] CrossBow Technology Inc. MICAz sensor node datasheet. Available at http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAz_Datasheet.pdf.

[Croc] CrossBow Technology Inc. TelosB sensor node datasheet. Available at http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/TELOSB_Datasheet.pdf.

[dMGSP08] Giacomo de Meulenaer, Franois Gosset, Franois-Xavier Standaert, and Olivier Pereira. On the Energy Cost of Communications and Cryptography in Wireless Sensor Networks. In *IEEE International Workshop on Security and Privacy in Wireless and Mobile Computing, Networking and Communications (SecPriWiMob'2008)*, pages 580–585, 10 2008.

[EYE] EYES. *European research project on self-organizing and collaborative energy-efficient sensor networks,* `http://www.eyes.eu.org/`.

[FKK96] A. Freier, P. Karlton, and P. Kocher. The SSL protocol version 3.0. Transport Layer Security Working Group, Internet draft available from `http://wp.netscape.com/eng/ssl3/draft302.txt`, November 1996.

[GL05] M. Girault and D. Lefranc. Server-aided verification: theory and practice. In Bimal Roy, editor, *Advances in Cryptology - Asiacrypt'05*, number 3788 in Lecture Notes in Computer Science, pages 605–623. Springer, 2005.

[GMF+05] V. Gupta, M. Millard, S. Fung, Y. Zhu, N. Gura, H. Eberle, and S. C. Shantz. Sizzle: A standards-based end-to-end security architecture for the embedded Internet. In *PERCOM '05: Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications*, pages 247–256, Washington, DC, USA, 2005. IEEE Computer Society.

[GPW+04] N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz. Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs. In *CHES*, pages 119–132, 2004.

[GST07] J. Großschädl, A. Szekely, and S. Tillich. The Energy Cost of Cryptographic Key Establishment in Wireless Sensor Networks. In Robert H. Deng and Pierangela Samarati, editors, *Proceedings of the 2nd ACM Symposium on Information, Computer and Communications Security (ASIACCS 2007)*, pages 380–382. ACM Press, 2007.

[GW08] Vipul Gupta and Michael Wurm. The energy cost of SSL in deeply embedded systems. Technical Report SMLI TR-2008-173, Sun Microsystems Laboratories, Mountain View, CA, USA, 2008.

[HMV03] D. Hankerson, A. Menezes, and S. Vanstone. *Guide to Elliptic Curve Cryptography*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.

[HNL07] M. Healy, T. Newe, and E. Lewis. Efficiently securing data on a wireless sensor network. In *Journal of Physics Conference Series*, volume 76, Issue 1, 2007.

[HV02] A. Hodjat and I. Verbauwhede. The energy cost of secrets in adhoc networks. In *Proc. IEEE Circuits and Systems Workshop on Wireless Communications and Networking*, page 4, 2002.

[IME09] IMEC. Ultra low-power radio. Scientific report available at `http://www.imec.be/ScientificReport/SR2009/HTML/1213343.html`, 2009.

[KSW04] Chris Karlof, Naveen Sastry, and David Wagner. Tinysec: a link layer security architecture for wireless sensor networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 162–175, New York, NY, USA, 2004. ACM.

[LDH06] Y. Law, J. Doumen, and P. Hartel. Survey and benchmark of block ciphers for wireless sensor networks. *ACM Transactions on Sensor Networks*, 2(1):65–93, 2006.

[LHKV04] B.C. Lai, D. Hwang, S. Kim, and I. Verbauwhede. Reducing radio energy consumption of key management protocols for wireless sensor networks. In *Proc. ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED 2004)*, pages 351–356. ACM Press, 2004.

[LHR10] Philippe Le-Huy and Sébastien Roy. Low-power wake-up radio for wireless sensor networks. *IEEE International Conference on Wireless and Mobile Computing, Networking and Communication*, 15(2):226–236, 2010.

[LN07] A. Liu and P. Ning. TinyECC: A configurable library for Elliptic Curve Cryptography in Wireless Sensor Networks. Technical Report TR-2007-36, North Carolina State University, Department of Computer Science, 2007.

[MHLC07] D. Moss, J. Hui, P. Levis, and J. Choi. CC2420 Radio Stack. TinyOS Core Working Group, draft available at `http://www.tinyos.net/tinyos-2.x/doc/pdf/tep126.pdf`, 2007.

[MWS04] David J. Malan, Matt Welsh, and Michael D. Smith. A public-key infrastructure for key distribution in tinyos based on elliptic curve cryptography. In *First IEEE International Conference on Sensor and Ad Hoc Communications and Networks*, 2004.

[NIS01] NIST. Recommendation for Block Cipher Modes of Operation. National Institute of Standards and Technology (NIST) Special Publication 800-38A, 2001.

[NT94] B. Neuman and T. Ts'o. Kerberos: An authentication service for computer networks. *IEEE Communications*, 32(9):33–38, 1994.

[PCG+05] J. Paek, K. Chintalapudi, R. Govindan, J. Caffrey, and S. Masri. A wireless sensor network for structural health monitoring: performance and experience. In *EmNets '05: Proceedings of the 2nd IEEE workshop on Embedded Networked Sensors*, pages 1–9, Washington, DC, USA, 2005. IEEE Computer Society.

[PHC04] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 95–107, New York, NY, USA, 2004. ACM Press.

[PLP06] K. Piotrowski, P. Langendoerfer, and S. Peter. How public key cryptography influences wireless sensor node lifetime. In *SASN '06: Proceedings of the fourth ACM workshop on Security of ad hoc and sensor networks*, pages 169–176, New York, NY, USA, 2006. ACM.

[SHCW04] V. Shnayder, M. Hempstead, B. Chen, and M. Welsh. PowerTOSSIM: Efficient power simulation for TinyOS applications. In *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2004.

[WGE+05] A. Wander, N. Gura, H. Eberle, V. Gupta, and S. C. Shantz. Energy analysis of public-key cryptography for wireless sensor networks. In *Pervasive Computing and Communications, 2005. PerCom 2005. Third IEEE International Conference on*, pages 324–328, 2005.

[YHE02] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient MAC protocol for wireless sensor networks. In *Proceedings 21st In-*

ternational Annual Joint Conference of the IEEE Computer and Communications Societies, New York, New York, USA, 2002.

CHAPTER 6

Cooperative Group Signatures on Constrained Devices

Group signatures allow any member of a group to sign messages on behalf of the group. They are anonymous and unlinkable for every verifier except for a given authority. They are very appealing for applications where the anonymity of users is strongly desired, e.g., in anonymous credentials, e-cash or e-vote. Unfortunately, their complexity is beyond the reach of devices with little computational abilities. In this chapter, we adapt to small constrained devices one of the most secure and efficient group signature scheme, namely XSGS proposed by Delerablée and Pointcheval at Vietcrypt 2006. We show that it can be efficiently implemented in a sensor node or in an RFID tag, in spite of the numerous elliptic curve point multiplications and the pairing evaluation to compute when signing. This is done by securely outsourcing a large part of the computation to an untrusted powerful intermediary (possibly an RFID reader). With this cooperation, XSGS can be executed in the MICAz (8-bit 7.37MHz ATmega128 microprocessor) and the TelosB (16-bit 4MHz MSP430 processor) sensor nodes in less than 200 ms.

6.1 Introduction

Group signatures have been introduced by Chaum and van Heyst at Eurocrypt 1991 [CvH91]. These signatures allow any member of a group to sign a document and any verifier to confirm that the signature has been computed by a group member. They are anonymous and unlinkable for every verifier except, when needed, for a given authority.

Group signatures are of great interest in various applications where the prover anonymity must be preserved. For instance, they represent a core building block of electronic cash systems in which customers are able to pay anonymously [CT03]. They are also very helpful in electronic voting or bidding. Furthermore, they allow the use of anonymous credentials to access services, resources or infrastructures and thus represent an attractive solution to enhance the protection of the users privacy.

Implementing these signature schemes on low-power devices, like RFID tags or contactless smart cards is very appealing for two main reasons. On the one hand, it can support the user mobility in many contexts, as in building access control or mobile payments. On the other hand, gathering the cryptographic secrets and functionalities into a small hardware component is in general recommended when performing cryptographic tasks on potentially insecure platforms as laptops or mobile phones. For this reason, embedding group signatures into a smart card largely increases the overall security of the deployed scheme.

However, fitting group signatures into small constrained devices appears to be a particularly challenging task, as the computation of such a signature typically requires numerous modular exponentiations or pairing evaluations. For instance, it is necessary to compute more than 10 elliptic curve point multiplications and 1 pairing to sign with the eXtremely Short Group Signature (XSGS [DP06]) scheme, one of the most efficient and powerful schemes available today (cf. Section 6.2.3 for a comparison).

This chapter describes a work by Coisel et al. [CdMCP10], in which we design a cooperative variant of the XSGS group signature scheme [DP06]. Our variant is efficient enough to be implemented in a weak device like an RFID tag. Its execution relies on the help of a more powerful intermediary, which can be the tag reader in an RFID scenario[1]. The trust we place in the intermediary is quite limited: we show that a compromised intermediary is not able to impersonate the tag. However, the anonymity of the tag is not guaranteed with respect to the intermediary. On the other hand, our solution ensures that, if the intermediary is honest, the resulting cooperative signatures keep the anonymity properties ensured by group signatures.

We observe that this model fits many practical applications. For instance, when the intermediary is a reader connected to a personal laptop, it makes little sense to expect that the laptop does not contain identity information, meaning that compromising the laptop would anyway result in a loss of anonymity.

[1] Of course, in such a case, we assume that the verification is not performed by the reader itself, but by a remote entity.

Other situations can be found, in which the user anonymity w.r.t. the intermediary is prevented for external reasons. For instance, let us consider an access control where the intermediary is a reader located at a gate. When accessing, a user may not remain anonymous as he can be recognized by a nearby person or monitored by a surveillance camera. On the other hand, for privacy reasons, the user prefers to remain anonymous with respect to the remote access control owner to avoid that its personal data is employed for other purposes than access control.

Our scheme is tractable for constrained devices like high-end RFID tags. We demonstrate this by documenting implementation results on two common wireless sensor nodes, the MICAz and the TelosB sensor nodes, the processor of the TelosB node being also used in contactless secure government electronic ID chips. In particular, the on-line phase of the protocol can be completed in less than 200 ms on both nodes. The off-line phase requires four to six seconds to be completed, but this can be mitigated by our proposition of a coupon mode (from [NMVR94] and the EU project CAFE ESPRIT 7023), which allows a node to precompute or preload (at home, for instance) up to 5000 coupons in advance, while satisfying the memory constraints of our devices. A refill procedure can be foreseen when the stock of coupons is depleted.

Related Work. The way to embed cryptography into low-power devices has been largely studied by the cryptographic community, as presented in Section 4.3. One solution is to make precomputations. In fact, it is sometimes possible to make some computations off-line and to store them, so that it is not necessary to perform them on-line anymore. This has the drawback of consuming a lot of memory space.

Another possibility is to modify the cryptographic mechanism to fit the device restrictions. This has already been done in the RFID case [GL04, FSW04] or when considering the integration of group signatures in a smart card [CG02, XY04]. This approach sometimes necessitates important modifications of the initial algorithm, and may imply some stronger (and questionable) assumptions such as, e.g., tamper-resistance [CG02, XY04]. With this last hypothesis, the security of the whole scheme relies on the hardness of tampering with a single tag, which is not reasonable in the RFID setting, considering the tags poor tamper resistance [HMF07].

In this work, we focus on another approach, which consists in studying how a more powerful entity can help a small device that is programmed to provide a group signature, as done in [MB02] for a scheme based on the Strong-RSA assumption. The introduction of an intermediary device in a signature

scheme must be carefully understood if one wants to avoid introducing severe security flaws in the system. Up to now, a formal security analysis of cooperative group signature schemes is still missing. A security model and some constructions exist for the verifier-side [GL05] in the context of the authentication process. The prover-side has been largely studied in the context of RSA [MKI88, BQ95] where the aim is to help the restricted device to perform a modular exponentiation. Another approach has also been taken in the CAFE project [CP92, CP93], which consists in designing schemes where a powerful prover interacts with a non-trusted smart card to perform some computations in such a way that the prover is unlinkable w.r.t. the smart card.

In the RFID context, lots of paper exists (e.g. [SRS04, ACdM05, SVW09b, SVW09a, ACSW10]) where the constrained device is helped by another entity to "re-encrypt" its data in order to remain untraceable to anybody. Unfortunately, although Armknecht et al. [ACSW10] describe a solution where the tag is untraceable even for the reader, none of these contributions are related to a group signature scheme, as formalized in [BMW03, BSZ05].

Our Contributions. With respect to the preceding related works, our contribution is threefold. To begin with, we propose the first complete security model for cooperative group signatures. Our model allows clarifying the exact level of trust that is placed into the intermediary, this trust directly impacting the amount of computation that can be outsourced by the prover device. The trust model we adopt fits natural application contexts. Then, we propose a new cooperative group signature scheme, based on the XSGS protocol [DP06], and prove its security in our model. Eventually, we demonstrate the efficiency of our scheme by describing two implementations on low-resource wireless sensor nodes. The efficiency gain against the original XSGS scheme is huge, as the low-power device has only one point multiplication to compute instead of ten or so[2] and one pairing in the original scheme. Using precomputation, the on-line phase of the group signature is realized in less than 200 ms on the two sensor nodes we tested, which is quite acceptable in practice.

The work presented in this chapter thus provides fundamental building blocks for the deployment of low-power privacy preserving applications, in contexts where the devices involved in the applications cannot perform heavy computations, which is the case most of the time for the moment.

[2]Precisely 11 when XSGS relies on the eXternal Diffie-Hellman (XDH) assumption or 13 otherwise (c.f. Section 6.4.3.

Outline. After a brief reminder on group signatures (Section 6.2), we extend the standard security properties to the cooperative setting (Section 6.3). We then describe XSGS, in its traditional form (Section 6.4). Next, we present our cooperative XSGS protocol and prove its security with respect to our new security definitions (Section 6.5). Afterward, we demonstrate that our protocol can be executed on small devices, by presenting and discussing the performance of its implementation on small wireless sensor devices, of which the controllers are also sometimes included in RFID tags (Section 6.6). For the sake of completeness, we eventually give another cooperative variant of XSGS, in which the signer remains anonymous w.r.t. the intermediary (Section 6.7). However, the efficiency gain of this stronger variant is not sufficient to allow an implementation on a weak device.

6.2 Definition of Group Signature Schemes

In [CvH91], Chaum and van Heyst introduce the notion of *group signature* schemes where any group member can sign a document on behalf of the group and any verifier can confirm that the signature was issued by a group member. Group signatures have the property of being anonymous and unlinkable for every verifier except for a given authority. There exist lots of contributions in the group signature context (e.g. [ACJT00, BBS04, FI05, DP06, BW07, Gro07]). As we later present a formal security model for group signature schemes, we first formally introduce in this section the concept of group signature and discuss some current constructions.

6.2.1 Generic Description of Group Signatures

Formally speaking, a group signature scheme $\mathcal{GS}$ is described by the following polynomial-time procedures, where λ is a security parameter.

- GENPARAM is a probabilistic algorithm which takes as input 1^λ and which outputs the public parameters of the system $\mathcal{PP} = (\mathsf{Gpk}, \mathsf{Rpk}, \mathsf{params})$ where Gpk is the public key of the group, Rpk is the public key of the opening manager and params are public parameters (e.g., mathematical groups, generators,...). It also outputs the group manager's secret key gmsk and the opening manager's secret key rsk to these respective entities.
- USERKEYGEN is a probabilistic algorithm which attributes to a user U_i a pair of secret/public key $(\mathsf{usk}_i, \mathsf{Upk}_i)$ respecting a Public Key Infrastructure (PKI).

- JOIN is a probabilistic protocol between the group manager and a new group member U_i to provide the latter with his group secret key $\mathsf{gsk}[i]$. The group manager makes an entry $\mathsf{Tab}[i]$ in the registration table Tab, with the entire transcript of the process.

- SIGN is a probabilistic algorithm which takes as input a secret signing key $\mathsf{gsk}[i]$ and a message m and returns the group signature σ on m.

- VERIF is a deterministic algorithm which takes as input a message m, and a candidate signature σ and returns either 1 if the signature is valid or 0 otherwise.

- OPEN is a deterministic algorithm which takes as input the opening manager secret key rsk, the registration table Tab, a message m and a signature σ and returns either an identity i together with a proof τ of this claim or 0 to indicate a failure.

- JUDGE is a deterministic algorithm which takes as input the registration table Tab, a message m, a signature σ, an identity i and a proof τ and returns 1 if the proof τ is a valid proof that user i has produced the signature σ and 0 otherwise.

6.2.2 Security Properties

Bellare et al. introduced in [BSZ05] a formal model, known as the BSZ model, listing the security properties that group signatures are expected to meet. We outline here these properties, that we will later adapt for the case of cooperative group signatures.

- **Correctness**: a signature produced by a valid user U_i must be accepted by a verifier (i.e., VERIF returns 1). Furthermore, the opening of this signature must return the identity of U_i and the judge must validate this opening.

- **Anonymity**: given several signatures of a user randomly chosen among two users, it is unfeasible to distinguish which of these two users have produced this set of signatures.

- **Traceability**: it is unfeasible to produce a valid signature which cannot be opened or where the proof outputted by OPEN cannot be verified. This property must be verified even if several users and also the group manager collude.

- **Non-Frameability**: it is unfeasible, even for the group manager and the opening manager, to claim falsely that a signature has been produced by a user.

To prove that a scheme ensures these properties, Bellare et al. [BSZ05] define for each of these properties an experiment played by an adversary. Depending on the concerned property, the adversary has several possibilities to interact with the system. For example, an adversary can corrupt some users and thus obtains their group secret keys. In some cases, the group manager can be corrupted and thus the adversary can play the role of this group manager during a JOIN procedure. All the possible interactions are realized through oracles[3] which are listed below. Moreover, a list of honest users $\mathcal{HU}$ and one of corrupted users $\mathcal{CU}$ are needed.

- $\mathcal{O}^{\mathsf{CreateU}}$: this oracle generates a new user i using the procedure USERKEYGEN.
- $\mathcal{O}^{\mathsf{AddU}}(i)$: this oracle adds a new user i in the group using the procedure USERKEYGEN and the interactive protocol JOIN. The identity of this new user is added to the list $\mathcal{HU}$. The public key Upk_i of the new user is outputted.
- $\mathcal{O}^{\mathsf{SJoin}}(i)$: during the request to this oracle, the adversary will play the role of the group manager during a JOIN protocol with a new honest user. First of all, the oracle generates a new user i with the procedure USERKEYGEN and simulates him during the protocol with the adversary. This new user is added to $\mathcal{HU}$.
- $\mathcal{O}^{\mathsf{UJoin}}$: this oracle simulates the group manager during a JOIN protocol where the adversary plays the role of the user.
- $\mathcal{O}^{\mathsf{CrptU}}(i)$: this oracle gives the total control of the user i to the adversary. In other words, the adversary obtains all the information related to this user (secret keys, certificates, random values, ...). The member i is moved from $\mathcal{HU}$ to $\mathcal{CU}$.
- $\mathcal{O}^{\mathsf{Reveal}}(i)$: this oracle outputs the secret key usk_i and gsk_i of the member i, who is added in $\mathcal{CU}$.
- $\mathcal{O}^{\mathsf{SignU}}(i, m)$: this oracle outputs the signature on m of the member i.

[3] An oracle is a theoretical black box in the terminology of provable security.

- $\mathcal{O}^{\mathsf{Open}}(m, \sigma)$: this oracle outputs the identity of the user which has produced σ.

- $\mathcal{O}^{\mathsf{Choose}_b}(m, i_0, i_1)$: if i_0 or i_1 have not been given as input to $\mathcal{O}^{\mathsf{CrptU}}$ (i.e.$i_0, i_1 \in \mathcal{HU}$), this oracle outputs the signature σ on the message m of the member i_b (where b is a bit). σ cannot be given as input to the $\mathcal{O}^{\mathsf{Open}}$ oracle. This last oracle does not correspond to an intuitive interaction of the adversary, but it is defined to ease the definition of the anonymity property in Section 6.3.3, as done in [BSZ05].

6.2.3 Some Group Signature Constructions

Currently, there are mostly three families of practical group signature constructions. The first family [Gro07], secure in the standard model, necessitates the use of the Groth-Sahai technique, formalized in [GS08], which makes it today not implementable in constrained devices.

The second one is based on the Strong-RSA assumption [ACJT00, CG04]. As it relies on big modulus and exponents, it necessitates the manipulation of big integers (except when using alternative mathematical tools [Joy09]) and is thus not suitable for RFIDs.

The third family is constructed around pairings and is proven secure under the q-Strong Diffie-Hellman (q-SDH) assumption in the random oracle model [BBS04, FI05, DP06]. This last family provides the shortest group signatures and seems thus more relevant for constrained devices.

Within this family, the BBS scheme [BBS04] only considers static groups while others [FI05, DP06] are secure in the dynamic case [BSZ05]. We here base our study on the XSGS protocol from [DP06], which is described in Section 6.4. In fact, our study is also relevant for the scheme in [FI05] but we have chosen the XSGS one as it includes a complete security study, which is not the case in [FI05][4].

The main drawback of XSGS is that it needs one pairing evaluation and many elliptic curve point multiplications to produce a group signature. But this is the case for many other group signature schemes. In fact, this complexity places XSGS as one of the most efficient group signature schemes which are today available.

[4] In order to reach the full anonymity property described in [BSZ05], the proposal in [DP06] uses the Naor-Yung methodology [NY89], and thus twice the same encryption scheme with the same message together with a proof. The scheme in [FI05] does not totally uses this method and the resulting security is not discussed in the paper.

Our purpose in this work is now to propose a secure (see next section) and efficient (see Sections 6.5 and 6.6) version of XSGS. Our solution is to outsource most of the computation load (i.e., the pairing and most of the point multiplications) to an intermediary, in which a limited trust is placed.

6.3 Security of Cooperative Group Signatures

A cooperative group signature [MB02] allows a group member, with constrained resources, to be helped by some powerful entity, called an intermediary, in the production of the group signature. The problem on which we focus here is that, in the cooperative context, the intermediary will certainly have at his disposal some information that is traditionally not available to the adversary. We thus give in this section all the assumptions regarding the intermediary and we adapt the security properties of a group signature scheme to the cooperative context. This work has not been totally done in [MB02].

6.3.1 Concept and Assumptions

We first assume that the intermediary does not know any secret information and thus, at the beginning of a protocol, the signer may transfer some data to the intermediary in order to decrease its computation complexity. Consequently, an adversary may obtain more information to break the security of the scheme, e.g., by eavesdropping on the communications. It is thus obvious that we must model all her new abilities.

In the cooperative setting, the "standard" adversary (meaning the adversary of the original group signature scheme) is improved in three different manners. First, the adversary can obtain from the intermediary all data that have been sent by the signer. Second, she can eavesdrop on all communications during a signature protocol (at least the shared data but potentially more information). Finally, she can impersonate the intermediary, and thus obtain all the exchanged information. Moreover, she can learn all the choices made by the intermediary during the protocol (e.g., random values). It is clear that this last adversary ability is more powerful than the two others. Consequently, we only formally model this one in the cooperative setting and thus introduce the new following oracle.

- $\mathcal{O}^{\mathsf{PartialSign}(i,m)}$: this oracle simulates for the adversary the behavior of the user i realizing the cooperative signature protocol of a message m. During this request, several exchanges between the oracle and the ad-

versary can be done as it simulates a real cooperative protocol execution between a signer and the adversary playing the role of the intermediary.

Remark 1. An adversary can also impersonate a signer to a legitimate intermediary. However, as all computations are based on public values, or on information sent by the signer, the adversary is obviously able to impersonate the intermediary at the same time. We thus decide to not model this point.

Now, based on this adversary's new ability, we must adapt the security properties of group signature schemes to the cooperative setting. As we have only sketched the security properties of a group signature schemes in Section 6.2.2, we formally define them in the following, based on the model of Bellare et al. [BSZ05].

6.3.2 Adaptation of the Correctness

The first security property concerns the correctness. Group signatures outputted by the cooperative protocol must be accepted and furthermore, a judge must be able to correctly open them. In our cooperative context, we decide that the intermediary may realize the connection with the "outside world". Thus, if he decides to send a false signature, or to stop the protocol, the signer cannot do anything to rectify this. Consequently, it seems impossible to ensure a cooperative correctness property where the adversary can actively participate during the experiment. Nevertheless, the cooperative protocol should at least ensure the correctness property of group signatures in the presence of a "standard" adversary.

Definition 1. *The* correctness predicate *of a group signature scheme, denoted $\mathcal{E}_{corr}^{GSS}$, is verified for a user i and a message m if and only if the following conditions are verified:*

$$\begin{aligned} & \textsc{Verif}(m, \textsc{Sign}(\mathit{gsk}[i], m)) = 1 \\ \wedge \quad & \textsc{Open}(\mathit{rsk}, \mathit{Tab}, m, \textsc{Sign}(\mathit{gsk}[i], m)) = (i, \tau) \\ \wedge \quad & \textsc{Judge}(\mathit{Tab}, m, \textsc{Sign}(\mathit{gsk}[i], m), i, \tau) = 1 \end{aligned}$$

We denote $\mathcal{E}_{corr}^{GSS}(i, m) = 1$ if this predicate is true, and $\mathcal{E}_{corr}^{GSS}(i, m) = 0$ otherwise.

A cooperative scheme ensures the correctness *property if there exists a negligible function $\epsilon(\lambda)$ such that :*

$$\forall (i, m) : \ \ Pr[\mathcal{E}_{corr}^{GSS}(i, m) = 0] < \epsilon(\lambda)$$

If the outputs of both the initial and the cooperative versions of a group signature scheme are constructed with identical operations, then the cooperative scheme ensures the correctness property if the initial is also correct in the BSZ model [BSZ05].

6.3.3 Adaptation of the Anonymity

To increase as much as possible the efficiency of the cooperative protocol, we allow the signer to divulge its identity to the intermediary. This makes it possible to transfer more data to the intermediary and thus to reduce the computational load for the signer. This loss of anonymity with respect to the intermediary is not necessarily a problem, for instance if we assume that the signer and the intermediary live in a personal environment. We indeed remark that in the standard model, group signatures are computationally very expensive for constrained devices, e.g., smart cards or RFID tags. Thus, the protocol is in practice completely realized on a powerful device such as a personal computer, which consequently already knows the user's identity. Therefore, allowing it to know the user identity does not introduce an additional threat as it should know it anyway in the standard protocol.

There are other situations where a weak signer is in the vicinity of a powerful intermediary toward which remaining anonymous is difficult or meaningless. It could be the case of a Trusted Platform Module (TPM) within a mobile phone. The RFID scenario provides another example, with access control. At the control gate, the users may be monitored by a surveillance camera or visually recognized (by a colleague for instance), such that their anonymity is locally prevented. On the other hand, they may wish to remain anonymous with respect to the access control authority, for privacy reasons. In this situation, the RFID reader located at the gate can act as the intermediary. Of course, it may not be in the same time the verifier, which is assumed to be distant. While the user anonymity is not well protected in this case, the cooperative group signatures still enhance the security of the scheme: the user secrets are better protected as they are located inside a contacless smart card.

By allowing the intermediary to know the user identity, the cooperative scheme should only verify the "standard" anonymity property: if the adversary does not learn the user identity thanks to the interactions with the intermediary, the anonymity should be guaranteed. In other words, if the initial group signature scheme provides anonymity (in the BSZ sense), then a cooperative version necessarily verifies this "weak" anonymity property. By doing this assumption, it is generally possible to transfer more data to the intermediary and thus to reduce the signer's complexity by a better factor.

Definition 2 (**Anonymity Property**). *A cooperative scheme ensures the* anonymity *property if there exists a negligible function* $\epsilon(\lambda)$ *such that :*

$\text{GENPARAM}(1^\lambda) \rightarrow \mathcal{PP};\ \forall b \in \{0,1\}$:

$$\Big| Pr\big[\mathcal{A}(\mathsf{gmsk}) \rightarrow 1 \mid b = 1\big] - Pr\big[\mathcal{A}(\mathsf{gmsk}) \rightarrow 1 \mid b = 0\big] \Big| < \epsilon(\lambda)$$

for any polynomial adversary $\mathcal{A}$*, who has access to* $\mathcal{O}^{\mathsf{CreateU}}$, $\mathcal{O}^{\mathsf{AddU}}$, $\mathcal{O}^{\mathsf{SJoin}}$, $\mathcal{O}^{\mathsf{UJoin}}$, $\mathcal{O}^{\mathsf{CrptU}}$, $\mathcal{O}^{\mathsf{Reveal}}$, $\mathcal{O}^{\mathsf{SignU}}$, $\mathcal{O}^{\mathsf{Open}}$, $\mathcal{O}^{\mathsf{Choose}_b}$.

Remark 2. In some cases, being unlinkable *w.r.t.* the intermediary is a really important issue and needs to be studied. For the completeness of the model, it is consequently possible to provide a stronger definition for the cooperative anonymity property, where the main difference with the above one is that the signer now does not trust the intermediary and wants to be anonymous and unlinkable also *w.r.t.* it. The adversary is thus additionally given access to the $\mathcal{O}^{\mathsf{PartialSign}}$ oracle in the above experiment.

6.3.4 Adaptation of the Traceability

Concerning the two remaining security properties, it is necessary to give to the adversary the access to the $\mathcal{O}^{\mathsf{PartialSign}(i,m)}$ oracle in the cooperative versions of the related experiments. We first focus on the traceability property.

Definition 3 (**Traceability Property**). *The* traceability predicate *of a group signature scheme, denoted* $\mathcal{E}_{trac}^{GSS}$*, is verified for* (m, σ) *if and only if the following conditions are verified:*

$$\text{VERIF}(m,\sigma) = 1 \ \wedge \ \Big[\text{OPEN}(m,\sigma,\mathsf{rsk}) = 0 \vee$$
$$\Big(\text{OPEN}(m,\sigma,\mathsf{rsk}) = (\mathsf{Upk},\tau) \wedge \text{JUDGE}(\sigma,m,\tau,\mathsf{Upk}) = 0\Big)\Big]$$

We denote $\mathcal{E}_{trac}^{GSS}(m,\sigma) = 1$ *if this predicate is true, and* $\mathcal{E}_{trac}^{GSS}(m,\sigma) = 0$ *otherwise.*

A cooperative scheme ensures the traceability *property if there exists a negligible function* $\epsilon(\lambda)$ *such that for any polynomial adversary* $\mathcal{A}$*, who has access to* $\mathcal{O}^{\mathsf{CreateU}}$, $\mathcal{O}^{\mathsf{AddU}}$, $\mathcal{O}^{\mathsf{SJoin}}$, $\mathcal{O}^{\mathsf{UJoin}}$, $\mathcal{O}^{\mathsf{CrptU}}$, $\mathcal{O}^{\mathsf{Reveal}}$, $\mathcal{O}^{\mathsf{SignU}}$, $\mathcal{O}^{\mathsf{Open}}$, $\mathcal{O}^{\mathsf{PartialSign}}$*:*

$$\text{GENPARAM}(1^\lambda) \rightarrow \mathcal{PP}; Pr\Big[\mathcal{A}(\mathsf{gmsk}) \rightarrow (m,\sigma) : \mathcal{E}_{trac}^{GSS}(m,\sigma) = 1\Big] < \epsilon(\lambda).$$

Note that the traceability predicate is verified even when the user, which possesses (usk, Upk), is corrupted, as in the standard security definition of a group signature scheme [BSZ05].

6.3.5 Adaptation of the Non-frameability

We next study the non-frameability property, for which we introduce a list Set which contains all valid signatures outputted during the experiment (namely all signatures realized by the $\mathcal{O}^{\mathsf{SignU}}$ oracle).

Definition 4 (Non-Frameability Property). *The* non-frameability predicate *of a group signature scheme, denoted $\mathcal{E}^{GSS}_{NonFra}$, is verified for (m, σ) if and only if the following conditions are verified, where $(\mathsf{Upk}_i, \tau) = \text{OPEN}(m, \sigma, \mathsf{rsk}, \mathsf{Tab})$:*

$$\text{VERIF}(m, \sigma) = 1 \wedge (m, \sigma, i) \notin \mathsf{Set} \wedge i \in \mathcal{HU} \wedge \text{JUDGE}(m, \sigma, \tau, \mathsf{Upk}_i, \mathsf{Tab}) = 1.$$

We denote $\mathcal{E}^{GSS}_{NonFra}(m, \sigma) = 1$ if this predicate is true, and $\mathcal{E}^{GSS}_{NonFra}(m, \sigma) = 0$ otherwise.

A cooperative scheme ensures the non-frameability *property if there exists a negligible function $\epsilon(\lambda)$ such that for any polynomial adversary $\mathcal{A}$, who has access to $\mathcal{O}^{\mathsf{AddU}}$, $\mathcal{O}^{\mathsf{CrptU}}$, $\mathcal{O}^{\mathsf{Reveal}}$, $\mathcal{O}^{\mathsf{SignU}}$, $\mathcal{O}^{\mathsf{Open}}$, $\mathcal{O}^{\mathsf{PartialSign}}$:*

$\text{GENPARAM}(1^\lambda) \rightarrow \mathcal{PP}$;

$$Pr\Big[\mathcal{A}(\mathsf{gmsk}, \mathsf{rsk}) \rightarrow (m, \sigma) : \mathcal{E}^{GSS}_{NonFra}(m, \sigma) = 1\Big] < \epsilon(\lambda).$$

6.4 XSGS Group Signature

In this section, we give some useful tools and next focus on the XSGS group signature scheme, of which we give a cooperative version in the next section. In the original paper [BBS04], the scheme is introduced using multiplicative notations. As we use elliptic curves for our implementation (Section 6.6), we here describe the scheme using additive notations.

6.4.1 Some Notations and Tools

Bilinear Groups.

We first give some words on the bilinear maps, following [BBS04]. Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be cyclic groups of prime order q (the former two being additively

written and $\mathbb{G}_T$ being multiplicatively written). Let ψ be an isomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$. G_1 (resp. G_2) is a generator of $\mathbb{G}_1$ (resp. $\mathbb{G}_2$). Finally, let e be a computable bilinear map $\mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ such that $e(G_1, G_2) \neq 1$ and for all $P_1 \in \mathbb{G}_1$, $P_2 \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}$, $e(a.P_1, b.P_2) = e(P_1, P_2)^{ab}$. $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \psi)$ is called a bilinear environment.

Zero-Knowledge Proofs of Knowledge.

Roughly speaking, a Zero Knowledge Proof of Knowledge (ZKPK), formalized by Feige, Fiat and Shamir in [FFS88], is an interactive protocol during which an entity $\mathcal{P}$ proves to a verifier $\mathcal{V}$ that he knows a set of secret values $\alpha_1, \ldots, \alpha_q$ verifying a given relation R without revealing anything else. These protocols are also used to prove that some public values are well-formed from secret values known by the prover. It is possible to transform these protocol into non-interactive proof of knowledge, generally called signature of knowledge, using the Fiat-Shamir heuristic presented in [FS86].

In the sequel, we denote by $\text{SOK}(\alpha_1, \ldots, \alpha_q : \mathsf{R}(\alpha_1, \ldots, \alpha_q))$ a signature of knowledge of the secrets $\alpha_1, \ldots, \alpha_q$ verifying the relation R. We also define π as the interactive protocol between a prover $\mathcal{P}$, on input $\alpha_1, \ldots, \alpha_q$ and R and a verifier $\mathcal{V}$ on input R and which allows $\mathcal{P}$ to prove that he knows the secrets in a zero-knowledge manner. The output of $\mathcal{V}$ is either 1 if the prover is accepted and 0 otherwise.

6.4.2 Decision Linear Problem and Encryption

The Decision Linear Problem, introduced in [BBS04], can be seen as an extension of the Decision Diffie-Hellman (DDH) problem, and is defined as follows.

Definition 5. *Given $G, G', H, \alpha.G, \beta.G', \gamma.H \in \mathbb{G}$ as input, the Decision Linear Problem consists to decide if $\gamma = \alpha + \beta$ or not.*

It is quite obvious that a solver of this problem can be adapted to solve the DDH problem. On the other hand, the opposite is not always true. A great advantage of this problem is that it is still a hard problem even in bilinear groups where the DDH problem is easy. Based on this problem, the authors of [BBS04] introduced a new encryption scheme called Linear Encryption. We describe here the procedures of this scheme.

- $\text{GENPARAM}(1^\lambda)$: let $\mathbb{G}$ be a group of prime order q. Select three generators G, G' and Rpk such that there exists $\mathsf{rsk}_1, \mathsf{rsk}_2 \in \mathbb{Z}_q$ which verify $\mathsf{Rpk} = \mathsf{rsk}_1.G = \mathsf{rsk}_2.G'$. The public key of the system is the tuple (G, G', Rpk) while the secret key is $(\mathsf{rsk}_1, \mathsf{rsk}_2)$.

- ENC(m): to encrypt the message $m \in \mathbb{G}$, this algorithm selects two random values $\alpha, \beta \in \mathbb{Z}_q$ and computes $T_1 = \alpha.G, T_2 = \beta.G', T_3 = m + (\alpha + \beta)\mathsf{Rpk}$. The encrypted message is (T_1, T_2, T_3).
- DEC(T_1, T_2, T_3): to decrypt a message, this algorithm computes $m = T_3 - \mathsf{rsk}_1.T_1 - \mathsf{rsk}_2.T_2$.

To define the parameters of this scheme verifying $\mathsf{Rpk} = \mathsf{rsk}_1.G = \mathsf{rsk}_2.G'$, a solution is described by the next steps:

- choose a random generator $G \in \mathbb{G}$;
- choose a random value $\mathsf{rsk} \in_R \mathbb{Z}_q$ and compute $G' = \mathsf{rsk}.G$;
- choose a first secret key $\mathsf{rsk}_1 \in_R \mathbb{Z}_q$ and compute $\mathsf{Rpk} = \mathsf{rsk}_1.G$;
- compute $\mathsf{rsk}_2 = \mathsf{rsk}_1/\mathsf{rsk} \pmod q$.

We thus have $\mathsf{Rpk} = \mathsf{rsk}_1.G$ by construction, but also $\mathsf{Rpk} = \mathsf{rsk}_2.G'$ as:

$$\mathsf{Rpk} = \mathsf{rsk}_2.G' = \frac{\mathsf{rsk}_1}{\mathsf{rsk}}.G' = \frac{\mathsf{rsk}_1}{\mathsf{rsk}}(\mathsf{rsk}.G) = \mathsf{rsk}_1.G.$$

6.4.3 The XSGS Group Signature Scheme

Here, we focus on the XSGS protocol, introduced by Delerablée and Pointcheval in [DP06], as it is one of the most secure group signature scheme and has also the advantage that the signatures are relatively short.

In the following description, we adapt XSGS to avoid relying on the eXternal Diffie-Hellman (XDH) assumption, as suggested in [DP06]. This is done for security reasons (see Section 8.1 of the extended version of [BBS04] for more details). The modification consists in replacing the double El Gamal encryption [Gam85] by the double Linear encryption [BBS04]. The resulting group signature is slightly bigger.

The XSGS scheme is described by the following procedures, where λ is a security parameter.

- GENPARAM(1^λ): this procedure generates the public parameters of the system and also the keys of the different entities as follows:
  - a bilinear environment $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \psi)$, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are additive groups and $\mathbb{G}_T$ a multiplicative group, all groups having a prime order q;

- a random generator $G_2 \in_R \mathbb{G}_2$ and a generator $G_1 \in \mathbb{G}_1$ such that $G_1 = \psi(G_2)$;
- the parameters for the double linear encryption, i.e., a generator $G \in_R \mathbb{G}_1$ and another generator $G' = \mathsf{rsk}.G$ where $\mathsf{rsk} \in_R \mathbb{Z}_q$;
- the secret keys of the opening judge $(\mathsf{rsk}_1, \mathsf{rsk}_3) \in_R \mathbb{Z}_q^2$, $\mathsf{rsk}_2 = \mathsf{rsk}_1/\mathsf{rsk}$, $\mathsf{rsk}_4 = \mathsf{rsk}_3/\mathsf{rsk}$ and the associated public keys $\mathsf{Rpk}_1 = \mathsf{rsk}_1.G = \mathsf{rsk}_2.G', \mathsf{Rpk}_2 = \mathsf{rsk}_3.G = \mathsf{rsk}_4.G'$;
- the secret key $\mathsf{gmsk} \in_R \mathbb{Z}_q$ of the group manager $\mathcal{GM}$ and the associated public key $\mathsf{GMpk} = \mathsf{gmsk}.G_2$;
- the parameters of Paillier's encryption scheme [Pai99] for EXTCOMMIT

The public parameters of the system are $\mathcal{PP} = \{\lambda, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \psi, G_1, G_2, G, G', \mathsf{GMpk}, \mathsf{Rpk}_1, \mathsf{Rpk}_2\}$.

- USERKEYGEN(1^λ): before that a user, denoted U_i, can join a group, he has to be registered in a Public Key Infrastructure (PKI). This procedure permits to ensure the unlinkability and the non-repudiation of the system. At the end of this procedure, the user obtain a couple of key $(\mathsf{usk}_i, \mathsf{Upk}_i)$. The value Upk_i is added in a table UPK, which is supposed public.

- JOIN[$\mathcal{U}_i(\mathsf{usk}_i, \mathsf{Upk}_i) \leftrightarrow \mathcal{GM}(\mathsf{UPK}, \mathsf{gmsk})$]: this interactive protocol between a user U_i and the group manager achieves the adhesion of the new user to the group. The user obtains a group certificate $\mathsf{cert}_i = (A_i, x_i)$, and his group secret key gsk_i. The group manager adds an entry $(\mathsf{Upk}_i, A_i, x_i, S)$ in Tab, where S is a signature of A_i made by the user U_i with his secret key usk_i. This interactive protocol is presented in Figure 6.1 where EXTCOMMIT denotes an extractable commitment done with Paillier's encryption scheme.

- SIGN($m, \mathsf{gsk}_i, \mathsf{cert}_i$): the signature of a message m is composed of the two following steps:

  - a double linear encryption, namely, the user randomly chooses $(\alpha_1, \beta_1, \alpha_2, \beta_2) \in_R \mathbb{Z}_q$ and computes the six values

$$T_1 = \alpha_1.G; \quad T_2 = \beta_1.G'; \quad T_3 = A_i + (\alpha_1 + \beta_1)\mathsf{Rpk}_1;$$
$$T_4 = \alpha_2.G; \quad T_5 = \beta_2.G'; \quad T_6 = A_i + (\alpha_2 + \beta_2)\mathsf{Rpk}_2;$$

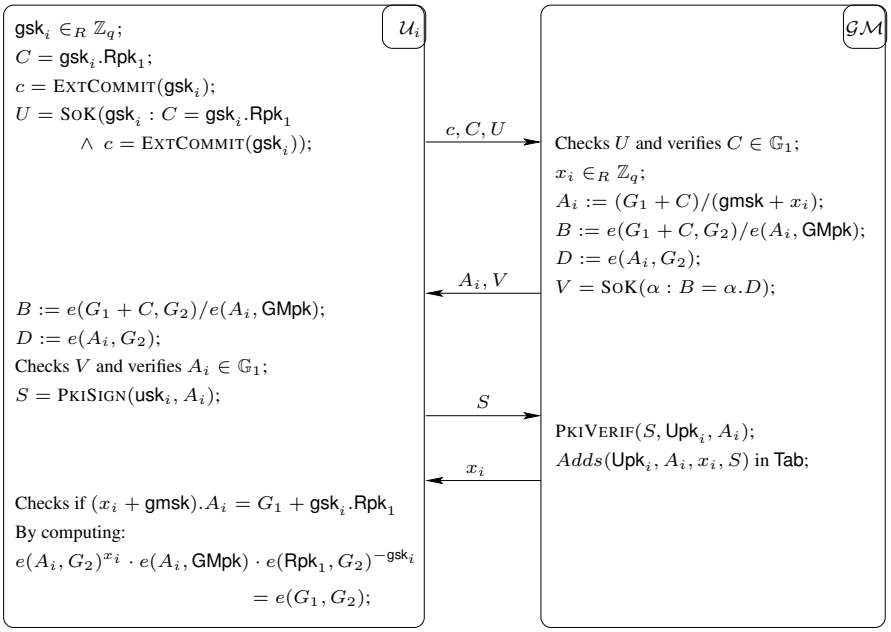


Figure 6.1: JOIN procedure of the XSGS protocol between a user $\mathcal{U}_i$ and the group manager $\mathcal{GM}$.

- the user computes the following signature of knowledge[5] U, where $z = (\alpha_1 + \beta_1).x_i + \mathsf{gsk}_i$:

$$
\begin{aligned}
U \;=\; & \textsc{SoK}\Big(\alpha_1, \beta_1, \alpha_2, \beta_2, x, z : \\
& T_1 = \alpha_1.G \wedge \\
& T_2 = \beta_1.G' \wedge \\
& T_4 = \alpha_2.G \wedge \\
& T_5 = \beta_2.G' \wedge \\
& T_3 - T_6 = (\alpha_1 + \beta_1)\mathsf{Rpk}_1 - (\alpha_2 + \beta_2)\mathsf{Rpk}_2 \wedge \\
& e(T_3, G_2)^{x_i} \cdot e(\mathsf{Rpk}_1, \mathsf{GMpk})^{-(\alpha_1+\beta_1)} \cdot e(\mathsf{Rpk}_1, G_2)^{-z} \\
& = e(G_1, G_2)/e(T_3, \mathsf{GMpk})\Big)(m).
\end{aligned}
$$

The user outputs the signature $\sigma = (T_1, T_2, T_3, T_4, T_5, T_6, U)$.

[5]The complete description of this signature of knowledge is detailed in Section 6.4.4.

- VERIF(m, σ): this procedure simply verifies the correctness of the signature of knowledge U, as detailed in Section 6.4.4.

- OPEN$(m, \sigma, (\mathsf{rsk}_1, \mathsf{rsk}_2, \mathsf{rsk}_3, \mathsf{rsk}_4), \mathsf{Tab})$: if the signature is valid, the opening judge computes $A_i = T_3 - (\mathsf{rsk}_1.T_1 + \mathsf{rsk}_2.T_2)$ and realizes the signature of knowledge $\tau = \text{SOK}\big(\mathsf{rsk}_1, \mathsf{rsk}_2 : A_i = T_3 - (\mathsf{rsk}_1.T_1 + \mathsf{rsk}_2.T_2) \wedge \mathsf{Rpk}_1 = \mathsf{rsk}_1.G \wedge \mathsf{Rpk}_1 = \mathsf{rsk}_2.G'\big)$. By using Tab, the judge can retrieve the key Upk_i associated to the user certificate A_i. Then he outputs Upk_i, S (i.e., $\text{PKISIGN}_{\mathsf{usk}_i}(A_i)$), A_i and τ.

- JUDGE$(m, \sigma, A_i, \tau, \mathsf{Upk}_i, \mathsf{Tab})$: this procedure verifies the correctness of the signature of knowledge τ. The signature S, stored in Tab, permits to check that the certificate A_i is the one that has been given to the user during the JOIN procedure. If both signatures (τ and S) are valid, the procedure outputs 1 else it outputs 0.

This protocol ensures all the security requirements of a group signature scheme under the q-SDH [BB08] and the decision linear assumptions (see Section 6.4.2). We refer the interested reader to [DP06] and [BBS04] for the security aspects of this scheme.

6.4.4 Focus on the Signature of Knowledge

During the signature of a message, a user must produce the signature of knowledge U. We detail here how this should be done.

- Choose $r_{\alpha_1}, r_{\beta_1}, r_{\alpha_2}, r_{\beta_2}, r_x, r_z \in_R \mathbb{Z}_q$

- Compute

$$
\begin{aligned}
P_1 &= r_{\alpha_1}.G;\\
P_2 &= r_{\beta_1}.G';\\
P_3 &= r_{\alpha_2}.G;\\
P_4 &= r_{\beta_2}.G';\\
P_5 &= (r_{\alpha_1} + r_{\beta_1})\mathsf{Rpk}_1 - (r_{\alpha_2} + r_{\beta_2})\mathsf{Rpk}_2;\\
P_6 &= e(T_3, G_2)^{r_x} \cdot e(\mathsf{Rpk}_1, \mathsf{GMpk})^{-(r_{\alpha_1}+r_{\beta_1})} \cdot e(\mathsf{Rpk}_1, G_2)^{-r_z}.
\end{aligned}
$$

- Compute $c = \mathcal{H}(m, T_1, T_2, T_3, T_4, T_5, T_6, P_1, P_2, P_3, P_4, P_5, P_6)$

- Compute

$$s_{\alpha_1} = r_{\alpha_1} + c.\alpha_1 \pmod q; s_{\beta_1} = r_{\beta_1} + c.\beta_1 \pmod q;$$
$$s_{\alpha_2} = r_{\alpha_2} + c.\alpha_2 \pmod q; s_{\beta_2} = r_{\beta_2} + c.\beta_2 \pmod q;$$
$$s_x = r_x + c.x_i \pmod q; s_z = r_z + c.z \pmod q.$$

The signature is the tuple $U = (c, s_{\alpha_1}, s_{\beta_1}, s_{\alpha_2}, s_{\beta_2}, s_x, s_z)$.

The verification of this signature of knowledge is done as follows. The verifier first computes:

$$\begin{aligned}
P_1 &= s_{\alpha_1}.G - c.T_1;\\
P_2 &= s_{\beta_1}.G' - c.T_2;\\
P_3 &= s_{\alpha_2}.G - c.T_4;\\
P_4 &= s_{\beta_2}.G' - c.T_5;\\
P_5 &= (s_{\alpha_1} + s_{\beta_1})\mathsf{Rpk}_1 - (s_{\alpha_2} + s_{\beta_2})\mathsf{Rpk}_2 - c.(T_3 - T_6);\\
P_6 &= e(T_3, G_2)^{s_x} \cdot e(\mathsf{Rpk}_1, \mathsf{GMpk})^{-(s_{\alpha_1}+s_{\beta_1})} \cdot e(\mathsf{Rpk}_1, G_2)^{-s_z}\\
&\quad \cdot (e(G_1, G_2)/e(T_3, \mathsf{GMpk}))^{-c}.
\end{aligned}$$

Finally the verifier validates the signature of knowledge if:

$$c = \mathcal{H}(m, T_1, T_2, T_3, T_4, T_5, T_6, P_1, P_2, P_3, P_4, P_5, P_6).$$

6.5 The Cooperative Version of XSGS

Our aim is now to adapt the previously described XSGS protocol in a secure cooperative manner such that it can be embedded in a constrained device. For this reason, we adapt the XSGS protocol such that the signer is not anonymous w.r.t. the intermediary. In this section, we thus describe a weak anonymous cooperative version of the XSGS scheme (denoted in the following Coop-XSGS), prove its security and assess its efficiency gain. We later give in Section 6.7 a strong anonymous cooperative variant of XSGS, of which the complexity remains prohibitive for small devices like RFID tags.

6.5.1 Protocol Description

At the beginning of the signature protocol, the signer transmits its certificate (A_i, x_i) (see Section 6.4 for details) to the intermediary, which performs all the computations related to it. The signer does not disclose his group secret

key and computes all the values based on it. The obtained cooperative version of the protocol is described in Figure 6.2. The efficiency gain from the signer point of view is huge as there only remains one point multiplication to compute instead of one pairing and at least 11 point multiplications in the original XSGS protocol.

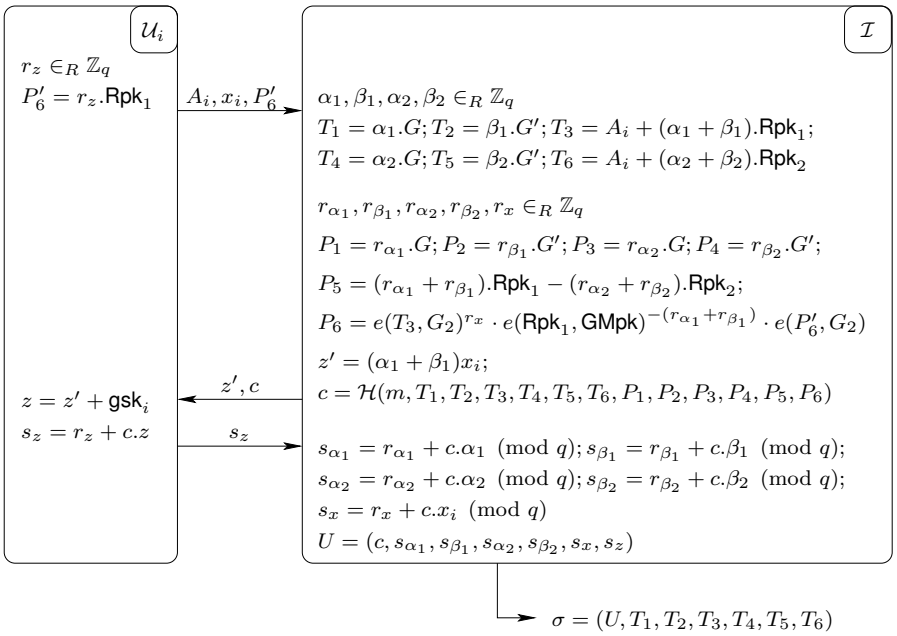


Figure 6.2: Sign procedure of the cooperative XSGS Protocol. A user $\mathcal{U}_i$ is assisted by a powerful intermediary $\mathcal{I}$, who performs the main part of the computations, to produce a signature σ.

6.5.2 Security Analysis

Intuitively, the transmission of the certificate to the intermediary does not seem to introduce any security flaw since both **Traceability** and **Non-Frameability** assume that even the group manager cannot break these properties. As this entity already knows all users certificates, it should be also hard for an active adversary to break these properties. Nevertheless we will formally prove these results.

First of all, we should prove that our cooperative protocol perfectly realizes a XS group signature, and thus that it verifies the correctness property.

Theorem 1. *The Coop-XSGS protocol ensures the Correctness property.*

Proof. Recall that for this property we assume that the intermediary has a honest behavior and executes perfectly his part of the protocol. It is obvious that values T_1 to T_6 are exactly similar to the ones of the standard protocol (see Section 6.4). On the contrary, the signature of knowledge U in the cooperative protocol slightly deviates from the standard one and we need to verify its correctness. More precisely, the deviation is on the P_6 value. Based on the pairing property (see Section 6.4.1), we have:

$$\begin{aligned} P_6 &= e(T_3, G_2)^{r_x} \cdot e(\mathsf{Rpk}_1, \mathsf{GMpk})^{-(r_{\alpha_1}+r_{\beta_1})} \cdot e(P_6', G_2) \\ &= e(T_3, G_2)^{r_x} \cdot e(\mathsf{Rpk}_1, \mathsf{GMpk})^{-(r_{\alpha_1}+r_{\beta_1})} \cdot (r_z.\mathsf{Rpk}_1, G_2) \\ &= e(T_3, G_2)^{r_x} \cdot e(\mathsf{Rpk}_1, \mathsf{GMpk})^{-(r_{\alpha_1}+r_{\beta_1})} \cdot e(\mathsf{Rpk}_1, G_2)^{r_z} \end{aligned}$$

Thus, the whole group signature is computed identically as in the standard protocol. Consequently, the cooperative protocol is correct. □

Theorem 2. *The Coop-XSGS protocol ensures the Anonymity property.*

Proof. As explained in Section 6.3 and since the scheme is correct (see above theorem), the proposed cooperative scheme ensures the standard anonymity property. □

Theorem 3. *The Coop-XSGS protocol ensures the Traceability property.*

Proof. This proof is relatively obvious since the adversary has no more information than the adversary in the standard model. Indeed, this security property is verified even when the adversary represents a collusion of members (thus knowing their group secret keys and certificates). Consequently, the cooperative version of the XSGS protocol trivially verifies the traceability property. □

Theorem 4. *The Coop-XSGS protocol ensures the Non-Frameabilty property.*

Proof. In the original non-frameability proof (see proof of Theorem 11 in [DP06]), the authors use the "unforgeability techniques" to retrieve the certificate and the group secret key used in a given signature outputted by an adversary. Thus, they build an algorithm which interacts with this adversary in order to break the discrete logarithm either in $\mathbb{G}_1$ or in $\mathbb{G}_2$ (depending of the retrieved group secret key).

This proof only works if the adversary does not know the group secret key of the targeted user. As the JOIN procedure does not leak any information

about it, their proof is correct. For the cooperative protocol, this proof can also be applied if and only if we prove that an active adversary cannot learn any information about this secret key.

For this purpose, we first highlight the fact that the protocol between the signer and the intermediary can be interpreted as a Schnorr protocol (see [Sch89] for further details) which has been proven to be a zero-knowledge proof of knowledge. As a consequence, the values P'_6 and s_z do not reveal any information about gsk_i.

It is next obvious that the intermediary has no information about the value r_z under the discrete logarithm assumption. Consequently, given a fixed value of s_z, whatever the value gsk_i is equal to, there exists one value r_z such that $s_z = r_z + ((\alpha_1 + \beta_1)x_i + \mathsf{gsk}_i)c$. As a result, a perfect simulation of $\langle P'_6, c, s_z \rangle$ can be realized as for the zero-knowledge property of Schnorr's protocol (see [Sch89]). Then, if r_z is uniformly chosen in $\mathbb{Z}_q$, the group secret key of the user is perfectly hidden in s_z. □

We thus have designed a secure cooperative version of the XSGS group signature. Theoretically speaking, this protocol appears to be really efficient from the signer point of view. In the following, we estimate the precise efficiency gain of the cooperative version w.r.t. the original protocol. We later demonstrate this efficiency in practice, by describing an implementation on wireless sensor nodes. Before this, we prove that it is not possible to further reduce the complexity of the signer in this cooperative protocol while remaining secure.

Theorem 5. *The Coop-XSGS protocol is the most efficient cooperative variant of XSGS satisfying the security properties of cooperative group signatures.*

Proof (sketch). First of all, we highlight the fact that the signer must not give any information about his group secret key gsk_i to the intermediary. Otherwise, the adversary will learn enough information to obviously break the cooperative non-frameability.

As a consequence, the signer must perform himself all the computations related to gsk_i, which include the computation of z. Indeed, as $z = z' + \mathsf{gsk}_i$ (z' being computed by the intermediary), it is obvious that z should remain secret. As a result, all the computations involving it should also be performed by the signer, which obviously includes the computation of s_z. Finally, the computation of P'_6 using r_z must be performed by the signer, since the intermediary necessary knows the value c (and $s_z = r_z + c \cdot z$, see [Sch89,CCT07,CKY09]). As a conclusion, if one of the remaining computations of the signer is dele-

gated to the intermediary, the cooperative scheme is no longer secure, which concludes the proof. □

6.5.3 Efficiency gain

Let us now examine the efficiency gain of Coop-XSGS with respect to the original protocol.

Comparing the efficiency of protocols in a theoretical way is in general not straightforward. Particularly, it is not as simple as comparing the numbers of point multiplications and other operations like pairings in each protocol. Indeed, theses numbers may not reflect the way of computing these operations. For instance, point multiplications can be considerably sped up when the base point is fixed, or if they can be combined in multiple point multiplications, as detailed in [MÖ1]. Moreover, for implementation reasons, there may be significant differences between the expected and achieved performances.

The most natural way to obtain the efficiency gain is to measure the performances of both protocols implementations. However, as we did not implement the original XSGS protocol on sensor nodes, we rely here on an imperfect assessment to get a first-order estimate of the efficiency gain of the cooperative variant.

The computational complexity of Coop-XSGS is simple: there is a single (fixed-base) point multiplication to compute, neglecting the modular arithmetic. For the original XSGS (its best variant relying on XDH), an analysis of the sign procedure in [DP06] indicates that 5 out of 11 point multiplications can in fact be combined in multiple point multiplications (1 double and 1 triple). The protocol requires thus the calculation of:

- 6 fixed-base point multiplications
- 1 double fixed-base point multiplication
- 1 triple variable-base point multiplication
- 1 pairing evaluation.

We aggregate these elements into a single complexity measure in the following way. We first observe that variable-base point multiplications are about twice slower than fixed-based ones (e.g., [LMK$^+$09]). We then assess the complexity of multiple point multiplications with respect to a simple one. Based on [MÖ1], the ratio for a double point multiplication is about 1.2 while it is 1.3 for a triple point multiplication. Finally, we estimate the complexity ratio

between a pairing and a fixed-base point multiplication to (very) roughly 4.5, based on the implementation results of [AOLD10] and [OSLD08]. All in all, the computational load for the original XSGS protocol is approximately equivalent to: $6 + 1.2 + 2 \cdot 1.3 + 4.5 \approx 14.3$ fixed-base point multiplications. This also gives the computational ratio between the original protocol and its cooperative variant. Again, this ratio is very approximate and it has to be confirmed by a practical comparison (we leave this task to a future work). Nevertheless, it already underlines the large benefit brought by the support from the intermediary.

6.6 Implementation of Coop-XSGS in Sensor Nodes

In this section, we study the performances in practice of the Coop-XSGS protocol. We begin by presenting our testing setup. Then, we describe our choice for the pairing parameters. Afterwards, we give the results of our implementations in wireless sensor nodes. Finally, we explore the tradeoff between computation and memory that can be achieved when storing precomputed point multiplications, or coupons.

6.6.1 Testing Setup

To assess the suitability of the cooperative XSGS protocol for small portable devices, we have implemented it using a wireless sensor node for the signer and a laptop for the powerful helping entity. Our aim is to mimic an RFID scenario where a tag or a contactless smart card interacts with a reader to produce the group signature. In this work, we use wireless sensor nodes for the signing entity as they are the platforms at our disposal and have roughly the same computing power as high-end RFID tags.

The two sensor nodes we consider here are the MICAz [Cro10a] and TelosB [Cro10b]. Their design features were already given in Section 4.2.1, but we recall that the MICAz is based on an 8-bit 7.37MHz ATmega128L microprocessor and the TelosB is equipped with a 16-bit 4MHz MSP430 processor.

These devices are conceptually quite close to contactless smart cards. For instance, the TI RF360 chip for contactless secure government electronic ID embeds the same MSP430 processor as the TelosB node [Tex10]. Therefore, although we consider an active device, the results of our implementations can easily be extended to platforms such as contactless smart cards.

The testing setup is shown in Figure 6.3. The device acting as the intermediary is a 64-bit 1.4 GHz Intel Core 2 Duo based laptop. It is linked with a

sensor node acting as the base station, or the interface with the signer's node.

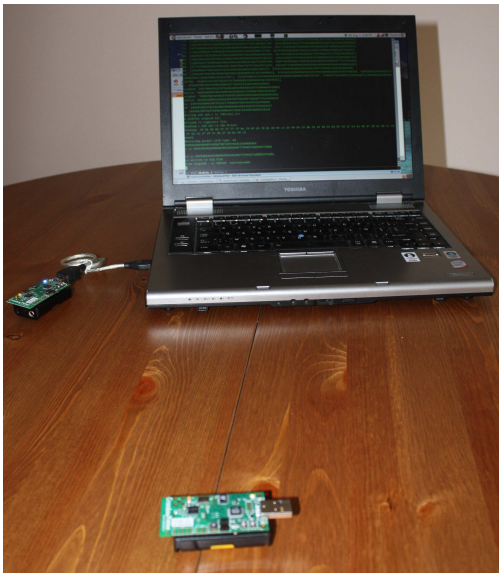


Figure 6.3: Testing setup of our implementation of the Coop-XSGS protocol. The signer sensor node interacts with the laptop (the intermediary) through a base station to produce the signature.

6.6.2 Pairing Parameters

Pairings involve fairly complex mathematics, as taught by Canetti and Rivest [Adi04]. Therefore, designers of cryptographic protocols and implementors would like to employ them as a black box. This is unfortunately not possible for the following reasons. First, the selected pairings may not satisfy all the assumptions on which the cryptographic scheme relies [GPS08]. Second, care must also be present when choosing the elliptic curves since not all the curves are pairing-friendly. In particular, many of the standard recommended curves [Qu99] cannot be used for the computation of pairings. And third, to reach an adequate security level for the pairing, the size of the field over which the curve is defined must be usually larger than in ECC.

There hopefully exist a certain number of recommendations concerning the use of pairings in cryptography (e.g., [GPS08] and [Lyn07]), which allow us to deal with them somehow abstractly. Especially, our choice for the pairing parameters was guided by the latter of these works, to which we refer the

interested reader for more information on the topic.

The signer computation mainly involves the elliptic curve point multiplication $P'_6 = r_z.\mathsf{Rpk}_1$. To reduce its complexity, we selected an asymmetric pairing type, as it allows to use a minimal field size on the signer side. Small-length inputs also reduce the storage and bandwidth costs for the weak signing entity.

For the elliptic curves defining the groups on which the pairing is applied, we selected the type D curves (following the classification of [Lyn07]), i.e., the ordinary curves with embedding degree 3, 4 or 6 known as MNT curves [MNT01]. This type of curve ensures an input length around 160-170 bits for an embedding degree 6 together with an efficient pairing computation [Lyn07]. The parameters of the used curve were taken from the Pairing-Based Cryptography (PBC) library [Lyn10]. They are labeled as the *d159* parameters, where 159 is the size of the curve base field. They ensure a security level for the pairing equivalent to the hardness of the discrete log problem on $6 \cdot 159 = 954$ bits. Parameters for a higher security level could be chosen if required.

6.6.3 Results

The protocol implemented follows the cooperative sign procedure described in Figure 6.2. The prover computations were coded in TinyOS [Tin10], a dialect of C, on both the MICAz and TelosB sensors. For the point multiplication giving P'_6, we relied on the freely available TinyECC library [LN08], which provides a reasonably efficient ECC implementation for sensor nodes. We had to extend it to support the MNT curves. We also implemented a fixed-base comb method for the point multiplication (Algorithm 3.44 in [HMV04]) in order to speed up the computations (at the cost of the storage of a 15 precomputed points in RAM).

On the intermediary side, the computations were implemented in C using the GMP library [GMP10] and the PBC library to achieve the pairings. They took place on a laptop equipped with a 64-bit 1.4 GHz Intel Core 2 Duo. The pairing computations were rather fast in this setting: 11 ms were sufficient to perform a Tate pairing.

The verify and open procedures of the protocol also take place on a powerful machine. Considering a similar computer as for the intermediary, these procedures can be achieved in less than 100 ms, as indicated in the lower part of Table 6.1.

Phase	Detail	Time (ms)
Off-line Sign		
	Prover (MICAz)	3800
	Prover (TelosB)	6400
	Intermediary	195
On-line Sign		
	Prover (MICAz)	7
	Prover (TelosB)	9
	Intermediary	65
	Communication	120
	Total	$<$ **200**
Verify		55
Open		35

Table 6.1: Running times of the various phases of the cooperative XSGS protocol. The on-line phase of the sign procedure takes less than 200 ms when either the MICAz or the TelosB is the prover device.

The Off-line phase. The off-line phase of the sign procedure refers to the calculations preceding the sending of the first message. For the signer, it represents the computation of the point multiplication P'_6. This operation can indeed be done prior to the interactions with the intermediary, either during idle time in case of an active device or precomputed and preloaded on the signer device (i.e., in coupon mode, as detailed below). Of course, P'_6 might still be computed close to the intermediary, i.e., in the on-line phase, e.g., in case of a passive RFID tag which does not employ coupons.

Table 6.1 gives the running times for the various parts of the cooperative XSGS protocol. It shows that the off-line phase lasts about 4 and 6 seconds on the MICAz and TelosB respectively. While already practical, these durations are expected to be significantly reduced with a more optimized implementation. While reasonably efficient, the TinyECC library, on which our implementation is based, is however significantly slower than recently proposed ECC implementations on sensor nodes. We therefore expect the point multiplication to be significantly faster using the same techniques as for instance the implementation proposed in [LMK$^+$09], which performs a fixed-base point multiplication in less than a second on a 192-bit elliptic curve group.

As a conclusion, even if the prover device has to compute the point multiplication P'_6 during the on-line phase, the whole procedure can be done within a few seconds (much less if a dedicated hardware ECC engine is available).

Mode	Memory	MICAz	TelosB
Coupon	RAM	1067 (26%)	1071 (10%)
	ROM	23764 (18%)	14470 (29%)
No Coupon	RAM	1699 (41%)	1739 (17%)
	ROM	36576 (28%)	19164 (39%)

Table 6.2: Storage requirements (in bytes) of our implementation. In coupon mode, 21 B must be added per stored coupon.

The On-line phase. The on-line phase of the protocol starts with the transmission of the protocol first message and ends when the signature is generated. It is performed in less than 200 ms with both sensor nodes as the signer. The communication delay and the intermediary computations are the main components of the on-line phase latency. The time required by the prover computations, i.e., the calculation of s_z, is marginal ($<$10 ms). The fast on-line phase of the cooperative protocol makes group signatures of practical interest for small devices like contactless smart cards.

Memory. The memory usage of the cooperative protocol is relatively modest (Table 6.2), even when the ECC point multiplication is performed on the node. Note that the given figures include the memory costs of the tiny operating system. This one already consumes a significant fraction of the used memory (about 700 B RAM and a little more than 10 kB ROM on both nodes).

6.6.4 Coupon mode

The point multiplication P_6' can be computed prior to the interactions between the signer and the intermediary. It can even be precomputed on a powerful machine and loaded on the signer device. This case corresponds to the coupon mode [NMVR94], where a *coupon* is a pair of (r_z, $r_z.\mathsf{Rpk}_1$) stored in the constrained device.

Using coupons, computations can be traded for memory occupation. Besides the latency savings (and energy for active devices), the coupon mode allows to avoid devoting resources to the implementation of the ECC point multiplication in the signer device. On the other hand, a sufficient amount of memory must be available in the signer device, since coupons are foreseen for a single use only.

A coupon (r_z, $r_z.\mathsf{Rpk}_1$) requires normally 60 B, i.e., three 20-B field elements, but it can be reduced to a little more than 20 B using point compression and a PRNG sequence for storing all the r_z, as done in [GJR07].

The storage requirement of our implementation in coupon mode is also given in Table 6.2. The RAM occupation includes a single coupon. Per additional coupon, 21 B must be accounted. As the coupons can be placed in RAM or ROM, their storage in both the MICAz and TelosB is not a problem. Considering the available memory resources, the MICAz and the TelosB could be filled with more than 5000 and 2500 coupons respectively, which is more than practical for many applications. For instance, in an access control application consuming a few coupons everyday, these amounts of coupons can ensure an autonomous protocol use for more than a year.

When the stock of coupons is depleted, a refill procedure can be foreseen. The computation of coupons involves only public parameters and it could be done in any powerful machine at the user disposal. We however underline that the refill procedure must be done in a fully secure way. Indeed, if an adversary manages to learn one of them, she becomes able to recover the user group secret key gsk_i and impersonate the signer (since r_z protects gsk_i).

6.7 Strong Cooperative Variant of XSGS

For the sake of completeness, we end this chapter by describing a strong anonymous cooperative variant of the XSGS scheme. As the signer must now remain anonymous w.r.t the intermediary, the certificate (A_i, x_i) can no longer be transmitted to the intermediary. It is thus obvious that this strong version is less efficient as most of the computations done by the intermediary are related to the user certificate. However, it is still possible for the intermediary to help the signer by computing one pairing without compromising the security. The strong anonymous cooperative variant for the XSGS scheme is presented in Figure 6.4.

It is quite obvious that the correctness of this scheme is ensured. Furthermore, as the intermediary can obtain less information during this protocol than in the previous version (see Section 6.5), it is also trivial that this strong variant ensures the cooperative traceability and the cooperative non-frameability properties.

From the anonymity point of view, the only information that an adversary can learn if she impersonates the intermediary is T_3. As this element belongs to the outputted signature, and based on the anonymity of the "standard" XSGS protocol, the cooperative variant for the XSGS presented in Figure 6.4 ensures the strong anonymity property.

Last but not least, we provide some intuition to show that it is not possible to define a more efficient version of this protocol while preserving all security

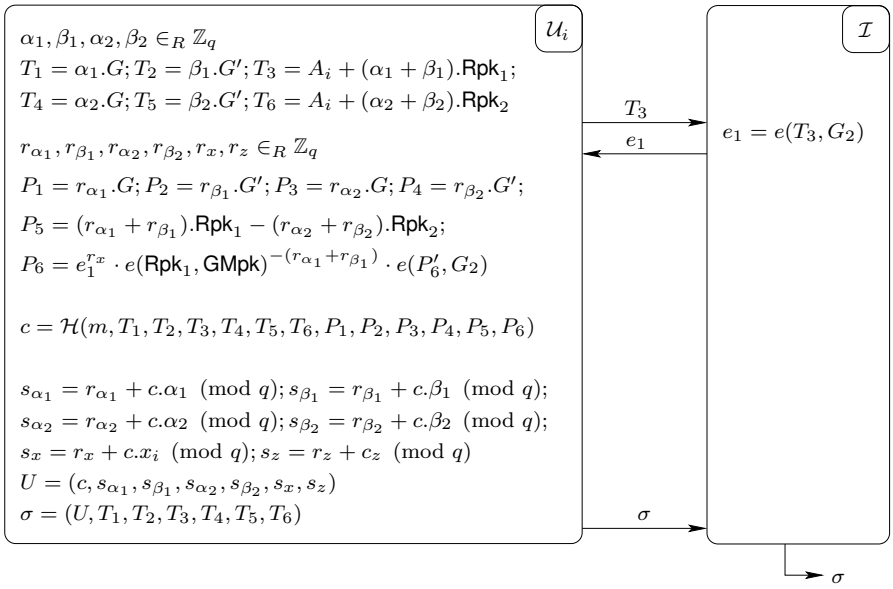


Figure 6.4: Sign procedure of the strong cooperative XSGS Protocol.

properties. Again, if the signer gives some information about gsk_i, the non-frameability will no longer be verified. The computation of the encryption implies the knowledge of the certificate and thus must be performed by the signer[6]. Finally, the proof of knowledge has to be done by the signer as all the values must be kept secret to preserve his anonymity [Sch89,CCT07,CKY09].

As a conclusion, we remark that this strong version of XSGS is more efficient than the standard XSGS as the intermediary plays a part by computing the pairing $e(T_3, G_2)$. However, as the signer has to perform the numerous remaining point multiplications, this scheme does not suit the constrained resources of a small embedded device.

[6]This is true for the whole encryption values. In fact, one may think that it is possible to give to the intermediary the knowledge of e.g., α without compromising the strong anonymity. This is not true since, in this case, we obtain, from the intermediary point of view, a kind of El Gamal encryption of A but, as we do not consider the XDH assumption (see Section 6.4.3), the DDH problem is not hard and El Gamal is consequently not secure.

Bibliography

[ACdM05] Giuseppe Ateniese, Jan Camenisch, and Breno de Medeiros. Untraceable RFID Tags via Insubvertible Encryption. In Vijay Atluri, Catherine Meadows, and Ari Juels, editors, *Conference on Computer and Communications Security – ACM CCS'05*, pages 92–101, Alexandria, Virginia, USA, November 2005. ACM, ACM Press.

[ACJT00] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In M. Bellare, editor, *CRYPTO'00*, volume 1880 of *LNCS*, pages 255–270. Springer, 2000.

[ACSW10] Frederik Armknecht, Liqun Chen, Ahmad-Reza Sadeghi, and Christian Wachsmann. Anonymous Authentication for RFID Systems. In *Workshop on RFID Security – RFIDSec'10*, Istanbul, Turkey, June 2010.

[Adi04] Ben Adida. Pairing-Based Cryptography. Lecture 25 of the course "Special Topics in Cryptography" by Ran Canetti and Ron Rivest, available online at `http://courses.csail.mit.edu/6.897/spring04/L25.pdf`, May 2004.

[AOLD10] D. F. Aranha, L. B. Oliveira, J. López, and R. Dahab. Efficient implementation of elliptic curve cryptography in wireless sensors. *Advances in Mathematics of Communications*, 4(2):169–187, 2010.

[BB08] Dan Boneh and Xavier Boyen. Short Signatures Without Random Oracles and the SDH Assumption in Bilinear Groups. *J. Cryptology*, 21(2):149–177, 2008.

[BBS04] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In M. K. Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, 2004.

[BMW03] M. Bellare, D. Micciancio, and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 614–629. Springer, 2003.

[BQ95] P. Béguin and J.-J. Quisquater. Fast Server-Aided RSA Signatures Secure Against Active Attacks. In *CRYPTO '95*, volume 963 of *LNCS*, pages 57–69. Springer, 1995.

[BSZ05] M. Bellare, H. Shi, and C. Zhang. Foundations of group signatures: The case of dynamic groups. In *CT-RSA 2005*, volume 3376 of *LNCS*, pages 136–153. Springer, 2005.

[BW07] X. Boyen and B. Waters. Full-domain subgroup hiding and constant-size group signatures. In *PKC 2007*, volume 4450 of *LNCS*, pages 1–15. Springer, 2007.

[CCT07] S. Canard, I. Coisel, and J. Traoré. Complex zero-knowledge proofs of knowledge are easy to use. In *ProvSec*, volume 4784 of *LNCS*, pages 122–137. Springer, 2007.

[CdMCP10] Iwen Coisel, Giacomo de Meulenaer, Sébastien Canard, and Olivier Pereira. Group Signatures are Suitable for Constrained Devices. To appear in the 13th Annual International Conference on Information Security and Cryptology (ICISC 2010), 2010.

[CG02] S. Canard and M. Girault. Implementing group signature schemes with smart cards. In *CARDIS '02*, pages 1–10. USENIX, 2002.

[CG04] Jan Camenisch and Jens Groth. Group signatures: Better efficiency and new theoretical aspects. In *SCN 2004*, volume 3352 of *Lecture Notes in Computer Science*, pages 120–133. Springer, 2004.

[CKY09] Jan Camenisch, Aggelos Kiayias, and Moti Yung. On the Portability of Generalized Schnorr Proofs. In *EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 425–442. Springer, 2009.

[CP92] David Chaum and Torben P. Pedersen. Wallet databases with observers. In *CRYPTO 92*, volume 740 of *Lecture Notes in Computer Science*, pages 89–105. Springer, 1992.

[CP93] Ronald Cramer and Torben P. Pedersen. Improved privacy in wallets with observers (extended abstract). In *EUROCRYPT 93*, pages 329–343, 1993.

[Cro10a] CrossBow. MICAz low-power wireless sensor module. Datasheet available at `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAz_Datasheet.pdf`, April 2010.

[Cro10b] CrossBow. TelosB low-power wireless sensor module. Datasheet available at `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/TELOSB_Datasheet.pdf`, April 2010.

[CT03] Sébastien Canard and Jacques Traoré. On fair e-cash systems based on group signature schemes. In *ACISP'03: Proceedings of the 8th Australasian conference on Information security and privacy*, pages 237–248, Berlin, Heidelberg, 2003. Springer-Verlag.

[CvH91] D. Chaum and E. van Heyst. Group signatures. In *EUROCRYPT*, pages 257–265, 1991.

[DP06] Cécile Delerablée and David Pointcheval. Dynamic fully anonymous short group signatures. In *VIETCRYPT 2006*, volume 4341 of *Lecture Notes in Computer Science*, pages 193–210. Springer, 2006.

[FFS88] U. Feige, A. Fiat, and A. Shamir. Zero-knowledge proofs of identity. *J. Cryptology*, 1(2):77–94, 1988.

[FI05] Jun Furukawa and Hideki Imai. An efficient group signature scheme from bilinear maps. In Colin Boyd and Juan Manuel González Nieto, editors, *ACISP*, volume 3574 of *Lecture Notes in Computer Science*, pages 455–467. Springer, 2005.

[FS86] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. M. Odlyzko, editor, *CRYPTO*, volume 263 of *LNCS*, pages 186–194. Springer, 1986.

[FSW04] M. Feldhofer, S.Dominikus, and J. Wolkerstorfer. Strong Authentication for RFID Systems Using the AES Algorithm. In *CHES 2004*, volume 3156 of *LNCS*, pages 357–370. Springer, 2004.

[Gam85] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[GJR07] Marc Girault, Loic Juniot, and Matt Robshaw. The Feasibility of On-the-Tag Public Key Cryptography. In *Workshop on RFID Security – RFIDSec'07*, Malaga, Spain, July 2007.

[GL04] M. Girault and D. Lefranc. Public key authentication with one (online) single addition. In *CHES 2004*, volume 3156 of *LNCS*, pages 413–427. Springer, 2004.

[GL05] M. Girault and D. Lefranc. Server-aided verification: Theory and practice. In *ASIACRYPT 2005*, volume 3788 of *LNCS*, pages 605–623. Springer, 2005.

[GMP10] GMP. The GNU Multiple Precision Arithmetic Library. Available online at `http://gmplib.org/`, April 2010.

[GPS08] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discrete Appl. Math.*, 156(16):3113–3121, 2008.

[Gro07] J. Groth. Fully anonymous group signatures without random oracles. In *ASIACRYPT 2007*, pages 164–180, 2007.

[GS08] J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In *EUROCRYPT'08*, LNCS. Springer, 2008.

[HMF07] Michael Hutter, Stefan Mangard, and Martin Feldhofer. Power and EM Attacks on Passive 13.56 MHz RFID Devices. In *CHES '07: Proceedings of the 9th international workshop on Cryptographic Hardware and Embedded Systems*, pages 320–333, Berlin, Heidelberg, 2007. Springer-Verlag.

[HMV04] D. Hankerson, A. Menezes, and S. Vanstone. *Guide to Elliptic Curve Cryptography*. Springer, 2004.

[Joy09] Marc Joye. On cryptographic schemes based on discrete logarithms and factoring. In *CANS 2009*, volume 5888 of *Lecture Notes in Computer Science*, pages 41–52. Springer, 2009.

[LMK+09] Christian Lederer, Roland Mader, Manuel Koschuch, Johann Großschdl, Alexander Szekely, and Stefan Tillich. Energy-Efficient Implementation of ECDH Key Exchange for Wireless Sensor Networks. In *Information Security Theory and Practices — WISTP 2009*, pages 112–127. Springer Verlag, LNCS 5746, September 2009.

[LN08] A. Liu and P. Ning. TinyECC: A configurable library for elliptic curve cryptography in wireless sensor networks. In *IPSN*, pages 245–256, April 2008.

[Lyn07] B. Lynn. *On the implementation of pairing-based cryptosystems.* PhD thesis, Stanford University, 2007.

[Lyn10] B. Lynn. PBC, the Pairing-Based Cryptography Library. Available online at `http://crypto.stanford.edu/pbc/`, April 2010.

[MÖ1] Bodo Möller. Algorithms for multi-exponentiation. In *SAC '01: Revised Papers from the 8th Annual International Workshop on Selected Areas in Cryptography*, pages 165–180, London, UK, 2001. Springer-Verlag.

[MB02] G. Maitland and C. Boyd. Co-operatively formed group signatures. In *CT-RSA 2002*, volume 2271 of *LNCS*, pages 218–235. Springer, 2002.

[MKI88] T. Matsumoto, K. Kato, and H. Imai. Speeding up secret computations with insecure auxiliary devices. In *CRYPTO '88*, volume 403 of *LNCS*, pages 497–506. Springer, 1988.

[MNT01] Atsuko Miyaji, Masaki Nakabayashi, and Shunzou TAKANO. New explicit conditions of elliptic curve traces for FR-reduction. IEICE Transactions on Fundamentals, E84-A(5):1234-1243, 2001.

[NMVR94] David Naccache, David M'Raïhi, Serge Vaudenay, and Dan Raphaeli. Can D.S.A. be Improved? Complexity Trade-Offs

with the Digital Signature Standard. In *EUROCRYPT 1994*, volume 950 of *Lecture Notes in Computer Science*, pages 77–85. Springer, 1994.

[NY89] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *STOC*, pages 33–43. ACM, 1989.

[OSLD08] L.B. Oliveira, M. Scott, J. Lopez, and R. Dahab. Tinypbc: Pairings for authenticated identity-based non-interactive key distribution in sensor networks. In *Networked Sensing Systems, 2008. INSS 2008. 5th International Conference on*, pages 173–180, June 2008.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT'99*, Lecture Notes in Computer Science, pages 223–238. Springer, 1999.

[Qu99] Minghua Qu. Standards for efficient cryptography SEC 2: Recommended Elliptic Curve Domain Parameters. Available online at `http://www.secg.org/collateral/sec2.pdf`, 1999.

[Sch89] C.-P. Schnorr. Efficient identification and signatures for smart cards. In G. Brassard, editor, *CRYPTO*, volume 435 of *LNCS*, pages 239–252. Springer, 1989.

[SRS04] Junichiro Saito, Jae-Cheol Ryou, and Kouichi Sakurai. Enhancing Privacy of Universal Re-encryption Scheme for RFID Tags. In Laurence Jang, Minyi Guo, Guang Gao, and Niraj Jha, editors, *Embedded and Ubiquitous Computing – EUC'04*, volume 3207 of *Lecture Notes in Computer Science*, pages 879–890, Aizu-Wakamatsu City, Japan, August 2004. Springer.

[SVW09a] Ahmad-Reza Sadeghi, Ivan Visconti, and Christian Wachsmann. Anonymizer-Enabled Security and Privacy for RFID. In *8th International Conference on Cryptology And Network Security – CANS'09*, Kanazawa, Ishikawa, Japan, December 2009. Springer.

[SVW09b] Ahmad-Reza Sadeghi, Ivan Visconti, and Christian Wachsmann. Efficient RFID Security and Privacy with Anonymizers. In *Work-*

shop on RFID Security – RFIDSec'09, Leuven, Belgium, July 2009.

[Tex10] Texas Instruments Inc. Texas Instruments Government Electronic Identification. Brochure available at `http://www.ti.com/rfid/docs/manuals/brochures/govid_trifold.pdf`, April 2010.

[Tin10] TinyOS. An open-source operating system designed for wireless embedded sensor networks . `http://www.tinyos.net/`, April 2010.

[XY04] S. Xu and M. Yung. Accountable ring signatures: A smart card approach. In *CARDIS'04*, pages 271–286. Kluwer, 2004.

Conclusion

Throughout this thesis, we dealt with problems related to the concept of power. We hope to have clarified our deliberately equivocal thesis title, referring to both the computational power and the electrical power.

In the first part, dedicated to cryptanalysis with specialized hardware, we began by studying how to improve ECM in reconfigurable hardware as a support for the factorization of a 1024-bit RSA modulus with the Number Field Sieve. The architecture we proposed featured a 15-fold improvement in cost-performance ratio with respect to previous works. More than three years after the publication of our results, our approach remains (to the author's knowledge) the best solution in this context. The amelioration was obtained by exploiting the power of embedded multipliers of modern FPGAs in a fully unrolled and pipelined architecture, achieving a throughput of one 135-bit modular multiplication per clock cycle.

We then investigated how to solve an ECDLP instance with ASICs. We considered the anomalous Koblitz curves, of special interest in cryptography because the Frobenius map can be employed to speed-up the computations. However, this "advantage" is also exploitable in cryptanalytic attacks: these curves simply offer less security per key bit. For the design of the arithmetic circuits, we weighed the options of the polynomial and normal-basis representations. As opposed to what was assumed in previous works, we concluded that the normal basis could lead to more efficient designs. Most importantly, our estimates of the cost of an attack breaking the ECC2K-130 challenge showed the insecurity of the corresponding key size. They contributed, together with the real distributed attack launched by the ECRYPT VAMPIRE working group,[7] to change the initial Certicom statement[8] from "The 131-bit Level I challenge is expected to be infeasible against realistic software and hardware attacks ..." to " The 131-bit Level I challenges will be require significantly more work,

[7] See `http://www.ecc-challenge.info/`

[8] See `http://www.certicom.com/images/pdfs/challenge-2009.pdf`

but may be within reach". Although this key size was not recommended, it was tempting to select it for an ECC implementation in a very constrained environment, typically for small embedded devices.

The second part covered cryptography for low-power devices. Our first work within this broad topic attempted to answer the following question: to which extent is it interesting for a small autonomous device to offload computations to another entity? This led us to investigate the relative energy costs of communication and computation in wireless sensor networks, and more generally the real cost of cryptographic protocols. As opposed to the few previous works, we highlighted the importance that communication could take, especially when the nodes needed to stay in listen mode. We also proposed a methodology to assess and compare the energy consumption of protocols, allowing the selection of the less consuming alternative to implement a given cryptographic task.

We finally moved to the study of how a powerful entity could help a very limited device in the production of group signatures, while limiting the trust placed in the helping party. We presented a cooperative version of XSGS, one of the most efficient group signature scheme which also features very interesting security properties. Our implementations on wireless sensor nodes proved its suitability for small embedded devices. The proposed protocol makes it possible to store the signer's secrets into a well-protected device like a smart card to enhance the security of group signatures; it also represents an interesting direction in the development of low-power privacy-preserving applications.

Besides the contributions described above, our works raises new questions and suggests ideas for future research.

We first list several relatively straightforward ideas for future works. For ECM, it would be interesting to extend our proof of concept to the implementation of Phase 2. In the same direction, an ECM implementation on an FPGA-based engine like COPACOBANA or on ASIC would also be very valuable. Concerning our work in progress about ECDLP for Koblitz curves, other research groups are currently exploring the path we initiated. Their suggested approach of a design based on a hybrid multiplier allows gathering the advantages of polynomial and normal bases. A further study on the topic should confirm the superiority of this option. Concerning the second part of the thesis, our work about the energy cost of protocols in wireless sensor networks could be completed by exploring other approaches for the Low Power Listening (LPL) to minimize the energy consumption, like synchronous LPL protocols. Also, cooperative versions of existing protocols could be studied. A

good starting point could be the very short group signature scheme recently proposed at SCN 2010 [BCN+10].

More generally, we demonstrate in this thesis the need of fully understanding the resources offered by the technology when designing hardware architectures. This approach provided surprisingly high improvements, in both cases of ECM and ECDLP, with respect to reasonably optimized implementations. It is thus worth the effort. Working with the device resources in mind is not only useful for implementations, it can also help to design efficient cryptographic primitives (as reported in Chapter 4) and protocols (cf. Chapter 6).

One goal of this thesis was to improve our knowledge concerning the security of cryptographic key lengths against attacks involving special-purpose hardware. There remains a lot of work in this area. It would be desirable to provide estimates for a wider range of key lengths. For instance, applications requiring a strong security for a small period of time would benefit from such estimates. Considering the relatively long development time of cryptanalytic hardware implementations, it would be very attractive to propose a methodology to provide sound security estimates without fully implementing the attacks. Partial implementations (like focusing on the arithmetic circuits only) or extrapolations based on existing implementations could be a solution. The methodology to employ should be discussed by the research community.

Another reflexion concerns the protocol level. In the context of lightweight cryptography, we observe that many highly optimized implementations are proposed. However, we feel that the protocol level deserves more consideration to provide robust cryptography to very constrained devices, considering the benefits brought by our contributions in Chapters 5 and 6.

Finally, as shown with Coop-XSGS, relaxing assumptions in specific application scenarios can be very beneficial to circumvent the technical challenges faced in contexts like cryptography for small embedded devices.

Bibliography

[BCN+10] Patrik Bichsel, Jan Camenisch, Gregory Neven, Nigel Smart, and Bogdan Warinschi. Get shorty via group signatures without encryption. In *Security and Cryptography for Networks - SCN 2010*, pages 381–398. Springer LNCS 6280, September 2010.

Publication list

- ✎ Sébastien Canard, Iwen Coisel, Giacomo de Meulenaer and Olivier Pereira, *Group Signatures are Suitable for Constrained Devices*, 13th Annual International Conference on Information Security and Cryptology (ICISC 2010), Lecture Notes in Computer Science, Springer, December 2010.

- ✎ Giacomo de Meulenaer and François-Xavier Standaert, *Stealthy Compromise of Wireless Sensor Nodes with Power Analysis Attacks*, 2nd International Conference on Mobile Lightweight Wireless Systems (MOBILIGHT 2010), Lecture Notes of ICST, Springer, May 2010.

- ✎ Nidal Aboudagga, Giacomo de Meulenaer, Mohamed Eltoweissy and Jean-Jacques Quisquater, *IMAPS: Imbricated Authentication Protocol Suite for Mobile Users and Groups*, IEEE Conference on Local Computer Networks (LCN 2009), IEEE Computer Society, October 2009.

- ✎ D. Bailey, B. Baldwin, L. Batina, D. Bernstein, P. Birkner, J. Bos, G van Damme, G. de Meulenaer, J. Fan, F. Gurkaynak, T. Gneysu, T. Kleinjung, T. Lange, N. Mentens, C. Paar, F. Regazzoni, P. Schwabe and L. Uhsadel, *The Certicom Challenges ECC2-X*, Workshop on Special Purpose Hardware for Attacking Cryptographic Systems (SHARCS 2009), September 2009.

- ✎ Giacomo de Meulenaer, Christophe Petit and Jean-Jacques Quisquater, *Hardware Implementations of a Variant of the Zémor-Tillich Hash Function: Can a Provably Secure Hash Function be very efficient ?*, IACR ePrint archive, Report 2009/229, May 2009.

✎ François-Xavier Standaert, Philippe Bulens, Giacomo de Meulenaer and Nicolas Veyrat-Charvillon, *Improving the Rules of the DPA Contest*, IACR eprint archive, Report 2008/517, December 2008.

✎ Giacomo de Meulenaer, Franois Gosset, François-Xavier Standaert and Jean-Jacques Quisquater, *On the Energy Cost of Communication and Cryptography in Wireless Sensor Networks*, (extended version), IEEE International Workshop on Security and Privacy in Wireless and Mobile Computing, Networking and Communications (SecPriWiMob 2008), October 2008.

✎ Giacomo de Meulenaer, Franois Gosset, François-Xavier Standaert, and Luc Vandendorpe, *On the Energy Cost of Communication and Cryptography in Wireless Sensor Networks*, Twenty-ninth Symposium on Information Theory in the Benelux, May 2008.

✎ Giacomo de Meulenaer, Franois Gosset, Guerric Meurice de Dormale and Jean-Jacques Quisquater, *Elliptic Curve Factorization Method : Towards Better Exploitation of Reconfigurable Hardware*, Workshop on Special Purpose Hardware for Attacking Cryptographic Systems (SHARCS 2007), September 2007.

✎ Giacomo de Meulenaer, Franois Gosset, Guerric Meurice de Dormale and Jean-Jacques Quisquater, *Integer Factorization Based on Elliptic Curve Method: Towards Better Exploitation of Reconfigurable Hardware*, IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM 2007), IEEE Computer Society Press, April 2007.