Efficient Cryptographic Algorithms and Protocols for Mobile Ad Hoc Networks

by

Xinxin Fan

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2010

Abstract

As the next evolutionary step in digital communication systems, mobile ad hoc networks (MANETs) and their specialization like wireless sensor networks (WSNs) have been attracting much interest in both research and industry communities. In MANETs, network nodes can come together and form a network without depending on any pre-existing infrastructure and human intervention. Unfortunately, the salient characteristics of MANETs, in particular the absence of infrastructure and the constrained resources of mobile devices, present enormous challenges when designing security mechanisms in this environment. Without necessary measures, wireless communications are easy to be intercepted and activities of users can be easily traced. This thesis presents our solutions for two important aspects of securing MANETs, namely efficient key management protocols and fast implementations of cryptographic primitives on constrained devices.

Due to the tight cost and constrained resources of high-volume mobile devices used in MANETs, it is desirable to employ lightweight and specialized cryptographic primitives for many security applications. Motivated by the design of the well-known Enigma machine, we present a novel ultra-lightweight cryptographic algorithm, referred to as Hummingbird, for resource-constrained devices. Hummingbird can provide the designed security with small block size and is resistant to the most common attacks such as linear and differential cryptanalysis. Furthermore, we also present efficient software implementations of Hummingbird on 4-, 8- and 16-bit microcontrollers from Atmel and Texas Instruments as well as efficient hardware implementations on the low-cost field programmable gate arrays (FPGAs) from Xilinx, respectively. Our experimental results show that after a system initialization phase Hummingbird can achieve up to 147 and 4.7 times faster throughput for a size-optimized and a speed-optimized software implementation, respectively, when compared to the state-of-the-art ultra-lightweight block cipher PRESENT on the similar platforms. In addition, the speed optimized Hummingbird encryption core can achieve a throughput of 160.4 Mbps and the area optimized encryption core only occupies 253 slices on a Spartan-3 XC3S200 FPGA device.

Bilinear pairings on the Jacobians of (hyper-)elliptic curves have received considerable attention as a building block for constructing cryptographic schemes in MANETs with new and novel properties. Motivated by the work of Scott, we investigate how to use efficiently computable automorphisms to speed up pairing computations on two families of non-supersingular genus 2 hyperelliptic curves over prime fields. Our findings lead to new variants of Miller's algorithm in which the length of the main loop can be up to 4 times shorter than that of the original Miller's algorithm in the best case. We also generalize Chatterjee *et al.*'s idea of encapsulating the computation of the line function with the group operations to genus 2 hyperelliptic curves, and derive new explicit formulae for the group operations in projective and new coordinates in the context of pairing computations.

Efficient software implementation of computing the Tate pairing on both a supersingular and a non-supersingular genus 2 curve with the same embedding degree of $k = 4$ is investigated. Combining the new algorithm with known optimization techniques, we show that pairing computations on non-supersingular genus 2 curves over prime fields use up to 55.8% fewer field operations and run about 10% faster than supersingular genus 2 curves for the same security level.

As an important part of a key management mechanism, efficient key revocation protocol, which revokes the cryptographic keys of malicious nodes and isolates them from the network, is crucial for the security and robustness of MANETs. We propose a novel self-organized key revocation scheme for MANETs based on the Dirichlet multinomial model and identity-based cryptography. Firmly rooted in statistics, our key revocation scheme provides a theoretically sound basis for nodes analyzing and predicting peers' behavior based on their own observations and other nodes' reports. Considering the difference of malicious behaviors, we proposed to classify the nodes' behavior into three categories, namely good behavior, suspicious behavior and malicious behavior. Each node in the network keeps track of three categories of behavior and updates its knowledge about other nodes' behavior with 3-dimension Dirichlet distribution. Based on its own analysis, each node is able to protect itself from malicious attacks by either revoking the keys of the nodes with malicious behavior or ceasing the communication with the nodes showing suspicious behavior for some time. The attack-resistant properties of the resulting scheme against false accusation attacks launched by independent and collusive adversaries are also analyzed through extensive simulations.

In WSNs, broadcast authentication is a crucial security mechanism that allows a multitude of legitimate users to join in and disseminate messages into the networks in a dynamic and authenticated way. During the past few years, several public-key based multi-user broadcast authentication schemes have been proposed in the literature to achieve immediate authentication and to address the security vulnerability intrinsic to μTESLA-like schemes. Unfortunately, the relatively slow signature verification in signature-based broadcast authentication has also incurred a series of problems such as high energy consumption and long verification delay. We propose an efficient technique to accelerate the signature verification in WSNs through the cooperation among sensor nodes. By allowing some sensor nodes to release the intermediate computation results to their neighbors during the signature verification, a large number of sensor nodes can accelerate their signature verification process significantly. When applying our faster signature verification technique to the broadcast authentication in a 4×4 grid-based WSN, a quantitative performance analysis shows that our scheme needs 15.5% $\sim$ 34.5% less energy and runs about 50% faster than the traditional signature verification method.

Acknowledgements

This thesis describes the research that I conducted during the past four years I worked as a research assistant at the Department of Electrical and Computer Engineering, University of Waterloo. These four years have been a tremendous period of development for me, both academically and personally, and I would like to take this opportunity to express my deep gratitude to many people who have made much of this time unforgettable.

First and foremost, I would like to thank my supervisor, Prof. Guang Gong, for accepting and supporting me as a Ph.D. student and allowing me to take the liberty to follow my research interests. Her advice and expertise in the field resolved many hurdles that I encountered throughout my research. I would like to thank Prof. Gong for her valuable friendship and camaraderie that we developed while working together. I would like to express my appreciation to Prof. Lein Harn at the University of Missouri, Kansas City, for serving as my external examiner and giving me many valuable suggestions and comments. I am also grateful to my thesis committee, Prof. Catherine Gebotys, Prof. Anwar Hasan and Prof. Alfred Menezes at the University of Waterloo, for helping me to improve the quality of my thesis. It is a privilege to have such a great committee.

I would like to thank Prof. David Jao at the University of Waterloo for all the fruitful discussions about elliptic and hyperelliptic curve cryptography when we worked on pairing computation on genus 2 hyperelliptic curves together. Thanks to Prof. Steven Galbraith at the University of Auckland for answering me so many questions about hyperelliptic curve pairings through emails and giving me valuable comments on my research work during the course of attending "Pairing-Based Cryptography" conference. I am also grateful to Prof. Mark Aagaard at the University of Waterloo for kindly sharing his experience with me on FPGA and ASIC design of cryptographic algorithms as well as the use of hardware synthesis and implementation tools.

I would like to thank Dr. Daniel Engels and Mr. Eric Smith from Revere Security Corporation for all the discussions on lightweight cryptographic primitives and their applications in securing RFID systems as well as the exciting research project about Hummingbird cipher we had. Their wealth of experience with applied cryptography is extremely helpful for me to bridge the gap between cryptography theoretical research and practical applications.

I want to express my deep gratitude to my former supervisor Prof. Yumin Wang at the Xidian University. Prof. Wang guided me to the exciting field of Cryptography and his professionalism, enthusiasm and encouragement are always inspiring. This thesis is also dedicated to him, especially on the occasion of his 74th birthday. My acknowledgements also go to Dr. Thomas Wollinger from the escrypt GmbH-Embedded Security for giving me many useful advices at the very beginning of my research and collaborating with me to develop fast group operations for hyperelliptic curve cryptosystems. I miss all those days we work together to write yet another research paper.

I would like to give special thanks to all my (former) colleagues and friends. I thank Dr. Kishan Gupta, Dr. Katrin Hoeper, Dr. Yassir Nawaz and Dr. Nam Yul Yu for their friendship and exchanging ideas. I also thank Dr. Honggang Hu, Dr. Hong Wen, Zhijun Li, Anuchart Tassanaviboon, Qi Chai, Zilong Wang, Yiyuan Luo, and Fei Huo for their support, for their friendship, and for having much fun with them. I want to thank all the members of the Communication Security (ComSec) Lab for always contributing to a good and warm group atmosphere and making ComSec an excellent place for doing research.

Last but not least, I would like to thank my family, my parents, and my friends for being patient with me and for their support. Special thanks go out to my wife Ning Zhang for her unconditional love and understanding during the past four years and especially during these last months writing the thesis.

To all of you thank you very much!

To my wife
To my parents
For their endless love and support!

Contents

List of Tables

List of Figures

Chapter 1

Introduction

With the rapid development in network technology, in particular wireless communications, the traditional centralized, fixed networks cannot satisfy enormous demands on network connectivity, data storage and information exchange any longer. New types of communication networks based on wireless and multi-hop communication have emerged to provide efficient solutions for the growing number of mobile wireless applications and services. A large family of the new types of wireless communication networks can be best represented by *mobile ad hoc networks (MANETs)*. In this paradigm, mobile devices self-organize to create a network by exploiting their wireless network interface, without a requirement for a pre-deployed infrastructure. While rapid technology progress for MANETs and their specialization like *wireless sensor networks (WSNs)* are making possible the era of seamless wireless connections where any mobile device would be able to connect to any other mobile device or network at any time and in any place, the intrinsic vulnerabilities of MANET structure also introduce a wide range of attacks and present new difficulties and challenges for the design of security mechanism. Without adequate security, MANETs should not be widely deployed in either military or commercial areas, and ubiquitous computing will only become pervasive nightmare.

In this thesis, we present our results for providing security solutions for various general-purpose and specialized MANETs, ranging from developing and implementing lightweight cryptographic primitives to designing and analyzing secure protocols. Our research will focus on efficient cryptographic algorithms and key management protocols, two paramount security issues for securing MANETs. This chapter starts with a brief introduction to the basic concept and applications of MANETs in Section 1.1, followed by a description of various types of attacks in Section 1.2. Section 1.3 presents the design goals of security mechanisms in MANETs. In Section 1.4, we describe the related work as well as a couple of research topics on which this thesis will focus. Finally, the outline of this thesis and a summary of our research contributions are given in Section 1.5.

1.1 Mobile Ad Hoc Networks

A MANET is a collection of autonomous nodes or terminals that communicate with each other by forming a multi-hop radio network and maintaining connectivity in a decentralized manner [135]. Due to the limited transmission range of each mobile node, it may be necessary for one mobile node to enlist the aid of other nodes in forwarding a package to its destination. Therefore, in such environment, every node in the network plays the role of a router by being able to determine the paths of transmitting packets to their destinations. Figure 1.1 illustrates an example of a MANET which contains two laptops, two PDAs and two digital cameras. Since node D is outside node A's transmission range, the data from A to D must be retransmitted by nodes B and C.

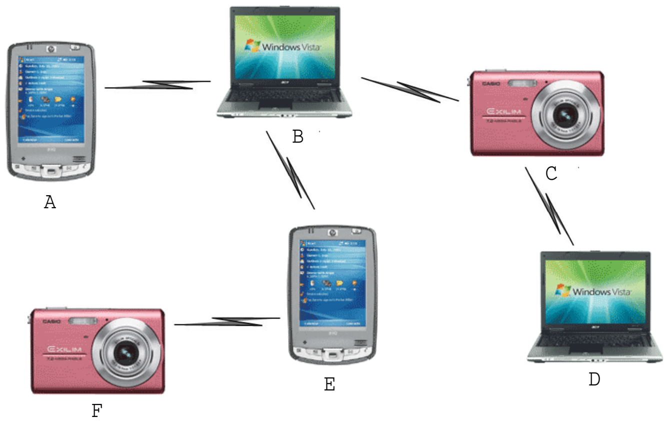


Figure 1.1: A Mobile Ad Hoc Network

MANETs inherit common characteristics found in wireless networks in general, and add characteristics specific to ad hoc networking. The MANETs generally have the following characteristics [140]:

- *Autonomous and infrastructureless*: A MANET does not depend on any established infrastructure or centralized administration. Each node operates in a distributed peer-to-peer mode, acts as an independent router, and generates independent data.
- *Dynamic Network Topology*: Each node is free to move about while communicating with other nodes. The topology of such an ad hoc network is dynamic in nature due to constant movement of the participating nodes, causing the intercommunication patterns among nodes to change continuously.
- *Wireless Connections and Multi-hop Routing*: Nodes communicate through wireless connections and share the same media (radio, infrared, etc.). In order to be able to

communicate with devices that are out of range, intermediate devices will forward data packets in a hop-by-hop fashion.

- *Resource and Energy Constrained Devices.* Most mobile devices are equipped with cheap and slow processors and limited storage capability. In addition, mobile devices generally rely on batteries as their power source. The use of complex algorithms there may not be possible.

- *Limited Physical Security.* The use of wireless communication and the exposure of the network nodes increase the possibility of attacks against the network. Due to the mobility of the nodes, the risk that nodes are physically compromised by theft, loss, or other means will probably be bigger than for traditional network nodes.

The dynamic and self-organizing nature of MANETs makes them particular useful in situations where rapid network deployments are required or it is prohibitively costly to deploy and manage network infrastructure. Some applications include:

- Attendees in a conference room sharing documents and other information via their laptops and hand-held computer;

- Armed forces creating a tactical network in an unfamiliar territory for communications and distribution of situational awareness information;

- Small sensor devices located in animals and other strategic locations that collectively monitor habitats and environmental conditions;

- Emergency services communicating in a disaster area and sharing video updates of locations among workers in the field, and back to headquarters.

Unfortunately, due to the inherent characteristics of MANETs mentioned above, securing such networks is particularly challenging and in many applications the traditional security solutions cannot be directly used. To provide potential solutions for protecting such networks, the primary task is to identify various types of attacks in MANETs, as discussed in the next section.

1.2 Attacks in Mobile Ad Hoc Networks

Having discussed the basic concept and the applications of MANETs, we look at some typical attacks in MANETs in this section. It is very important for protocol designers to keep in mind various attacks when designing the security mechanisms for MANETs.

Attacks against MANETs can be roughly classified into two major categories, namely external attacks and internal attacks, according to the domain of attacks. External attacks are carried out by nodes that do not belong to the domain of the network. These attackers try to join the network and access the resources without authorization. Unlike external attacks, internal attacks are from compromised nodes, which are actually part of the network. Compared with external attacks, internal ones are more serious because the attackers know valuable and secret information from compromised nodes and possess privileged access rights to the network resources. Furthermore, some attacks could be launched at multiple layers of MANETs. We list some typical attacks in MANETs as follows:

- *Eavesdropping*: eavesdropping is a very easy passive attack in the wireless communication environment. By placing an antenna at an appropriate location, an attacker can intercept and read the sensitive information that the victim transmits or receives without attracting the victim's attention. However, this attack can usually be prevented by encrypting the transmitted data.

- *Traffic Monitoring and Analysis*: Traffic monitoring and analysis can be deployed to collect information of network nodes and data transmission such as the identities and locations of nodes and the amount of data transmitted among them. These information could be exploited to launch further attacks.

- *Routing Attacks*: Attackers try to alter the routing information and data in the routing control packet. There are several attacks that fall into this category, such as rushing attacks [89] and wormhole attacks [90].

- *Location Disclosure Attack:* Attackers attempt to reveal information regarding the location of nodes or the structure of the network. By gathering the nodes' location information, the attacker can know which nodes are located in the routing path to the target node, and then plan the further attacks [31].

- *Resource Consumption Attack:* In this attack [162] a malicious node interacts with a victim with the intention of consuming its battery life by requesting excessive route discovery, or by forwarding unnecessary packets to that node.

- *Denial of Service Attack:* Denial of service (DoS) attacks can be easily applied to MANETs, where legitimate traffic cannot reach the target nodes since illegitimate traffic overwhelms the frequencies. DoS attacks can be launched from various layers, namely, physical layer, link layer, and network layer.

- *Sybil Attack:* In this attack [49] a single node attempts to create a large number of identities with malicious intent.

1.3 Security Services and Design Goals

Based on the characteristics of MANETs and a variety of attacks that we have described in the previous sections, we are now in a position to discuss the security services that are usually expected to be provided by the security mechanism in MANETs. Security services can be categorized into: *authentication*, *access control*, *confidentiality*, *integrity*, *non-repudiation*, and *availability*.

- *Authentication*: Authentication is the ability to verify a user's identity in an association and to assure the recipient that the message is from the source that it claims to be. Authentication is a fundamental mechanism to support access control.
- *Access Control*: Access Control is the ability to limit and control access to devices and applications via communication links. A user trying to gain access to the resource is first authenticated and then the corresponding access rights are granted.
- *Confidentiality*: Confidentiality ensures that the information transmitted over the network is unreadable to unauthorized users or nodes. Confidentiality can be achieved by using various encryption techniques.
- *Integrity:* Integrity is to be able to keep the data transmitted from being illegally modified or destroyed during the transmission.
- *Non-repudiation:* Non-repudiation guarantees that neither the sender nor the receiver of a message is able to deny the transmission.
- *Availability:* Availability is to keep the network service or resources available to legitimate users. It ensures the survivability of the network despite malicious incidents.

To provide the above security services, the first and also the paramount task is to design efficient key management mechanisms in MANETs because once provided, security services that employ cryptographic techniques can be implemented. Key management is the process that describes how the cryptographic keys are distributed to the network nodes and how these keys are further updated if required, revoked, and so on. The key management usually involves the following tasks [122]:

1. Generating the system parameters and registering the system users.
2. Creating, distributing, and installing the keying material.
3. Organizing the use of keying material.

4. Updating, revoking, and destroying the keying material.
5. Storing, recovering, and archiving the keying material.

In MANETs, the purpose of the first step is to initialize the system. In this step, the system administrator chooses a secure cryptosystem, verifies the credentials of the users, and provides a unique identity to each user of the system. This is then followed by creation and distribution of the keying material. In this step, the administrator generates the keying material for each user based on the cryptosystem selected and loads the system parameters and the keying material into the users' devices. In the third step, all the network nodes can communicate securely with each other by using the keys to encrypt the transmitted data. Since the attacker can manipulate compromised nodes to launch various attacks, the purpose of the fourth step is to revoke the compromised keys and therefore evict the attackers from the network. Furthermore, it might be necessary to update the keys of network nodes periodically to thwart some attacks. Finally, the fifth step could be needed in cases where the keying material must be saved, for example, for auditing purposes.

1.4 This Thesis and Related Work

Key management in MANETs is more difficult than that in traditional wireline network due to the salient characteristics of MANETs. Particularly, designing key management mechanisms for MANETs encompasses a very wide information security research domain, ranging from developing and implementing efficient cryptographic algorithms to formulating appropriate network security policies. This thesis will focus on the following subtopics:

(Ultra-)lightweight Symmetric Cipher

When deploying symmetric key-based key management schemes for MANETs, a symmetric cipher is usually employed to establish secure communication links among mobile devices during the operation of the network. However, those pervasive mobile devices usually have extremely constrained resources in terms of computational capabilities, memory, and power supply. Hence, classical cryptographic primitives designed for full-fledged computers might not be suited for resource-constrained mobile devices. Moreover, the tight cost constrains inherent in mass deployments of mobile devices also bring forward impending requirements for designing new cryptographic primitives that can perform strong authentication and encryption, and provide other security functionalities for ultralow-power applications in MANETs. This emerging research area is usually referred to as *lightweight cryptography.* The key issue of designing lightweight cryptographic algorithms is to deal with the trade-off among *security*, *cost*, and *performance* [144]. A host of lightweight symmetric ciphers that particularly target resource-constrained mobile devices have been published in the past few years. All the previous proposals can be roughly divided into the following three categories. The first category consists of highly optimized and compact hardware implementations for standardized block ciphers such as Advanced Encryption Standard (AES) [63, 64, 78], International Data Encryption Algorithm (IDEA) [113] and Extended Tiny Encryption Algorithm (XTEA) [98], whereas the proposals in the second category involve slight modifications of a classical block cipher like Data Encryption Standard (DES) [106] for lightweight applications. Finally, the third category features new low-cost designs, including lightweight block ciphers HIGHT [87], mCrypton [109], SEA [163], PRESENT [18] and KATAN and KTANTAN [26], as well as lightweight stream ciphers Grain [82], Trivium [27] and MICKEY [9]. In Chapter 2, we are going to present a novel ultra-lightweight cryptographic algorithm, referred to as Hummingbird, for resource-constrained mobile devices used in MANETs and investigate its performance across a wide range of resource-constrained software and hardware platforms.

Cryptographic Pairings on Genus 2 Hyperelliptic Curves

Identity-based cryptography (IBC) [157] has been extensively used to design asymmetric key-based key management schemes for MANETs in the past few years, see [46,85,151,178, 179] for example. To implement those key management protocols in resource-constrained mobile devices, it is essential to compute cryptographic pairings in an efficient manner. It used to be widely accepted that the limited resources associated with mobile devices make it impossible to execute public key cryptographic algorithms. However, recent studies have showed that even software implementations only of public-key cryptosystems such as elliptic curve cryptosystems (ECC) [76, 111, 121, 154, 170, 173], hyperelliptic curve cryptosystems (HECC) [173] and pairing-based cryptosystems (PBC) [137, 138, 158, 159, 175] are very viable and efficient on resource-constrained devices. For example, according to the state-of-the-art software implementation results on an 8-bit microcontroller ATmega128L, the generation and verification of a digital signature on a Koblitz elliptic curve defined over $\mathbb{F}_{2^{163}}$ take $0.36s$ and $0.63s$ [114], respectively, whereas the timing of computing an η_T pairing over $\mathbb{F}_{2^{239}}$ achieves about $1.93s$ [158]. The vast majority of the literature on pairing computation focuses solely on using elliptic curves. Note that Wollinger *et al.* [173] gave the first thorough comparison of the performance of ECC and HECC on a wide range of embedded processors including ARM, ColdFire and PowerPC. Their implementations demonstrated that HECC is suited for use in constrained environments and that the performance of HECC is better than ECC when special curve parameters are used. Therefore, an interesting question is that whether hyperelliptic curves can provide a viable alternative to using elliptic curves for pairing computation and achieve better performance. In [10, 80], the authors investigated pairing computation on *supersingular* hyperelliptic curves of genus 2 over binary and prime fields, respectively. Their implementation results showed that pairing computation on supersingular genus 2 curves over binary fields can outperform elliptic curves by using so-called η_T pairing. Moreover, for prime field case supersingular genus 2 curves are also viable candidates for practical use. In Chapter 3 we will propose efficient algorithms for pairing computation on two family of *non-supersingular* genus 2 curves and detail various techniques that lead to an efficient implementation.

Key Revocation in MANETs

In the context of wired networks, implementations of key revocation schemes are usually based on Public Key Infrastructures (PKIs) [88]. When the certificate of some user is to be revoked, a certificate authority (CA) adds that user's certificate information into a Certificate Revocation List (CRL) and puts it on an on-line trusted public repository or distributes it to other relevant users in some secure way. Based on such centralized structure, various solutions for the revocation have been proposed such as *Certificate Revocation System* (CRS) [123, 136], *Certificate Revocation Tree* (CRT) [101], and *Online Certificate Status Protocol* (OCSP) [120]. One common characteristic of all these methods is the need for good synchronization with the revocation source either by online status checks or by frequently refreshed certificates. Unfortunately, these conventional techniques are difficult to be applied to MANETs because of a number of unique features of MANETs such as the absence of an on-line CA and a centralized repository. Two main classes of solutions have been proposed for key revocation in MANETs. The schemes in the first category employ threshold cryptography and network nodes collaborate to revoke keys of malicious nodes [102, 117, 151, 179, 182]. Those in the second category are fully self-organized and each node has its own opinion about the network and other nodes' behavior [3, 30, 40, 86]. These solutions can be implemented with either certificate-based cryptography (CBC) [3,30,40,102,117,182] or identity-based cryptography (IBC) [86,151,179]. Furthermore, some novel ideas have also been proposed in the literature, which can be used to quickly remove malicious nodes from MANETs in particular application scenarios [37, 116, 129]. The main drawback of the existing key revocation systems is that they all classify the behavior of nodes in MANETs as either *good* and *bad* without any intermediate state. Such a binary behavior differentiation omits the actual cause and the degree of the misbehavior. In Chapter 4 we will describe a fine-grained key revocation scheme for MANETs based on the decentralized reputation system and Dirichlet multinomial model.

Broadcast Authentication in WSNs

In wireless sensor networks (WSNs), multi-user broadcast is an efficient and common communication paradigm, in which a host of network users will join in WSNs and disseminate messages (i.e., queries or commands) into the networks dramatically for obtaining the information of their interest [112, 145, 146]. Unfortunately, due to the nature of wireless communication in WSNs, adversaries can easily eavesdrop the traffic, impersonate other users, inject bogus data or alter the contents of legitimate messages during the multi-hop forwarding. Hence, authentication mechanisms need to be implemented in WSNs to protect broadcast messages from various malicious attacks. According to the cryptographic primitives employed, three categories of solutions have been proposed in the literature for addressing broadcast authentication in WSNs. Earlier research mainly focused on designing symmetric-key based broadcast authentication schemes. Typical examples are μTESLA [143] and its variants [50, 110, 112], which provide source authentication and message integrity by utilizing one-way hash chains and delayed disclosure of authentication keys. μTESLA-like schemes provide efficient broadcast authentication mechanisms for WSNs in terms of computational overhead and energy consumption. However, the inherent features of μTESLA-like schemes, such as the need for (loose) time synchronization and the delayed authentication, have made them vulnerable to a variety of attacks [134, 145, 146]. Moreover, scalability is another concern for symmetric-key based solutions [145]. The second category of solutions achieve broadcast authentication through the use of one-time signatures [32,142]. Unlike μTESLA, one-time signature based solutions do not need the time synchronization and authentication is also immediate. Unfortunately, such schemes have some undesirable features such as large key sizes and a limited number of usages per key, which make them only suitable for applications with infrequent messages at unpredictable times [115]. Considering the security and scalability of symmetric-key based broadcast authentication schemes, a couple of public-key based solutions have been proposed during the past few years [29, 145, 146, 176]. Public-key based broadcast authentication schemes have a common shortcoming: signature verification is much slower than the message authentication code verification used in symmetric-key based solutions. In Chapter 5 we will show how to speed up the signature verification for public-key based multi-user broadcast authentication schemes in WSNs.

1.5 Outline and Main Contributions

The outline and the main contributions of this thesis is the following:

- **Chapter 1** provides a general description of security in mobile ad hoc networks, including basic concept, typical attacks, security services and design goals. The motivation and the context of the work in this thesis are also presented.

- **Chapter 2** presents a novel ultra-lightweight cryptographic algorithm, referred to as Hummingbird, for resource-constrained devices used in MANETs. A preliminary security analysis of Hummingbird against the most common attacks to block ciphers and stream ciphers is provided. Efficient software implementations of Hummingbird on 4-, 8- and 16-bit microcontrollers are investigated. For the 4-bit microcontroller a speed optimized implementation is suggested, whereas for the 8-bit and 16-bit platforms both a speed optimized and a size optimized implementations are provided. A detailed performance comparison of Hummingbird and the state-of-the-art ultra-lightweight block cipher PRESENT on the similar platforms is conducted. Furthermore, four hardware architectures are devised for Hummingbird cipher, which stand for different design goals such as performance (i.e., area requirement and throughput) and supported functionalities (i.e., encryption-only or encryption/decryption). Efficient hardware implementations of Hummingbird on low-cost field programable gate arrays (FPGAs) and comparison with other cryptographic algorithms on similar hardware platforms are also described.

- **Chapter 3** shows how to speed up pairing computations on two families of non-supersingular genus 2 hyperelliptic curves over prime fields. We generalize Chatterjee *et al.*'s idea of encapsulating the computation of the line function with the group operations to genus 2 hyperelliptic curves, and derive new explicit formulae for the group operations in projective and new coordinates in the context of pairing computations. Moreover, we also propose new variants of Miller's algorithm by exploiting efficiently computable automorphisms on the two families of non-supersingular genus 2 curves in question. Various techniques that lead to an efficient implementation are also detailed. As a case study, we combine our new algorithm with various optimization techniques to efficiently implement the Tate pairing on a non-supersingular genus 2 curve $y^2 = x^5 + 9x$ over $\mathbb{F}_p$ with an embedding degree of $k = 4$. Detailed performance analysis and comparison with pairing computations on supersingular genus 2 curves are also provided.

- **Chapter 4** proposes a fully self-organized key revocation scheme for MANETs based on the Dirichlet multinomial model. The network and security modes as well as design goals are formulated. The behavior of network nodes is classified into three categories, namely good behavior, suspicious behavior, and malicious behavior. A detailed procedure for nodes analyzing and predicting peers' behavior based on their own observations and other nodes' reports is described. 3-dimension Dirichlet distribution is utilized by nodes to keep track of three categories of behavior defined and classified by an external trusted authority, and updates its knowledge about other nodes' behavior. Moreover, a deviation test is employed to filter potentially false statements from adversaries and Dempster-Shafer belief theory is used to integrate other nodes' reports. Our scheme differentiates between suspicious behavior and malicious behavior, and therefore nodes can make two different responses by either revoking malicious nodes' keys or ceasing the communication with suspicious nodes for some time based on the analysis of the collected information. Furthermore, the robustness of our key revocation scheme against the false accusation attack from independent and collusive adversaries are verified through extensive simulations.

- **Chapter 5** describes an efficient technique to accelerate the signature verification for public-key based multi-user broadcast authentication schemes in WSNs by exploiting the cooperation among sensor nodes. The system and adversary models are defined and the faster signature verification protocol is presented by using elliptic curve digital signature algorithm (ECDSA) as an example. The selection of appropriate system parameters is discussed and the security of the proposed technique is analyzed. As a case study, we also apply the faster signature verification technique to the broadcast authentication in a 4×4 grid-based WSN and analyze the performance of the proposed protocol with respect to communication and computation overheads. A detailed performance comparison with the traditional ECDSA signature verification is also conducted.
- **Chapter 6** summarizes the conclusions of our work and suggests possible directions for future research.

Chapter 2

Hummingbird: Ultra-Lightweight Cryptography for Resource-Constrained Devices

In this chapter we present a novel ultra-lightweight cryptographic algorithm, referred to as Hummingbird, for resource-constrained devices used in MANETs. Hummingbird has a *hybrid structure* of block cipher and stream cipher and was developed with both lightweight software and lightweight hardware implementations for constrained devices in mind. The hybrid model can provide the designed security with small block size and is therefore expected to meet the stringent response time and power consumption requirements for a large variety of embedded applications. We start with a brief overview of related work in Section 2.1, followed by the description of the Hummingbird cryptographic algorithm and its design rationale in Section 2.2. The security analysis of Hummingbird against common attacks such as differential and linear cryptanalysis is presented in Section 2.3. Sections 2.4 and 2.5 treat efficient software and hardware implementations of Hummingbird for 4-, 8- and 16-bit microcontrollers as well as low-cost field-programmable gate array (FPGA) devices, respectively. Finally, we close this chapter with further remarks on the operation modes of Hummingbird in Section 2.6. Please note that parts of this chapter are based on joint work with Daniel Engels, Honggang Hu, Guang Gong, and Eric Smith, and Hummingbird is originally designed by Daniel Engels, Peter Schweitzer, and Eric Smith. Partial contents of this chapter have been published in [57, 60].

2.1 Related Work

Quite a few lightweight symmetric ciphers that particularly target resource-constrained smart devices have been published in the past few years and those ciphers can be utilized as basic building blocks to design security mechanisms for MANETs. The existing solutions can be classified into the following three categories:

Compact Hardware Implementations for Standardized Block Ciphers

Feldhofer *et al.* [63,64] and Hämäläinen *et al.* [78] proposed low-power and compact ASIC encryption cores for 128-bit AES, respectively. In both designs, the 8-bit datapath is used for the round operations as well as for the on-the-fly key expansion and the S-box is implemented as combinatorial logic. Their low-cost AES implementations require only $3,400$ and $3,100$ gate equivalents (GE) and are able to encrypt a 128-bit data block within $1,032$ and 160 clock cycles, respectively.

Liu *et al.* [113] implemented a low-power encryption core for IDEA on SMIC 0.18μm process. Their implementation consumes about $4,660$ GE and can output 64-bit ciphertext per 320 clock cycles. Moreover, the average power is around 3μW when the supply voltage is 1.8V.

Kaps [98] presented efficient implementations of XTEA on FPGAs and ASICs for ultra-low power applications such as RFID tags and wireless sensor nodes. The compact hardware implementations of XTEA can encrypt a 64-bit data block within 112 clock cycles and require $3,490$ GE (ASIC) or 254 slices (FPGA).

Slight Modifications of a Classical Block Cipher

Leander *et al.* [106] suggested a lightweight DES variant called DESL (DES Lightweight). The key idea of the DESL design is to replace the eight original S-boxes with a single new one. The resulting serialized DESL ASIC implementation has an area requirement of $1,848$ GE and it can encrypt one 64-bit data block within 144 clock cycles.

New Low-Cost Block Ciphers and Stream Ciphers

Hong *et al.* [87] proposed a lightweight block cipher HIGHT with 64-bit block length and 128-bit key length. It has a 32-round iterative structure and targets resource-constrained devices like wireless sensor nodes and RFID tags. The authors claimed that a round-based implementation of HIGHT requires $3,048$ GE and a software implementation on 8-bit wireless sensor nodes is much faster than AES. Unfortunately, no details about software and hardware implementations are provided.

Lim and Korkishko [109] designed a lightweight block cipher mCrypton in 2006, which has a 64-bit block size and three key size options (64 bits, 96 bits and 128 bits). It consists of 13 rounds and specifically designed for resource-constrained smart devices. A round-based prototype implementation for an encryption-only core requires $2,420$ GE with a 64-bit key, $2,681$ GE with a 96-bit key and $2,949$ GE with a 128-bit key, respectively.

Standaert *et al.* [163] suggested the Scalable Encryption Algorithm (SEA) in 2006, which is a parametric block cipher targeted for small embedded applications. Different from other block ciphers, SEA takes the plaintext size n, the key length k as well as the processor word size b as parameters and, therefore, can be straightforwardly adapted to various implementation contexts and security requirements. On an 8-bit microcontroller ATtiny from Atmel, a software implementation of SEA with a block and key size of $n = 96$ bits, a word size of $b = 8$ bits and $n_r = 93$ rounds can encrypt one data block with $17,745$ clock cycles [163], whereas a hardware implementation with the same parameters needs $1,428$ clock cycles and $3,758$ GE [118].

Bogdanov *et al.* [18] described an ultra-lightweight SP-network based block cipher PRESENT in 2007, which has 32 rounds, a block size of 64 bits, and a key size of 80 or 128 bits. PRESENT is an aggressively hardware optimized block cipher which uses a single 4-bit S-box in both the datapath and the key scheduling. Particularly, a serial version of PRESENT can be implemented with as few as $1,000$ GE [147]. Moreover, the detailed software implementation results of PRESENT on a wide range of platforms such as 4-, 8-, 16- and 32-bit microcontrollers/CPUs can be found in [144].

Cannière *et al.* [26] proposed a new family of very efficient hardware-oriented block ciphers KATAN and KTANTAN. The KATAN family is composed of three block ciphers with 32, 48, or 64-bit block size, 80-bit key size and 254 rounds, whereas the KTANTAN family contains the other three ciphers with the same block sizes and achieves more compact hardware implementation by burning the key into device. In particular, the hardware implementation of the smallest cipher of the entire family, KTANTAN32, requires only 462 GE while achieving a throughput of 12.5 Kbit/sec at 100 KHz.

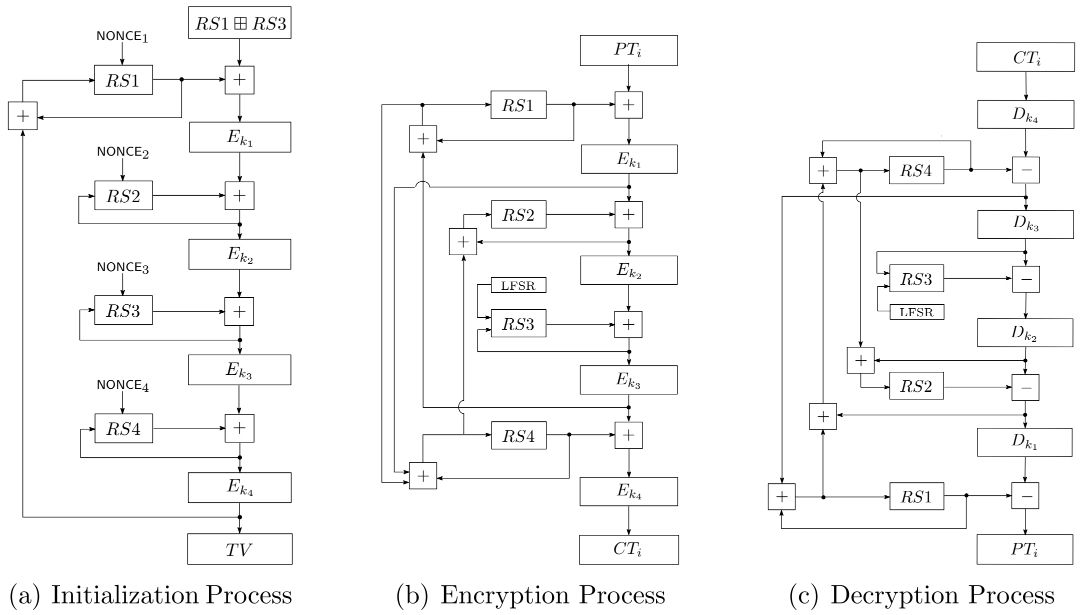
While many lightweight block ciphers have been proposed for resource-constrained devices, a number of lightweight stream ciphers that specifically targeted lightweight hardware implementation have been selected through the eSTREAM project [54]. Grain [82], Trivium [27] and MICKEY [9] are among the finalist of the hardware profile. While Grain and Trivium can be respectively implemented in only $1,294$ GE and $2,580$ GE, MICKEY requires $3,188$ GE [73]. Unlike block ciphers, the above three stream ciphers need a relatively long initialization phase (i.e., 321 clock cycles for Grain, $1,314$ clock cycles for Trivium and 267 clock cycles for MICKEY) before first use.

2.2 The Hummingbird Cryptographic Algorithm

Hummingbird is neither a block cipher nor a stream cipher, but a *rotor machine* equipped with novel rotor-stepping rules. The design of Hummingbird is based on an elegant combination of block cipher and stream cipher with 16-bit block size, 256-bit key size, and 80-bit internal state. The size of the key and the internal state of Hummingbird provides a security level which is adequate for many embedded applications. For clarity, we use the notation listed in Table 2.1 in the algorithm description. A top-level structure of the Hummingbird cryptographic algorithm is shown in Figure 2.1, which consists of four 16-bit block ciphers E_{k_i} or D_{k_i} ($i = 1, 2, 3, 4$), four 16-bit internal state registers RSi ($i = 1, 2, 3, 4$), and a 16-stage Linear Shift Feedback Register (LFSR). Moreover, the 256-bit secret key K is divided into four 64-bit subkeys k_1, k_2, k_3 and k_4 which are used in the four block ciphers, respectively.

Table 2.1: Notation

PT_i	the i-th 16-bit plaintext block, $i = 1, 2, \ldots, n$
CT_i	the i-th 16-bit ciphertext block, $i = 1, 2, \ldots, n$
K	the 256-bit secret key
$\mathbf{E}_K(\cdot)$	the encryption function of Hummingbird with 256-bit secret key K
$\mathbf{D}_K(\cdot)$	the decryption function of Hummingbird with 256-bit secret key K
k_i	the 64-bit subkey used in the i-th block cipher, $i = 1, 2, 3, 4$, such that $K = k_1 \| k_2 \| k_3 \| k_4$
$E_{k_i}(\cdot)$	a block cipher encryption algorithm with 16-bit input, 64-bit key k_i, and 16-bit output, i.e., $E_{k_i} : \{0,1\}^{16} \times \{0,1\}^{64} \to \{0,1\}^{16}, i = 1, 2, 3, 4$
$D_{k_i}(\cdot)$	a block cipher decryption algorithm with 16-bit input, 64-bit key k_i, and 16-bit output, i.e., $D_{k_i} : \{0,1\}^{16} \times \{0,1\}^{64} \to \{0,1\}^{16}, i = 1, 2, 3, 4$
RSi	the i-th 16-bit internal state register, $i = 1, 2, 3, 4$
LFSR	a 16-stage Linear Feedback Shift Register with the characteristic polynomial $f(x) = x^{16} + x^{15} + x^{12} + x^{10} + x^7 + x^3 + 1$
$\boxplus$	modulo 2^{16} addition operator
$\boxminus$	modulo 2^{16} subtraction operator
$\oplus$	exclusive-OR (XOR) operator
$m \lll l$	left circular shift operator, which rotates all bits of m to the left by l bits, as if the left and the right ends of m were joined.
$K_j^{(i)}$	the j-th 16-bit key used in the i-th block cipher, $j = 1, 2, 3, 4$, such that $k_i = K_1^{(i)} \| K_2^{(i)} \| K_3^{(i)} \| K_4^{(i)}$
$S_i(x)$	the i-th 4-bit to 4-bit S-box used in the block cipher, $S_i(x) : \mathbb{F}_2^4 \to \mathbb{F}_2^4, i = 1, 2, 3, 4$
NONCE_i	the i-th nonce which is a 16-bit random number, $i = 1, 2, 3, 4$
IV	the 64-bit initial vector, such that $IV = \mathsf{NONCE}_1 \| \mathsf{NONCE}_2 \| \mathsf{NONCE}_3 \| \mathsf{NONCE}_4$



(a) Initialization Process (b) Encryption Process (c) Decryption Process

Figure 2.1: A Top-Level Description of the Hummingbird Cryptographic Algorithm

2.2.1 Initialization Process

The overall structure of the Hummingbird initialization algorithm is shown in Figure 2.1(a). When using Hummingbird in practice, four 16-bit random nonces NONCE_i are first chosen to initialize the four internal state registers RSi $(i = 1, 2, 3, 4)$, respectively, followed by four consecutive encryptions on the message $RS1 \boxplus RS3$ by Hummingbird running in initialization mode (see Figure 2.1(a)). The final 16-bit ciphertext TV is used to initialize the LFSR. Moreover, the 13th bit of the LFSR is always set to prevent a zero register. The LFSR is also stepped once before it is used to update the internal state register $RS3$. We summarize the Hummingbird initialization process in the following Algorithm 1.

2.2.2 Encryption Process

The overall structure of the Hummingbird encryption algorithm is depicted in Figure 2.1(b). After a system initialization process, a 16-bit plaintext block PT_i is encrypted by first executing a modulo 2^{16} addition of PT_i and the content of the first internal state register $RS1$. The result of the addition is then encrypted by the first block cipher E_{k_1}. This procedure is repeated in a similar manner for another three times and the output of E_{k_4} is the corresponding ciphertext CT_i. Furthermore, the states of the four internal state registers will also be updated in an unpredictable way based on their current states, the

Algorithm 1 Hummingbird Initialization

```
Input: Four 16-bit random nonce $\mathsf{NONCE}_i$ $(i = 1, 2, 3, 4)$
Output: Initialized four rotors $RSi_4$ $(i = 1, 2, 3, 4)$ and LFSR
 1: $RS1_0 = \mathsf{NONCE}_1$                                    [Nonce Initialization]
 2: $RS2_0 = \mathsf{NONCE}_2$
 3: $RS3_0 = \mathsf{NONCE}_3$
 4: $RS4_0 = \mathsf{NONCE}_4$
 5: for $t = 0$ to 3 do
 6:     $V12_t = E_{k_1}((RS1_t \boxplus RS3_t) \boxplus RS1_t)$
 7:     $V23_t = E_{k_2}(V12_t \boxplus RS2_t)$
 8:     $V34_t = E_{k_3}(V23_t \boxplus RS3_t)$
 9:     $TV_t = E_{k_4}(V34_t \boxplus RS4_t)$
10:     $RS1_{t+1} = RS1_t \boxplus TV_t$
11:     $RS2_{t+1} = RS2_t \boxplus V12_t$
12:     $RS3_{t+1} = RS3_t \boxplus V23_t$
13:     $RS4_{t+1} = RS4_t \boxplus V34_t$
14: end for
15: LFSR = $TV_3$ | 0x1000                                       [LFSR Initialization]
16: return $RSi_4$ $(i = 1, 2, 3, 4)$ and LFSR
```

outputs of the first three block ciphers, and the state of the LFSR. Algorithm 2 describes the detailed procedure of Hummingbird encryption.

Algorithm 2 Hummingbird Encryption

```
Input: A 16-bit plaintext $PT_i$ and four rotors $RSi_t$ $(i = 1, 2, 3, 4)$
Output: A 16-bit ciphertext $CT_i$
 1: $V12_t = E_{k_1}(PT_i \boxplus RS1_t)$                       [Block Encryption]
 2: $V23_t = E_{k_2}(V12_t \boxplus RS2_t)$
 3: $V34_t = E_{k_3}(V23_t \boxplus RS3_t)$
 4: $CT_i = E_{k_4}(V34_t \boxplus RS4_t)$
 5: $\mathrm{LFSR}_{t+1} \leftarrow \mathrm{LFSR}_t$                [Internal State Updating]
 6: $RS1_{t+1} = RS1_t \boxplus V34_t$
 7: $RS3_{t+1} = RS3_t \boxplus V23_t \boxplus \mathrm{LFSR}_{t+1}$
 8: $RS4_{t+1} = RS4_t \boxplus V12_t \boxplus RS1_{t+1}$
 9: $RS2_{t+1} = RS2_t \boxplus V12_t \boxplus RS4_{t+1}$
10: return $CT_i$
```

2.2.3 Decryption Process

The overall structure of the Hummingbird decryption algorithm is illustrated in Figure 2.1(c). The decryption process follows the similar pattern as the encryption and a detailed description is shown in the following Algorithm 3.

Algorithm 3 Hummingbird Decryption

Input: A 16-bit ciphertext CT_i and four rotors RSi_t $(i = 1, 2, 3, 4)$
Output: A 16-bit plaintext PT_i

```
1: $V34_t = D_{k_4}(CT_i) \boxminus RS4_t$                    [Block Decryption]
2: $V23_t = D_{k_3}(V34_t) \boxminus RS3_t$
3: $V12_t = D_{k_2}(V23_t) \boxminus RS2_t$
4: $PT_i = D_{k_1}(V12_t) \boxminus RS1_t$
5: $\mathrm{LFSR}_{t+1} \leftarrow \mathrm{LFSR}_t$                    [Internal State Updating]
6: $RS1_{t+1} = RS1_t \boxplus V34_t$
7: $RS3_{t+1} = RS3_t \boxplus V23_t \boxplus \mathrm{LFSR}_{t+1}$
8: $RS4_{t+1} = RS4_t \boxplus V12_t \boxplus RS1_{t+1}$
9: $RS2_{t+1} = RS2_t \boxplus V12_t \boxplus RS4_{t+1}$
10: return $PT_i$
```

2.2.4 16-Bit Block Cipher

Hummingbird employs four identical block ciphers $E_{k_i}(\cdot)$ $(i = 1, 2, 3, 4)$ in a consecutive manner, each of which is a typical substitution-permutation (SP) network with 16-bit block size and 64-bit key as shown in the following Figure 2.2.

The block cipher consists of four regular rounds and a final round. The 64-bit subkey k_i is split into four 16-bit round keys $K_1^{(i)}, K_2^{(i)}, K_3^{(i)}$ and $K_4^{(i)}$ that are used in the four regular rounds, respectively. Moreover, the final round utilizes two keys $K_5^{(i)}$ and $K_6^{(i)}$ directly derived from the four round keys (see Fig. 2.2). While each regular round comprises of a key mixing step, a substitution layer, and a permutation layer, the final round only includes the key mixing and the S-box substitution steps. The key mixing step is implemented using a simple exclusive-OR operation, whereas the substitution layer is composed of four S-boxes with 4-bit inputs and 4-bit outputs as shown in Table 2.2.

The selected four S-boxes, denoted by $S_i(x) : \mathbb{F}_2^4 \rightarrow \mathbb{F}_2^4, i = 1, 2, 3, 4,$ are Serpent-type S-boxes [1] with additional properties (see Appendix A) which can ensure that the 16-bit block cipher is resistant to linear and differential attacks as well as interpolation attack.

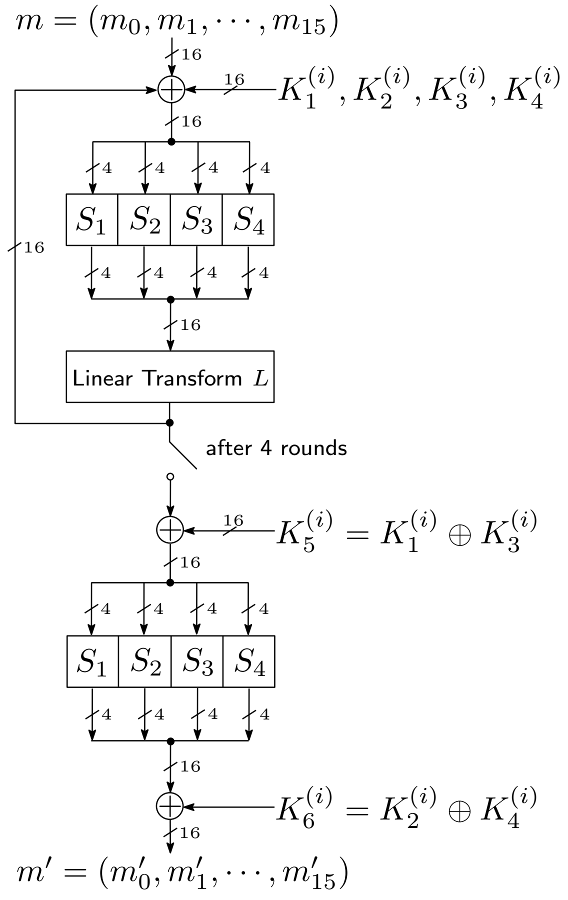

$m = (m_0, m_1, \cdots, m_{15})$
16
$K_1^{(i)}, K_2^{(i)}, K_3^{(i)}, K_4^{(i)}$
4
S_1 S_2 S_3 S_4
Linear Transform L
after 4 rounds
$K_5^{(i)} = K_1^{(i)} \oplus K_3^{(i)}$
$K_6^{(i)} = K_2^{(i)} \oplus K_4^{(i)}$
$m' = (m'_0, m'_1, \cdots, m'_{15})$

Figure 2.2: The Structure of Block Cipher in the Hummingbird Cryptographic Algorithm

Table 2.2: Four S-Boxes in Hexadecimal Notation

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S_1(x)$	8	6	5	F	1	C	A	9	E	B	2	4	7	0	D	3
$S_2(x)$	0	7	E	1	5	B	8	2	3	A	D	6	F	C	4	9
$S_3(x)$	2	E	F	5	C	1	9	A	B	4	6	8	0	7	3	D
$S_4(x)$	0	7	3	4	C	1	A	F	D	E	6	B	2	8	9	5

The permutation layer in the 16-bit block cipher is given by the linear transform $L : \{0,1\}^{16} \to \{0,1\}^{16}$ defined as follows:

$$L(m) = m \oplus (m \lll 6) \oplus (m \lll 10),$$

where $m = (m_0, m_1, \cdots, m_{15})$ is a 16-bit data block. We give a detailed description for the encryption process of the 16-bit block cipher in the following Algorithm 4. The decryption process can be easily derived from the encryption and therefore is omitted here.

Algorithm 4 16-bit Block Cipher Encryption $E_{k_i}(\cdot)$

```
Input: A 16-bit data block $m = (m_0, m_1, \cdots, m_{15})$ and a 64-bit subkey $k_i$ such that
       subkey $k_i = K_1^{(i)} \| K_2^{(i)} \| K_3^{(i)} \| K_4^{(i)}$
Output: A 16-bit date block $m' = (m'_0, m'_1, \cdots, m'_{15})$
 1: for $j = 1$ to 4 do
 2:     $m \leftarrow m \oplus K_j^{(i)}$                                    [key mixing step]
 3:     $A = m_0 \| m_1 \| m_2 \| m_3$,   $B = m_4 \| m_5 \| m_6 \| m_7$
        $C = m_8 \| m_9 \| m_{10} \| m_{11}, D = m_{12} \| m_{13} \| m_{14} \| m_{15}$
 4:     $m \leftarrow S_1(A) \| S_2(B) \| S_3(C) \| S_4(D)$                 [substitution layer]
 5:     $m \leftarrow m \oplus (m \lll 6) \oplus (m \lll 10)$               [permutation layer]
 6: end for
 7: $m \leftarrow m \oplus K_1^{(i)} \oplus K_3^{(i)}$
 8: $A = m_0 \| m_1 \| m_2 \| m_3$,   $B = m_4 \| m_5 \| m_6 \| m_7$
    $C = m_8 \| m_9 \| m_{10} \| m_{11}, D = m_{12} \| m_{13} \| m_{14} \| m_{15}$
 9: $m \leftarrow S_1(A) \| S_2(B) \| S_3(C) \| S_4(D)$
10: $m' \leftarrow m \oplus K_2^{(i)} \oplus K_4^{(i)}$
11: return $m' = (m'_0, m'_1, \cdots, m'_{15})$
```

Remark To further reduce the consumption of the memory, area and power of Hummingbird in both software and hardware implementations, four S-boxes used in Hummingbird can be replaced by a single S-box, which is repeated four times in the 16-bit block cipher. The compact version of Hummingbird can achieve the same security level as the original Hummingbird and will be implemented on both microcontrollers and FPGAs in this chapter.

2.2.5 Design Rationale

The design of the Hummingbird cryptographic algorithm is motivated by the well-known Enigma machine[1] and takes into account both security and efficiency simultaneously. The encryption/decryption process of Hummingbird can be viewed as the continuous running of a rotor machine, where four small block ciphers act as four virtual rotors which perform permutations on 16-bit words. The salient characteristics of Hummingbird lies in implementing extraordinarily large virtual rotors with custom block ciphers and using successively changing internal states to step each virtual rotor in various and unpredictable ways. Besides a novel cipher structure, Hummingbird is also designed to protect against the most common attacks such as linear and differential cryptanalysis, which will be discussed

[1]In Enigma machine each rotor has 26 contacts, whereas in Hummingbird each virtual rotor (i.e., a 16-bit block cipher) has $2^{16} = 65536$ contacts.

in detail in Section 2.3. Moreover, extremely simple arithmetic and logic operations are extensively employed in Hummingbird, which make it well-suited for resource-constrained environments.

2.3 Security Analysis

In this section, we analyze the security of the Hummingbird cryptographic algorithm by showing that it is resistant to the most common attacks to block ciphers and stream ciphers including birthday attack, differential and linear cryptanalysis, etc. Note that Hummingbird has a hybrid mode of block cipher and stream cipher, which can be considered as a finite state machine with the internal state $(RS1, RS2, RS3, RS4, \text{LFSR})$. However, the value of LFSR does not depend on those of $RS1$, $RS2$, $RS3$, and $RS4$. The purpose of using the LFSR is to guarantee the period of the internal state is at least 2^{16}.

A. Birthday Attack on the Initialization. For a fixed key, one may want to find two identical internal states $(RS1, RS2, RS3, RS4, \text{LFSR})$ initialized by two different IVs using the birthday attack. However, if we fix the key in the initialization procedure of the Hummingbird encryption scheme, the mapping $(RS1_t, RS2_t, RS3_t, RS4_t) \to (RS1_{t+1}, RS2_{t+1}, RS3_{t+1}, RS4_{t+1})$ is one-to-one. Hence the birthday attack does not work in this case.

B. Differential Cryptanalysis. Let $\mathbf{E}_K(\cdot)$ denote the encryption function of Hummingbird with the 256-bit key K. Recall that $E_{k_i}(\cdot)$, defined in Section 2.2.4, denotes the 16-bit block cipher encryption used in Hummingbird with the 64-bit subkey k_i. Then $\mathbf{E}_K(\cdot)$ is the composition of four $E_{k_i}(\cdot), i = 1, 2, 3, 4$. For a function $F(x)$ from $\mathbb{F}_2^m$ to $\mathbb{F}_2^m$, the differential between $F(x)$ and $F(x+a)$, where $+$ is the bit-wise addition, is denoted by $D_F(a,b)$ and defined as follows:

$$D_F(a,b) = \left|\{x \mid F(x) + F(x+a) = b,\ x \in \mathbb{F}_2^m\}\right|.$$

For many keys, we have computed the differentials of both $\mathbf{E}_K(\cdot)$ and $E_{k_i}(\cdot)$. Note that from Section 2.2.4 we know that there are five rounds in $E_{k_i}(\cdot)$. We list the differential $D_{E_{k_i}}(a,b)$ for each round in the following Table 2.3.

For substantially large amount of initial vectors IV and keys K, the differentials for both $E_{k_i}(\cdot)$ and $\mathbf{E}_K(\cdot)$ satisfy the following inequalities:

$$\max_{a,b\in\mathbb{F}_2^{16}, a\neq 0}\{D_{E_{k_i}}(a,b)\} \leq 20, \text{ and } \max_{a,b\in\mathbb{F}_2^{16}, a\neq 0}\{D_{\mathbf{E}_K}(a,b)\} \leq 20.$$

In other words, the differential of $\mathbf{E}_K(\cdot)$ has the same upper bound as $E_{k_i}(\cdot)$, the block cipher components in $\mathbf{E}_K(\cdot)$. We also tested the reduced version of Hummingbird for more instances of different pairs of (IV, K). From those experimental results, in general, the

Table 2.3: Differential Property of the 16-bit Block Cipher

# of Rounds	$\max_{a\neq 0,b} D_{E_{k_i}}(a,b)$
0	16384
1	1024
2	98
3	20
4	20

standard differential cryptanalysis method is not applicable to Hummingbird with practical time complexity.

C. Linear Cryptanalysis. For the linear cryptanalysis of $\mathbf{E}_K(\cdot)$, we need to consider the absolute value of the Walsh transform of $\mathbf{E}_K(\cdot)$, which is denoted by $|\hat{\mathbf{E}}_K(a,b)|$ and defined as follows,

$$\hat{\mathbf{E}}_K(a,b) = \sum_{x\in\mathbb{F}_2^{16}} (-1)^{\langle a,\mathbf{E}_K(x)\rangle+\langle b,x\rangle},\ \ a,b\in\mathbb{F}_2^{16} \text{ and } a\neq 0,$$

where $\langle x,y\rangle$ is the inner product of two binary vectors x and y (see Appendix A for the details). Unlike the case for the differential of $\mathbf{E}_K(\cdot)$ or $E_{k_i}(\cdot)$, we cannot perform an exhaustive computation for $|\hat{\mathbf{E}}_K(a,b)|$ for all $a,b\in\mathbb{F}_2^{16}$ and $a\neq 0$. The reason is that there are around 2^{48} instances for (a,b,x) that need to be verified for a pair of fixed IV and key K. For some fixed pairs of (IV,K), we have computed random subsets of (a,b) with size around 2^{20}. Those experimental results show that $|\hat{\mathbf{E}}_K(a,b)| \leq c\cdot\sqrt{2^{16}}$, where $c\leq 4.96875$. We list some data in the following Table 2.4.

Table 2.4: Linear Property of the 16-bit Block Cipher

$wt(a)$	$wt(b)$	Constant c
1	1	4.703125
1	2	4.359375
1	3	4.500000
2	1	4.390625
2	2	4.281250
2	3	4.828125
3	1	4.968750
3	2	4.718750
3	3	4.781250

We also conducted the experiments for an 8-bit version of the Hummingbird encryption scheme which means that all the rotors $RS_i, i=1,2,3,4$ and LFSR contain only 8 bits.

The Walsh transform of this reduced version of Hummingbird is bounded by $5 \cdot \sqrt{2^8}$ for many pairs of IV and key. This is the supporting evidence (albeit weak) that the absolute value of the Walsh transform of Hummingbird encryption function could be bounded by the square root of 2^{16} multiplying by a constant. Hence, Hummingbird seems to be resistant to linear cryptanalysis attack with practical time complexity.

D. Structural Attack. The Hummingbird encryption scheme may be viewed as a certain operation mode of a block cipher. For example, the ciphertext can be viewed as the internal state of a block cipher in CBC mode. In [15, 16], Biham investigated some operation modes of block ciphers. The author found that many triple modes are not as secure as one expected. In [17], Biham and Knudsen broke the ANSI X9.52 CBCM Mode. However, the internal state transition in Hummingbird encryption scheme is much more complicated than those studied by [15–17]. Hence, those attacks cannot be simply applied to the Hummingbird encryption scheme. In [169], by choosing IV, Wagner presented some new attacks on some modes proposed by Biham. Because IV initialization is used in the Hummingbird encryption scheme, Wagner's attacks are not applicable.

E. Algebraic Attack. For the Hummingbird encryption scheme, the degree of each S-box in the block cipher $E_{k_i}(\cdot)$ is maximized. Moreover, each block cipher $E_{k_i}(\cdot)$ consists of five rounds. Thus, there are totally 20 rounds for the Hummingbird encryption scheme, i.e., $\mathbf{E}_K(\cdot)$. Furthermore, the internal state transition involves modulo 2^{16} operation. Hence it is hard to apply efficient linearization techniques for algebraic attacks to Hummingbird.

F. Cube Attack. The success probability of cube attack is high if the degree of the internal state transition function in a stream cipher is low. For example, the degree of internal state transition function of Trivium grows slowly [47]. However, for the Hummingbird encryption scheme, the degree of the internal state transition function is very high. In addition, Hummingbird encryption scheme has a hybrid mode of block cipher and stream cipher. We have tested both the 16-bit block cipher $E_{k_i}(\cdot)$ used in the Hummingbird encryption scheme and the Hummingbird encryption function $\mathbf{E}_K(\cdot)$. We note that no linear equations of key bits can be used in the way as suggested in [47].

G. Slide and Related-Key Attack. Both slide attacks [19,20] and related-key attacks [14] need to exploit the weakness of key scheduling. However, there is no key scheduling in Hummingbird. In particular, the subkeys used in four small block ciphers are independent. In addition, the four rotors affect the output of each small block cipher in a nonlinear way. Hence, both slide attacks [19] and related-key attacks [14] cannot be applied to the Hummingbird.

H. Interpolation and Higher Order Differential Attack. Interpolation and higher order differential attacks [92, 104] can be applied to block ciphers with the low algebraic degree. As we discussed before for algebraic attack, the algebraic degree of the Hummingbird encryption is high. Hence it is difficult to apply interpolation and higher order differential attacks to the Hummingbird.

I. Complementation Properties. The DES has the following well-known complementation property, namely that if C is the ciphertext of the plaintext P under key K, then $\overline{C}$ is the ciphertext of $\overline{P}$ under key $\overline{K}$, where $\overline{x}$ is the bitwise complement of x. However, Hummingbird does not have this weakness due to the presence of the carry propagation resulting from four rotors.

2.4 Efficient Software Implementations on Low-Power Microcontrollers

One of the design goals of Hummingbird is to achieve good performance on typical embedded processors used in MANETs. This section presents software implementation results of the compact version of Hummingbird (i.e., a single 4×4 S-box is used four times in the 16-bit block cipher) across a wide range of low-power microcontrollers and compares the performance of Hummingbird and the other ultra-lightweight block cipher PRESENT [18] on the same or similar platforms. We first present implementation results of Hummingbird on a zero-power 4-bit MARC4 microcontroller from Atmel, which are among the only two implementations of cryptographic primitives on a 4-bit microcontroller so far. Then we implement Hummingbird on the 8-bit microcontroller ATmega128L from Atmel and the 16-bit microcontroller MSP430 from Texas instrument (TI), which are the most popular processors used in wireless sensor network platforms because of their low power design, multiple sensor interfaces, and widely available development tools. For achieving a better performance on the target 4-bit microcontroller, only a speed optimized implementation is considered, whereas two implementation variants are provided for both 8-bit and 16-bit microcontrollers, one of which is optimized for **code size** and the other for **speed**. Moreover, all implementation variants can perform both encryption and decryption.

2.4.1 Software Implementation on a 4-Bit Microcontroller

The Atmel MARC4 family of microcontrollers is small and lightweight with low power consumption and (optionally) integrated on-chip radio transmission. Their applications range from remote keyless entry and car tire pressure monitoring to wireless warning and alarm systems (tension, temperature, smoke detectors, etc.). In this subsection, we investigate efficient implementation of Hummingbird on a zero-power 4-bit ATAM893-D microcontroller of Atmel's MARC4 family and compare the performance of Hummingbird and PRESENT on the same platform.

4-Bit Microcontroller **ATAM893-D** and Development Tools

We chose the ATAM893-D [7], a member of Atmel's MARC4 family of 4-bit single-chip microcontrollers, as the target 4-bit platform due to its unique features such as high data throughput and low current consumption, which make it a perfect candidate for energy constrained wireless applications. The ATAM893-D mainly consists of a MARC4 4-bit CPU core, parallel I/O ports, the Universal Timer/Counter Communication Module (UTCM) with two 8-bit programmable multifunction/timer/counters, and the clock management module with integrated RC-, 32-kHz and 4-MHz crystal oscillators. A block diagram of the ATAM893-D can be found in [7].

The high performance and low power consumption of MARC4 microcontroller family are based on the usage of an advanced stack-based 4-bit CPU core (see [6, p.8]). The core contains 4KBytes program memory (ROM), 256×4-bit data memory (RAM), 12-bit wide Program Counter (PC), four 8-bit wide RAM address registers `SP`, `RP`, `X` and `Y`, 4-bit wide Condition Code Register (CCR), 4-bit Arithmetic Logic Unit (ALU), instruction decoder and interrupt controller. The key features of the MARC4 core include [6]:

- *RISC*[2] *Architecture*: 72 simple and highly-optimized instructions are provided and most of them are single-byte instructions.
- *HARVARD Structure*: Three independent buses (i.e., the instruction bus, the memory bus and the I/O bus) can be used in parallel to communicate between the ROM, the RAM, and attached peripherals.
- *Power Saving Modes*: The MARC4 core provides various power-saving functions and consumes 0.1μA in Deep-Sleep Mode, 0.6μA in Sleep Mode, 70μA in Power-down Mode and less than 1mA in Active Mode.
- *High-Level Programming Language*: The MARC4 instruction set is optimized for the high-level programming language qForth.
- *Stack-based Architecture*: The MARC4 inherits a stack-based architecture where the RAM is used to construct the Expression Stack (EXP) and the Return Stack (RET), which are addressed with the Expression Stack Pointer (`SP`) and the Return Stack Pointer (`RP`), respectively. All arithmetic, I/O and memory reference operations take their operands *from*, and return their result *to* the EXP. Both EXP and RET have a user-definable maximum depth.

The development tool for programming on MARC4 microcontrollers is the MARC4 Starter Kit (see Figure 5 in [168]) containing the following components:

[2]Reduced Instruction Set Computing

- *Hardware Components*: an *E-Lab ICP V24 Portable* programmer, an MARC4 *Programming board*, a ready-to-run application board *TEMIC T4xCx92*.
- *Software Components*: a Windows-based *editor*, an integrated qForth *compiler*, an integrated *simple* MARC4 *core simulator* (only core, no peripheral modules), an integrated *Help Function* with qForth *dictionary* and an *Atmel-wm ICP* programmer software.

4-Bit Implementation of the Hummingbird Cryptographic Algorithm

The MARC4 is a zero address machine with a compact and efficient instruction set. The instructions contain only the operation to be performed but no source or destination address information. Therefore, it is a formidable task to implement the Hummingbird on the target platform since all operations and function calls must be organized and scheduled according to the stack-based architecture of the MARC4 core. Figure 2.3 depicts the hierarchical function call graph in the implementation of Hummingbird where 11 functions have been implemented on the ATAM893-D microcontroller. The implementation details of the functions in Figure 2.3 will be presented below using the qForth programming language [3].

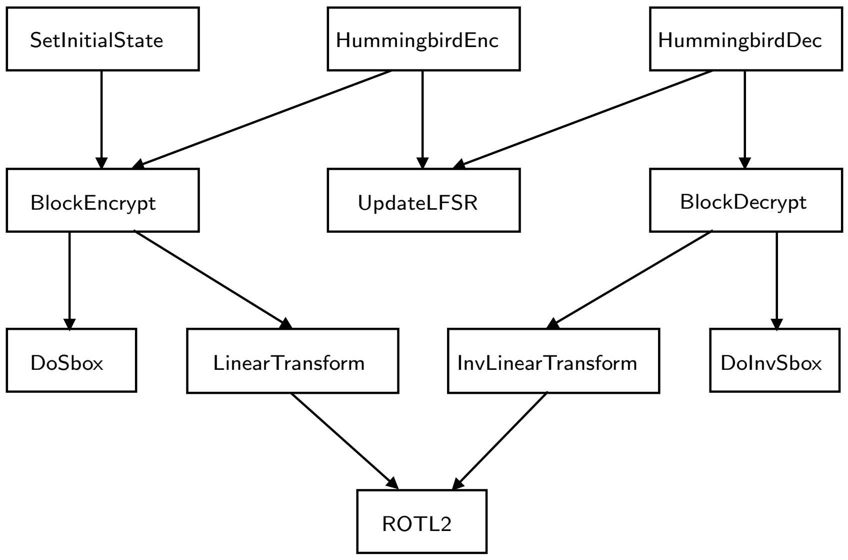


Figure 2.3: The Function Call Graph in Hummingbird Implementation

Both the Expression and Return Stacks are located in RAM, whose size is variable and must be defined by the programmer. By analyzing various operations in the Hummingbird

[3]To fully understand the code and its effects in this section, the reader is referred to [6].

algorithm and the function call graph in Figure 2.3, we define the maximum depth of the EXP and the RET to be 9 and 7, respectively. Moreover, the following variables are used in the implementation of Hummingbird:

```
Text0 -- Text3: 16-bit plaintext/ciphertext block;
RSi0  --  RSi3: 16-bit rotor RSi, i = 1, 2, 3, 4;
LFSR0 -- LFSR3: 16-stage LFSR;
V120  --  V123: 16-bit output of block cipher E_{k_1};
V230  --  V233: 16-bit output of block cipher E_{k_2};
V340  --  V343: 16-bit output of block cipher E_{k_3};
T0    --  T3  : temporary variables;
S0    --  S3  : temporary variables;
KeySelection  : 4-bit variable for the subkey selection.
```

Initializing the internal state registers of Hummingbird can be divided into three phases (see Figure 2.1(a)). These phases are 1) Initializing the rotor RSi with a randomly chosen nonce NONCE_i $(i = 1, 2, 3, 4)$, 2) Encrypting the message $RS1 \boxplus RS3$ using the four block ciphers consecutively and updating the states four rotors for four rounds, 3) Using the final 16-bit ciphertext as the initial state of LFSR with the 13th bit set. The initialization of the internal state registers is implemented by the following qForth word[4] **:SetInitialState** in which the qForth word **:BlockEncrypt** implements the 16-bit block cipher encryption.

```
: SetInitialState
  4 #DO                        \ Four rounds encryption and rotor stepping
    [...]
    CLR_BCF                    \ Clear BRANCH and CARRY flag
    V340 @ RS40 @ + Text0 !    \ Compute V34 + RS4
    [...]
    4 KeySelection !           \ Use the 4th 64-bit key k_4
    BlockEncrypt               \ 16-bit block cipher encryption
    Text0 @ T0 !               \ Store the ciphertext in T0 to T3
    [...]
    CLR_BCF                    \ Clear BRANCH and CARRY flag
    RS10 @ T0 @ + RS10 !       \ Step rotor RS1(t + 1) = RS1(t) + T
    [...]
  #LOOP
  T0 @ LFSR0 !                 \ Set the initial state of LFSR
  [...]
  T3 @ 0001b OR LFSR3 !        \ Set the 13th bit of LFSR
;
```

The encryption of one 16-bit plaintext block PT_i with a given 256-bit secret key K consists of two stages (see Figure 2.1(b)). PT_i is first encrypted using four block ciphers in a similar manner as the initialization of the internal state registers. Then the states of the four rotors and the LFSR will be updated in terms of the process in Figure 2.1(b). The encryption is implemented by the qForth word **:HummingbirdEnc** which calls two subwords **:BlockEncrypt** and **:UpdateLFSR**.

[4]A word in qForth is a synonym of a subroutine in other programming languages [6].

```
: HummingbirdEnc
  [...]
  CLR_BCF                     \ Clear BRANCH and CARRY flag
  V120 @ RS20 @ + Text0 !     \ Compute V12 + RS2
  [...]
  2 KeySelection !            \ Use the 2nd 64-bit key k_2
  BlockEncrypt                \ 16-bit block cipher encryption
  Text0 @ V230 !              \ Store the ciphertext in V230 to V233
  [...]
  CLR_BCF                     \ Clear BRANCH and CARRY flag
  RS30 @ V230 @ + RS30 !      \ Step rotor RS3(t + 1) = RS3(t) + V23 + LFSR
  [...]
  UpdateLFSR                  \ Step the Galois LFSR to the next state
  RS30 @ LFSR0 @ + RS30 !
  [...]
;
```

In Hummingbird, a 16-stage LFSR in Galois configuration (see Figure 2.4) is utilized because it offers minimal latency and higher speed for both software and hardware implementation. In the Galois configuration, when the state of LFSR needs to be updated, bits that are not taps are shifted one position to the right unchanged. The taps, on the other hand, are XORed with the output bit before they are stored in the next position. The state update of the Galois LFSR can be efficiently implemented with the qForth word **:UpdateLFSR** as follows:

```
: UpdateLFSR
  LFSR0 @ LFSR1 @ LFSR2 @ LFSR3 @     \ Put LFSR into the stack
  SHR Temp3 !                         \ LFSR >> 1
  [...]
  LFSR0 @ 0001b AND 0001b =           \ Test whether LFSR & 0x1 = 0x1
  IF                                  \ Yes, LFSR = (LFSR >> 1) ^ 0xca44
    [...]
    Temp3 @ 1100b XOR LFSR3 !
  ELSE                                \ No, LFSR = LFSR >> 1
    [...]
    Temp3 @ LFSR3 !
  THEN
;
```

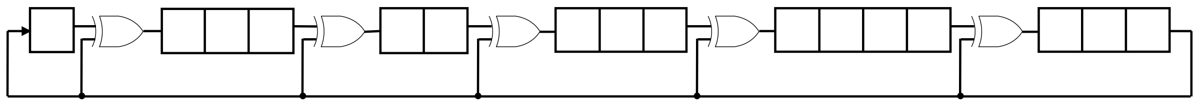


Figure 2.4: A 16-bit Galois LFSR with Characteristic Polynomial $f(x) = x^{16} + x^{15} + x^{12} + x^{10} + x^7 + x^3 + 1$

To implement the 16-bit block cipher encryption E_{k_i} as shown in Figure 2.2, the 4×4 S-box S_1 and the 64-bit subkey $k_i = K_1^{(i)} \| K_2^{(i)} \| K_3^{(i)} \| K_4^{(i)}$ are first stored in some specific areas of the ROM of the MARC4 core with the qForth command **ROMCONST** as follows:

```
\ 16 nibbles for storing S-BOX used in the Compact Hummingbird
ROMCONST SBOX 8 , 6 , 5 , 15 , 1 , 12 , 10 , 9 ,
             14 , 11 , 2 , 4 , 7 , 0 , 13 , 3 , AT 340h

\ 64-bit subkey k_1 and two derived round keys used in block cipher E_{k_1}
ROMCONST Key1 7eh , 2bh , 16h , 15h , aeh , 28h , a6h , d2h ,
              d0h , 03h , b0h , c7h , AT 360h
[...]
```

Note that two derived round keys $K_5^{(i)} = K_1^{(i)} \oplus K_3^{(i)}$ and $K_6^{(i)} = K_2^{(i)} \oplus K_4^{(i)}$ are also stored in the ROM for efficient implementation. Moreover, the qForth command **ROMByte@** is used to fetch an 8-bit constant from the ROM each time. The four regular rounds of the encryption algorithm can be implemented in a similar way. More specifically, after a simple key mixing step, a message is divided into four 4-bit chunks and each of them is then substituted by a single 4×4 S-box (see **:DoSbox** below). Subsequently, the 16-bit message is further permuted by the linear transform L (see **:LinearTransform** below). Unlike the regular round, the final round only consists of two key mixing steps with round keys $K_5^{(i)}$ and $K_6^{(i)}$ and an S-box substitution step.

```
: BlockEncrypt
  KeySelection @            \ Select a 64-bit subkey
  CASE
    1 OF
      Key1 ROMByte@         \ Fetch the 1st 8-bit of k_1
      Text0 @ XOR Text0 !   \ Add round key
      Text1 @ XOR Text1 !
      [...]
      DoSbox                \ S-box substitution
      LinearTransform       \ Linear transform Text ^ (Text << 6) ^ (Text << 10)
      [...]
    ENDOF
    [...]
  ENDCASE
;
```

The S-box substitution can be efficiently implemented by first pushing the base address (12 bits) of the look-up table into the EXP. The value of the 4-bit chunk is then used as the offset to form the address of the substitution result.

```
: DoSbox
  SBOX Text0 @ +           \ Look-up table addressing
  ROMByte@                 \ S-box substitution
  Text0 !                  \ Store SBOX[Text0] to Text0
  DROP                     \ Drop high nibble which is 0
  [...]
;
```

The linear transform L involves rotating a 16-bit message by 6 and 10 positions to the left, respectively, which is performed by two steps: 1) rotate the message by 4 and 8 positions to the left, respectively; 2) rotate the resulting message by 2 positions to the left. For the MARC4 4-bit architecture, the first step can be easily implemented by re-addressing memory pointers, which is equivalent to change the order that a 16-bit message is pushed into the EXP, as shown in the following qForth word **:LinearTransform**.

```
: LinearTransform
  Text2 @ Text1 @ Text0 @ Text3  \ Push |Text0|Text1|Text2|Text3| onto stack << 4
  ROTL2                          \ T = Text << 6
  Text0 @ T0 @ XOR S0 !          \ S = Text ^ (Text << 6)
  [...]
  Text1 @ Text0 @ Text3 @ Text2  \ Push |Text0|Text1|Text2|Text3| onto stack << 8
  ROTL2                          \ T = Text << 10
  S0 @ T0 @ XOR Text0 !          \ Text = Text ^ (Text << 6) ^ (Text << 10)
  [...]
;
```

The second step is implemented with the qForth word **:ROTL2** (see below), which continuously rotates a 16-bit message by one position to the left twice with the assistance of the MARC4 Conditional Code Register (CCR).

```
: ROTL2
  SHL T3 !                \ Rotate a 16-bit message by one position to the left
  ROL T2 !                \ Text3 << 1
  ROL T1 !                \ Text2 << 1
  ROL T0 !                \ Text1 << 1
  0111b CCR@ <            \ Test whether there is a carry
  IF T3 @ 1 XOR T3 !      \ If it is, insert 0001b into T3
  ELSE T3 @ 0 XOR T3 !    \ Timing-attack resistance
  THEN
  [...]
;
```

The decryption procedure of Hummingbird is quite similar to the encryption process (see Figure 2.1(c)). From the function call graph in Figure 2.3, we note that both encryption and decryption share the same qForth words **:UpdateLFSR** and **:ROTL2**, which further reduces the code size and enables us to fit an implementation that is capable of encryption and decryption in spite of the harsh memory constrains of the MARC4 controllers (i.e., 4KBytes ROM and 128Bytes RAM). Due to the similarity of the encryption and decryption, we omit the description of the implementation of the decryption process here.

Experimental Results and Comparisons

To the best of our knowledge, the only published implementation of cryptographic algorithms on 4-bit microcontrollers is about the ultra-lightweight block cipher PRESENT [168]. Hence, we compare the performance of the optimized implementations of PRESENT and Hummingbird here. Table 2.5 summarizes the memory consumption and cycle count of two ciphers on the target 4-bit platform. From Table 2.5, we note that Hummingbird encryption and decryption algorithms require roughly equal lines of code[5], which is longer than those of PRESENT. The reason is that the loop unrolling technique is employed to optimize the performance of the algorithm in our implementation, which increases the code size significantly. Moreover, both Hummingbird and PRESENT use almost the same number of RAM nibbles as the stack space[6]. In addition, the design of Hummingbird is based on a hybrid mode of block cipher and stream cipher, which results in a relatively long initialization process when compared to the block cipher PRESENT. After the initialization phase, the encryption of a 16-bit plaintext block with Hummingbird requires 5,773 cycles, whereas it takes 55,734 cycles for PRESENT to encrypt 64-bit plaintext. Furthermore, the decryption of one ciphertext block of 16/64 bits with Hummingbird/PRESENT requires 5,212 and 65,574 cycles on the target 4-bit microcontroller, respectively.

Table 2.5: Memory Consumption and Cycle Count Comparison

encryption/ decryption	ROM [lines of code]	Stack Depth [EXP/RET]	Init. [cycles]	Cycles/block [cycles]
PRESENT-enc [168]	841	25/4	230	55,734
PRESENT-dec [168]	945	25/4	230	65,574
Hummingbird-enc	1532	9/7	22,949	5,773
Hummingbird-dec	1559	9/7	22,949	5,212

Table 2.6 estimates the running time and throughput of PRESENT and Hummingbird on the 4-bit ATAM893-D microcontroller clocked at 16KHz, 500KHz and 2MHz, respectively. As one can see from Table 2.6 that after an initialization phase the Hummingbird encryption and decryption algorithms can achieve nearly 1.41 and 2.14 times faster throughput than the state-of-the-art ultra-lightweight block cipher PRESENT at each given clock frequency, respectively.

Combining the cost of the system initialization, Table 2.7 and Table 2.8 describe the overall performance of PRESENT and Hummingbird for encrypting and decrypting messages with different length on the target 4-bit microcontroller, respectively. From Tables 2.7 and 2.8, we note that the longer the message length, the larger performance improvement of Hummingbird over PRESENT. Moreover, for typical embedded applications where most of transmitted messages are usually very short, the advantage of using Hummingbird instead of PRESENT is self-evident as well.

[5]We count the lines of code twice for the qForth words shared by the Hummingbird encryption and decryption algorithms.

[6]While the depth (in RAM nibbles) of the EXP is exactly the number allocated, the RET depth is expressed as $\text{RET}_{\text{value}} = \text{RET}_{\text{depth}} \times 4$.

Table 2.6: Timing and Throughput Comparison at Three Clock Frequencies

encryption/ decryption	Clock Freq. [KHz]	Clock Source [int./ext.]	Timing [ms]	Throughput [bits/sec]
PRESENT-enc [168]	2,000	int.	27.9	2,297
	500	ext.	111.5	574
	16	ext.	3,483	18.4
PRESENT-dec [168]	2,000	int.	32.8	1,952
	500	ext.	131.1	488
	16	ext.	4,098	15.6
Hummingbird-enc	2,000	int.	2.89	5,543
	500	ext.	11.55	1,385.8
	16	ext.	360.8	44.3
Hummingbird-dec	2,000	int.	2.61	6,139.7
	500	ext.	10.4	1,534.9
	16	ext.	325.8	49.1

Table 2.7: The Overall Encryption Performance Comparison

Message Length	Clock Freq. [KHz]	PRESENT-enc [168] [ms]	Hummingbird-enc [ms]	Performance Improvement
64-bit	2,000	28.02	23.03	17.8%
	500	111.96	92.1	
	16	3,497.38	2,877.51	
128-bit	2,000	55.92	34.59	38.1%
	500	223.46	138.3	
	16	6,980.38	4,320.71	
192-bit	2,000	83.82	46.15	44.9%
	500	334.96	184.5	
	16	10,463.38	5,763.91	

2.4.2 Software Implementation on a 8-Bit Microcontroller

The ATmega128L from Atmel is a high-performance and low-power microcontroller that has been widely deployed in wireless sensor nodes MICAz for various applications requiring an extremely low power consumption for extended battery life. In this subsection, we report efficient implementation of Hummingbird on the ATmega128L microcontroller and compare the performance of Hummingbird and PRESENT on similar platforms.

Table 2.8: The Overall Decryption Performance Comparison

Message Length	Clock Freq. [KHz]	PRESENT-dec [168] [ms]	Hummingbird-dec [ms]	Performance Improvement
64-bit	$2,000$	32.92	21.91	33.4%
	500	131.56	87.5	
	16	$4,112.38$	$2,737.51$	
128-bit	$2,000$	65.72	32.35	50.8%
	500	262.66	129.1	
	16	$8,210.38$	$4,040.71$	
192bit	$2,000$	98.52	42.79	56.6%
	500	393.76	170.7	
	16	$12,308.38$	$5,343.91$	

8-Bit Microcontroller ATmega128L and Development Tools

The 8-bit ATmega128L microcontroller features the AVR enhanced RISC architecture. The processor is equipped with 133 powerful and highly-optimized instructions and most of them can be executed with one clock cycle. Moreover, ATmega128L comes with 128 KBytes of In-System Self-Programmable Flash, 4 KBytes EEPROM and 8 KBytes Internal SRAM. Optionally it can handle up to 64 KBytes of external memory space. Its clock frequency can run from 0 to 8 MHz and the power supplies can go from 2.7 to 5.5 V. In addition, at a frequency of 4 MHz with a power supply of 5 V ATmega128L draws 5.5 mA current when active, 2.5 mA in Idle Mode and less than 15 μA in Power-down Mode.

In order to implement and test the performance of Hummingbird on the target platform, we use a combination of the integrated development environment AVR Studio 4.17 [5] from Atmel and the open-source WinAVR-20090313 tool kit [171] for our purpose. While AVR Studio is used as an editor and a simulator, the WinAVR provides a GNU GCC compiler with the according libraries and a linker.

Size Optimized Implementation on the Target 8-bit Platform

Note that the final round of the 16-bit block cipher in Hummingbird requires two derived round keys $K_5^{(i)} = K_1^{(i)} \oplus K_3^{(i)}$ and $K_6^{(i)} = K_2^{(i)} \oplus K_4^{(i)}$ (see Figure 2.2). For a size optimized implementation it is wise to calculate the above two keys $K_5^{(i)}$ and $K_6^{(i)}$ on-the-fly, which can save the storage requirements by 16 bytes. Moreover, the single S-box is implemented as a byte array with 16 elements, in which the lower half of a byte is used to store the value of the Hummingbird S-box and the higher half of a byte is padded with zeros. The S-box look-up of 16-bit block is conducted sequentially and 4 bits are processed each time.

To generate the code with minimal size on the ATmega128L microcontroller, we set the optimization level to be "OPT = s" for GCC compiler in WinAVR-20090313.

Performance Results for Size Optimized Implementation

Table 2.9 summarizes the memory consumption and cycle count of two ultra-lightweight ciphers Hummingbird and PRESENT on 8-bit microcontrollers for the size optimized implementation.

Table 2.9: Memory Consumption and Cycle Count Comparison (Size Optimized Implementation on 8-bit Microcontrollers)

Cipher	Key Size [bit]	Block Size [bit]	8-bit Micro-controller	Flash Size [bytes]	Hex Code Size [Kbytes]	SRAM Size [bytes]	Init. [cycles]	Enc. [cycles/ block]	Dec. [cycles/ block]
Hummingbird	256	16	ATmega128L	1,308	3.68	0	14,735	3,664	3,868
PRESENT [144]	80	64	ATmega163	1,474	–	32	–	646,166	634,614

From Table 2.9 we note that the code size of Hummingbird is about 13% smaller than that of PRESENT on the 8-bit microcontrollers. In addition, Hummingbird needs a relatively long initialization process when compared to the block cipher PRESENT because of the hybrid structure of block cipher and stream cipher adopted in Hummingbird. However, after an initialization procedure, Hummingbird encryption algorithm can achieve the throughput of 17.5 Kbps at a frequency of 4 MHz on the 8-bit microcontroller. Under the same settings, the throughput of Hummingbird decryption algorithm can amount to 16.5 Kbps. Note that the throughput of PRESENT encryption and decryption algorithms is only 0.396 Kbps and 0.403 Kbps in this case. Therefore, for the size optimized implementation, the throughput of Hummingbird is about 40 times faster than that of PRESENT on the target 8-bit platforms. Considering the cost of the initialization phase in Hummingbird, we compare the overall performance of Hummingbird and PRESENT for encrypting and decrypting messages with different length in the following Tables 2.10 and 2.11, respectively.

Table 2.10: The Overall Encryption Performance Comparison at 4 MHz (Size Optimized Implementation on 8-bit Microcontrollers)

Message Length	PRESENT-enc [144] [ms]	Hummingbird-enc [ms]	Performance Improvement
64-bit	161.54	7.35	95.5%
128-bit	323.08	11.01	96.6%
192-bit	484.62	14.68	96.9%

Table 2.11: The Overall Decryption Performance Comparison at 4 MHz (Size Optimized Implementation on 8-bit Microcontrollers)

Message Length	PRESENT-dec [144] [ms]	Hummingbird-dec [ms]	Performance Improvement
64-bit	158.65	7.55	95.2%
128-bit	317.31	11.42	96.4%
192-bit	475.96	15.29	96.8%

For the size optimized implementation, Tables 2.10 and 2.11 show that one can achieve around 95.2% $\sim$ 96.9% performance improvements when using Hummingbird instead of PRESENT to encrypt or decrypt message blocks with length 64-bit, 128-bit, and 192-bit.

Speed Optimized Implementation on the Target 8-bit Platform

For a speed optimized implementation, we precompute and store all required round keys $K_5^{(i)}$ and $K_6^{(i)}$ (see Figure 2.2) in an array and this precomputation procedure requires additional 16 bytes of data memory and has to done once when a new key is used. Furthermore, in order to accelerate the implementation of S-box layer in Hummingbird, we use a more efficient technique that combines two identical 4×4 S-boxes $S(x)$'s to form a larger 8×8 S-box $S_{8\times8}(x)$ such that $S_{8\times8}(x_1 \| x_2) = S(x_1) \| S(x_2)$, where x_1 and x_2 are 4-bit inputs to the two 4×4 S-boxes $S(x)$'s, respectively. Using the S-box $S_{8\times8}(x)$ significantly reduces the time for the S-box loop-up at the cost of 512 bytes of data memory (Note that both $S_{8\times8}(x)$ and $S_{8\times8}^{-1}(x)$ have 256 entries of each one byte). To generate the code with maximal speed, we set the optimization level to be "OPT = 3" for GCC compiler in WinAVR-20090313.

Performance Results for Speed Optimized Implementation

Table 2.12 summarizes the memory consumption and cycle count of two ultra-lightweight ciphers Hummingbird and PRESENT on 8-bit microcontrollers for the speed optimized implementation.

From Table 2.12 we note that the code size of Hummingbird is about 78% larger than that of PRESENT on the 8-bit microcontrollers. The main reason is that the -O3 option of the GCC compiler aggressively optimizes for speed by unrolling all loops in the code, which drastically increase the size of the code. Assuming that the microcontrollers operate at the frequency of 4 MHz, Hummingbird encryption algorithm can achieve the throughput of 45.7 Kbps on the 8-bit microcontroller, which is about 71.3% faster than that of PRESENT

Table 2.12: Memory Consumption and Cycle Count Comparison (Speed Optimized Implementation on 8-bit Microcontrollers)

Cipher	Key Size [bit]	Block Size [bit]	8-bit Micro-controller	Flash Size [bytes]	Hex Code Size [Kbytes]	SRAM Size [bytes]	Init. [cycles]	Enc. [cycles/block]	Dec. [cycles/block]
Hummingbird	256	16	ATmega128L	10,918	30.5	0	8,182	1,399	1,635
PRESENT [144]	80	64	ATmega163	2,398	–	528	–	9,595	9,820

on the similar platform. Based on the same assumption, the throughput of Hummingbird decryption algorithm can amount to 39.1 Kbps on the 8-bit microcontroller, which is around 50% faster than that of PRESENT on the similar platform. Combining the overhead of the initialization phase in Hummingbird, we compare the overall performance of Hummingbird and PRESENT for encrypting and decrypting messages with different length in the following Tables 2.13 and 2.14 for the speed optimized implementation.

Table 2.13: The Overall Encryption Performance Comparison at 4 MHz (Speed Optimized Implementation on 8-bit Microcontrollers)

Message Length	PRESENT-enc [144] [ms]	Hummingbird-enc [ms]	Performance Improvement
64-bit	2.40	3.38	-28.9%
128-bit	4.80	4.72	1.7%
192-bit	7.20	6.06	15.8%

Table 2.14: The Overall Decryption Performance Comparison at 4 MHz (Speed Optimized Implementation on 8-bit Microcontrollers)

Message Length	PRESENT-dec [144] [ms]	Hummingbird-dec [ms]	Performance Improvement
64-bit	2.46	3.68	-33.2%
128-bit	4.92	5.32	-7.5%
192-bit	7.38	6.95	5.8%

For the speed optimized implementation, Tables 2.13 shows that on 8-bit microcontrollers Hummingbird encryption is about 28.9% slower than PRESENT encryption when the message length is 64 bits. Furthermore, Hummingbird decryption (see Table 2.14) is about 33.2% and 7.5% slower than PRESENT decryption for messages with length 64-bit and 128-bit, respectively. The main reason is that Hummingbird has a hybrid structure of block cipher and stream cipher which involves a relatively long initialization process when compared to the block cipher PRESENT.

2.4.3 Software Implementation on a 16-Bit Microcontroller

The MSP430 family of microcontrollers from Texas Instrument (TI) has widespread applications in the embedded system community due to their extremely low power consumption. In particular, the 16-bit MSP430F1611 has been integrated into wireless sensor nodes TELOSB and TMote Sky for energy constrained wireless applications. This subsection addresses efficient implementation of Hummingbird on the MSP430F1611 microcontroller and compare the performance of Hummingbird and PRESENT on similar platforms.

16-Bit Microcontroller MSP430 and Development Tools

The 16-bit microcontroller MSP430F1611 is different in many ways from the Atmel chip. It has a traditional *von-Neumann* architecture and all special function registers (SFRs), peripherals, RAM, and Flash/ROM share the same address space. Moreover, it comes with 48 KBytes Flash memory and 10 KBytes RAM. The clock frequency of MSP430F1611 ranges from 0 to 8 MHz and the power supplies can go from 1.8 to 3.6 V. In particular, the MSP430F1611 microcontroller features the ultralow power consumption. At a frequency of 1 MHz and a voltage supply of 2.2 V the chip draws 200 μA current in Active Mode, 0.7 μA in Real-time Clock Mode, and 0.1 μA in Off Mode (RAM Retention). Although the instruction set of MSP430F1611 only contains 27 instructions, 7 different addressing modes provide great flexibility in data manipulation.

We use CrossWorks for MSP430 Version 2 from Rowley Associates [149] to implement and simulate Hummingbird on the target platform. The CrossWorks for MSP430 bundles an ANSI C compiler, macro assembler, linker/locator, libraries, core simulator, flash downloader, JTAG debugger, and an integrated development environment CrossStudio. Different optimization levels can be set to generate codes with either smallest size or fastest speed.

Size Optimized Implementation on the Target 16-bit Platform

For a size optimized implementation we take the strategy of computing two round keys $K_5^{(i)}$ and $K_6^{(i)}$ on-the-fly again in order to save 16 bytes of storage. Similar to the code size optimized 8-bit implementation, the single S-box is implemented as a byte array with 16 elements and the S-box look-up for a 16-bit data block is realized sequentially for both the encryption and the decryption routines. Moreover, to generate the code with minimal size, we also choose "Minimize Size" as the optimization strategy in CrossStudio.

Performance Results for Size Optimized Implementation

Table 2.15 summarizes the memory consumption and cycle count of two ultra-lightweight ciphers Hummingbird and PRESENT on 16-bit microcontrollers for the size optimized implementation.

Table 2.15: Memory Consumption and Cycle Count Comparison (Size Optimized Implementation on 16-bit Microcontrollers)

Cipher	Key Size [bit]	Block Size [bit]	16-bit Micro-controller	Flash Size [bytes]	Hex Code Size [Kbytes]	SRAM Size [bytes]	Init. [cycles]	Enc. [cycles/block]	Dec. [cycles/block]
Hummingbird	256	16	MSP430F1611	1,064	2.95	0	9,667	2,414	2,650
PRESENT [144]	80	64	C167CR	–	9.67	–	–	1,442,556	1,332,062

From Table 2.15 one can find that the code size of Hummingbird is about 69% smaller than that of PRESENT on the 16-bit microcontrollers. In addition, after an initialization process, Hummingbird encryption and decryption algorithms can achieve the throughput of 26.5 Kbps and 24.2 Kbps at a frequency of 4 MHz on the 16-bit microcontrollers, respectively. As a result, for the size optimized implementation, the throughput of Hummingbird encryption and decryption algorithms is about 148 and 124 times faster than that of PRESENT on the target 16-bit platforms, respectively. Considering the cost of the initialization phase in Hummingbird, we compare the overall performance of Hummingbird and PRESENT for encrypting and decrypting messages with different length in the following Tables 2.16 and 2.17.

Table 2.16: The Overall Encryption Performance Comparison at 4 MHz (Size Optimized Implementation on 16-bit Microcontrollers)

Message Length	PRESENT-enc [144] [ms]	Hummingbird-enc [ms]	Performance Improvement
64-bit	360.64	4.83	98.7%
128-bit	721.28	7.24	98.9%
192-bit	1,081.92	9.66	99.1%

For the size optimized implementation, Tables 2.16 and 2.17 show that about 98.5% to 99.1% performance improvements can be achieved by using Hummingbird instead of PRESENT to encrypt or decrypt message blocks with length 64-bit, 128-bit, and 192-bit.

Table 2.17: The Overall Decryption Performance Comparison at 4 MHz (Size Optimized Implementation on 16-bit Microcontrollers)

Message Length	PRESENT-dec [144] [ms]	Hummingbird-dec [ms]	Performance Improvement
64-bit	333.02	5.07	98.5%
128-bit	666.03	7.72	98.8%
192-bit	1,081.92	9.66	99.1%

Speed Optimized Implementation on the Target 16-bit Platform

For a speed optimized implementation, we precompute the required round keys $K_5^{(i)}$ and $K_6^{(i)}$ at a cost of additional 16 bytes of storage. Similar to the speed optimized 8-bit implementation, S-box layer is implemented by combining two small 4×4 S-boxes into a larger 8×8 S-box, which needs 512 bytes of data memory. To generate the code with maximal speed, we also choose "Maximize Speed" as the optimization strategy in CrossStudio.

Performance Results for Speed Optimized Implementation

Table 2.18 summarizes the memory consumption and cycle count of Hummingbird and PRESENT on 16-bit microcontrollers for the speed optimized implementation.

Table 2.18: Memory Consumption and Cycle Count Comparison (Speed Optimized Implementation on 16-bit Microcontrollers)

Cipher	Key Size [bit]	Block Size [bit]	16-bit Micro-controller	Flash Size [bytes]	Hex Code Size [Kbytes]	SRAM Size [bytes]	Init. [cycles]	Enc. [cycles/ block]	Dec. [cycles/ block]
Hummingbird	256	16	MSP430F1611	1,360	3.76	0	4,824	1,220	1,461
PRESENT [144]	80	64	C167CR	–	92.2	–	–	19,464	33,354

From Table 2.18 one can find that the code size of Hummingbird is about 96% smaller than that of PRESENT on the 16-bit microcontrollers. Assuming that the microcontrollers operate at the frequency of 4 MHz, Hummingbird encryption and decryption algorithms can achieve the throughput of 52.5 Kbps and 43.8 Kbps on the 16-bit microcontrollers, respectively, after a initialization procedure. Therefore, for the speed optimized implementation, the throughput of Hummingbird encryption and decryption algorithms is about 2.5 and 4.7 times faster than that of PRESENT on the similar platforms, respectively. Combining the overhead of the initialization phase in Hummingbird, we compare the overall performance of Hummingbird and PRESENT for encrypting and decrypting messages with different length in the following Tables 2.19 and 2.20 for the speed optimized implementation.

Table 2.19: The Overall Encryption Performance Comparison at 4 MHz (Speed Optimized Implementation on 16-bit Microcontrollers)

Message Length	PRESENT-enc [144] [ms]	Hummingbird-enc [ms]	Performance Improvement
64-bit	4.87	2.43	50.1%
128-bit	9.68	3.65	62.3%
192-bit	14.61	4.87	66.7%

Table 2.20: The Overall Decryption Performance Comparison at 4 MHz (Speed Optimized Implementation on 16-bit Microcontrollers)

Message Length	PRESENT-dec [144] [ms]	Hummingbird-dec [ms]	Performance Improvement
64-bit	8.34	2.67	67.9%
128-bit	16.68	4.13	75.2%
192-bit	25.02	5.59	77.6%

For the speed optimized implementation and different length of messages, Table 2.19 shows that on 16-bit microcontrollers Hummingbird encryption is about 50.1% to 66.7% faster than PRESENT encryption, whereas Hummingbird decryption is around 67.9% to 77.6% more efficient than that of PRESENT in our experiment. The main reason is that the size (i.e., 16 bits) of the data block and the internal state registers in Hummingbird is perfectly suited to the architecture of 16-bit microcontrollers.

2.5 Efficient Hardware Implementations on Low-Cost FPGAs

Besides having good performance and low memory consumption on various embedded processors, Hummingbird is also expected to achieve small footprint and high throughput in different hardware platforms. This section describes efficient FPGA implementations of a stand-alone Hummingbird component. We implement an encryption only core and an encryption/decryption core on the low-cost Xilinx FPGA series Spartan-3 and compare our results with other reported (ultra-)lightweight block cipher implementations on the same series. Moreover, a speed-optimized and an area-optimized hardware architectures are also described in this section. Note that the choice of different kinds of I/O interfaces has a significant influence on the performance of hardware implementation and is highly application specific. Therefore, we do not implement any specific I/O logic in order to obtain the accurate performance profile of a plain Hummingbird encryption/decryption core as well as provide enough flexibility for various applications. Again, only the compact version of Hummingbird (i.e., a single 4-bit S-box $S(x)$ is applied 4 times in parallel in each round of the 16-bit block cipher) is chosen for implementation on the target platform in order to reduce the required hardware resources and to achieve better performance.

2.5.1 Target Platform and Design Tools

FPGAs are composed of configurable logic blocks (CLB) and a programmable interconnection network. We implement both encryption and decryption modules in VHDL for the low-cost Spartan-3 XC3S200 (Package FT256 with speed grade -5) FPGA device from Xilinx[7] [174]. We use the integrated FPGA development environment Aldec Active-HDL 8.2sp1 for writing, debugging and simulating VHDL codes. Furthermore, Synopsys Synplify Pro C-2009.06-SP1 and Xilinx ISE Design Suite v11.1 are employed for the design synthesis and implementation, respectively.

2.5.2 Selection of a "Hardware-Friendly" S-Box

A "hardware-friendly" S-box is the S-box that can be efficiently implemented in the target hardware platform with a small area requirement. Four 4×4 S-boxes $S_i(x) : \mathbb{F}_2^4 \rightarrow \mathbb{F}_2^4$ ($i = 1, 2, 3, 4$) have been carefully selected in Hummingbird according to certain security criteria (see Section 2.2.4). To implement the compact version of Hummingbird, we need to choose a "hardware-friendly" S-box from four S-boxes listed in Table 2.2. Let $x = (x_3 \| x_2 \| x_1 \| x_0)$ be the 4-bit input to the S-box and let $S_i(x) = (S_i^{(3)}(x) \| S_i^{(2)}(x) \| S_i^{(1)}(x) \| S_i^{(0)}(x))$ denote the 4-bit output of the i-th S-box ($i = 1, 2, 3, 4$). By using the Boolean minimization tool *Espresso* [58] we can obtain the following minimal Boolean function representations (BFR) for the four S-boxes in Hummingbird as shown in Table 2.21, where $\overline{x}_i$ denotes the inversion of bit x_i, $\cdot$ denotes a logical AND and $+$ denotes a logical OR. Note that each S-box can be implemented in hardware by using either a look-up table (LUT) or the Boolean function representations (i.e., combinatorial logic). The exact efficiency of the above two approaches significantly depends on specific hardware platforms and synthesis tools. Therefore, for each proposed architecture of the 16-bit block cipher we investigate two implementation strategies (i.e., BRR and LUT) for the four S-boxes in Sections 2.5.3 and 2.5.4, respectively, and select one that results in the most area-efficient implementation of the 16-bit block cipher.

[7]Basic building blocks of Xilinx FPGA are CLB slices and each slice on a Spartan-3 device contains two sets of a look-up table (LUT) followed by a flip-flop.

Table 2.21: Boolean Function Representations for S-boxes in Hummingbird

S-boxes	Minimal Boolean Function Representations
$S_1(x)$	$S_1^{(0)}(x) = x_3 \cdot \overline{x}_2 \cdot \overline{x}_1 \cdot x_0 + \overline{x}_3 \cdot \overline{x}_2 \cdot x_1 + x_3 \cdot x_2 \cdot x_1 \cdot x_0 + x_2 \cdot \overline{x}_1 \cdot \overline{x}_0 + x_3 \cdot x_2 \cdot x_1 \cdot \overline{x}_0 + \overline{x}_3 \cdot x_1 \cdot x_0$
	$S_1^{(1)}(x) = x_3 \cdot \overline{x}_2 \cdot \overline{x}_1 \cdot x_0 + x_3 \cdot x_2 \cdot x_1 \cdot x_0 + \overline{x}_3 \cdot x_2 \cdot x_1 \cdot \overline{x}_0 + x_3 \cdot \overline{x}_2 \cdot \overline{x}_0 + \overline{x}_3 \cdot \overline{x}_2 \cdot x_0 + x_3 \cdot \overline{x}_1 \cdot \overline{x}_0$
	$S_1^{(2)}(x) = \overline{x}_3 \cdot \overline{x}_2 \cdot x_1 + \overline{x}_2 \cdot x_1 \cdot x_0 + x_3 \cdot x_2 \cdot x_1 \cdot \overline{x}_0 + \overline{x}_3 \cdot \overline{x}_2 \cdot x_0 + x_3 \cdot \overline{x}_1 \cdot \overline{x}_0 + \overline{x}_3 \cdot x_2 \cdot \overline{x}_1 \cdot x_0$
	$S_1^{(3)}(x) = x_3 \cdot \overline{x}_2 \cdot \overline{x}_1 \cdot x_0 + \overline{x}_3 \cdot x_2 \cdot x_1 \cdot \overline{x}_0 + \overline{x}_2 \cdot \overline{x}_1 \cdot \overline{x}_0 + x_3 \cdot x_2 \cdot x_1 \cdot \overline{x}_0 + \overline{x}_3 \cdot x_1 \cdot x_0 + \overline{x}_3 \cdot x_2 \cdot \overline{x}_1 \cdot x_0$
$S_2(x)$	$S_2^{(0)}(x) = \overline{x}_3 \cdot x_2 \cdot \overline{x}_1 \cdot x_0 + x_3 \cdot \overline{x}_2 \cdot x_1 \cdot \overline{x}_0 + x_3 \cdot x_2 \cdot x_1 \cdot x_0 + \overline{x}_3 \cdot \overline{x}_2 \cdot x_0 + x_2 \cdot \overline{x}_1 \cdot \overline{x}_0 + x_3 \cdot \overline{x}_1 \cdot \overline{x}_0$
	$S_2^{(1)}(x) = \overline{x}_3 \cdot \overline{x}_2 \cdot x_1 \cdot \overline{x}_0 + \overline{x}_3 \cdot \overline{x}_2 \cdot \overline{x}_1 \cdot x_0 + \overline{x}_3 \cdot x_2 \cdot x_0 + x_3 \cdot \overline{x}_2 \cdot \overline{x}_1 \cdot x_0 + x_3 \cdot \overline{x}_2 \cdot x_1 \cdot x_0 + x_3 \cdot \overline{x}_1 \cdot \overline{x}_0$
	$S_2^{(2)}(x) = \overline{x}_3 \cdot \overline{x}_2 \cdot x_1 \cdot \overline{x}_0 + \overline{x}_3 \cdot \overline{x}_2 \cdot \overline{x}_1 \cdot x_0 + x_3 \cdot \overline{x}_2 \cdot x_1 \cdot x_0 + x_3 \cdot x_1 \cdot \overline{x}_0 + x_2 \cdot \overline{x}_1 \cdot \overline{x}_0 + x_3 \cdot x_2 \cdot \overline{x}_1$
	$S_2^{(3)}(x) = \overline{x}_3 \cdot x_2 \cdot \overline{x}_1 \cdot x_0 + x_3 \cdot \overline{x}_2 \cdot x_1 \cdot \overline{x}_0 + x_3 \cdot x_2 \cdot x_1 \cdot x_0 + \overline{x}_3 \cdot x_1 \cdot \overline{x}_0 + x_3 \cdot \overline{x}_2 \cdot \overline{x}_1 \cdot x_0 + x_3 \cdot x_2 \cdot \overline{x}_1$
$S_3(x)$	$S_3^{(0)}(x) = x_3 \cdot x_2 \cdot x_1 \cdot \overline{x}_0 + x_2 \cdot \overline{x}_1 \cdot x_0 + x_3 \cdot \overline{x}_2 \cdot \overline{x}_1 \cdot \overline{x}_0 + \overline{x}_3 \cdot x_1 \cdot \overline{x}_0 + x_3 \cdot x_2 \cdot x_0 + \overline{x}_3 \cdot \overline{x}_2 \cdot x_1$
	$S_3^{(1)}(x) = x_3 \cdot x_2 \cdot \overline{x}_1 \cdot x_0 + x_3 \cdot x_2 \cdot x_1 \cdot \overline{x}_0 + \overline{x}_3 \cdot x_2 \cdot x_1 \cdot x_0 + \overline{x}_3 \cdot \overline{x}_2 \cdot \overline{x}_1 + x_3 \cdot \overline{x}_2 \cdot \overline{x}_1 \cdot \overline{x}_0 + \overline{x}_2 \cdot x_1 \cdot \overline{x}_0$
	$S_3^{(2)}(x) = \overline{x}_3 \cdot x_2 \cdot \overline{x}_1 \cdot \overline{x}_0 + \overline{x}_2 \cdot \overline{x}_1 \cdot x_0 + \overline{x}_2 \cdot x_1 \cdot \overline{x}_0 + x_3 \cdot x_2 \cdot x_0 + \overline{x}_3 \cdot \overline{x}_2 \cdot x_1$
	$S_3^{(3)}(x) = \overline{x}_3 \cdot \overline{x}_2 \cdot \overline{x}_1 \cdot x_0 + \overline{x}_3 \cdot x_2 \cdot \overline{x}_1 \cdot \overline{x}_0 + \overline{x}_3 \cdot x_2 \cdot x_1 \cdot x_0 + x_3 \cdot x_1 \cdot x_0 + x_3 \cdot \overline{x}_2 \cdot \overline{x}_1 \cdot \overline{x}_0 + \overline{x}_3 \cdot x_1 \cdot \overline{x}_0$
$S_4(x)$	$S_4^{(0)}(x) = x_3 \cdot \overline{x}_2 \cdot \overline{x}_1 \cdot \overline{x}_0 + x_2 \cdot x_1 \cdot x_0 + \overline{x}_3 \cdot \overline{x}_2 \cdot x_1 \cdot \overline{x}_0 + x_3 \cdot \overline{x}_2 \cdot x_1 \cdot x_0 + x_3 \cdot x_2 \cdot x_1 \cdot \overline{x}_0 + \overline{x}_3 \cdot \overline{x}_1 \cdot x_0$
	$S_4^{(1)}(x) = \overline{x}_3 \cdot \overline{x}_2 \cdot x_1 \cdot \overline{x}_0 + x_3 \cdot x_2 \cdot \overline{x}_1 \cdot \overline{x}_0 + x_3 \cdot \overline{x}_2 \cdot x_1 \cdot x_0 + \overline{x}_2 \cdot \overline{x}_1 \cdot x_0 + \overline{x}_3 \cdot x_2 \cdot x_1 + x_3 \cdot \overline{x}_2 \cdot x_1 \cdot \overline{x}_0$
	$S_4^{(2)}(x) = x_3 \cdot \overline{x}_2 \cdot \overline{x}_1 \cdot \overline{x}_0 + x_2 \cdot x_1 \cdot x_0 + \overline{x}_3 \cdot x_2 \cdot \overline{x}_1 \cdot \overline{x}_0 + \overline{x}_2 \cdot \overline{x}_1 \cdot x_0 + x_3 \cdot \overline{x}_2 \cdot x_1 \cdot \overline{x}_0 + \overline{x}_3 \cdot \overline{x}_2 \cdot x_0$
	$S_4^{(3)}(x) = x_3 \cdot \overline{x}_2 \cdot \overline{x}_1 \cdot \overline{x}_0 + \overline{x}_3 \cdot x_2 \cdot \overline{x}_1 \cdot \overline{x}_0 + x_3 \cdot \overline{x}_1 \cdot x_0 + x_3 \cdot \overline{x}_2 \cdot x_1 \cdot x_0 + \overline{x}_3 \cdot x_2 \cdot x_1 + x_3 \cdot x_2 \cdot x_1 \cdot \overline{x}_0$

2.5.3 Speed Optimized Hardware Architecture

In this subsection we present a speed-optimized hardware architecture for Hummingbird encryption/decryption cores, where the encryption or decryption can be performed with four clock cycles. The main goal of the design is to achieve a high speed and throughput. To this end, we first propose a loop-unrolled architecture for the 16-bit block cipher, followed by a detailed description of the data path architectures of encryption/decryption cores.

Loop-Unrolled Architecture of 16-bit Block Cipher

The loop-unrolled architecture for the 16-bit block cipher is illustrated in Figure 2.5. In this architecture, only one 16-bit block of data is processed at a time. However, five rounds are cascaded and the whole encryption can be performed in a single clock cycle. The loop-unrolled architecture consists of 8 XORs, 20 S-boxes, and 4 permutation layers for the datapath. After the given 16-bit block is XORed with the first round key, the obtained result is split into four 4-bit chunks and each of them is then processed by a 4-bit S-box in parallel. The linear transform L performs a permutation on the output of the S-box layer for each of four regular rounds. The final round only includes the S-box layer and four XOR operations and the output ciphertext is stored into a 16-bit flip-flop (FF).

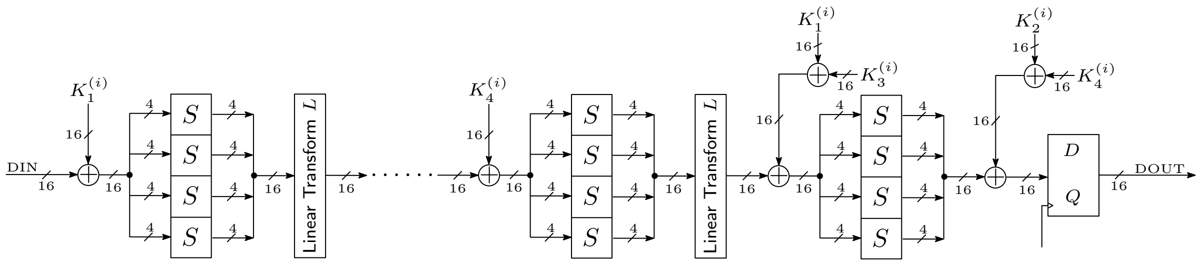


Figure 2.5: Loop-Unrolled Architecture of 16-bit Block Cipher

To select a "hardware-friendly" S-box for the compact version of Hummingbird, we implement the loop-unrolled architecture of the 16-bit block cipher on the target FPGA platform and test one S-box candidate from Table 2.2 each time. Table 2.22 summarizes the area requirement when using different S-boxes and implementation strategies. All experimental results are from post-place and route analysis.

Table 2.22: Area Requirement Comparison for the Loop-Unrolled Architecture of 16-bit Block Cipher on the Spartan-3 XC3S200 FPGA (Using four S-boxes and two implementation strategies)

S-box	Implementation Strategy	# LUTs	# FFs	Total Occupied Slices
$S_1(x)$	LUT	186	16	107
	BFR	186	16	109
$S_2(x)$	LUT	193	16	112
	BFR	186	16	107
$S_3(x)$	LUT	**186**	**16**	**101**
	BFR	186	16	106
$S_4(x)$	LUT	190	16	104
	BFR	187	16	109

When comparing different S-boxes and implementation strategies, Table 2.22 shows that the loop-unrolled architecture occupies the minimal number of slices provided that the S-box $S_3(x)$ is employed and implemented by a LUT. Therefore, the S-box $S_3(x)$ is chosen for efficient implementation of speed optimized Hummingbird encryption/decryption cores that are described in detail in the following subsections.

Speed Optimized Hummingbird Encryption Core

The top-level description and the I/O interface of a speed optimized Hummingbird encryption core are illustrated in Figure 2.6 and Figure 2.7, respectively. As can be seen from

Figure 2.7, the speed optimized Hummingbird encryption core has 119 pins and therefore can be implemented on the Spartan-3 XC3S200 FPGA that features $200,000$ systems gates and a package (FT256) with 173 I/O pins.

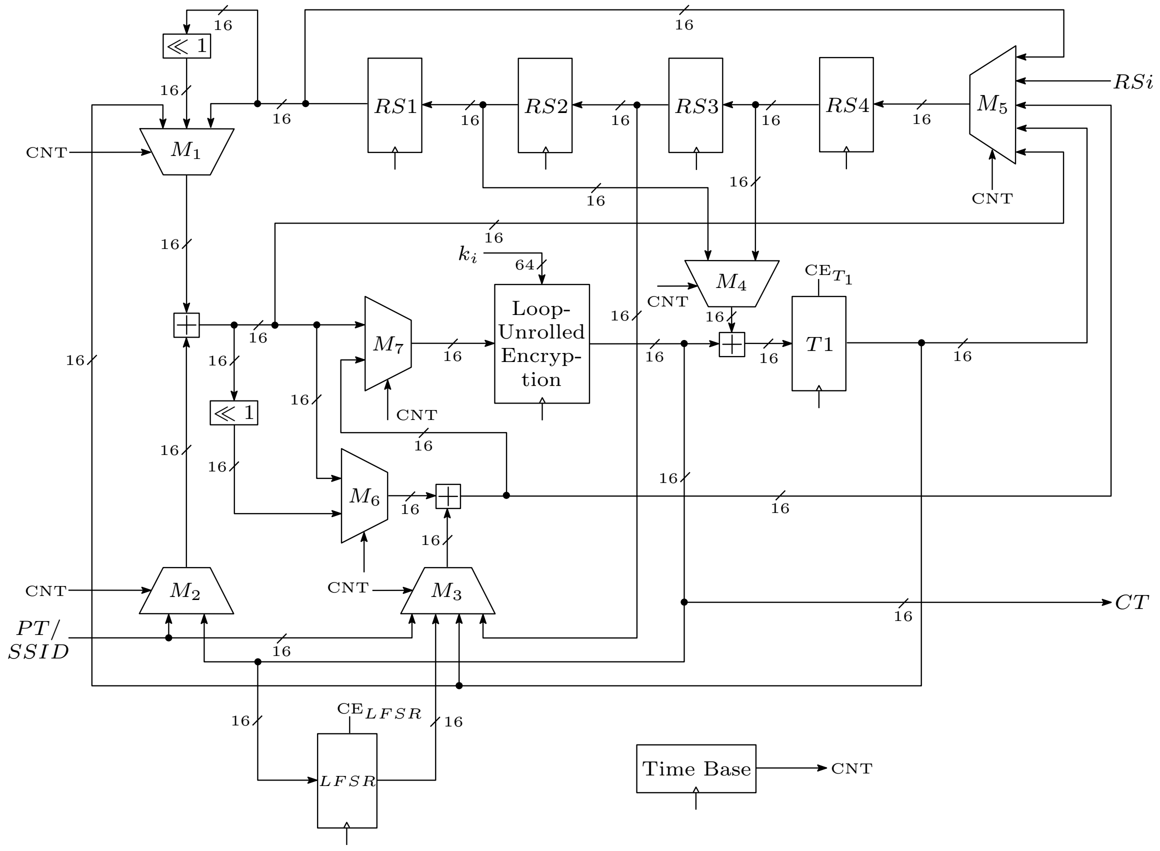


Figure 2.6: The Datapath of Speed Optimized Hummingbird Encryption Core Using the Loop-Unrolled Architecture of 16-bit Block Cipher

The speed optimized Hummingbird encryption core works as follows. After the chip enable signal CE changes from '0' to '1', the initialization process (see Figure 2.1(a)) begins and four rotors RSi ($i = 1, 2, 3, 4$) are first initialized by four 16-bit random nonce through the interface RSi(15:0) within four clock cycles. From the fifth clock cycle, the core starts encrypting $RS1 \boxplus RS3$ for four times (see Algorithm 1) and each iteration requires four clock cycles to finish encryptions by four 16-bit block ciphers as well as the internal state updating. During the above procedure, the 64-bit subkeys k_i ($i = 1, 2, 3, 4$) are read from an external register through the interfaces KEY1(15:0) to KEY4(15:0) and under the control of the signal KEYSEL(1:0). Moreover, depending on the value of a round counter, the multiplexer M_5 chooses the correct computation results to update four rotors and other multiplexers select appropriate inputs to feed the 16-bit block cipher. Note that in order

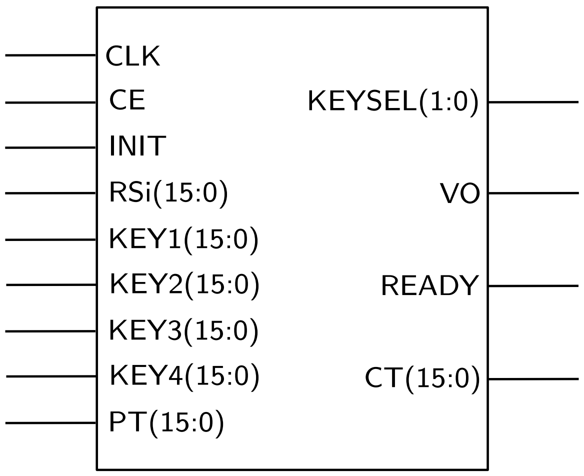


Figure 2.7: The I/O Interface of Hummingbird Encryption Core

to save chip area for the encryption-only core the full update of the rotor $RS2$ involves successive encryptions of two plaintext blocks. More specifically, the rotor $RS2$ is updated by $V12_t$ and $RS4_{t+1}$ (see Algorithm 2) when encrypting two successive plaintext blocks, respectively. Once the initialization process is done after 20 clock cycles, the READY signal changes from '0' to '1' and the first 16-bit plaintext block is read from an external register for encryption. With another four clock cycles, the corresponding ciphertext is output from the encryption core and the valid output signal VO becomes high level. Therefore, the proposed speed optimized Hummingbird encryption core can encrypt one 16-bit plaintext block per 4 clock cycles, after an initialization process of 20 clock cycles.

Speed Optimized Hummingbird Encryption/Decryption Core

We depict the top-level architecture and the I/O interface of a speed optimized Hummingbird encryption/decryption core in the following Figure 2.8 and Figure 2.9, respectively. As can be seen from Figure 2.9, the speed optimized Hummingbird encryption/decryption core has 143 pins and therefore can also be implemented on the Spartan-3 XC3S200 FPGA.

The Hummingbird encryption/decryption core supports the following four operation modes: i) encryption only; ii) decryption only; iii) encryption followed by decryption; and iv) decryption followed by encryption. Both encryption and decryption routines share the same initialization procedure that first takes 4 clock cycles to load four random nonce into rotors through multiplexers M_5 and M_{11}, followed by 16 clock cycles for four iterations. The architecture of the encryption/decryption core is quite similar to that of the encryption-only core except the following several aspects. Firstly, the rotor RS_2 completes the update when encrypting two successive plaintext blocks in the encryption-only core, whereas all

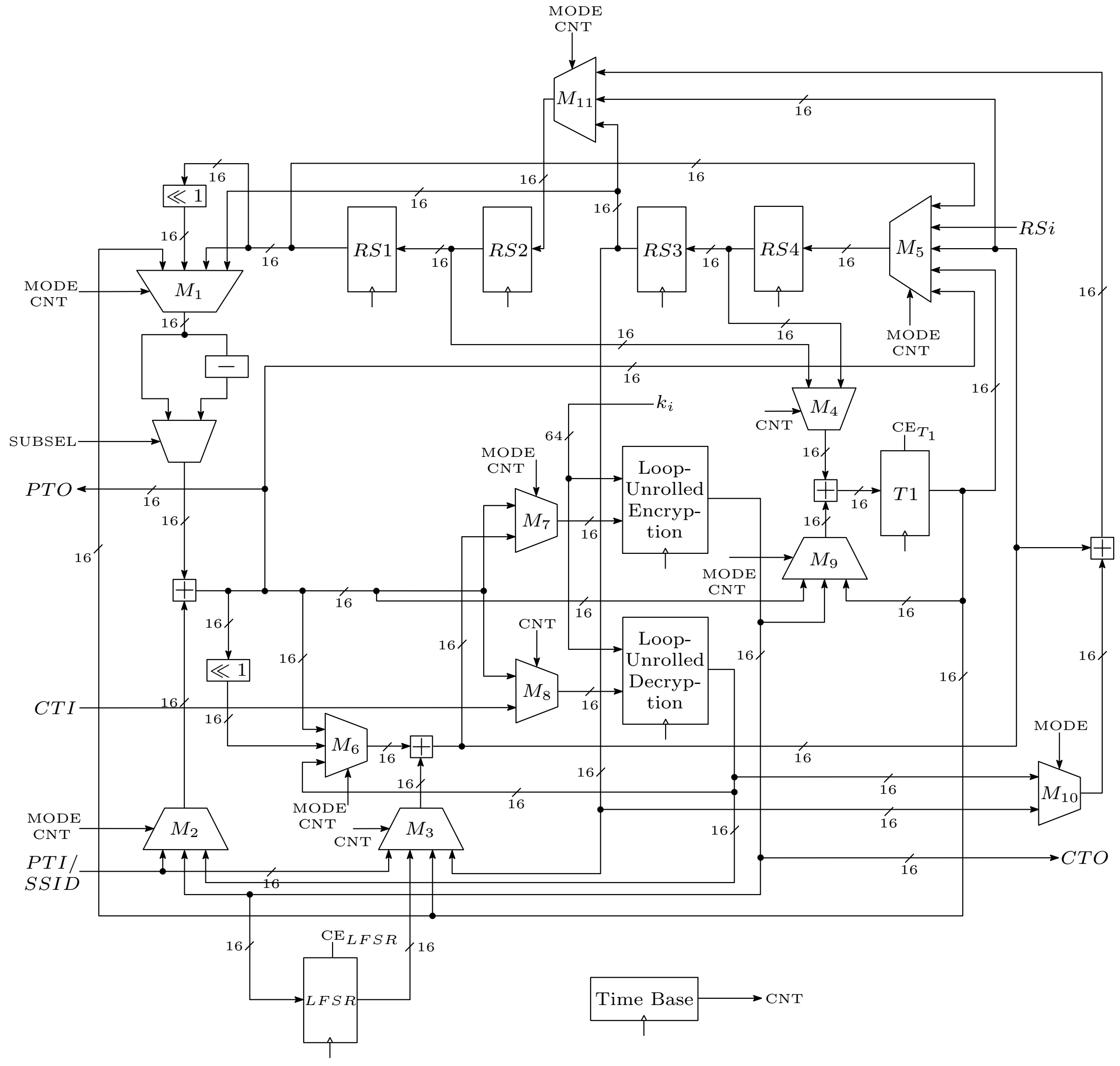


Figure 2.8: The Datapath of Speed Optimized Hummingbird Encryption/Decryption Core Using the Loop-Unrolled Architecture of 16-bit Block Cipher

rotors are fully updated each time a plaintext block is encrypted or decrypted in order to support the four operation modes in the encryption/decryption core. For this purpose, two multiplexers M_{10} and M_{11} are introduced to fully update the rotor $RS2$ after each encryption/decryption. Secondly, an adder that can perform both modulo 2^{16} addition and subtraction is included, which executes the corresponding arithmetic according to the operation modes of the core. Thirdly, two multiplexers M_7 and M_8 are used to feed correct values to the encryption and decryption routines of the 16-bit block cipher, respectively.

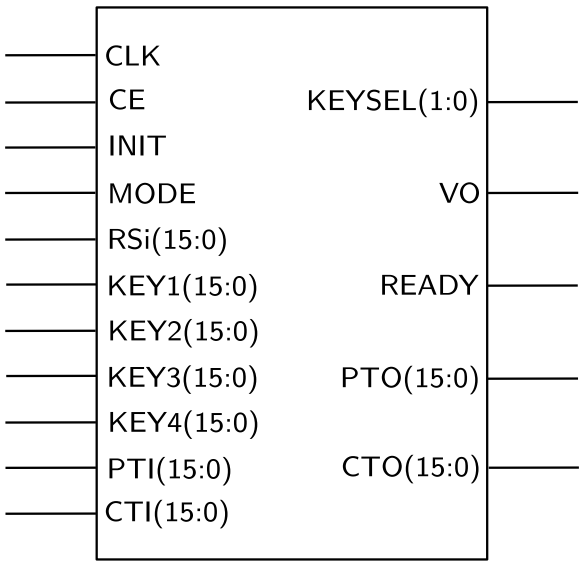


Figure 2.9: The I/O Interface of Hummingbird Encryption/Decryption Core

Finally, all the other multiplexers select appropriate inputs based on the value of a round counter as well as the operation modes. The workflow of the encryption/decryption core is also similar to that of encryption-only core. Hence, the speed optimized Hummingbird encryption/decryption core can encrypt or decrypt one 16-bit plaintext or ciphertext block per 4 clock cycles, after an initialization process of 20 clock cycles.

2.5.4 Area Optimized Hardware Architecture

We describe an area-optimized architecture for Hummingbird encryption/decryption cores in this subsection, which require 16 clock cycles to perform the encryption or decryption. Different from the prior speed-optimized design, the area-optimized architecture features a more compact and energy-efficient solution. A round-based architecture for the 16-bit block cipher is first presented. According to the round-based design of the 16-bit block cipher, we devise the corresponding hardware architecture for the area-optimized Hummingbird encryption/decryption cores.

Round-based Architecture of 16-bit Block Cipher

To further reduce the chip area and power consumption, we propose a round-based architecture that repeatedly uses only one round function block as shown in Figure 2.10. In this architecture, four regular rounds share the common hardware resources of one substitution and permutation layer and the final round is composed of another substitution layer and four XORs. Hence, there are totally 5 XORs, 8 S-boxes, and one permutation layer for the datapath. Furthermore, three 16-bit multiplexers are introduced for different purposes: i) a 4-to-1 multiplexer M_1 is utilized to switch among the required round keys; ii) a 2-to-1 multiplexer M_2 is employed to choose between the input and the computation result of each round; and iii) a 2-to-1 multiplexer M_3 is used to export either the computation result of each round or the final ciphertext that is then stored into a 16-bit register. For the round-based architecture, the whole encryption can be performed in four clock cycles.

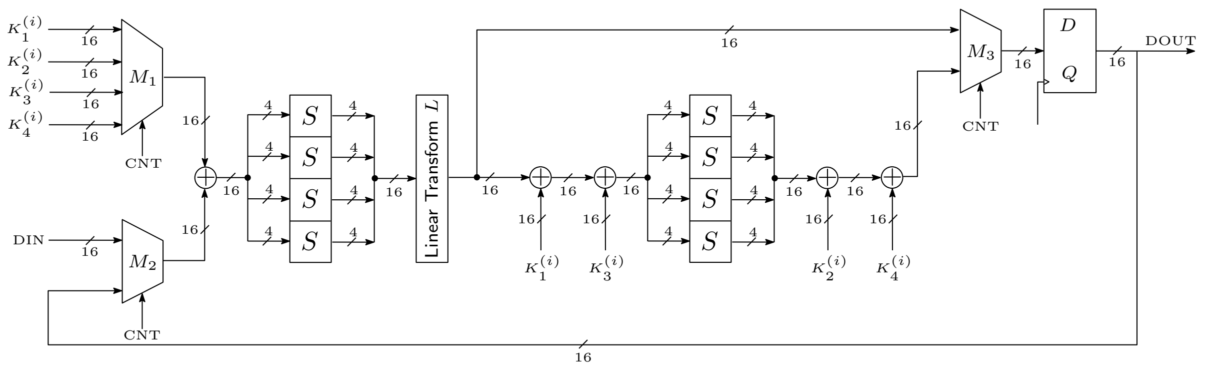


Figure 2.10: Round-based Architecture of 16-bit Block Cipher

Similar to the case of the loop-unrolled architecture, we also implement the round-based architecture of the 16-bit block cipher on the Spartan-3 XC3S200 FPGA and test its area requirement when using four S-boxes and two implementation options, respectively. Table 2.23 summarizes our experimental results that are from post-place and route analysis on the target platform.

Table 2.23: Area Requirement Comparison for the Round-based Architecture of 16-bit Block Cipher on the Spartan-3 XC3S200 FPGA (Using four S-boxes and two implementation strategies)

S-box	Implementation Strategy	# LUTs	# FFs	Total Occupied Slices
$S_1(x)$	LUT	158	16	92
	BFR	158	16	85
$S_2(x)$	LUT	156	16	92
	BFR	**152**	**16**	**82**
$S_3(x)$	LUT	154	16	86
	BFR	161	16	92
$S_4(x)$	LUT	159	16	92
	BFR	163	16	93

From Table 2.23 we note that the round-based architecture of the 16-bit block cipher can achieve the minimal area on the Spartan-3 XC3S200 FPGA by employing the S-box $S_2(x)$ in each round and implementing it with the boolean function representations. Consequently, the S-box $S_2(x)$ is selected for efficient implementation of area optimized Hummingbird encryption/decryption cores that are addressed in the following subsections.

Area Optimized Hummingbird Encryption Core

The top-level description of an area optimized Hummingbird encryption core is depicted in Figure 2.11. Moreover, the area optimized Hummingbird encryption core has the same I/O interface (see Figure 2.7) as that of the speed optimized encryption unit.

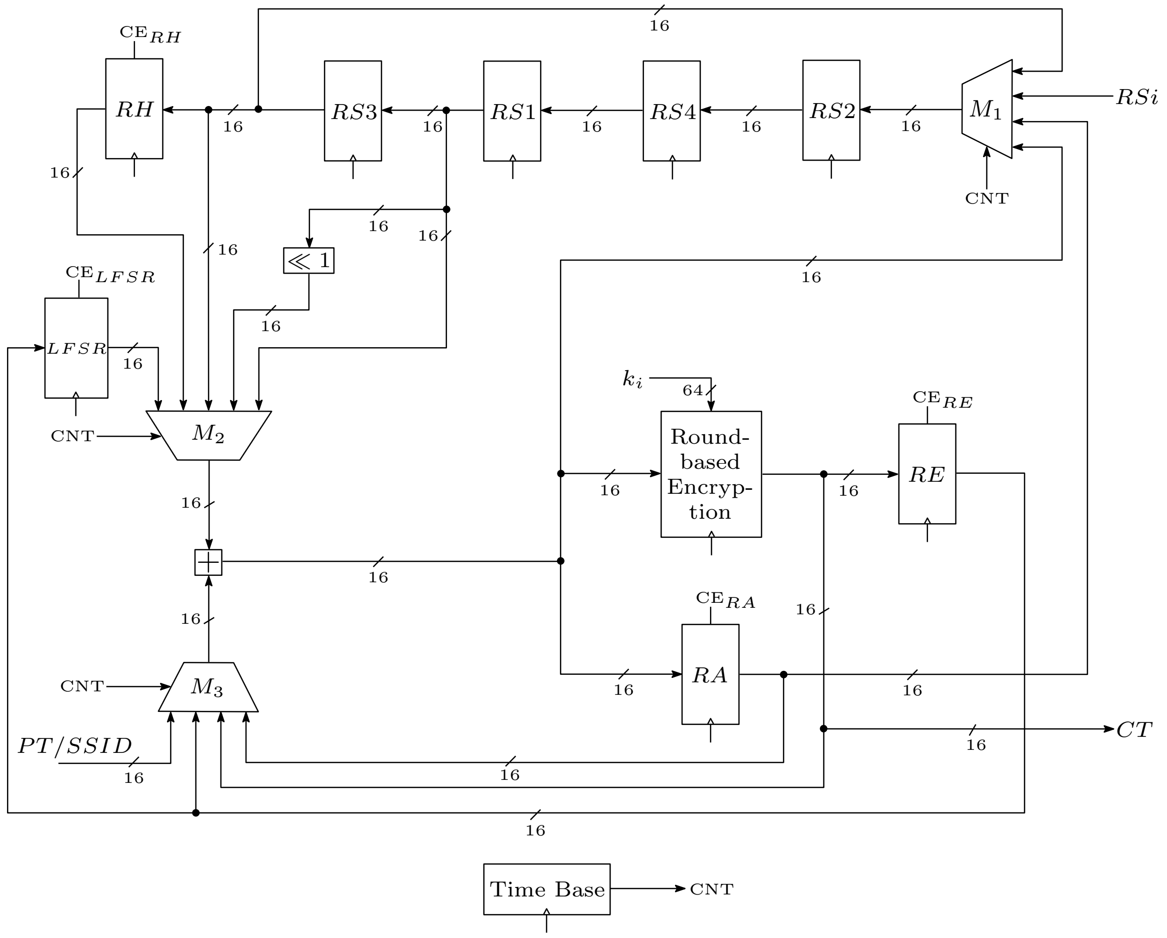


Figure 2.11: The Datapath of Area Optimized Hummingbird Encryption Core Using the Round-based Architecture of 16-bit Block Cipher

The area optimized Hummingbird encryption core works as follows. Once the chip is enabled (i.e., CE = '1'), the initialization process (see Figure 2.1(a)) starts and four rotors RSi ($i = 1, 2, 3, 4$) are first initialized by four 16-bit random nonce through the interface RSi(15:0) within four clock cycles. Moreover, the value $RS3 \boxplus SSID$ is also stored into the register RA in the fourth clock cycle, where $SSID$ denotes the identity of current session[8]. The core then executes four iterations (see Algorithm 1) to encrypt the message $RS1 \boxplus RS3$ from the fifth clock cycle. Each iteration takes 20 clock cycles to complete encryptions by four 16-bit block ciphers and the internal state updating as well. Depending on the value of a round counter, the 64-bit subkeys k_i ($i = 1, 2, 3, 4$) are read from an external register through the interfaces KEY1(15:0) to KEY4(15:0) and under the control of the signal KEYSEL(1:0). In addition, multiplexers M_1, M_2 and M_3 and temporary registers RH, RA and RE also choose the corresponding inputs under the control of the round counter. While the multiplexer M_1 takes care of the update of four rotors, M_2 and M_3 select appropriate operands to form the correct input of the 16-bit block cipher. After 80 clock cycles, the system initialization is finished and the READY signal becomes high level. The first 16-bit plaintext block is then read from an external register for encryption. For another 16 clock cycles, the corresponding ciphertext is output from the encryption core and the valid output signal VO changes from '0' to '1'. Therefore, the proposed area optimized Hummingbird encryption core is able to encrypt one 16-bit plaintext block per 16 clock cycles, after an initialization process of 84 clock cycles.

Area Optimized Hummingbird Encryption/Decryption Core

We show the top-level architecture of an area optimized Hummingbird encryption/decryption core in Figure 2.12. Note that both area and speed optimized encryption/decryption cores have the same I/O interface (see Figure 2.9).

Like the speed optimized Hummingbird encryption/decryption core, the area optimized one also supports four operation modes (see Section 2.5.3). Moreover, both encryption and decryption routines share the same initialization procedure that first takes 4 clock cycles to load four random nonce into rotors through multiplexers M_1, followed by 80 clock cycles for four iterations. Three temporary registers RH, RA and RE store the required data under the control of the operation mode selection signal MODE and a round counter. In addition, the encryption/decryption core also consists of an 16-bit adder that can perform both modulo 2^{16} addition or substraction. Depending on the current operation mode and the value of the round counter, multiplexers M_2 and M_3 choose appropriate operands that will be used by the 16-bit adder to generate the correct inputs for the 16-bit encryption

[8]Note that session identity $SSID$ is useful when using Hummingbird in some authentication protocols, see [57] for an example. If Hummingbird is only used as an encryption engine, $SSID$ is not necessary and only $RS3$ is stored in RA in the fourth clock cycle.

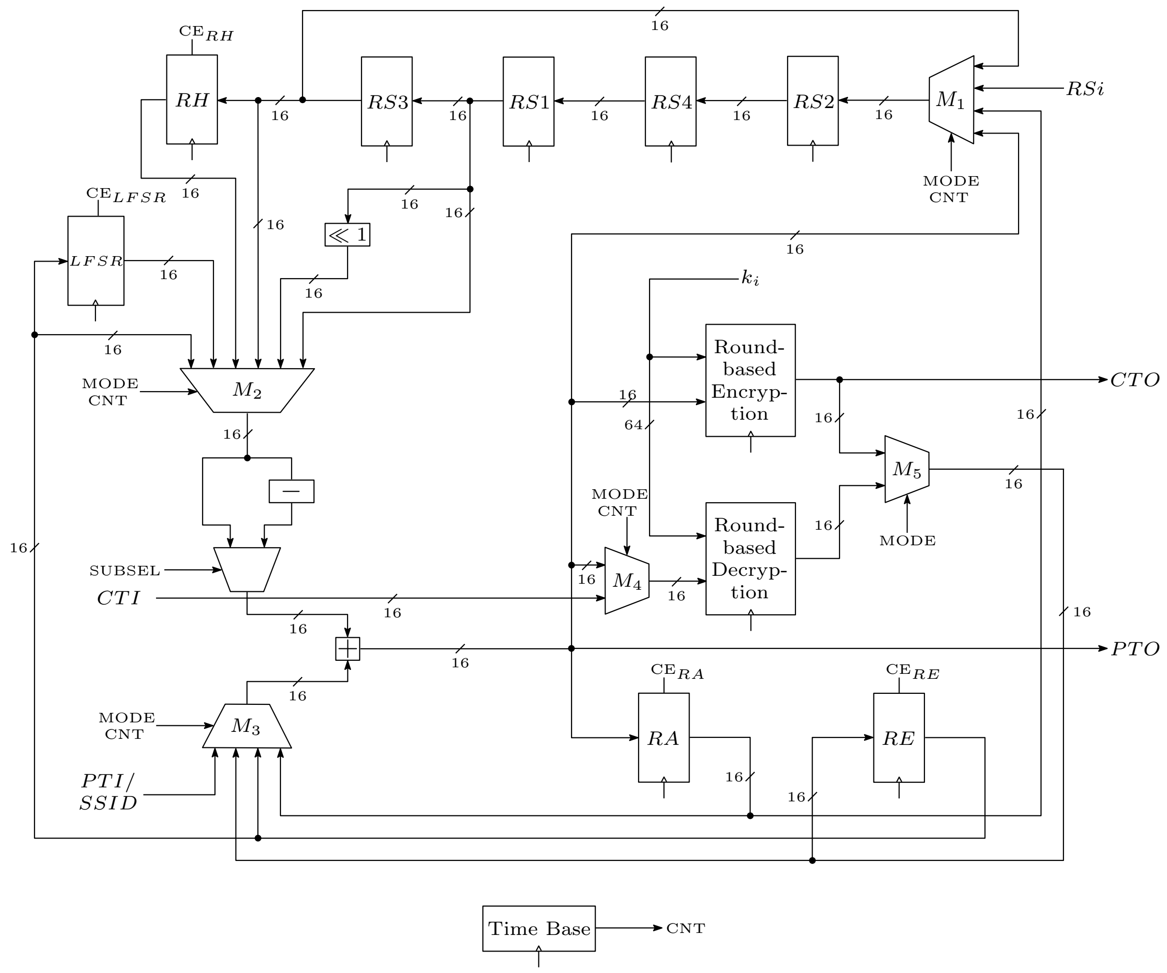


Figure 2.12: The Datapath of Area Optimized Hummingbird Encryption/Decryption Core Using the Round-based Architecture of 16-bit Block Cipher

or decryption module. All rotors will be fully updated through the multiplexer M_1 after each encryption/decryption of a 16-bit plaintext/ciphertext block. While the plaintex and ciphertext block will be read from the interfaces PTI(15:0) and CTI(15:0), the corresponding ciphertext and plaintext will be output through CTO(15:0) and PTO(15:0), respectively. Furthermore, two multiplexers M_4 and M_5 are also included in the encryption/decryption core, where M_4 drives the required inputs to the 16-bit decryption module and M_5 selects output of the 16-bit encryption or decryption module. Since the encryption/decryption core is just a simple extension of the encryption-only core, both of them follow a quite similar workflow. Therefore, the area optimized Hummingbird encryption/decryption core can encrypt or decrypt one 16-bit plaintext or ciphertext block per 16 clock cycles, after an initialization process of 84 clock cycles.

2.5.5 Implementation Results and Comparisons

A summary of our implementation results is presented in Table 2.24, where the area requirements (in slices), the maximum work frequency, and the throughput are provided. All experimental results were extracted after place and route with the ISE Design Suite v11.1 from Xilinx on a xc3s200-5ft256 Spartan-3 platform with speed grade −5. In addition, to achieve better performance, we set Place & Route Effort Lever (Overall) to be "High" and Place & Route Extra Effort to be "Continue on Impossible".

Table 2.24: Implementation Results for Compact Version of Hummingbird on the Spartan-3 XC3S200 FPGA

Opt.	Mode	S-box & its	# LUTs	# FFs	Total Occupied	Max. Freq.	# CLK Cycles		Throughput	Efficiency
	(Enc/Dec)	Implementation			Slices	(MHz)	Init.	Enc/Dec	(Mbps)	(Mbps/# Slices)
Speed	Enc	$S_3(x)$ with LUT	473	120	273	40.1	20	4	160.4	0.59
	Enc/Dec		1,024	145	558	32.2			128.8	0.23
Area	Enc	$S_2(x)$ with BFR	411	131	253	66.1	84	16	66.1	0.26
	Enc/Dec		651	152	363	61.4			61.4	0.17

For the Hummingbird encryption-only and encryption/decryption cores, Table 2.24 shows that the throughput of the speed optimized implementation is 1.42 and 1.1 times larger than that of the area optimized implementation at the cost of additional 20 and 195 slices on the target platform, respectively.

Table 2.25: Performance Comparison of FPGA Implementations of Cryptographic Algorithms

Cipher	Key Size	Block Size	FPGA Device	Total Occupied Slices	Max. Freq. (MHz)	Throughput (Mbps)	Efficiency (Mbps/# Slices)
Hummingbird	256	16	Spartan-3 XC3S200-5	273	40.1	160.4	0.59
PRESENT [144]	80	64	Spartan-3 XC3S400-5	176	258	516	2.93
	128	64		202	254	508	2.51
PRESENT [74]	80	64	Spartan-3E XC3S500	271	–	–	–
XTEA [98]	128	64	Spartan-3 XC3S50-5	254	62.6	36	0.14
			Virtex-5 XC5VLX85-3	9,647	332.2	20,645	2.14
ICEBERG [164]	128	64	Virtex-2	631	–	1,016	1.61
SEA [119]	126	126	Virtex-2 XC2V4000	424	145	156	0.368
AES [34]	128	128	Spartan-2 XC2S30-6	522	60	166	0.32
AES [72]			Spartan-3 XC3S2000-5	17,425	196.1	25,107	1.44
			Spartan-2 XC2S15-6	264	67	2.2	0.01
AES [148]			Spartan-2 XC2V40-6	1,214	123	358	0.29
AES [25]			Spartan-3	1,800	150	1700	0.9

Table 2.25 describes the performance comparison of our Hummingbird implementation with existing FPGA implementations of block ciphers PRESENT [74, 144], XTEA [98], ICEBERG [164], SEA [119] as well as AES [25,34,72,148]. Note that numerous AES hardware architectures, ranging from compact to high speed, have been proposed in literature and we

only focus on those implementations using low-cost Spartan series FPGA devices with speed grade -5 and above for the purpose of comparison. Moreover, the implementation figures of ICEBERG and SEA are only available on Virtex-2 series FPGAs. We also would like to point out that it is quite difficult to provide a fair comparison among different implementations on FPGAs, taking into account the diversity of FPGA devices and packages, speed grade level, and synthesis and implementation tools. Therefore, we also include additional information such as implementation platform and speed grade level in Table 2.25.

Our experimental results show that in the context of low-cost FPGA implementation Hummingbird can achieve larger throughput with smaller area requirement, when compared to block ciphers XTEA, ICEBERG, SEA and AES. However, the implementation of the ultra-lightweight block cipher PRESENT is more efficient than that of Hummingbird, although a slightly large (and hence more expensive) device Spartan-3 XC3S400 is required. The main reason is due to the complex internal state updating procedure in Hummingbird cipher. As a result, the control unit is more complicated and the delay of the critical path is much longer in Hummingbird hardware architecture than those in PRESENT core.

2.6 Encryption Modes and Conclusions

In this chapter we present a novel ultra-lightweight cryptographic algorithm, Hummingbird, which is a combination of block cipher and stream cipher. There are two modes related to Hummingbird as follows: (a) **Enigma Mode**: this is the mode where Hummingbird is used as a word-based cipher (16-bit word) where the plaintext is transitioned through a series of rotors. The ciphertext is dependent on the plaintext; (b) **Stream Mode**: this is the mode of Hummingbird where two values in the internal state ($RS1 \boxplus RS3$) are fed into the input of Hummingbird. The output is a keystream that is XORed with plaintext.

The hybrid structure adopted in Hummingbird can provide the designed security with small block size which is expected to meet the stringent response time and power consumption requirements in a large variety of embedded applications. We show that Hummingbird seems to be resistant to the most common attacks to block ciphers and stream ciphers including birthday attacks, differential and linear cryptanalysis, structure attacks, algebraic attacks, cube attacks, etc.

Efficient software implementations of Hummingbird on 4-, 8- and 16-bit microcontrollers are also investigated. 4-bit microcontrollers have extremely low power consumption (typically $1 - 70\mu A$), making them interesting platforms for implementing various security solutions. Our speed optimized implementation shows that after a system initialization process Hummingbird can achieve up to 2.14 times faster throughput than the state-of-the-art ultra-lightweight block cipher PRESENT on a 4-bit ATAM893-D microcontroller. When comparing the performance of Hummingbird and PRESENT on similar 8-bit platforms, the throughput of Hummingbird is about 40 and 0.7 times faster for a size-optimized and a speed-optimized implementations, respectively. Moreover, since the block size of Hummingbird is perfectly suited to the architecture of 16-bit microcontrollers, Hummingbird can achieve up to 148 and 4.7 times faster throughput than PRESENT for a size-optimized and a speed-optimized implementations, respectively.

On the side of hardware, we propose speed optimized and area optimized architectures, respectively. Furthermore, for different optimization goals two hardware cores are designed, one of which only supports encryption and the other can perform both encryption and decryption. We implement all four hardware cores on the low-cost Spartan-3 XC3S200 FPGA device. Our experimental results show that the speed optimized encryption core can achieve a throughput of 160.4 Mbps, whereas the area optimized core only occupies 253 slices on the target platform. When compared to FPGA implementations of other (lightweight) cryptographic algorithms, Hummingbird cipher also demonstrates better trade-off between throughput and area requirement.

To sum up, our software and hardware implementation results highlight that Hummingbird was designed with both lightweight software and lightweight hardware implementations for constrained devices in mind. Consequently, Hummingbird is well suited for implementing various security mechanisms on mobile devices used in MANETs.

Chapter 3

Efficient Pairing Computation on Genus 2 Hyperelliptic Curves over Prime Fields

In this chapter we propose new variants of Miller's algorithm to speed up pairing computations on two families of non-supersingular genus 2 hyperelliptic curves over prime fields. The achieved acceleration is mainly based on the marriage of two techniques developed for genus 2 curves: 1) efficiently computable automorphisms; and 2) encapsulated group operations. First we will review related work and our motivation in Section 3.1 before we move to a brief introduction of mathematical background in Section 3.2. Then we recall supersingular genus 2 curves over prime fields which have been used for pairing computations, and introduce two families of non-supersingular genus 2 curves with efficiently computable automorphisms in Section 3.3. The new variants of Miller's algorithm and explicit formulae for encapsulated group operations are presented in Section 3.4 and Section 3.5, respectively. Section 3.6 details various techniques for efficiently implementing the Tate pairing on a non-supersingular genus 2 curve with embedding degree 4, analyzes the computational cost of our new algorithm and gives experimental results. Finally, this chapter is concluded in Section 3.7. The research results in this chapter have been published in [61, 62].

3.1 Related Work and Motivation

Miller proposed the first algorithm [125] for computing the Weil pairing on elliptic curves. In practice, the Tate pairing shows better performance than that of the Weil pairing and therefore is widely used. While many important techniques have been proposed to accelerate the computation of the Tate pairing and its variants on elliptic curves [10, 11, 81], the subject of pairing computations on hyperelliptic curves is also receiving an increasing amount of attention. Choie and Lee [35] investigated the implementation of the Tate pairing on supersingular genus 2 hyperelliptic curves over prime fields. Later on, Ó hÉigeartaigh and Scott [80] improved the implementation of [35] significantly by using a new variant of Miller's algorithm combined with various optimization techniques. Duursma and Lee [53] presented a closed formula for the Tate pairing computation on a very special family of supersingular hyperelliptic curves. Barreto *et al.* [10] generalized the results of [53] and proposed the Eta pairing approach for efficiently computing the Tate pairing on supersingular genus 2 curves over binary fields. In [108], Lee *et al.* considered the Eta pairing computation on general divisors on supersingular genus 3 hyperelliptic curves with the form of $y^2 = x^7 - x \pm 1$. Recently, the Ate pairing, which is an extension of the Eta pairing to the setting of ordinary curves, has been generalized to hyperelliptic curves [75] as well. Although the Eta and Ate pairings hold the record for speed at the present time, we will focus our attention on the Tate pairing in this chapter. The main reason is that the Tate pairing is uniformly available across a wide range of hyperelliptic curves and subgroups, whereas the Eta pairing is only defined for supersingular curves and the Ate pairing incurs a huge performance penalty in the context of ordinary genus 2 curves [75, Table 6].

Motivated by previous work in [152, 165, 181], we consider pairing computations on two families of non-supersingular genus 2 hyperelliptic curves over prime fields. We first explicitly give efficiently computable automorphisms (also isogenies) and the dual isogenies on the divisor class group of these curves. We then exploit the automorphism in lieu of the order of the torsion group to construct the rational functions required in Miller's algorithm, and shorten the length of the main loop in Miller's algorithm as a result. Based on the new construction of the rational functions, we propose new variants of Miller's algorithm for computing the Tate pairing on certain non-supersingular genus 2 curves over prime fields. In the best case, the length of the loop in our new variant can be up to 4 times shorter than that of the original Miller's algorithm. In addition, we also address the efficient implementation of the Tate pairing on genus 2 hyperelliptic curves over large prime fields in projective coordinates in this chapter, which is a natural generalization of Chatterjee *et al.*'s work [33] to genus two curves. More specifically, we derive new explicit formulae for the group operations for genus 2 hyperelliptic curves in projective and new (weighted projective) coordinates and show how to encapsulate the computation of the line function with the group operations. Finally, we generate a non-supersingular pairing-friendly genus 2 curve with embedding degree 4 and compare the performance of our new algorithm with that of the variant proposed by Ó hÉigeartaigh and Scott [80] for supersingular genus 2 curves.

3.2 Mathematical Background

In this section, we present a brief introduction to the theory of genus 2 hyperelliptic curves over prime fields, the definition of the Tate pairing and Miller's algorithm to compute it, restricting attention to the material which is relevant to this work. For more details, the reader is referred to [8].

3.2.1 Genus 2 Hyperelliptic Curves over Prime Fields

Let $\mathbb{F}_q$ be a finite field of characteristic $p \neq 2$, $q = p^n$, and let $\overline{\mathbb{F}}_q$ denote the algebraic closure of $\mathbb{F}_q$. Let $\mathbb{F}_q(C)/\mathbb{F}_q$ be a quadratic function field defined via an equation

$$C : y^2 = f(x), \tag{3.1}$$

where $f(X) = x^5 + f_4x^4 + f_3x^3 + f_2x^2 + f_1x + f_0 \in \mathbb{F}_q[x]$ is a monic and square-free polynomial of degree 5. The curve $C/\mathbb{F}_q$ associated with this function field is called a *hyperelliptic curve of genus 2 defined over* $\mathbb{F}_q$. For our purpose it is enough to consider a point P as an ordered pair $P = (x, y) \in \overline{\mathbb{F}}_q^2$ which satisfies $y^2 = f(x)$. Besides these tuples there is one point $\mathcal{O}$ at infinity. The inverse of P is defined as $-P = (x, -y)$. We call a point P that satisfies $P = -P$ a *ramification point.* Note that for $p \neq 5$ the transform $x \to x - f_4/5$ makes the coefficient of x^4 in $f(x)$ zero.

Unlike elliptic curves, points on a hyperelliptic curve do not form a group. Rather than points, divisors are employed. A *divisor* D of $C(\overline{\mathbb{F}}_q)$ is an element of the free abelian group over the points of $C(\overline{\mathbb{F}}_q)$, e.g., $D = \sum_{P \in C(\overline{\mathbb{F}}_q)} n_P P$ with $n_P \in \mathbb{Z}$ and $n_P = 0$ for almost all points P. The *degree* of a divisor D is defined as $\deg(D) = \sum_{P \in C(\overline{\mathbb{F}}_q)} n_P$. The *support* $\mathrm{supp}(D)$ of a divisor D is the set of points P with $n_P \neq 0$ and we define $\mathrm{ord}_P(D) = n_P$. We say that a divisor D is defined over $\mathbb{F}_q$ if $D^\sigma = D$, where $D^\sigma = \sum_{P \in C(\overline{\mathbb{F}}_q)} n_P P^\sigma$, for all automorphisms σ of $\overline{\mathbb{F}}_q$ over $\mathbb{F}_q$. The *divisor class group* $\mathcal{J}_C(\mathbb{F}_q)$ is defined by the quotient group $\mathrm{Div}_C^0(\mathbb{F}_q)/\mathrm{Prin}_C(\mathbb{F}_q)$, where Div_C^0 is a group of degree zero divisors and Prin_C is a group of principal divisors on C, which is a finite formal sum of the zeros and poles.

Each divisor class in $\mathcal{J}_C(\mathbb{F}_q)$ can be represented uniquely by a so-called *reduced divisor* [28], i.e. a divisor of the form $\sum_{i=1}^m (P_i) - m(\mathcal{O}), m \leq 2$ with $P_i = (x_i, y_i) \in C(\overline{\mathbb{F}}_q), P_i \neq \mathcal{O}$ and $P_i \neq -P_j$ for $i \neq j$. Mumford [130] showed that a reduced divisor can be represented by means of two polynomials $u(x), v(x) \in \mathbb{F}_q[x]$, where $u(x)$ and $v(x)$ satisfy the following three conditions: (i) $u(x)$ is monic, (ii) $\deg v(x) < \deg u(x) \leq 2$, and (iii) $u(x) \mid v(x)^2 - f(x)$. In the remainder of this chapter, we will use the notation $[u, v]$ for the divisor class represented by $u(x)$ and $v(x)$. For a genus 2 hyperelliptic curve, we have commonly $[u, v] = [x^2 + u_1x + u_0, v_1x + v_0]$. Cantor's algorithm [28] describes how to perform the group addition of two reduced divisors in Mumford's representation. However, explicit formulae [105] are used to implement the group operations in practice.

3.2.2 Tate Pairing on Hyperelliptic Curves

Let C be a hyperelliptic curve of genus 2 over $\mathbb{F}_q$ given by equation (3.1). We say that a subgroup of the divisor class group $\mathcal{J}_C(\mathbb{F}_q)$ has *embedding degree* k if the order n of the subgroup divides $q^k - 1$, but does not divide $q^i - 1$ for any $0 < i < k$. For our purpose, n should be a (large) prime with $n \mid \#\mathcal{J}_C(\mathbb{F}_q)$ and $\gcd(n, q) = 1$. Let $\mathcal{J}_C(\mathbb{F}_{q^k})[n]$ be the n-torsion group and $\mathcal{J}_C(\mathbb{F}_{q^k})/n\mathcal{J}_C(\mathbb{F}_{q^k})$ be the quotient group. Then the Tate pairing is a well defined, non-degenerate, bilinear map [67]:

$$\langle \cdot, \cdot \rangle_n : \mathcal{J}_C(\mathbb{F}_{q^k})[n] \times \mathcal{J}_C(\mathbb{F}_{q^k})/n\mathcal{J}_C(\mathbb{F}_{q^k}) \to \mathbb{F}_{q^k}^*/(\mathbb{F}_{q^k}^*)^n,$$

defined as follows: let $D_1 \in \mathcal{J}_C(\mathbb{F}_{q^k})[n]$, with $\mathrm{div}(f_{n,D_1}) = nD_1$ for some rational function $f_{n,D_1} \in \mathbb{F}_{q^k}(C)^*$. Let $D_2 \in \mathcal{J}_C(\mathbb{F}_{q^k})/n\mathcal{J}_C(\mathbb{F}_{q^k})$ with $\mathrm{supp}(D_1) \cap \mathrm{supp}(D_2) = \emptyset$ (to ensure a non-trivial pairing value). The Tate pairing of two divisor classes $\overline{D}_1$ and $\overline{D}_2$ is then defined by

$$\langle \overline{D}_1, \overline{D}_2 \rangle_n = f_{n,D_1}(D_2) = \prod_{P \in C(\overline{\mathbb{F}}_q)} f_{n,D_1}(P)^{\mathrm{ord}_P(D_2)}.$$

Note that the Tate pairing as detailed above is only defined up to n-th powers. One can show that if the function f_{n,D_1} is properly normalized, we only need to evaluate the rational function f_{n,D_1} at the effective part of the reduced divisor D_2 in order to compute the Tate pairing [11, 75].

In practice, the fact that the Tate pairing is only defined up to n-th power is usually undesirable, and many pairing-based protocols require a unique pairing value. Hence one defines the *reduced* pairing as

$$\langle \overline{D}_1, \overline{D}_2 \rangle_n^{(q^k-1)/n} = f_{n,D_1}(D_2)^{(q^k-1)/n} \in \mu_n \subset \mathbb{F}_{q^k}^*,$$

where $\mu_n = \{u \in \mathbb{F}_{q^k}^* \mid u^n = 1\}$ is the group of n-th roots of unity. In the rest of this chapter we will refer to the extra powering required to compute the reduced pairing as the *final exponentiation*. One of the important properties of the reduced pairing we will use in this chapter is that for any positive integer N satisfying $n \mid N$ and $N \mid q^k - 1$ we have

$$\langle D_1, D_2 \rangle_n^{(q^k-1)/n} = \langle D_1, D_2 \rangle_N^{(q^k-1)/N}. \tag{3.2}$$

3.2.3 Miller's Algorithm

The main task involved in the computation of the Tate pairing $\langle \overline{D}_1, \overline{D}_2 \rangle_n$ is to construct a rational function f_{n,D_1} such that $\mathrm{div}(f_{n,D_1}) = nD_1$. In [125] (see also [126]), Miller described a polynomial time algorithm, known universally as Miller's algorithm, to construct the function f_{n,D_1} and compute the Weil pairing on elliptic curves. However, the algorithm can be easily adapted to compute the Tate pairing on hyperelliptic curves.

Let $G_{iD_1,jD_1} \in \mathbb{F}_{q^k}(C)^*$ be a rational function with $\mathrm{div}(G_{iD_1,jD_1}) = iD_1 + jD_1 - (iD_1 \oplus jD_1)$ where $\oplus$ is the group law on $\mathcal{J}_C$ and $(iD_1 \oplus jD_1)$ is reduced. Miller's algorithm constructs the rational function f_{n,D_1} based on the following iterative formula:

$$f_{i+j,D_1} = f_{i,D_1} f_{j,D_1} G_{iD_1,jD_1} .$$

Algorithm 5 shows the basic version of Miller's algorithm for computing the reduced Tate pairing on hyperelliptic curves according to the above iterative relation. Essentially, computing the Tate pairing with Miller's algorithm amounts to performing a scalar multiplication of a reduced divisor and evaluating certain intermediate rational functions which appear in the process of the divisor class addition. A more detailed version of Miller's algorithm for hyperelliptic curves can be found in [75].

Algorithm 5 Miller's Algorithm for Hyperelliptic Curves (basic version)

```
Input: $\overline{D}_1 \in \mathcal{J}_C(\mathbb{F}_{q^k})[n], \overline{D}_2 \in \mathcal{J}_C(\mathbb{F}_{q^k})$, represented by $D_1$ and $D_2$ with
       $\mathrm{supp}(D_1) \cap \mathrm{supp}(D_2) = \emptyset$
Output: $\langle D_1, D_2 \rangle_n^{(q^k-1)/n}$
 1: $f \leftarrow 1, T \leftarrow D_1$
 2: for $i = \lfloor \log_2(n) \rfloor - 1$ downto 0 do
 3:     $\triangleright$ Compute $T'$ and $G_{T,T}(x,y)$ such that $T' = 2T - \mathrm{div}(G_{T,T})$
 4:       $f \leftarrow f^2 \cdot G_{T,T}(D_2), \overline{T} \leftarrow [2]\overline{T}$
 5:     if $n_i = 1$ then
 6:         $\triangleright$ Compute $T^{'}$ and $G_{T,D_1}(x,y)$ such that $T^{'} = T + D_1 - \mathrm{div}(G_{T,D_1})$
 7:           $f \leftarrow f \cdot G_{T,D_1}(D_2), \overline{T} \leftarrow \overline{T} \oplus \overline{D}_1$
 8:     end if
 9: end for
10: return $f^{(q^k-1)/n}$
```

Choie and Lee [35] described how to explicitly find the rational function $G(x,y)$ in the Algorithm 5 for the case of genus 2 hyperelliptic curves. Their results can be summarized as follows: Let $D_1 = [u_1, v_1]$ and $D_2 = [u_2, v_2]$ be the two reduced divisors in $\mathcal{J}_C(\mathbb{F}_{q^k})$ that are being added. In the composition stage of Cantor's algorithm, we compute the polynomial $\delta(x)$ which is the greatest common divisor of u_1, u_2 and $v_1 + v_2 + h$ and a divisor $D_3^{'} = [u_3^{'}, v_3^{'}]$, which is in the same divisor class as $D_3 = [u_3, v_3] = \overline{D}_1 + \overline{D}_2$. At this point, two cases may occur:

- If the divisor $D_3^{'}$ is already reduced following the composition stage, then the rational function $G(x,y) = \delta(x)$.

- If this is not the case, then the rational function $G(x,y) = \delta(x) \cdot \frac{y - v_3'(x)}{u_3(x)}$.

In the most frequent cases[1] $\delta = 1$ and thus $G(x,y) = \frac{y - v_3'(x)}{u_3(x)}$.

3.3 Supersingular Curves and Non-supersingular Curves

In this section, we first recall the supersingular genus 2 curves over $\mathbb{F}_p$ which have been used to implement the Tate pairing. Then, by making a small change to the definition of these curves, we produce two families of non-supersingular genus 2 curves over $\mathbb{F}_p$ with efficiently computable automorphisms which provide potential advantages for pairing computations.

3.3.1 Supersingular Genus 2 Curves over $\mathbb{F}_p$

Theoretically, supersingular genus 2 hyperelliptic curves exist over $\mathbb{F}_p$ with an embedding degree of $k = 6$ [150]. However, only supersingular genus 2 curves with an embedding degree of $k = 4$ are known to the cryptographic community at the present time [36]. In [35,80], the authors investigated the efficient implementation of the Tate pairing on supersingular genus 2 curves with embedding degree 4. The curve used in their implementation is defined by the following equation:

$$C_1 : y^2 = x^5 + a, \quad a \in \mathbb{F}_p^* \text{ and } p \equiv 2, 3 \pmod 5.$$

On these supersingular curves a *modified pairing* $\langle D_1, \psi(D_1)\rangle_n^{(p^k-1)/n}$ is computed, where the map $\psi_1(\cdot)$ is a *distortion map* that maps elements in $C_1(\mathbb{F}_p)$ to $C_1(\mathbb{F}_{p^4})$. The distortion map ψ_1 is given by $\psi_1(x,y) = (\xi_5 x, y)$, where ξ_5 is a primitive 5-th root of unity in $\mathbb{F}_{p^4}$. We also note that another family of supersingular genus 2 curves over $\mathbb{F}_p$ with embedding degree 4 [36] is also suitable for implementing pairings. Such curves are given by an equation of the form

$$C_2 : y^2 = x^5 + ax, \quad a \in \mathbb{F}_p^* \cap \overline{QR}_p \text{ and } p \equiv 5 \pmod 8,$$

where $\overline{QR}_p$ denotes the set of quadratic non-residues modulo p. The distortion map for the curve C_2 is defined by $\psi_2(x,y) = (\xi_8^2 x, \xi_8 y)$, where ξ_8 is a primitive 8-th root of unity in $\mathbb{F}_{p^4}$.

[1]For addition, the inputs are two co-prime polynomials of degree 2, and for doubling the input is a square free polynomial of degree 2.

3.3.2 Non-Supersingular Genus 2 Curves over $\mathbb{F}_p$

Motivated by the work in [152, 165, 181], we consider now the following two families of non-supersingular genus 2 hyperelliptic curves over $\mathbb{F}_p$:

$$\begin{aligned} C_1' : y^2 = x^5 + a, \quad & a \in \mathbb{F}_p^* \quad \text{and} \quad p \equiv 1 \pmod 5, \\ C_2' : y^2 = x^5 + ax, \quad & a \in \mathbb{F}_p^* \quad \text{and} \quad p \equiv 1 \pmod 8. \end{aligned}$$

Curves of this form exist which are pairing friendly (see Section 3.4). Note that the only difference between the curves C_i and C_i' ($i = 1, 2$) is the congruence condition applied to the characteristic p. Although distortion maps do not exist on these non-supersingular curves, both C_1' and C_2' have efficiently-computable *endomorphisms*. In fact, these endomorphisms also induce efficient *automorphisms* on the divisor class groups of C_1' and C_2', which have been used to accelerate the scalar multiplication for hyperelliptic curve cryptosystems [139] and to attack the discrete logarithm problems on the Jacobians [52,70]. In the next section, we will show how to speed up the computation of the Tate pairing using these efficiently computable automorphisms on the curves C_1' and C_2'. We first recall some basic facts which will be used in this work.

For the curve C_1', the morphism ψ_1 defined by $P = (x, y) \mapsto \psi_1(P) = (\xi_5 x, y)$ (see Section 3.3.1 and notice $\xi_5 \in \mathbb{F}_p$ now) induces an efficient non-trivial automorphism of order 5 on the divisor class group $\mathcal{J}_{C_1'}(\mathbb{F}_p)$ [139]. The formulae for ψ_1 on the Jacobian are given by

$$\begin{aligned} \psi_1 : [x^2 + u_1x + u_0, v_1x + v_0] &\mapsto [x^2 + \xi_5 u_1 x + \xi_5^2 u_0, \xi_5^{-1} v_1 x + v_0] \\ [x + u_0, v_0] &\mapsto [x + \xi_5 u_0, v_0] \\ \mathcal{O} &\mapsto \mathcal{O}. \end{aligned}$$

The characteristic polynomial of ψ_1 is given by $P(t) = t^4 + t^3 + t^2 + t + 1$. Since the automorphism ψ_1 is also an *isogeny*, we can construct its *dual isogeny* as follows:

$$\begin{aligned} \widehat{\psi_1} : [x^2 + u_1x + u_0, v_1x + v_0] &\mapsto [x^2 + \xi_5^{-1} u_1 x + \xi_5^{-2} u_0, \xi_5 v_1 x + v_0] \\ [x + u_0, v_0] &\mapsto [x + \xi_5^{-1} u_0, v_0] \\ \mathcal{O} &\mapsto \mathcal{O}. \end{aligned}$$

Note that $\psi_1 \circ \widehat{\psi_1} = [1]$ and $\#\,\mathrm{Ker}\ \psi_1 = \deg[1] = 1$, and $\widehat{\psi_1}$ is also a non-trivial automorphism on the curve C_1'.

Let $D \in \mathcal{J}_{C_1'}(\mathbb{F}_p)$ be a reduced divisor of a large prime order n. From [139], we know that the automorphisms ψ_1 and $\widehat{\psi_1}$ act respectively as multiplication maps by $[\lambda_1]$ and $[\widehat{\lambda_1}]$ on the subgroup $\langle D \rangle$ of $\mathcal{J}_{C_1'}(\mathbb{F}_p)$, where λ_1 and $\widehat{\lambda_1}$ are the two roots of the equation

$t^4+t^3+t^2+t+1 \equiv 0 \pmod{n}$. Furthermore, it is easily seen that $[\lambda_1]D = \psi_1(D)$ can be obtained with only three or one field multiplications in $\mathbb{F}_p$ for a *general divisor* and a *degenerate divisor*, respectively. In the genus 2 context, a general divisor has two finite points in the support, whereas a degenerate divisor has only a single finite point in its support.

Similarly, for the curve C_2', the morphism ψ_2 defined by $P=(x,y) \mapsto \psi_2(P) = (\xi_8^2 x, \xi_8 y)$ (see Section 3.3.1 and notice $\xi_8 \in \mathbb{F}_p$ now) induces an efficient non-trivial automorphism of order 8 on the divisor class group $\mathcal{J}_C(\mathbb{F}_p)$ as follows [139].

$$\begin{aligned} \psi_2 : [x^2+u_1x+u_0, v_1x+v_0] &\mapsto [x^2+\xi_8^2u_1x+\xi_8^4u_0, \xi_8^{-1}v_1x+\xi_8v_0] \\ [x+u_0, v_0] &\mapsto [x+\xi_8^2u_0, \xi_8v_0] \\ \mathcal{O} &\mapsto \mathcal{O}. \end{aligned}$$

The characteristic polynomial of ψ_2 is given by $P(t) = t^4+1$ and the dual isogeny of ψ_2 is defined as follows

$$\begin{aligned} \widehat{\psi_2} : [x^2+u_1x+u_0, v_1x+v_0] &\mapsto [x^2+\xi_8^{-2}u_1x+\xi_8^4u_0, \xi_8v_1x+\xi_8^{-1}v_0] \\ [x+u_0, v_0] &\mapsto [x+\xi_8^{-2}u_0, \xi_8^{-1}v_0] \\ \mathcal{O} &\mapsto \mathcal{O}. \end{aligned}$$

It is not difficult to show that $\psi_2 \circ \widehat{\psi_2} = [1]$ and $\#\,\mathrm{Ker}\ \psi_2 = \deg[1] = 1$, and $\widehat{\psi_2}$ is also a non-trivial automorphism on the curve C_2'. Let $D \in \mathcal{J}_{C_2'}(\mathbb{F}_p)$ be a reduced divisor of a large prime order n. Then the automorphism ψ_2 acts as a multiplication map by λ_2 on the subgroup $\langle D \rangle$ of $\mathcal{J}_{C_2'}(\mathbb{F}_p)$, where λ_2 is a root of the equation $t^4+1 \equiv 0 \pmod{n}$. Moreover, $[\lambda_2]D = \psi_2(D)$ can be computed with four or two field multiplications in $\mathbb{F}_p$ for a general divisor and a degenerate divisor, respectively.

3.4 Efficient Pairings on Non-supersingular Genus 2 Curves

In this section we investigate efficient algorithms for computing the Tate pairing on the two families of genus 2 hyperelliptic curves C_1' and C_2' defined in Section 3.3.2. We show how to use the efficiently-computable automorphisms on these curves to shorten the length of the loop in Miller's algorithm. As a result, we propose new variants of Miller's algorithm for certain non-supersingular genus 2 curves over large prime fields.

3.4.1 Pairing Computation with Efficient Automorphisms

In this subsection, we present the main results of this chapter in the following theorems and show their correctness. The pairing computation on the curve C_1' is first examined.

Theorem 3.4.1 *Let C_1' be a non-supersingular genus 2 hyperelliptic curve over $\mathbb{F}_p$ as above, with embedding degree $k > 1$ and automorphisms ψ_1 and $\widehat{\psi}_1$ defined as above. Let $D_1 = [u_1(x), v_1(x)] \in \mathcal{J}_{C_1'}(\mathbb{F}_p)$ be a reduced divisor of prime order n, where $n^2 \nmid \#\mathcal{J}_{C_1'}(\mathbb{F}_p)$. Let $[\lambda_1]$ act as the multiplication map on the subgroup $\langle D_1 \rangle$ defined as above such that $[\lambda_1]D_1 = \psi_1(D_1)$. Suppose $m \in \mathbb{Z}$ is such that $mn = \lambda_1^4 + \lambda_1^3 + \lambda_1^2 + \lambda_1 + 1$. Let rational functions $\frac{c_1(x,y)}{d_1(x,y)}, \frac{c_2(x,y)}{d_2(x,y)}, \frac{c_3(x,y)}{d_3(x,y)} \in \mathbb{F}_p(C_1')^*$ respectively satisfy the following three relations:*

$$\begin{aligned}
[\lambda_1]D_1 + [\lambda_1^2]D_1 - \left([\lambda_1]D_1 \oplus [\lambda_1^2]D_1\right) &= \operatorname{div}\left(\frac{c_1(x,y)}{d_1(x,y)}\right), \\
\left[\lambda_1^3\right] D_1 + [\lambda_1^4]D_1 - \left([\lambda_1^3]D_1 \oplus [\lambda_1^4]D_1\right) &= \operatorname{div}\left(\frac{c_2(x,y)}{d_2(x,y)}\right), \\
\left[\lambda_1 + \lambda_1^2\right] D_1 + [\lambda_1^3 + \lambda_1^4]D_1 - \left([\lambda_1 + \lambda_1^2]D_1 \oplus [\lambda_1^3 + \lambda_1^4]D_1\right) &= \operatorname{div}\left(\frac{c_3(x,y)}{d_3(x,y)}\right).
\end{aligned}$$

Let $g(x,y) = \frac{c_1(x,y)\cdot c_2(x,y)\cdot c_3(x,y)}{d_1(x,y)\cdot d_2(x,y)}$. Then for $D_2 \in \mathcal{J}_{C_1'}(\mathbb{F}_{p^k})$, we have

$$\begin{aligned}
\langle D_1, D_2 \rangle_n^{\frac{m(p^k-1)}{n}} &= \Big[f_{\lambda_1,D_1}^{\lambda_1^3+\lambda_1^2+\lambda_1+1}(D_2) \cdot f_{\lambda_1,D_1}^{\lambda_1^2+\lambda_1+1}\left(\widehat{\psi}_1(D_2)\right) \cdot f_{\lambda_1,D_1}^{\lambda_1+1}\left(\widehat{\psi}_1^2(D_2)\right) \cdot \\
&\quad f_{\lambda_1,D_1}\left(\psi_1^2(D_2)\right) \cdot g(D_2)\Big]^{\frac{p^k-1}{n}}.
\end{aligned}$$

Note that the condition that λ_1 satisfies $\lambda_1^4 + \lambda_1^3 + \lambda_1^2 + \lambda_1 + 1 \equiv 0 \pmod{n}$ guarantees the existence of the integer m. Moreover, the pairing will be non-degenerate if $n \nmid m$ and $\operatorname{supp}(D_1) \cap \operatorname{supp}(D_2) = \emptyset$. We split the proof of the Theorem 3.4.1 into the following lemmas.

Lemma 3.4.2 *With notation as above, we have*

$$\langle D_1, D_2 \rangle_n^{\frac{m(p^k-1)}{n}} = \left(f_{\lambda_1^4+\lambda_1^3+\lambda_1^2+\lambda_1,D_1}(D_2) \cdot u_1(D_2)\right)^{\frac{p^k-1}{n}}.$$

Proof. From the important property of the reduced pairing (see equation (3.2)), we have

$$\langle D_1, D_2 \rangle_n^{\frac{m(p^k-1)}{n}} = \langle D_1, D_2 \rangle_{mn}^{\frac{p^k-1}{n}} = f_{mn,D_1}(D_2)^{\frac{p^k-1}{n}}.$$

From the condition that $mn = \lambda_1^4 + \lambda_1^3 + \lambda_1^2 + \lambda_1 + 1$, we get

$$\langle D_1, D_2 \rangle_n^{\frac{m(p^k-1)}{n}} = f_{mn,D_1}(D_2)^{\frac{p^k-1}{n}} = f_{\lambda_1^4+\lambda_1^3+\lambda_1^2+\lambda_1+1,D_1}(D_2)^{\frac{p^k-1}{n}}.$$

Since $[\lambda_1^4 + \lambda_1^3 + \lambda_1^2 + \lambda_1]D_1 = -D_1$, we obtain the following relation

$$D_1 + [\lambda_1 + \lambda_1^2 + \lambda_1^3 + \lambda_1^4]D_1 = D_1 + (-D_1) = \mathrm{div}(u_1(x)).$$

Therefore, we have

$$\begin{aligned}
&\mathrm{div}\left(f_{\lambda_1^4+\lambda_1^3+\lambda_1^2+\lambda_1+1,D_1}\right) = (\lambda_1^4 + \lambda_1^3 + \lambda_1^2 + \lambda_1)D_1 + D_1 \\
&\quad = \mathrm{div}\left(f_{\lambda_1^4+\lambda_1^3+\lambda_1^2+\lambda_1,D_1}\right) + D_1 + [\lambda_1 + \lambda_1^2 + \lambda_1^3 + \lambda_1^4]D_1 \\
&\quad = \mathrm{div}\left(f_{\lambda_1^4+\lambda_1^3+\lambda_1^2+\lambda_1,D_1} \cdot u_1(x)\right),
\end{aligned}$$

which proves the result. ∎

The next lemma shows the relation between $\mathrm{div}\left(f_{\lambda_1^4+\lambda_1^3+\lambda_1^2+\lambda_1,D_1} \cdot u_1(x)\right)$ and the divisors $\mathrm{div}\left(f_{\lambda_1,[\lambda_1^i]D_1}\right)$ for $i = 0, 1, 2$, and 3.

Lemma 3.4.3 *With notation as above, we have*

$$\begin{aligned}
&\mathrm{div}\left(f_{\lambda_1^4+\lambda_1^3+\lambda_1^2+\lambda_1,D_1} \cdot u_1(x)\right) = \\
&\quad \mathrm{div}\left(f_{\lambda_1,D_1}^{\lambda_1^3+\lambda_1^2+\lambda_1+1} \cdot f_{\lambda_1,[\lambda_1]D_1}^{\lambda_1^2+\lambda_1+1} \cdot f_{\lambda_1,[\lambda_1^2]D_1}^{\lambda_1+1} \cdot f_{\lambda_1,[\lambda_1^3]D_1} \cdot g(x,y)\right).
\end{aligned}$$

Proof. We first note the following relation

$$\begin{aligned}
\mathrm{div}\left(f_{\lambda_1^4+\lambda_1^3+\lambda_1^2+\lambda_1,D_1}\right) &= (\lambda_1^4 + \lambda_1^3 + \lambda_1^2 + \lambda_1)D_1 - [\lambda_1^4 + \lambda_1^3 + \lambda_1^2 + \lambda_1]D_1 \\
&= \mathrm{div}\left(f_{\lambda_1^4+\lambda_1^3,D_1}\right) + \mathrm{div}\left(f_{\lambda_1^2+\lambda_1,D_1}\right) + [\lambda_1 + \lambda_1^2]D_1 + \\
&\quad [\lambda_1^3 + \lambda_1^4]D_1 - \left([\lambda_1 + \lambda_1^2]D_1 \oplus [\lambda_1^3 + \lambda_1^4]D_1\right) \\
&= \mathrm{div}\left(f_{\lambda_1^4+\lambda_1^3,D_1}\right) + \mathrm{div}\left(f_{\lambda_1^2+\lambda_1,D_1}\right) + \mathrm{div}\left(\frac{c_3(x,y)}{d_3(x,y)}\right) \\
&= \mathrm{div}\left(f_{\lambda_1^4+\lambda_1^3,D_1} \cdot f_{\lambda_1^2+\lambda_1,D_1} \cdot \frac{c_3(x,y)}{d_3(x,y)}\right)
\end{aligned}$$

Since $[\lambda_1^4 + \lambda_1^3 + \lambda_1^2 + \lambda_1]D_1 = -D_1$, we get $d_3(x,y) = u_1(x)$. Therefore, we have

$$\mathrm{div}\left(f_{\lambda_1^4+\lambda_1^3+\lambda_1^2+\lambda_1,D_1} \cdot u_1(x)\right) = \mathrm{div}\left(f_{\lambda_1^4+\lambda_1^3,D_1} \cdot f_{\lambda_1^2+\lambda_1,D_1} \cdot c_3(x,y)\right). \tag{3.3}$$

Similarly, we can obtain the following two equalities

$$\begin{aligned}
&\operatorname{div}\left(f_{\lambda_1^4+\lambda_1^3,D_1}\right) = (\lambda_1^4+\lambda_1^3)D_1 - [\lambda_1^4+\lambda_1^3]D_1 \\
&= \operatorname{div}\left(f_{\lambda_1^4,D_1}\right) + \operatorname{div}\left(f_{\lambda_1^3,D_1}\right) + [\lambda_1^4]D_1 + [\lambda_1^3]D_1 - \big([\lambda_1^3]D_1 \oplus [\lambda_1^4]D_1\big) \\
&= \operatorname{div}\left(f_{\lambda_1^4,D_1}\right) + \operatorname{div}\left(f_{\lambda_1^3,D_1}\right) + \operatorname{div}\left(\frac{c_2(x,y)}{d_2(x,y)}\right) \\
&= \operatorname{div}\left(f_{\lambda_1^4,D_1}\cdot f_{\lambda_1^3,D_1}\cdot \frac{c_2(x,y)}{d_2(x,y)}\right)
\end{aligned}$$

and

$$\begin{aligned}
&\operatorname{div}\left(f_{\lambda_1^2+\lambda_1,D_1}\right) = (\lambda_1^2+\lambda_1)D_1 - [\lambda_1^2+\lambda_1]D_1 \\
&= \operatorname{div}\left(f_{\lambda_1^2,D_1}\right) + \operatorname{div}\left(f_{\lambda_1,D_1}\right) + [\lambda_1^2]D_1 + [\lambda_1]D_1 - \big([\lambda_1]D_1 \oplus [\lambda_1^2]D_1\big) \\
&= \operatorname{div}\left(f_{\lambda_1^2,D_1}\right) + \operatorname{div}\left(f_{\lambda_1,D_1}\right) + \operatorname{div}\left(\frac{c_1(x,y)}{d_1(x,y)}\right) \\
&= \operatorname{div}\left(f_{\lambda_1^2,D_1}\cdot f_{\lambda_1,D_1}\cdot \frac{c_1(x,y)}{d_1(x,y)}\right)
\end{aligned}$$

Some easy calculations (see Lemma 2 in [10]) give us

$$\operatorname{div}\left(f_{\lambda_1^4,D_1}\right) = \operatorname{div}\left(f_{\lambda_1,D_1}^{\lambda_1^3}\cdot f_{\lambda_1,[\lambda_1]D_1}^{\lambda_1^2}\cdot f_{\lambda_1,[\lambda_1^2]D_1}^{\lambda_1}\cdot f_{\lambda_1,[\lambda_1^3]D_1}\right) \tag{3.4}$$

$$\operatorname{div}\left(f_{\lambda_1^3,D_1}\right) = \operatorname{div}\left(f_{\lambda_1,D_1}^{\lambda_1^2}\cdot f_{\lambda_1,[\lambda_1]D_1}^{\lambda_1}\cdot f_{\lambda_1,[\lambda_1^2]D_1}\right) \tag{3.5}$$

$$\operatorname{div}\left(f_{\lambda_1^2,D_1}\right) = \operatorname{div}\left(f_{\lambda_1,D_1}^{\lambda_1}\cdot f_{\lambda_1,[\lambda_1]D_1}\right) \tag{3.6}$$

Combining the equations (3.3)–(3.6) proves the result. ∎

The following lemma shows how to evaluate functions $f_{\lambda_1,[\lambda_1^i]D_1}$ ($i = 1, 2, 3$) at the image divisor D_2 by using the function f_{λ_1,D_1}.

Lemma 3.4.4 *With notation as above, we have (up to a scalar multiple in $\mathbb{F}_p^*$)*

$$\begin{aligned}
f_{\lambda_1,[\lambda_1]D_1}(D_2) &= f_{\lambda_1,D_1}(\widehat{\psi}_1(D_2)), \\
f_{\lambda_1,[\lambda_1^2]D_1}(D_2) &= f_{\lambda_1,D_1}(\widehat{\psi}_1^2(D_2)), \\
f_{\lambda_1,[\lambda_1^3]D_1}(D_2) &= f_{\lambda_1,D_1}(\psi_1^2(D_2)).
\end{aligned}$$

Proof. Using the pullback property (see Silverman [160] p. 33) and following the same proof as the Lemma 1 in [10], we obtain (up to a scalar multiple in $\mathbb{F}_p^*$)

$$\begin{aligned}
f_{\lambda_1,[\lambda_1]D_1} \circ \psi_1 &= f_{\lambda_1,D_1}, \\
f_{\lambda_1,[\lambda_1^2]D_1} \circ \psi_1^2 &= f_{\lambda_1,D_1}, \\
f_{\lambda_1,[\lambda_1^3]D_1} \circ \psi_1^3 &= f_{\lambda_1,D_1}.
\end{aligned}$$

Using the relations between the isogeny ψ_1 and its dual isogeny $\widehat{\psi_1}$ (see Section 3.3.2), we have

$$\begin{aligned}
f_{\lambda_1,[\lambda_1]D_1} \circ \psi_1 \circ \widehat{\psi_1} &= f_{\lambda_1,[\lambda_1]D_1} = f_{\lambda_1,D_1} \circ \widehat{\psi_1}, \\
f_{\lambda_1,[\lambda_1^2]D_1} \circ \psi_1^2 \circ \widehat{\psi_1^2} &= f_{\lambda_1,[\lambda_1^2]D_1} = f_{\lambda_1,D_1} \circ \widehat{\psi_1^2}, \\
f_{\lambda_1,[\lambda_1^3]D_1} \circ \psi_1^3 \circ \widehat{\psi_1^3} &= f_{\lambda_1,[\lambda_1^3]D_1} = f_{\lambda_1,D_1} \circ \widehat{\psi_1^3} = f_{\lambda_1,D_1} \circ \psi_1^2,
\end{aligned}$$

which proves the results. ∎

With the above three lemmas, we can now prove the statement of Theorem 3.4.1 as follows:

Proof of Theorem 3.4.1 For $D_1 \in \mathcal{J}_{C_1'}(\mathbb{F}_p)[n]$ and $D_2 \in \mathcal{J}_{C_1'}(\mathbb{F}_{p^k})$, Lemma 3.4.4 shows that up to a scalar multiple in $\mathbb{F}_p^*$ we have

$$\begin{aligned}
f_{\lambda_1,[\lambda_1]D_1}(D_2) &= f_{\lambda_1,D_1}(\widehat{\psi_1}(D_2)), \\
f_{\lambda_1,[\lambda_1^2]D_1}(D_2) &= f_{\lambda_1,D_1}(\widehat{\psi_1^2}(D_2)), \\
f_{\lambda_1,[\lambda_1^3]D_1}(D_2) &= f_{\lambda_1,D_1}(\psi_1^2(D_2)).
\end{aligned}$$

Now, substituting the above three equalities into Lemma 3.4.3 implies that

$$\begin{aligned}
f_{\lambda_1^4+\lambda_1^3+\lambda_1^2+\lambda_1,D_1}(D_2) \cdot u_1(D_2) &= f_{\lambda_1,D_1}^{\lambda_1^3+\lambda_1^2+\lambda_1+1}(D_2) \cdot f_{\lambda_1,D_1}^{\lambda_1^2+\lambda_1+1}\left(\widehat{\psi_1}(D_2)\right) \cdot \\
&\quad f_{\lambda_1,D_1}^{\lambda_1+1}\left(\widehat{\psi_1^2}(D_2)\right) \cdot f_{\lambda_1,D_1}\left(\psi_1^2(D_2)\right) \cdot g(D_2).
\end{aligned}$$

Finally, substituting the above equation into Lemma 3.4.2 gives the result of Theorem 3.4.1. ∎

Next, we consider how to use the efficiently-computable automorphism ψ_2 to accelerate the computation of the Tate pairing on the curve C_2'. The following Theorem 3.4.5 describes our result.

Theorem 3.4.5 *Let C_2' be a non-supersingular genus 2 hyperelliptic curve over $\mathbb{F}_p$ as above, with embedding degree $k > 1$ and automorphisms ψ_2 and $\widehat{\psi}_2$ defined as above. Let $D_1 = [u_1(x), v_1(x)] \in \mathcal{J}_{C_2'}(\mathbb{F}_p)$ be a reduced divisor of prime order n, where $n^2 \nmid \#\mathcal{J}_{C_2'}(\mathbb{F}_p)$. Let $[\lambda_2]$ act as the multiplication map on the subgroup $\langle D_1 \rangle$ defined as above such that $[\lambda_2]D_1 = \psi_2(D_1)$. Suppose $m \in \mathbb{Z}$ is such that $mn = \lambda_2^4 + 1$. Then for $D_2 \in \mathcal{J}_{C_2'}(\mathbb{F}_{p^k})$, we have*

$$\begin{aligned}\langle D_1, D_2 \rangle_n^{\frac{m(p^k-1)}{n}} &= \left[f_{\lambda_2, D_1}^{\lambda_2^3}(D_2) \cdot f_{\lambda_2, D_1}^{\lambda_2^2}\left(\widehat{\psi}_2(D_2)\right) \cdot f_{\lambda_2, D_1}^{\lambda_2}\left(\widehat{\psi}_2^2(D_2)\right) \cdot \right. \\ &\qquad \left. f_{\lambda_2, D_1}\left(\widehat{\psi}_2^3(D_2)\right) \cdot u_1(D_2) \right]^{\frac{p^k-1}{n}}.\end{aligned}$$

Proof. The proof is similar to that of Theorem 3.4.1. Therefore, we omit it here. ∎

From Theorem 3.4.1 and Theorem 3.4.5, we note that the pairing computation on curve C_2' is more efficient than that on curve C_1' (i.e., more exponentiation computations are needed on curve C_1' than those on curve C_2'). Hence, the following discussions only focus on the curve C_2'.

3.4.2 A New Variant of Miller's Algorithm

In this subsection, we propose a new variant of Miller's algorithm for the family of genus 2 hyperelliptic curves C_2' over $\mathbb{F}_p$ with efficiently computable automorphisms ψ_2 and $\widehat{\psi}_2$. From Theorem 3.4.5, we obtain the following Algorithm 6 for computing the Tate pairing on such curves C_2', which is a variant of Miller's Algorithm (see Algorithm 5 in Section 3.2.3). For the curve C_1', we can also obtain a similar variant of Miller's algorithm as in Algorithm 6, based on Theorem 3.4.1.

3.5 Encapsulated Computation on Genus 2 Curves

In this section we generalize the idea of encapsulated add-and-line and encapsulated double-and-line proposed in [33] to genus 2 hyperelliptic curves over large prime fields. Note that, in the process of computing Tate pairings, one inversion is required for each divisor class addition and doubling, and the calculation of the inversion of an element in large characteristic is usually quite expensive. Therefore, to avoid inversions, we need to derive efficient inversion-free explicit formulae for genus 2 hyperelliptic curves in the context of pairing computations. In the remainder of this chapter, we use I to denote a field inversion, M a field multiplication, and S a field squaring, respectively.

Algorithm 6 Computing the Tate Pairing with Efficient Automorphisms

Input: $\overline{D}_1 = [u_1, v_1] \in \mathcal{J}_{C_2'}(\mathbb{F}_p)[n], \overline{D}_2 \in \mathcal{J}_{C_2'}(\mathbb{F}_{p^k})$, represented by D_1 and D_2 with $\mathrm{supp}(D_1) \cap \mathrm{supp}(D_2) = \emptyset$, $\lambda_2 = (e_r, e_{r-1}, \ldots, e_0)_2$, where $e_i \in \{0, 1\}$

Output: $\langle D_1, D_2 \rangle_n^{m(p^k-1)/n}$

```
1: $T \leftarrow D_1, f_1 \leftarrow 1, f_2 \leftarrow 1, f_3 \leftarrow 1, f_4 \leftarrow 1, f_5 \leftarrow u_1(D_2)$
2: for $i = r-1$ downto 0 do
3:     $\triangleright$ Compute $T'$ and $G_{T,T}(x,y)$ such that $T' = 2T - \mathrm{div}(G_{T,T})$
4:      $\overline{T} \leftarrow [2]\overline{T}, f_1 \leftarrow f_1^2 \cdot G_{T,T}(D_2), f_2 \leftarrow f_2^2 \cdot G_{T,T}(\widehat{\psi}_2(D_2))$
5:      $f_3 \leftarrow f_3^2 \cdot G_{T,T}(\widehat{\psi}_2^2(D_2)), f_4 \leftarrow f_4^2 \cdot G_{T,T}(\widehat{\psi}_2^3(D_2))$
6:     if $e_i = 1$ then
7:         $\triangleright$ Compute $T'$ and $G_{T,D_1}(x,y)$ such that $T' = T + D_1 - \mathrm{div}(G_{T,D_1})$
8:          $\overline{T} \leftarrow \overline{T} \oplus \overline{D}_1, f_1 \leftarrow f_1 \cdot G_{T,D_1}(D_2), f_2 \leftarrow f_2 \cdot G_{T,D_1}(\widehat{\psi}_2(D_2))$
9:          $f_3 \leftarrow f_3 \cdot G_{T,D_1}(\widehat{\psi}_2^2(D_2)), f_4 \leftarrow f_4 \cdot G_{T,D_1}(\widehat{\psi}_2^3(D_2))$
10:    end if
11: end for
12: $f \leftarrow ((f_1^{\lambda_2} \cdot f_2)^{\lambda_2} \cdot f_3)^{\lambda_2} \cdot f_4 \cdot f_5$
13: $f \leftarrow f^{(p^k-1)/n}$
14: return $f$
```

Lange [105] presented efficient explicit formulae for the group operations on genus 2 curves using various systems of coordinates. In the projective coordinate system, the quintuple $[U_1, U_0, V_1, V_0, Z]$ corresponds to the affine class $[x^2+U_1/Zx+U_0/Z, V_1/Zx+V_0/Z]$ in Mumford representation [130], whereas the sextuple $[U_1, U_0, V_1, V_0, Z_1, Z_2]$ stands for the affine class $[x^2 + U_1/Z_1^2x + U_0/Z_1^2, V_1/(Z_1^3Z_2)x + V_0/(Z_1^3Z_2)]$ in the new coordinate system. Lange's formulae are designed to be used in the context of computing scalar multiplications, and do not explicitly calculate all of the rational functions required in Miller's algorithm. However, one can extract the rational functions required from the formulae in [105] at the cost of 3 extra field multiplications.

Choie and Lee [35] modified Lange's explicit formulae in affine coordinates to reduce the cost of extracting the rational functions required in Miller's algorithm. The formulae presented in [35] require $1I + 23M + 3S$ and $1I + 23M + 5S$ in $\mathbb{F}_p$ for divisor class addition[2] and doubling, respectively, thereby saving 2 field multiplications over the previous method. Ó hÉigeartaigh and Scott [80] further optimized the doubling formula proposed in [35] for supersingular genus 2 curves over $\mathbb{F}_p$ of the form $y^2 = x^5 + a$ by saving 1 multiplication and 1 squaring.

[2]We note that the addition formula in [35] requires $3S$ instead of $2S$ as claimed. Indeed, each of Steps 1, 4, and 6 in [35, Table 5] requires a separate squaring.

Based on the above explicit formulae in affine coordinates, we derive new explicit mixed-addition and doubling formulae in the projective and new coordinate systems in the context of pairing computations, respectively. Since the explicit formulae in new coordinates are more efficient than those in projective coordinates, we use new coordinates to represent divisor classes in the main presentation. The mixed-addition and doubling formulae in projective coordinates can be found in the Appendix B. We will explain how to encapsulate the group operations and the line computations in the following subsections. To increase performance, we also enlarge the set of coordinates to $[U_1, U_0, V_1, V_0, Z_1, Z_2, z_1, z_2]$ as in [105], where $z_1 = Z_1^2$ and $z_2 = Z_2^2$.

3.5.1 Encapsulated Divisor Addition and Line Computation

In this subsection, we show how to encapsulate the computation of the line function with the divisor class addition in new coordinates. Given two divisor classes $\overline{E}_1 = [U_{11}, U_{10}, V_{11}, V_{10}, 1, 1, 1, 1]$ and $\overline{E}_2 = [U_{21}, U_{20}, V_{21}, V_{20}, Z_{21}, Z_{22}, z_{21}, z_{22}]$ in new coordinates as inputs, Table 3.1 describes an explicit mixed-addition formula which calculates a divisor class $\overline{E}_3 = [u_3(x), v_3(x)]$ and the rational function $l(x)$ such that $E_1 + E_2 = E_3 + \text{div}\left(\frac{y-l(x)}{u_3(x)}\right)$ in the most common case. Our new explicit formula requires $36M + 5S$ for computing the divisor class addition in new coordinates. Table 3.2 summarizes the computational cost of calculating the divisor class addition and extracting the line function in various coordinate systems. From Table 3.2 we note that in the context of pairing computations our mixed-addition formulae can save $5M$ in the projective coordinate system and $3M$ in the new coordinate system, respectively, when compared to the formulae given by Lange [105].

In the new coordinate system, the rational function $c(x, y) = y - l(x)$ that is required in Miller's algorithm has the following form:

$$c(x, y) = y - \left(\frac{s_1'}{rz_{23}}x^3 + \frac{l_2}{rz_{24}}x^2 + \frac{l_1}{rz_{24}}x + \frac{l_0}{rz_{24}}\right),$$

where $s_1', l_2, l_1, l_0, r, z_{23}$ and $z_{24} = z_{21}z_{23}$ are computed in Table 3.1. By defining the auxiliary rational function $c'(x, y) = (rz_{24})c(x, y)$, we obtain

$$c'(x, y) = (rz_{24})y - ((s_1'z_{21})x^3 + l_2x^2 + l_1x + l_0).$$

Note that the result of evaluating the function $c(x, y)$ at an image divisor D_2 will be raised to the power $(q^k - 1)/n$ $(k > 1)$ in the last step of Miller's algorithm. For efficiency reasons, the first input to the Tate pairing is usually restricted to the 1-eigenspace of the Frobenius endomorphism on $\mathcal{J}_C[n]$. Therefore, we have the following relation

$$c(D_2)^{(q^k-1)/n} = ((c'(D_2)/(rz_{24}))^{q-1})^{(q^{k-1}+q^{k-2}+\ldots+1)/n} = c'(D_2)^{(q^k-1)/n}.$$

Table 3.1: Mixed-Addition Formula on a Genus 2 Curve over $\mathbb{F}_p$ (New Coordinates)

Input	Genus 2 HEC $C : y^2 = x^5 + f_3x^3 + f_2x^2 + f_1x + f_0$ $\overline{E}_1 = [U_{11}, U_{10}, V_{11}, V_{10}, 1, 1, 1, 1]$ and $\overline{E}_2 = [U_{21}, U_{20}, V_{21}, V_{20}, Z_{21}, Z_{22}, z_{21}, z_{22}]$	
Output	$\overline{E}_3 = [U_{31}, U_{30}, V_{31}, V_{30}, Z_{31}, Z_{32}, z_{31}, z_{32}] = \overline{E}_1 \oplus \overline{E}_2$ $l(x)$ such that $E_1 + E_2 = E_3 + \text{div}\left(\frac{y-l(x)}{u_3(x)}\right)$	

Step	Expression	Cost
1	**Compute resultant and precomputations:** $z_{23} = Z_{21}Z_{22}, z_{24} = z_{21}z_{23}, \tilde{U}_{11} = U_{11}z_{21}, y_1 = \tilde{U}_{11} - U_{21}$ $y_2 = U_{20} - U_{10}z_{21}, y_3 = U_{11}y_1, y_4 = y_2 + y_3, r = y_2y_4 + y_1^2U_{10}$	$7M, 1S$
2	**Compute almost inverse of $\boldsymbol{u_2}$ mod $\boldsymbol{u_1}$:** $inv_1 = y_1, inv_0 = y_4$	–
3	**Compute $\boldsymbol{s'}$:** $w_0 = V_{10}z_{24} - V_{20}, w_1 = V_{11}z_{24} - V_{21}, w_2 = inv_0w_0$ $w_3 = inv_1w_1, s'_1 = y_1w_0 + y_2w_1, s'_0 = w_2 - U_{10}w_3$	$7M$
4	**Precomputations:** $\tilde{r} = rz_{23}, R = \tilde{r}^2, Z_{31} = s'_1Z_{21}, Z_{32} = \tilde{r}Z_{21}$ $z_{31} = Z_{31}^2, z_{32} = Z_{32}^2, \tilde{s}'_0 = s'_0z_{21}$	$4M, 3S$
5	**Compute $\boldsymbol{l}$:** $l_2 = s'_1U_{21} + \tilde{s}'_0, l_0 = s'_0U_{20} + rV_{20}$ $l_1 = (s'_1 + s'_0)(U_{21} + U_{20}) - s'_1U_{21} - s'_0U_{20} + rV_{21}$	$5M$
6	**Compute $\boldsymbol{U_3}$:** $w_1 = \tilde{U}_{11} + U_{21}, U_{31} = s'_1(2\tilde{s}'_0 - s'_1y_1) - z_{32}, l'_1 = l_1s'_1$ $U_{30} = \tilde{s}'_0(s'_0 - 2s'_1U_{11}) + s_1'^2(y_3 - \tilde{U}_{10} - U_{20}) + 2l'_1 + Rw_1$	$7M, 1S$
7	**Compute $\boldsymbol{V_3}$:** $w_1 = l_2s'_1 - U_{31}, V_{30} = U_{30}w_1 - z_{31}(l_0s'_1)$ $V_{31} = U_{31}w_1 + z_{31}(U_{30} - l'_1)$	$6M$
Sum		$36M, 5S$

Table 3.2: Divisor Class Addition in Different Systems and in Odd Characteristic

Reference	Coordinate Type	Addition	Mixed Addition	Extracting Line Function $l(x)$
Miyamoto *et al.* [128]	Affine	$1I, 24M, 2S$	–	no cost
	Projective	$54M$	–	no cost
Lange [105]	Affine	$1I, 22M, 3S$	–	$3M$
	Projective	$47M, 4S$	$40M, 3S$	$3M$
	New	$47M, 7S$	$36M, 5S$	$3M$
Choie and Lee [35]	Affine	$1I, 23M, 3S$	–	no cost
Our work	Projective	–	**$38M, 3S$ Table B.1**	**no cost**
	New	–	**$36M, 5S$ Table 3.1**	**no cost**

The above relation means that in new coordinates we can work with the rational function $c'(x,y)$ instead of $c(x,y)$ without altering the value of the resulting Tate pairing. For the same reason we also work with the rational function $u'_3(x) = z_{31}x^2 + U_{31}x + U_{30}$ instead of $u_3(x) = x^2 + \frac{U_{31}}{z_{31}}x + \frac{U_{30}}{z_{31}}$ for both divisor addition and divisor doubling.

3.5.2 Encapsulated Divisor Doubling and Line Computation

In this subsection, we describe how to encapsulate the computation of the line function with the divisor class doubling in new coordinates. Given a divisor class $\overline{E}_1 = [U_{11}, U_{10}, V_{11}, V_{10}, Z_{11},\ Z_{12}, z_{11}, z_{12}]$ in new coordinates as an input, Table 3.3 describes an explicit doubling formula which calculates a divisor class $\overline{E}_3 = [u_3(x), v_3(x)]$ and the rational function $l(x)$ such that $2E_1 = E_3 + \operatorname{div}\left(\frac{y-l(x)}{u_3(x)}\right)$ in the most common case. Our new explicit formula needs $35M + 7S$ to double a divisor class in new coordinates. Table 3.4 summarizes the computational cost of doubling a divisor class and extracting the line function in various coordinate systems. From Table 3.4 we note that in the context of pairing computations our doubling formulae can save $2M$ in both projective and new coordinates, when compared to the formulae given by Lange [105].

In the new coordinate system, the rational function $c(x,y) = y - l(x)$ that is required in Miller's algorithm has the following form:

$$c(x,y) = y - \left(\frac{s_1}{s'_1 Z_{32}}x^3 + \frac{l_2}{Z_{31}Z_{32}}x^2 + \frac{l_1}{Z_{31}Z_{32}}x + \frac{l_0}{Z_{31}Z_{32}}\right),$$

where $s_1, s'_1, l_2, l_1, l_0, Z_{31}$ and Z_{32} are available in Table 3.3. By defining the auxiliary

Table 3.3: Doubling Formula on a Genus 2 Curve over $\mathbb{F}_p$ (New Coordinates)

Input	Genus 2 HEC $C : y^2 = x^5 + f_3x^3 + f_2x^2 + f_1x + f_0$ $\overline{E}_1 = [U_{11}, U_{10}, V_{11}, V_{10}, Z_{11}, Z_{12}, z_{11}, z_{12}]$	
Output	$\overline{E}_3 = [U_{31}, U_{30}, V_{31}, V_{30}, Z_{31}, Z_{32}, z_{31}, z_{32}] = [2]\overline{E}_1$ $l(x)$ such that $2E_1 = E_3 + \text{div}\left(\frac{y-l(x)}{u_3(x)}\right)$	

Step	Expression	Cost
1	**Compute resultant:** $w_0 = V_{11}^2, w_1 = U_{11}^2, w_2 = V_{10}z_{11}$ $w_3 = w_2 - U_{11}V_{11}, r = U_{10}w_0 + V_{10}w_3$	$4M, 2S$
2	**Compute almost inverse:** $inv'_1 = -V_{11}, inv'_0 = w_3$	–
3	**Compute $\boldsymbol{k'}$:** $z'_{11} = z_{11}^2, w_3 = f_3z'_{11} + w_1, \tilde{U}_{10} = U_{10}z_{11}$ $k'_1 = z_{12}(2(w_1 - \tilde{U}_{10}) + w_3), z''_{11} = z_{11}z'_{11}$ $k'_0 = z_{12}(U_{11}(4\tilde{U}_{10} - w_3) + f_2z''_{11}) - w_0$	$7M, 1S$
4	**Compute $\boldsymbol{s'}$:** $w_0 = k'_0inv'_0, w_1 = k'_1inv'_1, s'_1 = w_2k'_1 - V_{11}k'_0, s'_0 = w_0 - \tilde{U}_{10}w_1$	$5M$
5	**Precomputations:** $Z_{31} = s'_1z_{11}, z_{31} = Z_{31}^2, w_0 = rz_{11}, w_1 = w_0Z_{12}$ $Z_{32} = 2w_1Z_{11}, z_{32} = Z_{32}^2, w_2 = w_1^2, R = rZ_{31}$ $S_0 = s_0'^2, S = s'_0Z_{31}, s_0 = s'_0s'_1, s_1 = s'_1Z_{31}$	$8M, 4S$
6	**Compute $\boldsymbol{l}$:** $l_2 = s_1U_{11} + s_0z_{11}, V'_{10} = RV_{10}, l_0 = s_0U_{10} + 2V'_{10}$ $V'_{11} = RV_{11}, l_1 = (s_1 + s_0)(U_{11} + U_{10}) - s_1U_{11} - s_0U_{10} + 2V'_{11}$	$6M$
7	**Compute $\boldsymbol{U_3}$:** $U_{30} = S_0 + 4(V'_{11} + 2w_2U_{11}), U_{31} = 2S - z_{32}$	$1M$
8	**Compute $\boldsymbol{V_3}$:** $w_0 = l_2 - U_{31}, w_1 = w_0U_{30}, w_2 = w_0U_{31}$ $V_{31} = w_2 + z_{31}(U_{30} - l_1), V_{30} = w_1 - z_{31}l_0$	$4M$
Sum		$35M, 7S$

Table 3.4: Divisor Class Doubling in Different Systems and in Odd Characteristic

Reference	Coordinate Type	Doubling	Extracting Line Function $l(x)$
Miyamoto *et al.* [128]	Affine	$1I, 23M, 4S$	no cost
	Projective	$53M$	no cost
Lange [105]	Affine	$1I, 22M, 5S$	$3M$
	Projective	$38M, 6S$	$3M$
	New	$34M, 7S$	$3M$
Choie and Lee [35]	Affine	$1I, 23M, 5S$	no cost
Ó hÉigeartaigh and Scott [80]	Affine	$1I, 22M, 4S$	no cost
Our work	Projective	$\mathbf{39M, 6S}$ **Table B.2**	**no cost**
	New	$\mathbf{35M, 7S}$ **Table 3.3**	**no cost**

rational function $c'(x, y) = (Z_{31}Z_{32})c(x, y)$, we obtain

$$c'(x, y) = (Z_{31}Z_{32})y - ((s_1z_{11})x^3 + l_2x^2 + l_1x + l_0),$$

where z_{11} is also available in Table 3.3. With the same argument as the case of the mixed-addition, we have the relation $c(D_2)^{(q^k-1)/n} = c'(D_2)^{(q^k-1)/n}$ for an image divisor D_2. Therefore, we can simply work with the rational function $c'(x, y)$ instead of $c(x, y)$ without altering the value of the resulting Tate pairing in the new coordinate system.

3.6 Implementing the Tate Pairing with Efficient Automorphisms

In this section, we describe various techniques that enable the efficient implementation of the Tate pairing on a non-supersingular genus 2 curve of type C_2' over $\mathbb{F}_p$. Furthermore, we also analyze the computational cost of our new algorithm in detail and give timings for our implementation.

3.6.1 Curve Generation

While generating suitable parameters for supersingular genus 2 hyperelliptic curves over prime fields is easy, it seems to be more difficult to generate pairing-friendly non-supersingular genus 2 curves over $\mathbb{F}_p$ because of the complicated algebraic structure of hyperelliptic curves. Only a few results have appeared in the literature to address this issue [65,69,83,99] and there is ongoing research in this direction. In [65], Freeman proposed the first explicit construction of pairing-friendly genus 2 hyperelliptic curves over prime fields with ordinary Jacobians by modeling on the Cocks-Pinch method for the elliptic curve case [38]. In the appendix of [65], we find two examples which belong to the curve family C_1' considered in this chapter. Unfortunately, the curve parameters in the two examples are too large to be optimal for efficient implementation. In a recent paper [99], Kawazoe and Takahashi presented two different approaches for explicitly constructing pairing-friendly genus 2 curves of the type $y^2 = x^5 + ax$ over $\mathbb{F}_p$. One is an analogue of the Cocks-Pinch method and the other is a cyclotomic method. Their findings are based on the closed formulae [68, 77] for the order of the Jacobian of hyperelliptic curves of the above type. In this chapter we will restrict to the case $p \equiv 1 \pmod 8$ and generate a suitable non-supersingular pairing-friendly genus 2 curve C_2' with embedding degree 4 using the theorems in [99]. The reason that we only consider curves with embedding degree 4 in this section is to facilitate performance comparisons between supersingular and non-supersingular genus 2 curves. However, we would like to point out that non-supersingular curves with higher embedding degree are available from [99] and that our method is also applicable to such curves.

To find good curve parameters which are suitable for applying our new algorithm, we use the following searching strategies. From Theorem 3.4.5 we note that the subgroup order n should satisfy $mn = \lambda_2^4 + 1$ for an integer m. Assume that we require the (160/1024) bit security level. Then n is a prime around 160 bits and λ_2 is at least 40 bits[3]. Furthermore, since the bit length of λ_2 determines the length of the loop in Algorithm 6, λ_2 should be taken as small as possible. Based on these observations, we first check all λ_2's of the form $\lambda_2 = 2^a, a \in \{41, 42, \ldots, 60\}$. We found two λ_2's, namely $\lambda_2 = 2^{58}$ and 2^{59}, for which $\lambda_2^4 + 1$ has a prime factor of 164 bits and 162 bits, respectively. However, using the above two primes as the order n of the subgroup and running the algorithms of [99], we could not find any curve. Therefore, we consider the slightly more complicated choice of $\lambda_2 = 2^a + 2^b$, where $a \in \{41, 42, \ldots, 50\}, b \in \{1, 2, \ldots, 50\}$ and $a > b$. After a couple of trials, we found that choosing $\lambda_2 = 2^{43} + 2^{10}$ generates a non-supersingular pairing-friendly genus 2 hyperelliptic curve whose Jacobian has embedding degree 4 with respect to a 163-bit prime n. The curve is given by the equation

$$C_2^* : y^2 = x^5 + 9x$$

over $\mathbb{F}_p$, for a 329-bit prime p, where the hexadecimal representations of n and p are as

[3]If we require the (256/3072) bit security level, we can use the method in [99] to generate a suitable non-supersigular pairing-friendly genus 2 hyperelliptic curve with embedding degree 6. In this case, n is a prime around 256 bits, p is about 512 bits and λ_2 is at least 64 bits.

follows:

```
n  =  00000006 a37991af 81ddfa3a ead6ec83 1ca0fc44 75d5add9 (163 bits)
p  =  0000016b 953ca333 acf202b3 0476f30f ff085473 6d0a0be4
      c542fa48 66e5afba 7bc6cd6d 21ca9fad eef796f1          (329 bits)
```

In the following five subsections, we will detail various techniques required to efficiently implement the calculation of the Tate pairing on the curve C_2^*.

3.6.2 Finite Field Arithmetic

As the embedding degree of the curve C_2^* in our implementation is $k = 4$, we first discuss how to construct the finite field $\mathbb{F}_{p^4}$. Rather than construct $\mathbb{F}_{p^4}$ as a direct quartic extension of $\mathbb{F}_p$, the best way is to define the field $\mathbb{F}_{p^4}$ as a quadratic extension of $\mathbb{F}_{p^2}$, which is in turn a quadratic extension of $\mathbb{F}_p$. Since the p is congruent to 5 modulo 12 in our implementation, the field $\mathbb{F}_{p^2}$ can be constructed by the irreducible binomial $x^2 + 3$ and the field $\mathbb{F}_{p^4}$ can be constructed as a quadratic extension of $\mathbb{F}_{p^2}$ by the irreducible binomial $x^2 - \sqrt{-3}$. Letting $\beta = -3$, elements of the field $\mathbb{F}_{p^2}$ can be represented as $a + b\sqrt{\beta}$, where $a, b \in \mathbb{F}_p$, whereas elements of the field $\mathbb{F}_{p^4}$ can be represented as $c + d\sqrt[4]{\beta}$, where $c, d \in \mathbb{F}_{p^2}$. Under this tower construction, a multiplication of two elements and a squaring of one element in $\mathbb{F}_{p^4}$ cost $9M$ and $6M$ in $\mathbb{F}_p$, respectively [80].

3.6.3 Encapsulated Group Operations

Applying the explicit formulae in Section 3.5 to the curve C_2^* defined above, we can calculate the divisor class addition and doubling with $36M + 5S$ and $32M + 6S$ in $\mathbb{F}_p$ in new coordinates, respectively.

3.6.4 Using Degenerate Divisors and Denominator Elimination

For a hyperelliptic curve of genus $g > 1$, using a degenerate divisor as the image divisor is more efficient than using a general divisor when evaluating line functions. Frey and Lange [66] discussed in detail when it is permissible to choose a degenerate divisor as the second argument of Miller's algorithm. They also note that, when the embedding degree k is even, one might choose the second pairing argument from a set $S = \{(x, y) \in C(\mathbb{F}_{q^k}) \mid x \in \mathbb{F}_{q^{k/2}}, y \in \mathbb{F}_{q^k} \backslash \mathbb{F}_{q^{k/2}}\}$. Note that the point in the set S is on the quadratic twist of

$C/\mathbb{F}_{q^{k/2}}$. When considering C_2^* as a curve defined over $\mathbb{F}_{p^2}$, we can define a quadratic twist of C_2^* over $\mathbb{F}_{p^2}$, denoted by $C_{2,t}^*$, as follows

$$C_{2,t}^* : y^2 = x^5 + 9c^4x,$$

where $c \in \mathbb{F}_{p^2}$ is a quadratic non-residue over $\mathbb{F}_{p^2}$. It is known that $C_{2,t}^*(\mathbb{F}_{p^4}) \cong C_2^*(\mathbb{F}_{p^4})$ and the isomorphism of $C_{2,t}^*(\mathbb{F}_{p^4})$ and $C_2^*(\mathbb{F}_{p^4})$ also induces an isomorphism ϕ of $\mathcal{J}_{C_{2,t}^*}(\mathbb{F}_{p^4})$ and $\mathcal{J}_{C_2^*}(\mathbb{F}_{p^4})$ [103]. As in [62] we first generate a degenerate divisor class $\overline{D}_t = [x - x_t, y_t] \in \mathcal{J}_{C_{2,t}^*}(\mathbb{F}_{p^2})$ on the twisted curve $C_{2,t}^*/\mathbb{F}_{p^2}$. Then the isomorphism ϕ will map $\overline{D}_t$ to a degenerate divisor class $\overline{D}_2 = \phi(\overline{D}_t) = [x - c^{-1}x_t, c^{-5/2}y_t] \in \mathcal{J}_{C_2^*}(\mathbb{F}_{p^4})$ on the curve C_2^* over $\mathbb{F}_{p^4}$. Note that the denominator elimination technique applies in this case since $x - c^{-1}x_t$ is defined over $\mathbb{F}_{p^2}$. Furthermore, we do not need to compute $f_5 = u_1(D_2) \in \mathbb{F}_{p^2}$ in Algorithm 6 either, for the same reason.

3.6.5 Evaluating Line Functions at A Degenerate Divisor

Here we consider the evaluation of line functions at a degenerate divisor $D_2 = [x - x_2, y_2] \in \mathcal{J}_{C_2^*}(\mathbb{F}_{p^4})$ generated by the method in Section 3.6.4, where $x_2 = c^{-1}x_t \in \mathbb{F}_{p^2}$ and $y_2 = c^{-5/2}y_t \in \mathbb{F}_{p^4} \backslash \mathbb{F}_{p^2}$. Moreover, we further assume that in this work $c = \sqrt{-3}$ is taken as a quadratic non-residue over $\mathbb{F}_{p^2}$. Therefore, y_2 has only two non-zero coefficients instead of four in a polynomial basis representation of $\mathbb{F}_{p^4}$. Furthermore, since the denominator elimination technique applies in this case, we only need to evaluate the numerators of the rational functions at D_2. From Section 3.5 we know that in new coordinates we can respectively work with the numerators $c_1(x, y) = (Z_{31}Z_{32})y - ((s_1z_{11})x^3 + l_2x^2 + l_1x + l_0)$ for group doubling and $c_2(x, y) = (\tilde{r}z_{21})y - ((s_1'z_{21})x^3 + l_2x^2 + l_1x + l_0)$ for group addition, where $Z_{31}, Z_{32}, \tilde{r}, z_{11}, z_{21}, s_1, s_1', l_2, l_1$ and l_0 are from Table 3.1 and Table 3.3. Note that in Algorithm 6 we need to evaluate the function $c_i(x, y), i = 1$ or 2 at four image divisors $D_2 = [x - x_2, y_2], \widehat{\psi}_2(D_2) = [x - \xi_8^{-2}x_2, \xi_8^{-1}y_2], \widehat{\psi}_2^2(D_2) = [x - \xi_8^4x_2, \xi_8^{-2}y_2] = [x + x_2, \xi_8^{-2}y_2]$ and $\widehat{\psi}_2^3(D_2) = [x - \xi_8^2x_2, \xi_8^{-3}y_2]$ for each iteration of the loop. Hence we have the following relations

$$\begin{aligned}
c_i(D_2) &= (\tilde{r}z_{11})y_2 \quad\quad - [((s_1'z_{11})x_2^3 + l_1x_2) \quad + (l_2x_2^2 + l_0)],\\
c_i(\widehat{\psi}_2(D_2)) &= ((\tilde{r}z_{11})y_2)\xi_8^{-1} - [((s_1'z_{11})x_2^3 - l_1x_2)\xi_8^2 - (l_2x_2^2 - l_0)],\\
c_i(\widehat{\psi}_2^2(D_2)) &= ((\tilde{r}z_{11})y_2)\xi_8^{-2} + [((s_1'z_{11})x_2^3 + l_1x_2) \quad - (l_2x_2^2 + l_0)],\\
c_i(\widehat{\psi}_2^3(D_2)) &= ((\tilde{r}z_{11})y_2)\xi_8^{-3} + [((s_1'z_{11})x_2^3 - l_1x_2)\xi_8^2 + (l_2x_2^2 - l_0)].
\end{aligned}$$

We assume that x_2^2, x_2^3, ξ_8^{-1} and ξ_8^2 are precomputed with $7M + 2S$ over $\mathbb{F}_p$. Since x_2, x_2^2 and x_2^3 are in $\mathbb{F}_{p^2}$ and y_2 has only two non-zero coefficients in the polynomial basis representation of $\mathbb{F}_{p^4}$, $c_i(D_2)$ can be computed with $10M$ over $\mathbb{F}_p$. By reusing the intermediate computation results, we can compute $c_i(\widehat{\psi}_2(D_2)), c_i(\widehat{\psi}_2^2(D_2))$ and $c_i(\widehat{\psi}_2^3(D_2))$ with $4M$, $2M$ and $2M$ over $\mathbb{F}_p$, respectively. Therefore, the total cost of evaluating the function $c_i(x, y)$ at the degenerate divisor D_2 is $18M$ over $\mathbb{F}_p$ per iteration, with a precomputation of $7M + 2S$. For the case of evaluating the rational functions at a general divisor, the reader is referred to [62].

3.6.6 Final Exponentiation

For a genus 2 curve with an embedding degree of $k = 4$, the output of Miller's algorithm needs to be raised to the power of $(p^4 - 1)/n$. Calculating this exponentiation can follow two steps as shown in [80]. Letting $f \in \mathbb{F}_{p^4}$ be the output of Miller's algorithm, the first step is to compute f^{p^2-1} which can be obtained with a conjugation with respect to $\mathbb{F}_{p^2}$ and $1I + 1M$ in $\mathbb{F}_{p^4}$. The remaining exponentiation to $(p^2 + 1)/n$ is an expensive operation which can be efficiently computed with the Lucas laddering algorithm [153] for the curve C_2^* in question.

3.6.7 Performance Analysis and Comparison

In this subsection, we analyze the computational complexity of the Algorithm 6 for calculating the Tate pairing on non-supersingular genus 2 hyperelliptic curves C_2' and compare the performance of pairing computations on supersingular and non-supersingular genus 2 curves over prime fields with the same embedding degree of $k = 4$.

We first analyze the algebraic complexity of the pairing computation on curves C_2' with our new algorithm (see Section 3.4.2). Recall that n is the subgroup order and λ_2 is a root of the equation $\lambda^4 + 1 \equiv 0 \bmod n$. We assume that the embedding degree k is even and the line functions in Algorithm 6 are evaluated at a degenerate divisor D_2 instead of a general divisor for efficiency reasons. We also assume that λ_2 has a random Hamming weight, meaning that about $\left(\frac{1}{2}\log_2 \lambda_2\right)$ additions take place in Algorithm 6 on average. Then the algebraic cost for computing the Tate pairing is given by (without including the cost of the final exponentiation)

$$T_1 + (\log_2 \lambda_2)(T_D + T_G + 4T_{sk} + 8T_{mk}) + \left(\frac{1}{2}\log_2 \lambda_2\right)(T_A + T_G + 8T_{mk}) + T_{10},$$

where

1. T_1 – the cost of precomputing f_5 in Step 1 of Algorithm 6.

2. T_D – the cost of doubling a general divisor.

3. T_A – the cost of adding two general divisors.
4. T_G – the cost of evaluating rational function $G(x, y)$ at the image divisors $D_2, \widehat{\psi}_2(D_2), \widehat{\psi}_2^2(D_2)$ and $\widehat{\psi}_2^3(D_2)$.
5. T_{sk} – the cost of squaring in $\mathbb{F}_{p^k}$.
6. T_{mk} – the cost of multiplication in $\mathbb{F}_{p^k}$.
7. T_{10} – the cost of computing f in Step 10 of Algorithm 2.

When applying various optimization techniques detailed in previous subsections to the particular curve C_2^*, we can further reduce the above cost of computing the Tate pairing to

$$43 \cdot (T_D + T_G + 4T_{sk} + 4T_{mk}) + (T_A + T_G + 4T_{mk}) + T_{10},$$

where $T_D = 32M + 6S, T_A = 36M + 5S, T_G = 18M, T_{sk} = 6M, T_{mk} = 9M$ and $T_{10} = 828M$. Furthermore, we also need $7M + 2S$ for precomputations (see Section 3.6.5). Note that all multiplications and squarings here are over $\mathbb{F}_p$. Therefore, the total cost of computing the Tate pairing with our optimizations is given by $5655M + 265S$ in $\mathbb{F}_p$.

In [35, 62, 80], the authors examined the implementation of the Tate pairing on a family of supersingular genus 2 hyperelliptic curves C_1 (see Section 3.3.1) over prime fields with embedding degree 4 in affine and projective coordinates, respectively. We compare the performance of pairing computations on the supersingular curve C_1 and the non-supersingular curve C_2^* in the following Table 3.5. Note that both curves have the same embedding degree of $k = 4$.

Table 3.5: Performance Comparison of Pairing Computation on Curves C_1 and C_2^*

Reference	Curve Equation	Coordinate Type	Cost
Choie and Lee [35]	$C_1 : y^2 = x^5 + a$, $a \in \mathbb{F}_p^*, p \equiv 2, 3 \bmod 5$	Affine	$240I, 17688M, 2163S$
Ó hÉigeartaigh & Scott [80]		Affine	$162I, 10375M, 645S$
Fan, Gong and Jao [62]		Projective	$13129M, 967S$
		New	$12487M, 971S$
This work	$C_2^* : y^2 = x^5 + 9x$, $p \equiv 1 \bmod 8$	New	$\boldsymbol{5655M, 265S}$

From Table 3.5, we note that for the same security level the computation of the Tate pairing on the non-supersingular genus 2 curve C_2^* is algebraically about 55.8% faster than on the supersingular genus 2 curve C_1, under the assumption that field squarings have cost $S = 0.8M$. Therefore, our algorithm improves previous work for pairing computations on genus 2 hyperelliptic curves over prime fields by a considerable margin.

3.6.8 Experimental Results

For verifying our theoretical analysis in Section 3.6.7, we report implementation results of computing the Tate pairing on the supersingular genus 2 curve C_1 and non-supersingular genus 2 curve C_2^* in this section. Both curves are defined over $\mathbb{F}_p$ and have an embedding degree of $k = 4$. The code was written in C and Microsoft Developer Studio 6 was used for compilation and debugging on a Core 2 Duo™@2.67 GHz processor. For the curve C_1 and the (160/1024) bit security level we use the curve parameters that are generated in [80], where the subgroup order $n = 2^{159} + 2^{17} + 1$ is a Solinas prime [161] and the characteristic p of the finite field $\mathbb{F}_p$ is a 256-bit prime. Recall that the curve C_2^* is defined over a prime field of size 329 bits (see Section 3.6.1). The operations in the above two prime fields are implemented with various efficient algorithms in [79]. Table 3.6 shows the timings of our finite field library and the corresponding IM-ratio. From Table 3.6 we note that the IM-ratios are sufficiently large for the two prime fields in our implementation that using new coordinates and encapsulated group operations can provide better performance than using affine coordinates in this case.

Table 3.6: Timings of Prime Field $\mathbb{F}_p$ Library

Curve	# of bits of p	Multiplication (M)	Squaring (S)	Inversion (I)	IM-ratio
C_1	256	$0.84\mu s$	$0.78\mu s$	$41.9\mu s$	53.7
C_2^*	329	$1.40\mu s$	$1.30\mu s$	$64.6\mu s$	46.1

Table 3.7 summarizes previous work and our experimental results for the implementation of the Tate pairing on the curve C_1 and C_2^* for the (160/1024) bit security level. All of the timings are given in milliseconds.

From Table 3.7, we note that in our implementation the pairing computation on the curve C_2^* is about 10% faster than that on the curve C_1, in contrast to the algebraic complexity analysis in Section 3.6.7. The reason is that the sizes of the fields over which both curves are defined are different. Observe that the curve C_2^* is defined over a larger prime field than C_1, which significantly decreases the speed of computing the final exponentiation of the Tate pairing when the curve C_2^* is used. This explains why our new algorithm only obtains a 10% performance improvement in the implementation. Unfortunately, under current techniques for generating pairing-friendly non-supersingular genus 2 hyperelliptic

Table 3.7: Experimental Results – (160/1024) Security Level

Reference	Curve	Coordinate Type	Subgroup Order	Running Time (ms)
Choie and Lee [35]	C_1	Affine	Random	515
Ó hÉigeartaigh and Scott [80]	C_1	Affine	$n = 2^{159} + 2^{17} + 1$	16
This work	C_1	New	$n = 2^{159} + 2^{17} + 1$	14.6
	C_2^*	New	$\lambda_2 = 2^{43} + 2^{10}$	13.1

curves, we cannot find such a curve of the form $y^2 = x^5 + ax$ defined over a 256-bit prime field with an embedding degree of $k = 4$. Nevertheless, despite the unequal field size, our implementation on the curve C_2^* is still slightly faster, even though strictly speaking a direct comparison between fields of different size is complicated as many factors could affect the comparison one way or another.

3.7 Conclusions

In this chapter, we have proposed new variants of Miller's algorithm for computing the Tate pairing on two families of non-supersingular genus 2 hyperelliptic curves over prime fields with efficiently computable automorphisms. We describe how to use the automorphisms to unroll the main loop of Miller's algorithm. In the best case, the length of the loop in our new variant can be up to 4 times shorter than that of the original Miller's algorithm.

Furthermore, we also show how to efficiently implement pairing computations on genus 2 hyperelliptic curves over prime fields in projective coordinates. We generalize Chatterjee *et al.*'s idea of encapsulated double-and-line computation and add-and-line computation to genus 2 curves in projective and new coordinates, respectively. We also show that some of the operations in the encapsulated method do not need to be computed since they are eliminated by the final exponentiation. Our new explicit formulae are applicable to pairing computations on both supersingular and non-supersingular genus 2 curves.

As a case study, we combine our new algorithm with various optimization techniques in the literature to efficiently implement the Tate pairing on a non-supersingular genus 2 curve $y^2 = x^5 + 9x$ over $\mathbb{F}_p$ with an embedding degree of $k = 4$. We also analyze the performance for the new algorithm in detail. When compared with pairing computations on supersingular genus 2 curves at the same security level, our new algorithm can obtain 55.8% performance improvements algebraically. However, the size of the field where the non-supersingular curve is defined is 1.285 times larger than that of the field used for supersingular curves, which somewhat offsets these gains. Nevertheless, our experimental results show that using the non-supersingular genus 2 curve one can still obtain a 10% performance improvement over the supersingular curve.

Chapter 4

Key Revocation Based on Dirichlet Multinomial Model for MANETs

The absence of an online trusted authority makes the issue of key revocation in MANETs particularly challenging. In this chapter we present a novel self-organized key revocation scheme based on the Dirichlet multinomial model and identity-based cryptography (IBC). Our key revocation scheme offers a theoretically sound basis for a node in MANETs to predict the behavior of other nodes based on its own observations and reports from peers. In our scheme, each node keeps track of three categories of behavior defined and classified by an external trusted authority, and updates its knowledge about other nodes' behavior with 3-dimension Dirichlet distribution. Differentiating between suspicious behavior and malicious behavior enables nodes to make multilevel response by either revoking keys of malicious nodes or ceasing the communication with suspicious nodes for some time to gather more information for making further decision. Furthermore, we also analyze the attack-resistant properties of our key revocation scheme through extensive simulations in the presence of independent and collusive adversaries, respectively. First we review existing solutions and describes the motivation for our work in Section 4.1. Section 4.2 gives a short introduction to mathematical tools used in this chapter including cryptographic pairings and Dirichlet multinomial model. Section 4.3 formulates the network and security models and presents our design goals. Section 4.4 gives a detailed description of our key revocation scheme, followed by simulations and analysis of our key revocation scheme under false statement attacks by independent and collusive adversaries in Section 4.5. This chapter is finally concluded in Section 4.6. Partial contents of this chapter have been published in [59].

4.1 Related Work and Motivation

Revoking cryptographic keys or certificates of malicious nodes is crucial for the security and robustness of MANETs. Namely, good nodes can isolate malicious ones from the network by ceasing the further communication with them and ignoring any message received from them. Therefore, if cryptographic keys or certificates are issued by an authority, it must possible, whenever necessary (e.g., key compromise), for the authority to revoke them, and essentially evict malicious nodes from the network. In the context of wired networks, implementations of key revocation schemes are usually based on Public Key Infrastructures (PKIs). When the certificate of some user is to be revoked, the certificate authority (CA) adds user's certificate information into a Certificate Revocation List (CRL) and puts it on an on-line trusted public repository or distributes it to other relevant users in some secure way. However, these conventional techniques are difficult to be applied to MANETs because of unique features of MANETs such as the absence of an on-line CA and a centralized repository. Two main classes of solutions have been proposed for key revocation in MANETs. The schemes in the first category employ threshold cryptography and network nodes collaborate to revoke keys of malicious nodes. Those in the second category are fully self-organized and each node has its own opinion about the network and other nodes' behavior. These solutions can be implemented with either the certificate-based cryptography (CBC) or identity-based cryptography (IBC). Furthermore, some novel ideas have also been proposed in the literature, which can be used to fast remove malicious nodes from MANETs in particular application scenarios.

4.1.1 Threshold Cryptography Based Key Revocation Schemes

Key revocation schemes that depend on the use of a centralized trusted third party (TTP) are not well suited to the ad hoc network scenario due to several reasons. Firstly, a CA will be a vulnerable point in the network, especially if it is not distributed. If the CA is compromised, the security of the entire network is breached. More importantly, the CA should be available all the time in order to manage keys of network nodes. A typical approach to solve these problems is to distribute the services of a centralized CA to a set of network nodes using threshold cryptography [156]. And then these network nodes can collaboratively carry out the key revocation. Although this kind of key revocation schemes do not require the establishment of any infrastructure, the use of threshold cryptography may cause tremendous computation and communication overhead on the network. We next describe several key revocation schemes based on threshold cryptography for MANETs. In the following discussion, N denotes the overall number of network nodes and t and n are two positive integers satisfying $t \leq n < N$.

Partially Distributed Authority

In the seminal paper by Zhou and Hass [182], the authors use CBC and a (t, n) threshold scheme to distribute the services of the CA to a set of specialized server nodes in MANETs. The system contains three types of nodes, namely, client, server and combiner nodes. The client nodes are the normal users of the network, whereas the server and combiner nodes are part of the CA. The system can tolerate $t - 1$ compromised servers due to the use of threshold cryptography. Although the authors mentioned that the servers can collaborate to revoke the certificates of the malicious nodes, no algorithms about the certificate revocation are provided.

In [179], Zhang *et al.* designed a key management mechanism called IKM for MANETs by combining IBC and threshold cryptography. The authors described a novel construction method of ID-based public/private keys. In IKM, an external TTP distributes its functionality to n Distributed Private Key Generators (D-PKGs) and bootstraps the network with identity-based cryptography. More specifically, the TTP generates two master keys and only introduces the information of one of master keys into the network using a (t, n) threshold scheme. Therefore, each node has two pairs of keys, a static one issued by the TTP and one that depends on the current time interval issued by the on-line D-PKGs. By this means, IKM guarantees high-level resilience to node compromise attacks. Keys are updated periodically through the broadcasts of the D-PKGs. In their key revocation protocol, each node observing malicious behavior of other nodes securely reports its signed accusations to the preassigned D-PKGs with ID-based signature and encryption schemes. When the number of accusations against a malicious node reaches the revocation threshold in a predetermined time window, the t D-PKGs will collaborate to revoke the key of the malicious node with an ID-based (t, n)-threshold signature scheme. Although the design of IKM minimizes the damage from node compromise attacks, the procedure of key revocation involves a lot of expensive pairing computations, which means that the network nodes in MANETs should have sufficient computational and power resources.

Fully Distributed Authority

In [102] and [117], a localized key management scheme was proposed for MANETs. The authors use a (t, N) threshold scheme to distribute the capabilities of the CA to all nodes in MANETs. In addition, the authors also briefly described a localized certificate revocation scheme in a single paragraph. In their scheme, each node monitors the behavior of its one-hop neighboring nodes. The key revocation will happen in two cases. One is when node A observes that one of its neighbors is misbehaving. Node A will directly mark its neighbor as "convicted". Furthermore, node A also disseminates its signed accusations to its m-hop neighborhoods with a RSA signature scheme, where m is a design parameter denoting the range of the accusation propagation. The other case is when node A receives an accusation against node B from node C. Node A first verifies whether the accuser C can be trusted by checking its own certificate revocation lists. If it is, it verifies the signature of the accuser C and updates its CRLs accordingly. Otherwise node A ignores the accusation received from C. When the number of accusations for one node reaches a predefined network-wide revocation threshold, the certificate of that node will be revoked. Furthermore, each node only holds each entry in a CRL for some time T_{cert} (which is defined as the valid period of a certificate) so that it will not provide the certificate update service for a convicted node that still have a valid certificate. Therefore, the convicted node will be evicted from the network after a period of T_{cert}. Although the proposed key revocation scheme meets some requirements of MANETs, it is vulnerable to the Sybil attack [49] where an malicious node can create a large number of identities to collect enough shares and reconstruct the CA's private key [30].

In [151], Saxena *et al.* proposed ID-GAC, an elegant identity-based access control scheme for ad hoc groups such as MANETs. ID-GAC uses a (t, N) threshold scheme to share the master key among all network nodes before the deployment. In particular, the authors presented a membership revocation mechanism based on Membership Revocation Lists (MRLs), which are the analogues of the CRLs used in the traditional PKI. Their scheme is basically similar to those in [102] and [117] except that the signed accusations are broadcasted to the entire network and a identity-based threshold signature scheme is used to update key shares. The disadvantages of ID-GAC include: (i) Broadcasting accusations to the entire network introduces a lot of communication load; (ii) The procedure of key revocation needs many expensive pairing computations which impose a large computational overhead for mobile devices in MANETs; and (iii) ID-GAC suffers from the same undesirable security drawback as the schemes in [102] and [117].

4.1.2 Self-Organized Key Revocation Schemes

The above key revocation schemes use threshold cryptography and therefore some nodes need to collaborate to revoke keys of malicious nodes, whereas several fully self-organized schemes are also proposed in the literature. In self-organized key revocation schemes, each node has its own view about the network and decides whether cryptographic keys of other network nodes should be revoked based on its own observations and the information collected from peers in MANETs. These self-organized key revocation schemes do not require any infrastructure and on-line access to trusted authorities.

In [30], Capkun *et al.* presented a self-organized key management scheme for MANETs. This scheme uses a PGP-like trust model [183] and allows nodes to certify each other. When a user want to revoke a certificate that he issued, the user sends an explicit revocation statement to the nodes that regularly update that certificate. The disadvantage of this scheme is that the assumption of a transitive trust might be too strong for MANETs.

The scheme proposed in [40] by Crépeau and Davis is the first self-organized certificate revocation scheme for MANETs. Their protocol uses a weighted accusation scheme and provides the protection against the potentially false accusation attacks from malicious nodes. The scheme is further improved and extended in [3]. Here, the value of a node's trustworthiness determines the weight of its accusation. The weight of node A's accusations depends on the number of accusations against node A, as well as the number of additional accusations made by node A. The authors presented a method for actually quantifying the trustworthiness of the nodes in MANETs satisfying that accusations from the trustworthy nodes will have higher weight than those from less trustworthy nodes. The underline principle of their scheme is that the weight of a node's accusation is zero if the trustworthiness of the node is the minimum possible value (i.e., the node receives the maximum number of accusations from peers) and the node made the maximum number of accusations that is allowed at the same time. All accusations are frequently broadcasted throughout the entire network. Moreover, the newly joining nodes will receive the signed profile tables of the existing members of the MANET from peers in order to obtain the up-to-date information about the behavior profile of other nodes. The certificate of a node is revoked when the sum of the weighted accusations against the node is equal to or greater than a configurable threshold (revocation quotient). Their scheme does not need the time synchronization and any access to on-line CAs after each node gets the certificate from an off-line CA during network initialization. Furthermore, one-way hash chain is employed to provide the authentication of the data origins and the integrity check of massages. Crépeau *et al.*'s key revocation scheme uses the idea from game theory very tactfully and provides a strong protection against certificates being wrongfully revoked through false accusations attacks from malicious nodes. However their scheme also has the following disadvantages: (i) Broadcasting accusations to the entire network introduces tremendous communication overhead; (ii) Using μTESLA-based techniques [142] to authenticate accusations is vulnerable to denial-of-service attacks [134]; (iii) The newly joining nodes need to verify a large amount of profile tables received from peers, which means that these nodes should have sufficient computational and power resources.

In [86], Hoeper and Gong presented self-organized key revocation and key renew schemes for MANETs. These schemes are further analyzed in [84]. The authors introduced a new format involving node's identity, key expiry date, and key version number for ID-based public keys such that new keys can be issued for the same identity after the previous key has been revoked. In their revocation scheme, each node uses a neighborhood watch mechanism to monitor nodes within its communication range. Upon detection of malicious behavior, these observations are then securely propagated to an m-hop neighborhood (m denotes the range of the accusation propagation) using the pre-shared secret key obtained from a non-interactive ID-based key agreement protocol. Furthermore, when a node realizes that its private key has been compromised, the node will generate a harakiri message and propagate this message to all its m-hop neighbors. Node A will consider node B's public key as revoked if at least one of the following three conditions is true: (i) A observes the malicious behavior of B through the neighborhood watch; (ii) A receives a harakiri message from B declaring that its private key has been compromised; (iii) A receives at least δ accusations against B from trustworthy nodes within node A's m-hop neighborhood, where δ is a predetermined revocation threshold. The authors also used the majority vote with parameter ε to mitigate the influence of false accusation attacks from colluding l-hop neighbors ($2 \leq l \leq m$). In addition, newly joining nodes can simply join the network and start the key revocation scheme without first verifying a large number of past accusations.

4.1.3 Other Key Revocation Schemes

In [116], Luo *et al.* proposed a certificate management scheme called **DICTATE** (**DI**stributed **C**er**T**ification **A**uthority with probabilis**T**ic fr**E**shness) for special MANETs that have intermittent connections to a mother certificate authority (mCA), which is a trusted authority connected to the backbone and known to all network nodes. Their scheme is based on the combination of an offline mCA that issues initial certificates for nodes and a number of online distributed servers (dCAs) that process certificate update and query requests. To prevent the key compromise, the scheme requires nodes' certificates to be updated periodically. Otherwise the certificates will become invalid. Periodically, there is a check time, at which the dCAs (physically) go back to the mCA for purgation (only distributed CA servers should go through this procedure and clients can still perform their remote operations). During the check time, the mCA, through out-of-band mechanisms, detects compromised servers and has them re-initiated or substituted by new ones. It also refreshes the secret shared among the dCAs.

In [37], Clulow and Moore introduced the concept of the suicide for solving the problem of the credential revocation in self-organizing systems for the first time. Their work is further extended and analyzed in [129]. The basic idea of the suicide attack is extremely simple: a node who observes another node's malicious behavior simply broadcasts a signed message to the entire network and claims both of them to be dead. Therefore, a good node that unilaterally removes a malicious node from MANETs sacrifices its own participation in the future network operations as well. This radical strategy can fast isolate the malicious nodes from the network and is ideally suited to highly mobile networks or special-purpose MANETs, for example those deployed in a military battlefield, in which all entities belong to a group and have the common benefits and goals. The suicide protocol is fully self-organized and incurs the low communication and storage overhead. The authors also described possible implementations and proposed various countermeasures to mitigate the abuse of this mechanism. More specifically, when a centralized trusted authority is available, it will broadcast the suicide note received from an accuser to the entire network. If there is no trusted authority in MANETs but nodes are capable of performing asymmetric primitives, an accuser will broadcast its signed suicide node accompanied with its public key certificate. To mitigate the abuse of the suicide mechanism, the authors also suggested that accusers attach a timestamp for their suicide nodes in order to resolve potential conflicts. Furthermore, each node should wait a long enough period for duplicate suicide notes and only accept the one with the earliest timestamp.

Besides the suicide attacks, Moore *et al.* also proposed reelection-based key revocation schemes for relatively static MANETs. Reelection is in fact a kind of access control mechanism in which good nodes collaborate to periodically renew their own membership. If a malicious node cannot update its network access token, it will be evicted from the network automatically. The authors describe two implementation options of the reelection protocol. The first one is based on the combination of threshold secret sharing and one-way hash chain techniques. Prior to the deployment, an external authority uses a one-way hash chain to determine a set of network tokens that nodes will use to demonstrate their membership in each time period. For each node, the authority then distributes the shares of its network tokens, and the anchor of the hash chain to all its neighbors. Hash tree based authentication mechanism is employed by each node to verify the received shares from its neighbors. The network access token of a malicious node will be automatically revoked when a number of its neighbors delete the stored shares of that token. Another way to implement the reelection is to use the buddy list. The basic idea is that each node periodically and locally broadcasts a buddy list of its approved neighbors, and the receivers cross-reference these lists to determine whether to trust a node or not. A μTESLA-like broadcast authentication mechanism is used for nodes to authenticate received buddy lists.

4.1.4 Motivation

We note that the key revocation procedure involves observations and interactions among nodes. Therefore, it is closely related to reputation and trust of nodes in the network. This observation allows us to design a key revocation scheme based on the decentralized reputation system. Reputation systems have been investigated extensively in the past and used successfully in many commercial online applications [97]. They provide a mechanism for rating participants of transactions by having buyers and sellers compute each other reputation scores, and therefore stimulate good behavior as well as sanction bad behavior. In the context of MANETs, reputation systems have emerged as a promising mechanism for ensuring cooperation and fairness, and thwarting node failures and malicious attacks [22, 24, 124]. However, the previous reputation systems all classify the behavior of nodes in MANETs as either *good* or *bad* without any intermediate state. Such a binary behavior differentiation omits the actual cause and the degree of the misbehavior. Note that some misbehavior may just happen accidently (for example, a node cannot forward packages due to temporary congestion of the network) and last only for a short time. When a node shows this kind of accidental misbehavior, it might not mean that the node has been compromised by an attacker. Therefore, in this case it is more reasonable to keep collecting information about the behavior of this node instead of immediately characterizing it as malicious and excluding it from the network.

To provide more flexibility and precision for nodes analyzing peers' behavior and making different response based on results of the analysis, we present a novel self-organized key revocation scheme based on IBC and Dirichlet multinomial model in this chapter. In our scheme, depending on different application scenarios, an external TTP classifies nodes' behavior into three categories during the network initialization phase, namely good behavior, suspicious behavior, and malicious behavior. Each node keeps track of peers' behavior with a neighborhood watch scheme or by analyzing other nodes' reports, and then updates its own knowledge about peers' behavior with 3-dimensional Dirichlet distribution and makes the corresponding response. Furthermore, a deviation test is employed to filter potentially false statements from adversaries and Dempster-Shafer belief theory [155] is used to integrate other nodes' reports. While a node revokes the keys of nodes showing malicious behavior once enough evidence has been collected, it also shields itself from suspicious behavior of peers by ceasing the communication with them and continues gathering information for further decisions.

4.2 Mathematical Background

In this section, we present a brief introduction to IBC, bilinear pairing, and Dirichlet multinomial model, which form the basis of our design in this work. For a detailed treatment, the reader is referred to references mentioned below.

4.2.1 IBC and Bilinear Pairing

The concept of IBC is due to Shamir [157]. In an ID-based cryptosystem, a user's public key is an easily calculated function of his identity, while his private key can be computed by a TTP. Recently, IBC has been used to design efficient key management protocols for MANETs [85,179]. All these protocols use so-called bilinear pairings. Due to the important role of bilinear pairings in IBC, we give a brief introduction about the concept of bilinear pairings below.

Let r be a positive integer. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be additively-written abelian groups of order r with identity $\mathcal{O}$, and let $\mathbb{G}_T$ be a multiplicatively-written cyclic group of order r with identity 1. A *bilinear pairing* on $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ is a map

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$$

that satisfies the following additional properties:

1. **Bilinearity:** For $\forall P, P' \in \mathbb{G}_1$ and $\forall Q, Q' \in \mathbb{G}_2$ we have $e(P+P', Q) = e(P, Q)e(P', Q)$ and $e(P, Q + Q') = e(P, Q)e(P, Q')$.
2. **Non-degeneracy:** For $\forall P \in \mathbb{G}_1$ with $P \neq \mathcal{O}$, there is some $Q \in \mathbb{G}_2$ such that $e(P, Q) \neq 1$. Furthermore, for $\forall Q \in \mathbb{G}_2$ with $Q \neq \mathcal{O}$, there is some $P \in \mathbb{G}_1$ such that $e(P, Q) \neq 1$.
3. **Computability:** $e(P, Q)$ can be efficiently computed for all $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2$.

In practice, the abelian groups $\mathbb{G}_1$ and $\mathbb{G}_2$ are implemented using a divisor class group on certain (hyper-)elliptic curves and the cyclic group $\mathbb{G}_T$ is implemented using a multiplicative subgroup of a finite field. Most pairing applications rely on the hardness of the so-called *Bilinear Diffie-Hellman Problem* (BDHP)[1]. For more details, the reader is referred to [8].

4.2.2 Dirichlet Multinomial Model

The Dirichlet distribution, often denoted by $\mathsf{Dir}(\vec{\alpha})$, is a family of continuous multivariate probability distributions parameterized by the vector $\vec{\alpha}$ of positive reals which captures a sequence of observations of the possible outcomes in a state space. The Dirichlet distribution is defined as follows: Let $\Theta = \{\theta_1, \ldots, \theta_k\}$ be a state space consisting of k mutually disjoint events. Let $\vec{p} = (p(\theta_1), \ldots, p(\theta_k))$ be a continuous random vector taking values in the k-dimension simplex[2] with the joint probability density function

$$f(\vec{p} \mid \vec{\alpha}) = \frac{\Gamma\left(\sum_{i=1}^{k} \alpha(\theta_i)\right)}{\prod_{i=1}^{k} \Gamma\left(\alpha(\theta_i)\right)} \prod_{i=1}^{k} p(\theta_i)^{\alpha(\theta_i)-1},$$

where $\Gamma(x) = \int_0^\infty t^{x-1}e^{-t}dt$ is the Gamma function. Then $\vec{p}$ is said to have a k-dimension Dirichlet distribution with parameter vector $\vec{\alpha} = (\alpha(\theta_1), \ldots, \alpha(\theta_k))$ ($\alpha(\theta_i) > 0$ for $i = 1, \ldots, k$). The Dirichlet distribution is the multivariate generalization of the Beta distribution and the probability expectation value of any of the k random variables is defined as:

$$\mathbb{E}(p(\theta_i) \mid \vec{\alpha}) = \frac{\alpha(\theta_i)}{\sum_{i=1}^{k} \alpha(\theta_i)}.$$

[1]Let e be a bilinear pairing on $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$. The *Bilinear Diffie-Hellman Problem* (BDHP) is the following: given $P, P_1 = [a]P, P_2 = [b]P \in \mathbb{G}_1, Q \in \mathbb{G}_2$ such that $e(p, Q) \neq 1$, compute $e([ab]P, Q)$.

[2]The k-dimension simplex is such that if $\vec{p} = (p(\theta_1), \ldots, p(\theta_k))$ then $p(\theta_i) \geq 0$ and $\sum_{i=1}^{k} p(\theta_i) = 1$.

Since the Dirichlet distribution is a conjugate priori of the multinomial distribution, the posteriori distribution is also Dirichlet and can be calculated as follows [95]:

$$f(\vec{p} \mid \vec{r}, \vec{a}) = \frac{\Gamma\left(\sum_{i=1}^{k} r(\theta_i) + \mathcal{C}a(\theta_i)\right)}{\prod_{i=1}^{k} \Gamma\left(r(\theta_i) + \mathcal{C}a(\theta_i)\right)} \prod_{i=1}^{k} p(\theta_i)^{(r(\theta_i)+\mathcal{C}a(\theta_i)-1)}, \tag{4.1}$$

where $a(\theta_i)$ is a *base rate* vector over the state space Θ satisfying $a(\theta_i) \geq 0$ and $\sum_{i=1}^{k} a(\theta_i) = 1$, $\mathcal{C}$ is a *priori* constant which is equal to the cardinality of the state space over which a uniform distribution is assumed ($\mathcal{C}$ is usually set to 2), and the vector $r(\theta_i)$ is a *posteriori* evidence over the state space Θ. Given the Dirichlet distribution of Equation (4.1), the probability expectation of any of the k variables can now be written as:

$$\mathbb{E}(p(\theta_i) \mid \vec{r}, \vec{a}) = \frac{r(\theta_i) + Ca(\theta_i)}{C + \sum_{i=1}^{k} r(\theta_i)}.$$

Dirichlet multinomial model [71] provides a flexible mechanism for constructing reputation system for e-commerce applications. The basic idea behind a Dirichlet reputation system [95] is to compute reputation values by statically updating Dirichlet probability density function. Given the *a priori* reputation values, the *a posteriori* reputation value is calculated to increase the precision of a belief by combining the *a priori* knowledge and the new observations. For more details about the Dirichlet multinomial model and the Dirichlet reputation system, the reader is referred to [71, 95].

4.3 System Models and Design Goals

In this section, we formulate the network model and the security model as well as design assumptions and goals.

4.3.1 Network Model

We consider a general MANET consisting of an unconstrained number of networking nodes with a random mobility pattern, i.e., nodes moving independently within a given field or keeping stationary in a location for a period of time. In addition, the network topology also changes dynamically when a particular network event, such as node join, leave or failure, occurs. Each node has limited transmission and reception capabilities. Mobile nodes that are within each other's radio range communicate directly via bandwidth-constrained, error-prone insecure wireless links, while those that are far apart rely on other nodes to relay their messages in a multi-hop fashion. As requirements of many network tasks and protocols, each node must be unambiguously identified by a unique identity. It can be a MAC address or an IP address.

In order for nodes monitoring various behavior of their direct neighbors within the communication range, we assume that communication links are bidirectional in the network and nodes are in promiscuous mode. Both assumptions are common in many low-layer MANETs protocols such as DSR [93] and AODV [141] routing protocols. Furthermore, for disseminating accusation messages securely with IBC in our key revocation scheme, we assume that nodes know identities of their neighbors up to m-hop (m is a design parameter denoting the range of the accusation propagation). Identifiers of neighbor nodes can be obtained by running some neighborhood discovery protocols, which are part of many existing routing protocols and therefore can be reused. Moreover, we also assume that embedded processors of mobile nodes can perform public-key algorithms related to IBC. We would like to point out that all the above assumptions are quite common and reasonable for most application scenarios of MANETs. Hence, our design does not introduce additional burdens into the network.

4.3.2 Security Model

We term as an *adversary* or *attacker* any node whose behavior deviates from the legitimate MANET protocols. We assume that each node in MANET is installed an Intrusion Detection System [127] which can detect predefined misbehavior of nodes. The main purpose of a key revocation scheme is to revoke keys of malicious nodes and finally isolates them from the network. Most previous schemes [117, 151, 179] are vulnerable to potentially false statement attacks in which malicious nodes accuse other nodes in a MANET at their own will. Therefore, we need to evaluate the influence of false statement attacks mounted by malicious nodes on our key revocation scheme in details. The analysis of other types of attacks aimed at the different layers of MANETs, though important, is out of the scope of this work.

The false accusation attack can be independently or collaboratively initiated by some adversaries. We consider the following two attack scenarios in this chapter:

- **Attacks by independent adversaries**: in this attack scenario, each adversary independently chooses attack targets and propagates false accusations against victims through the network in order to accelerate keys of target nodes to be revoked by other nodes in the MANET. Note that in this case it is possible that an adversary also accuse other adversaries, except for accusing well-behaving nodes.
- **Attacks by collusive adversaries**: in this attack scenario, collusive adversaries know each other and they choose one or several well-behaving nodes as common

attack objects. These adversaries always report positive observations about their friends and negative ones about the chosen victims. In this way, the adversaries can not only prolong their lifetime in the MANET, but also speed up the procedure of revoking keys of the victims.

Furthermore, we also assume that adversaries always attempt to maximize their influence by propagating extremely positive or extremely negative observations to the network. Detailed simulations and analysis of our scheme against the above two types of attacks are presented in Section 4.5.

4.3.3 Design Goals

From our point of view, an ideal key revocation scheme for MANETs should have the following properties:

- It should be fully self-organized.
- It should be flexible enough to deal with the information that nodes collect through their own observations and interactions with peers, and to make corresponding responses based on results of the analysis.
- It should be able to efficiently revoke keys of malicious nodes when they show the behavior that the network cannot tolerate.
- It should be robust enough to thwart false statement attacks mounted by independent adversaries or a number of collusive adversaries.
- It should be efficient in terms of communication, computation and storage overhead.

4.4 Protocol Description

In this section, we describe our key revocation scheme in detail. We first provide an overview of our key revocation scheme in Section 4.4.1. And then we present the detailed procedure of our protocol in Sections 4.4.2 to 4.4.6.

4.4.1 Overview

Our fully self-organized key revocation scheme is within the framework of Bayesian data analysis. We employ Dirichlet multinomial model and explicitly use probability to quantify the uncertainty about nodes' behavior. Each node in a MANET gradually updates its knowledge about peers' behavior through interactions among them, and finally makes multilevel response based on the analysis of collected information. Furthermore, IBC is used to secure the information transmission during interactions of nodes. Our scheme consists of five parts: network initialization, neighborhood watch, authenticated information dissemination, filter of false statements, and multilevel response for malicious nodes.

In the network initialization, an external TTP first generates a set of secure system parameters for IBC. And then the TTP completes the registration of nodes by preloading each node with appropriate key materials according to the expire date and the version number of every key. Moreover, nodes' behavior is classified by the TTP into three categories: good behavior set, suspicious behavior set and malicious behavior set.

To protect the MANET from adversaries, each node overhears the wireless channel in the promiscuous mode, and monitors various behavior of its one-hop neighbors at all time with the neighborhood watch scheme. Each node records its observation and updates the knowledge about the behavior of all its one-hop neighbors. In addition, since nodes may change their behavior over time, a discount factor is introduced for the case that nodes can forget past observations gradually.

Each node not only uses direct observations to update its knowledge about one-hop neighbors' behavior, but also distributes these information to all its m-hop neighbors in some secure way. The data integrity and the authenticity of the message origin are implemented with a keyed-hash function where the key is derived from the bilinear pairing in a non-interactive fashion.

After one node receives an observation report from the other node, it first decides whether the sender can be trusted by checking the sender's key status. And then the receiver verifies the authenticity of the report with the pre-shared key between two nodes. Although merging other nodes' observations can accelerate the estimation about some subject's behavior, using all the receiving reports without hesitation will result in potentially false statement attacks from adversaries. Hence, we set two defence lines to thwart these attacks. Firstly, a deviation test based on the statistical pattern of reports is used to filter out false statements to some extent. Furthermore, if the sender's report passes the deviation test of the receiver, we will use Dempster-Shafer belief theory [155] to update the receiver's current knowledge about the behavior of the subject in question with this report.

In our key revocation model, each node considers that their peers show good behavior, suspicious behavior and malicious behavior with different probabilities. For approximating to these unknown parameters, a node uses 3-dimension Dirichlet distribution as the prior distribution of the unknown parameters, updates this distribution by either node's direct observations or its counterparts' reports, then estimates two parameters with posteriori expected probabilities and compares these values to predefined thresholds, and finally makes multilevel response based on results of comparisons. A high level description of our key revocation scheme is shown in the following Algorithm 7.

Algorithm 7 Self-Organized Key Revocation for MANETs

```
1: Network Initialization
     ▷ Generation of system parameters
     ▷ Registration of network nodes
     ▷ Classification of node behavior
2: Neighborhood Watch
     ▷ Monitor neighbors' behavior and generate observation matrix
     ▷ Update key status of nodes with direct observations
3: Authenticated Information Dissemination
     ▷ Disseminate nodes' direct observations to all $m$-hop neighbors in an authenticated
       way by using a keyed-hash function
4: Filter of False Statements
     ▷ Filter out potentially false statements statistically
     ▷ Update key status of nodes based on Dempster-Shafer theory
5: Multilevel Response for Malicious Nodes
     ▷ Revoke keys of nodes showing malicious behavior
     ▷ Cease communication with nodes showing suspicious behavior and keep observing
       their behavior for further decision
```

4.4.2 Step 1. Network Initialization

Our scheme assumes that an external TTP bootstraps the MANET with IBC and classifies the behavior of nodes. More specifically, the external TTP will complete the following tasks during network initialization:

Generation of system parameters

The TTP generates secure system parameters $\langle q, k, C/\mathbb{F}_q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e\rangle$ as described in Section 4.2.1. Note that we take $\mathbb{G}_1 = \mathbb{G}_2$ in this chapter. The TTP also generates a random master key $s \in \mathbb{Z}_n^*$ and a random generator $P \in \mathbb{G}_1$, and sets his public key $P_{pub} = sP \in \mathbb{G}_1$. Finally, the TTP chooses a cryptographic secure hash function: $H : \{0,1\}^* \to \mathbb{G}_1$. The TTP publishes all of these parameters except his master key.

Registration of network nodes

For the purpose of key revocation, we use the public key format $Q_i = H(ID_i \parallel \mathsf{date} \parallel \mathsf{version})$ for each node with identity ID_i as introduced in [86], where date is the expiry date of the key and version is its version number. After the user ID_i shows his credential and passes the authentication of the TTP, the TTP will derive his public key Q_i and generate the corresponding ID-based private key $d_i = sQ_i$.

Classification of node behavior

MANETs are complex and dynamic systems and the behavior of a node might change at any time for various reasons. In our model, the state space Θ includes three mutually disjoint events: good behavior θ_g, suspicious behavior θ_s and malicious behavior θ_m, namely $\Theta = \{\theta_g, \theta_s, \theta_m\}$. To keep track of various observable behavior in the lifetime of the MANET, the TTP classifies nodes' behavior into three categories, namely good behavior set $\mathbb{B}_g$, suspicious behavior set $\mathbb{B}_s$ and malicious behavior set $\mathbb{B}_m$. The set $\mathbb{B}_g$ includes behavior complying with descriptions of the MANET protocols such as finding a path for a packet correctly and relaying control and data packets for others at the best effort. The set $\mathbb{B}_s$ contains accidental misbehavior that temporarily and slightly deteriorate the performance of MANETs, for example node failures due to the network congestion or a lack of resources, whereas intentional misbehavior such as dropping data packets or broadcasting fake routing information, which seriously degrade the performance of MANETs, are comprised in the set $\mathbb{B}_m$. The practical classification of the sets $\mathbb{B}_g, \mathbb{B}_s$ and $\mathbb{B}_m$ depends on the network policy, the detection ability of nodes and the concrete application scenarios.

We note that all previous key revocation schemes for MANETs [3,86,117,129,151,179] only classify nodes' behavior as either good or malicious. The main motivation that we consider suspicious behavior is based on the observation that nodes show some misbehavior for a short time just by accident. For example, many reasons might cause a node not to forward packages for others such as network congestion or malicious attacks. Therefore, when a node observes that one of its neighbors cannot relay packages for some time, it is more reasonable for the node to cease the communication with that neighbor and keep observing its behavior instead of revoking its key and excluding it from the network immediately. By introducing suspicious behavior, we give nodes that misbehave by accident a chance to return to normal. As a result, our method provides more precise estimation about nodes' behavior than that with a simple binary behavior classification. Figure 4.1 demonstrates possible state transitions among different types of nodes in the lifetime of the MANET.

- A good node may behave suspiciously due to various reasons, such as network congestion and misconfiguration. Moreover, it is also possible that a good node is compromised by an adversary and becomes a malicious node.

- A suspicious node can become cooperative again when the network congestion is resolved or the node is reconfigured. Furthermore, a suspicious node might behave maliciously due to being compromised or energy depletion.

- A malicious node may reorder, delay and drop control and data packets, or disrupt legitimate path selections by broadcasting fake route replies.

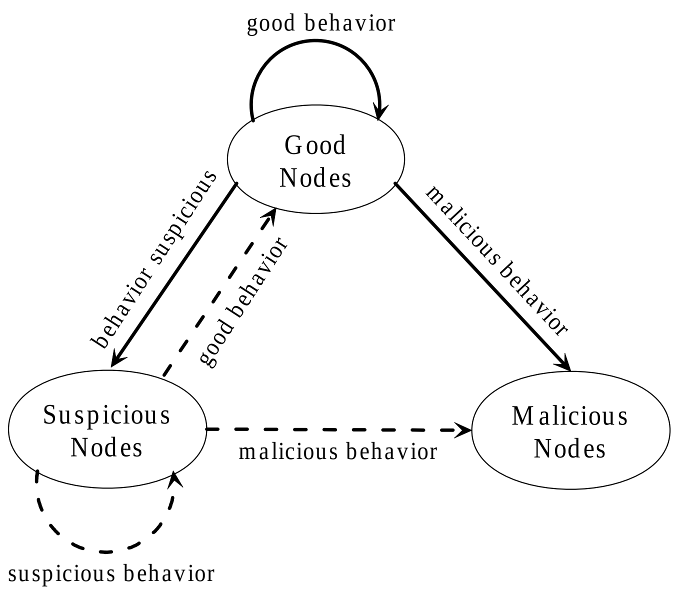


Figure 4.1: State Transition Diagram among Different Types of Nodes

Since nodes' behavior must fall into one of the above three categories, nodes can analyze and predict peers' behavior with 3-dimension Dirichlet distribution $\mathsf{Dir}(\alpha_g, \alpha_s, \alpha_m)$, where $(\alpha_g, \alpha_s, \alpha_m)$ is a parameter vector which keeps track of nodes' behavior appearing in sets $\mathbb{B}_g, \mathbb{B}_s$ and $\mathbb{B}_m$, respectively. Moreover, after the network initialization phase, each node ID_i is preloaded the following materials:

- **System Parameters**: $\langle q, k, C/\mathbb{F}_q, \mathbb{G}_1, \mathbb{G}_T, e, H, P, P_{pub} \rangle$.

- **Public / Private Key Pair**: $\langle Q_i, d_i \rangle$.

- **Behavior Classification:** $\mathbb{B}_g, \mathbb{B}_s$ and $\mathbb{B}_m$.

4.4.3 Step 2. Neighborhood Watch

A neighborhood watch mechanism is a localized monitoring scheme, the main aim of which is to observe behavior of nodes and decide whether they are conformed to descriptions of the MANET protocols. In the neighborhood watch scheme, each node ID_i monitors all its one-hop neighbors and records three categories of behavior each time they occur. We do not limit types of node behavior in this work and any new type of observable behavior can be added to the corresponding set $\mathbb{B}_g, \mathbb{B}_s$ or $\mathbb{B}_m$.

Without loss of generality, we use the notation $\mathcal{N}_i^{(1)}$ to denote the set of one-hop neighbors of node ID_i. Let $N_i^{(1)}$ be the cardinality of the set $\mathcal{N}_i^{(1)}$. Note that $\mathcal{N}_i^{(1)}$, and so $N_i^{(1)}$, will be dynamically changed with time due to the mobility of nodes in the MANET. We use the parameter vector $\left(\gamma_{j,g}^i, \gamma_{j,s}^i, \gamma_{j,m}^i\right)$ of 3-dimension Dirichlet distribution to record node ID_i's direct experience with the node ID_j. Initially, the parameter vector is set to $(Ca(\theta_g), Ca(\theta_s), Ca(\theta_m))$, where $(a(\theta_g), a(\theta_s), a(\theta_m))$ is the base rate vector and C is the prior constant (see Section 4.2.2). Node ID_i makes one individual observation for each node $ID_j \in \mathcal{N}_i^{(1)}$ periodically. We set binary variables $\beta_{j,g}^i, \beta_{j,s}^i$ and $\beta_{j,m}^i$ to be 1 if the node ID_i's observation about the node ID_j's behavior is classified into the sets $\mathbb{B}_g, \mathbb{B}_s$ or $\mathbb{B}_m$, respectively, and 0 otherwise. According to new observations about behavior of all its one-hop neighbors, node ID_i first updates its direct experience for each $ID_j \in \mathcal{N}_i^{(1)}$ with the following formulae:

$$\begin{aligned}
\gamma_{j,g}^i &:= \mu\gamma_{j,g}^i + \beta_{j,g}^i, \\
\gamma_{j,s}^i &:= \mu\gamma_{j,s}^i + \beta_{j,s}^i, \\
\gamma_{j,m}^i &:= \mu\gamma_{j,m}^i + \beta_{j,m}^i,
\end{aligned}$$

where the weight $\mu \in [0,1]$ is a discount factor for past observations (typically, μ is very close to 1). Node ID_i then updates its own *observation matrix* OM^i with new information. Assume that node ID_i has obtained direct experience with N_i nodes in the network up to the current time instance, node ID_i's observation matrix is as follows:

$$OM^i = \begin{bmatrix}
ID_1 & \gamma_{1,g}^i & \gamma_{1,s}^i & \gamma_{1,m}^i \\
\vdots & \vdots & \vdots & \vdots \\
ID_{N_i^{(1)}} & \gamma_{N_i^{(1)},g}^i & \gamma_{N_i^{(1)},s}^i & \gamma_{N_i^{(1)},m}^i \\
\vdots & \vdots & \vdots & \vdots \\
ID_{N_i} & \gamma_{N_i,g}^i & \gamma_{N_i,s}^i & \gamma_{N_i,m}^i
\end{bmatrix}.$$

We use the parameter vector $(\alpha_{j,g}^i, \alpha_{j,s}^i, \alpha_{j,m}^i)$ of 3-dimension Dirichlet distribution to keep track of node ID_i's global knowledge about node ID_j's behavior. Note that the vector $(\alpha^i_{j,g}, \alpha^i_{j,s}, \alpha^i_{j,m})$ will be updated by both node ID_i's direct experience and reports from other nodes. Initially, the parameter vector is also set to $(Ca(\theta_g), Ca(\theta_s), Ca(\theta_m))$. After node ID_i makes a direct observation about node ID_j's behavior, its global knowledge about node ID_j's behavior will be updated with the following formulae:

$$\begin{aligned}\alpha^i_{j,g} &:= \mu\alpha^i_{j,g} + \beta^i_{j,g},\\ \alpha^i_{j,s} &:= \mu\alpha^i_{j,s} + \beta^i_{j,s},\\ \alpha^i_{j,m} &:= \mu\alpha^i_{j,m} + \beta^i_{j,m}.\end{aligned}$$

Upon obtaining new information about all its one-hop neighbors, node ID_i also updates corresponding rows in its *node status matrix*, NSM^i, which indicates node ID_i's opinion about key status of other nodes. Let N be the total number of nodes in the MANET. Furthermore, we assume that node ID_i has obtained the knowledge of key status of M_i nodes until the current time instance by observing its one-hop neighbors and collecting information from others. Without loss the generality, we also assume that the first $N_i^{(1)}$ rows of NSM^i include information of node ID_i's one-hop neighbors at current time instance. Under the above assumptions, node ID_i's *node status matrix* NSM^i is as follows:

$$NSM^i = \begin{bmatrix} ID_1 & (t^i_1, v^i_1) & R^i_1 & \alpha^i_{1,g} & \alpha^i_{1,s} & \alpha^i_{1,m} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ ID_{N_i^{(1)}} & (t^i_{N_i^{(1)}}, v^i_{N_i^{(1)}}) & R^i_{N_i^{(1)}} & \alpha^i_{N_i^{(1)},g} & \alpha^i_{N_i^{(1)},s} & \alpha^i_{N_i^{(1)},m} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ ID_{M_i} & (t^i_{M_i}, v^i_{M_i}) & R^i_{M_i} & \alpha^i_{M_i,g} & \alpha^i_{M_i,s} & \alpha^i_{M_i,m} \\ ID_{M_i+1} & ? & ? & Ca(\theta_g) & Ca(\theta_s) & Ca(\theta_m) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ ID_N & ? & ? & Ca(\theta_g) & Ca(\theta_s) & Ca(\theta_m) \end{bmatrix},$$

where t^i_j and v^i_j represent the expiry date and the version number of the current public key Q_j of the node ID_j, respectively. $R^i_j \in \{-1, 0, 1\}$ denotes key status of node ID_j from the point of view of node ID_i, and "?" means node ID_i does not obtain any information about behavior of nodes $ID_k, k \in \{M_i + 1, \ldots, N\}$ until the current time instance. Note that R^i_j being -1, 0 or 1 indicates that the status of node ID_j's key is "Revoked", "Suspicious" or "Trustworthy", respectively. After each node ID_i updates the first $N_i^{(1)}$ rows of NSM^i with the neighborhood watch scheme, it will use the method described in Section 4.4.6 to decide whether key status of its one-hop neighbors need to be changed. For nodes whose key status have been marked as "Suspicious", node ID_i will cease the communication with those nodes. Furthermore, node ID_i also keeps observing behavior of suspicious nodes and receiving other nodes' reports to make further decisions.

4.4.4 Step 3. **Authenticated Information Dissemination**

Periodically, node ID_i securely disseminates its direct experience about other nodes' behavior to all its m-hop neighbors. Let $\mathcal{N}_i^{(m)}$ be the set of m-hop neighbors of node ID_i. Node ID_i then sends its observation matrix OM^i to each node $ID_j \in \mathcal{N}_i^{(m)}$ with the following format:

$$om_j^i = ((ID_i, ID_j, OM^i), h_{K_{i,j}}((ID_i, ID_j, OM^i))),$$

where $K_{i,j}$ is the pre-shared key between a pair of nodes ID_i and ID_j, and $h_{K_{i,j}}(\cdot)$ is a secure hash function taking $K_{i,j}$ as the input key. With the aid of the cryptographic pairing (see Section 4.2.1), the pre-shared key $K_{i,j}$ can be separately calculated by nodes ID_i and ID_j in a non-interactive fashion during the phase of a neighbor discovery as follows:

$$K_{i,j} = e(d_i, Q_j) = e(sQ_i, Q_j) = e(Q_i, sQ_j) = e(Q_i, d_j),$$

where $\langle Q_i, d_i \rangle$ and $\langle Q_j, d_j \rangle$ are the public/private key pair of nodes ID_i and ID_j, respectively. Furthermore, both data integrity and authenticity of messages are simultaneously guaranteed by the keyed-hash function $h_{K_{i,j}}(\cdot)$. Therefore, an attacker cannot change content of the observation matrix.

Note that we directly use the pairwise pre-shared secret key $K_{i,j}$ to secure communications among nodes in order to eliminate the communication overhead of establishing session keys. Although we can also use the lightweight ID-based key exchange protocol proposed in [85] to generate a different session key for each interaction, we need three-round communications between two nodes in this case.

4.4.5 Step 4. **Filter of False Statements**

Each time node ID_i receives an observation matrix om_i^j from node ID_j, node ID_i will perform the following information processing and integration algorithm shown in Figure 4.2.

In Step 4.1, node ID_i checks the status of node ID_j's key in the node status matrix NSM^i. If $R_j^i = 1$, then node ID_i considers node ID_j to be trustworthy and continues the next step; otherwise node ID_i will discard the observation matrix received from node ID_j and stop.

In Step 4.2, node ID_i verifies the authenticity of the message om_i^j using the pre-shared key $K_{i,j}$, as described in Section 4.4.4. If the message passes the authentication, node ID_i will further analyze reliability of node ID_j's observation in Step 4.3, otherwise node ID_i knows that the received message does not come from node ID_j, and therefore just discards it and stops.

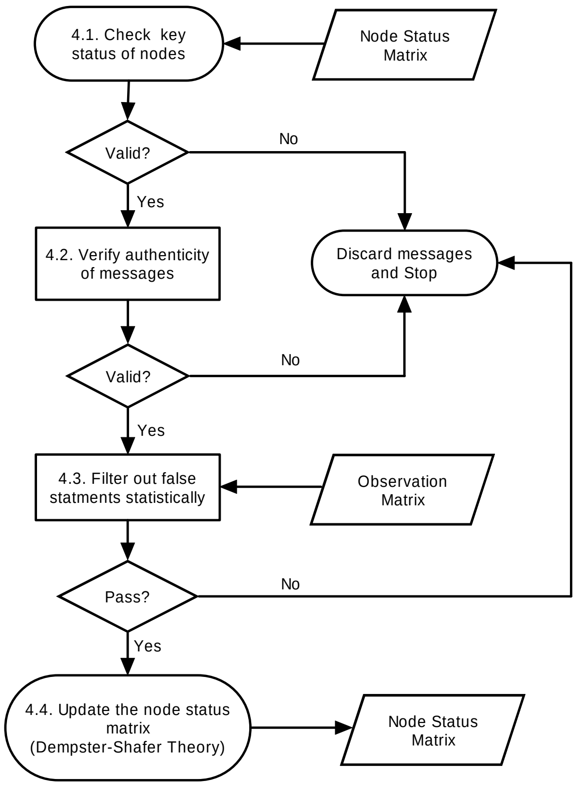

4.1. Check key status of nodes
Node Status Matrix
Valid?
No
Yes
4.2. Verify authenticity of messages
Discard messages and Stop
Valid?
No
Yes
4.3. Filter out false statments statistically
Observation Matrix
Pass?
No
Yes
4.4. Update the node status matrix (Dempster-Shafer Theory)
Node Status Matrix

Figure 4.2: Information Processing and Integration Algorithm

Due to the possibility that nodes are compromised and then arbitrarily report their observations under the control of attackers, messages that node ID_i receives from its counterparts might be spurious. Therefore, the main purpose of Step 4.3 is to avoid or mitigate the influence of false statements from malicious nodes to some degree. In the context of key revocation, attackers' goals are twofold by manipulating observations of compromised nodes. On the one hand, attackers can choose one or many good nodes and report unfairly negative observations about victims' behavior in order to revoke their keys. On the other hand, if attackers know each other and collude in MANETs, they will also propagate unfairly positive observations about their confederates' behavior for the purpose of keeping their keys valid and further damaging the operation of the network. Two efficient statistical filtering techniques based on Beta distribution have been proposed to protect Bayesian reputation systems from liars by Whitby *et al.* [172] and Buchegger *et al.* [22,23], respectively. Their methods are based on the assumption that the statistical pattern of dishonest reports is different from that of truthful ones. In addition, the difference between these two techniques is that Whitby *et al.*'s method uses quantiles of Beta distribution, whereas Buchegger *et al.*'s method employs a deviation test for the compatibility of received messages. Since our key revocation scheme is based on the Dirichlet distribution and it is difficult to define the quantile in the multivariate case, we only generalize the idea of the deviation test suggested by Buchegger *et al.* [22,23] to Dirichlet multinomial model here.

In Step 4.3, node ID_i extracts orderly each row from the node ID_j's observation matrix OM^j and performs a deviation test for the compatibility of node ID_j's observations. More specifically, when node ID_i extracts the k-th row from OM^j, it computes the following two posteriori expected probabilities with which node ID_k shows behavior in $\mathbb{B}_s$ and $\mathbb{B}_m$, respectively:

$$\begin{aligned}
\mathbb{E}\left(p(\theta_s) \mid \vec{\gamma}_k^j, \vec{a}\right) &= \frac{\gamma_{k,s}^j + Ca(\theta_s)}{C + \gamma_{k,g}^j + \gamma_{k,s}^j + \gamma_{k,m}^j}, \\
\mathbb{E}\left(p(\theta_m) \mid \vec{\gamma}_k^j, \vec{a}\right) &= \frac{\gamma_{k,m}^j + Ca(\theta_m)}{C + \gamma_{k,g}^j + \gamma_{k,s}^j + \gamma_{k,m}^j},
\end{aligned}$$

where $\vec{\gamma}_k^j = \left(\gamma_{k,g}^j, \gamma_{k,s}^j, \gamma_{k,m}^j\right)$ represents node ID_j's direct experience about node ID_k's behavior, and $\vec{a} = (a(\theta_g), a(\theta_s), a(\theta_m))$ is the default base rate vector. And then, node ID_i takes the row corresponding to node ID_k from its node status matrix NSM^i and separately calculates two expected probabilities based on its own knowledge about node ID_k's behavior as follows:

$$\begin{aligned}
\mathbb{E}\left(p(\theta_s) \mid \vec{\alpha}_k^i, \vec{a}\right) &= \frac{\alpha_{k,s}^i + Ca(\theta_s)}{C + \alpha_{k,g}^i + \alpha_{k,s}^i + \alpha_{k,m}^i}, \\
\mathbb{E}\left(p(\theta_m) \mid \vec{\alpha}_k^i, \vec{a}\right) &= \frac{\alpha_{k,m}^i + Ca(\theta_m)}{C + \alpha_{k,g}^i + \alpha_{k,s}^i + \alpha_{k,m}^i},
\end{aligned}$$

where $\vec{\alpha}_k^i = \left(\alpha_{k,g}^i, \alpha_{k,s}^i, \alpha_{k,m}^i\right)$ denotes node ID_i's global knowledge about node ID_k's behavior. After obtaining the above four expected probabilities, node ID_i executes the following deviation tests:

$$\begin{aligned}
\left|\mathbb{E}\left(p(\theta_s) \mid \vec{\alpha}_k^i, \vec{a}\right) - \mathbb{E}\left(p(\theta_s) \mid \vec{\gamma}_k^j, \vec{a}\right)\right| &\leq \varepsilon_1, \\
\left|\mathbb{E}\left(p(\theta_m) \mid \vec{\alpha}_k^i, \vec{a}\right) - \mathbb{E}\left(p(\theta_m) \mid \vec{\gamma}_k^j, \vec{a}\right)\right| &\leq \varepsilon_2,
\end{aligned}$$

where $\varepsilon_1, \varepsilon_2 \in (0,1)$ are two deviation thresholds determined by a system designer. If node ID_j's report about node ID_k's behavior cannot pass the above deviation tests, node ID_i considers that report as incompatible and just discards it. Otherwise, node ID_i uses node ID_j's report to update its knowledge about the behavior of the node ID_k in Step 4.4.

Note that the simplistic information integration method used in [23] is vulnerable to false statement attacks from an adversary, as analyzed theoretically in [132]. Therefore, we set up the second defense line to thwart false statement attacks by integrating other nodes' reports based on Dempster-Shafer belief theory [155]. In [94], Jøsang constructed a bijective mapping between Dirichlet distributions and Dempster-Shafer belief functions. Therefore, we first map node ID_i's global knowledge and node ID_j's report about node ID_k's behavior (two Dirichlet distributions) to two belief distribution functions, respectively. Then we use the technique of belief discounting [96] to update node ID_i's opinion about node ID_k's behavior as a result of node ID_j's report. Finally we map the resulting belief function to a Dirichlet distribution. In this way, the reports from different nodes are given different weight based on their respective reputation. Suppose that

$$\nu = \frac{C\alpha^i_{j,g}}{\left(C + \alpha^i_{j,s} + \alpha^i_{j,m}\right)\left(C + \gamma^j_{k,g} + \gamma^j_{k,s} + \gamma^j_{k,m}\right) + C\alpha^i_{j,g}}.$$

Then node ID_i uses node ID_j's report to update its global knowledge about node ID_k's behavior with the following equations:

$$\begin{aligned}\alpha^i_{k,g} &:= \mu\alpha^i_{k,g} + \nu\gamma^j_{k,g},\\ \alpha^i_{k,s} &:= \mu\alpha^i_{k,s} + \nu\gamma^j_{k,s},\\ \alpha^i_{k,m} &:= \mu\alpha^i_{k,m} + \nu\gamma^j_{k,m}.\end{aligned}$$

4.4.6 Step 5. Multilevel Response for Malicious Nodes

Each time node ID_i updates its knowledge about node ID_k's behavior in the MANET by either the neighborhood watch scheme or other nodes' reports, it checks whether ID_k's behavior are still within boundaries of its misbehavior tolerance and the status of node ID_k's key needs to be changed. Note that node ID_k's key status R^i_k in the node status matrix NSM^i directly determines how node ID_i treats node ID_k.

To minimize the squared-error loss for the deviation from the true probabilities $p(\theta_m)$ and $p(\theta_s)$ with which node ID_k shows respectively malicious and suspicious behavior, we choose posteriori expected probabilities $\mathbb{E}\left(p(\theta_m) \mid \vec{\alpha}^i_k, \vec{a}\right)$ and $\mathbb{E}\left(p(\theta_s) \mid \vec{\alpha}^i_k, \vec{a}\right)$ as estimators as usually done. As soon as node ID_i obtains the updated vector $\vec{\alpha}^i_k$ describing node ID_k's behavior, it will response as follows:

1. Node ID_i computes the posteriori expected probability $\mathbb{E}\left(p(\theta_m) \mid \vec{\alpha}^i_k, \vec{a}\right)$. If $\mathbb{E}(p(\theta_m) \mid \vec{\alpha}^i_k, \vec{a}) \geq t_{rev}$, i.e., it is equal to or larger than a predetermined revocation threshold t_{rev}, node ID_i sets $R^i_k = -1$ and stops. Otherwise it goes to the next step. Here, $R^i_k = -1$ denotes that node ID_i believes that node ID_k has been compromised and revokes its key. Once node ID_i revokes node ID_k's key, it will cease any communication with node ID_k until node ID_k receives a new key from the TTP.

2. Node ID_i calculates the posteriori expected probability $\mathbb{E}\left(p(\theta_s) \mid \vec{\alpha}_k^i, \vec{a}\right)$. If $\mathbb{E}(p(\theta_s) \mid \vec{\alpha}_k^i, \vec{a}) \geq t_{sus}^k$, i.e., it is equal to or larger than a predetermined suspicion threshold t_{sus}^k, node ID_i sets $R_k^i = 0$. Note that $R_k^i = 0$ means that node ID_i suspects that node ID_k has been compromised, and so node ID_i will shield itself against suspicious behavior of node ID_k by terminating the communication with it. Furthermore, to make further decision, node ID_i continues collecting information to update its knowledge about node ID_k's behavior. Possible state transitions of the suspicious node ID_k are described with dash lines in Figure 4.1. Note that three cases might happen for node ID_k: a) Node ID_k just shows suspicious behavior by accident, and therefore behaves normally after a short time. In this case, it will become a good node and be trusted by node ID_i again. b) Node ID_k continues behaving suspiciously. In this case, all nodes will finally mark node ID_k to be suspicious and terminate to communicate with it. Hence, node ID_k will be evicted from the network. c) Node ID_k shows malicious behavior. In this case, the key of node ID_k will be revoked once the posterior expected probability $\mathbb{E}\left(p(\theta_s) \mid \vec{\alpha}_k^i, \vec{a}\right)$ reaches the revocation threshold. In addition, to react faster than before when node ID_k behaves suspiciously again in the above case a), node ID_i also decreases the suspicion threshold of node ID_k as follows:
$$t_{sus}^k := \xi t_{sus}^k,$$
where $\xi \in (0, 1)$ is a fading factor of the suspicion threshold of a node. Furthermore, we also introduce a parameter t_{max} which denotes the maximum number of state transitions between good nodes and suspicious nodes (see Figure 4.1). Once the state transition has appeared t_{max} times for node ID_k, node ID_i will revoke its key immediately by setting $R_k^i = -1$ and terminate any further communication with node ID_k until node ID_k receives a new key from the TTP.

4.5 Performance Evaluation

In this section, we evaluate the performance of our key revocation scheme through extensive simulations, the goal of which is to demonstrate attack-resistant properties of our scheme under the existence of independent adversaries and collusive adversaries, respectively. Furthermore, we also show the advantages of classifying nodes' behavior into three categories over the simple binary differentiation.

4.5.1 Simulation Setup

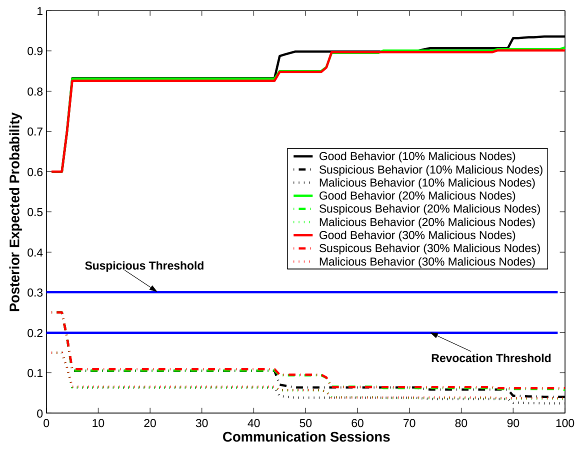
we have implemented our key revocation scheme with the C programming language on Microsoft Visual Studio platform. The performance evaluations are based on the simulations of 100 wireless nodes that form a MANET over a square (600m $\times$ 600m) space and interact 100 times. We use the "random waypoint" model [21] to simulate the mobility of nodes in the MANET. For each node, We set the maximum speed as 10m/s and maximum travel time as 20s. The communication range of each node is set to be 100 m. Furthermore, we assume that the base rate vector $\vec{a} = (a(\theta_g), a(\theta_s), a(\theta_m))$ is $(0.6, 0.25, 0.15)$, which denotes the prior uncertainty that honest nodes show good behavior, suspicious behavior and malicious behavior, respectively. We also assume that the discount factor μ is 0.999, both deviation thresholds ε_1 and ε_2 are 0.1, the revocation threshold t_{rev} is 0.2, and the suspicious threshold t^k_{sus} is set to be 0.3. The simulation is repeated for a number of communication sessions. In each session, each node moves to a new position and observe the behavior of its neighbors. Moreover, in some sessions nodes also flood their observations to all m-hop neighbors.

To simulate false statement attacks from adversaries, before running the simulation, we randomly select a certain fraction of the network population as suspicious nodes and malicious nodes, respectively. More specifically, we assume that 20% of all network nodes will show suspicious behavior for different reasons. Among those suspicious nodes, we further assume that half of them, named *type-I suspicious nodes*, show suspicious behavior just by accident (for example, a node drops packages due to the network congestion.) and behave normally after some time (due to the improvements of the network environment), whereas the other half of suspicious nodes, called *type-II suspicious nodes*, show suspicious behavior followed by malicious behavior. Note that type-I suspicious nodes are basically good and therefore record their observations honestly, whereas type-II suspicious nodes are basically malicious and so we assume that they record a suspicious behavior or a malicious behavior with probability $\frac{1}{2}$, respectively, for selected attack objects in each communication session. Considering two types of suspicious nodes in the simulations enables us to demonstrate the following two cases (also see Figure 4.1): a) Type-I suspicious nodes can get trustworthy again by good nodes after they are marked as suspicious; b) Keys of type-II suspicious nodes will be finally revoked. Furthermore, we also change the fraction of malicious nodes, ranging from 10% to 30%. Based on the above parameters and assumptions, we simulate two attack scenarios described in Section 4.3.2.

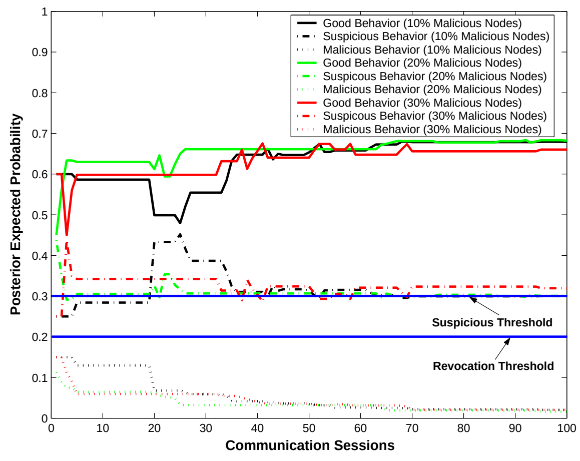
4.5.2 False Statement Attacks by Independent Adversaries

In this section, we evaluate the impact of false statement attacks launched by independent adversaries on our key revocation scheme. In this attack scenario, we further assume that each adversary selects 10% of all network nodes as attack objects, randomly and independently. These adversaries record a malicious behavior for the selected attack objects in each communication session and flood their accusations to all one-hop neighbors each 5 communication sessions.

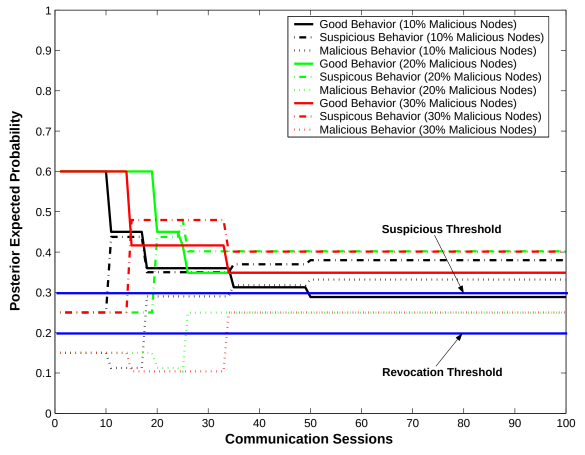
Note that we are concerned with the influence of false accusation attacks on good nodes' opinion about the key status of other nodes. Therefore, we randomly sample two good nodes, a type-I suspicious node, a type-II suspicious node, and a malicious node. We then keep track of the opinion of one good node about the key status of other four nodes. Figure 4.3 shows the attack-resistance properties of our key revocation scheme against independent adversaries when their population increases from 10% to 30%. Although we randomly sample several nodes, we would like to point out that the opinion of other good nodes follows the similar curves as Figure 4.3.



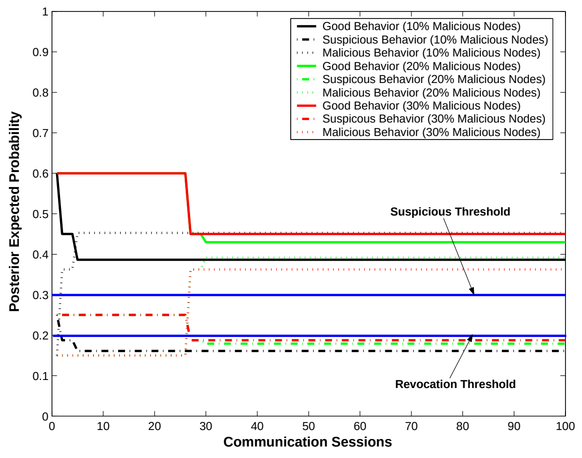
(a) A good node's opinion about the key status of the other good node



(b) A good node's opinion about the key status of a type-I suspicious node



(c) A good node's opinion about the key status of a type-II suspicious node



(d) A good node's opinion about the key status of a malicious node

Figure 4.3: Simulation Results for False Statement Attacks by Independent Adversaries

Figure 4.3(a) describes a good node's opinion about the key status of the other good node. We note that from the point of view of a good node the posterior expected probabilities that the other good node shows suspicious behavior and malicious behavior never exceed the corresponding suspicious threshold and revocation threshold. Therefore, the keys of good nodes never get wrongly revoked by other good nodes under the false statement attacks by independent adversaries.

For a type-I suspicious node, Figure 4.3(b) shows that from the point of view of a good node the posterior expected probability that a type-I suspicious node shows malicious behavior is always less than the revocation threshold. Hence, the key of the type-I suspicious node will not be revoked unless it have altered their states between good and suspicious for t_{max} times (also see Figure 4.1). In particular, when that node show suspicious behavior followed by good behavior, the key of that node will be first marked as suspicious once the posterior expected probability $\mathbb{E}\left(p(\theta_s) \mid \vec{\alpha}_k^{i,new}, \vec{a}\right)$ exceeds the suspicious threshold. Then that node becomes trustworthy by the good node again after it behaves normally for some time. Note that if one uses the simple binary differentiation for nodes' behavior, the keys of type-I suspicious nodes will be revoked immediately. However, the type-I suspicious nodes only misbehave temporarily and are basically good in our simulations. Therefore, our scheme provides more accurate estimation about nodes' behavior than that in the binary case.

For a type-II suspicious node who show suspicious behavior followed by malicious behavior, Figure 4.3(c) indicates that a good node will first mark its key as suspicious when the posterior expected probability $\mathbb{E}\left(p(\theta_s) \mid \vec{\alpha}_k^{i,new}, \vec{a}\right)$ exceeds the suspicious threshold. After gathering enough evidence about malicious behavior of the type-II suspicious node, the good node will finally revoke its key. In addition, Figure 4.3(d) shows that a good node can correctly revoke the key of a malicious node in the presence of independent adversaries.

From the simulation results in Figure 4.3, we note that our key revocation scheme can efficiently isolate malicious nodes from the network and also demonstrates strong robustness against the false statement attacks from independent adversaries even in a highly hostile environment (10% type-II suspicious nodes and 30% malicious nodes).

4.5.3 False Statement Attacks by Collusive Adversaries

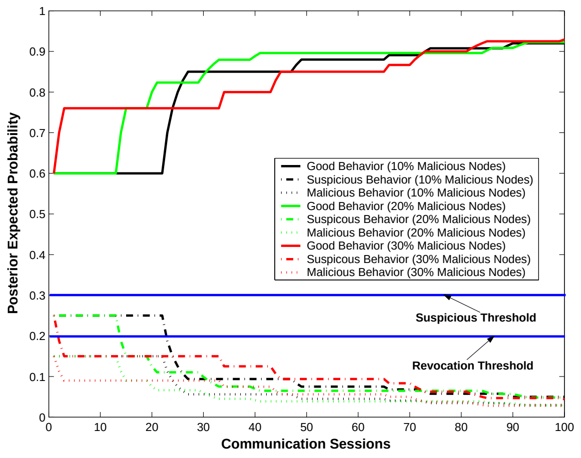
In this section, we study whether false statement attacks from collusive adversaries will affect our key revocation scheme. To this end, we assume that all malicious nodes choose 10% good nodes as common targets instead of randomly and independently selecting attack objects. In this attack scenario, all malicious nodes not only record malicious behavior for the selected 10% good nodes but also record good behavior for other malicious nodes in each communication session. Furthermore, they also propagate their false statements to all one-hop neighbors each 5 communication sessions.

Here, we check the opinion of a good node about the key status of other nodes under the collusive false statement attacks. Similar to the case of independent adversaries, we randomly select two good nodes (one of them is the attack object of the collusive adversaries), a type-I suspicious node, a type-II suspicious node, and a malicious node again, and keep track of the opinion of a good node. Figure 4.4 shows the attack-resistance properties of our key revocation scheme against collusive adversaries when the number of malicious nodes increases from 10% to 30%. We want to emphasize again that in our key revocation scheme each node has its own view about the key status of other nodes. Although we observe that all good nodes have similar opinion about other nodes' key status in our simulations, it is impossible for us to show all good nodes' opinion due to space limitations. Therefore, we randomly sample several nodes from different categories.

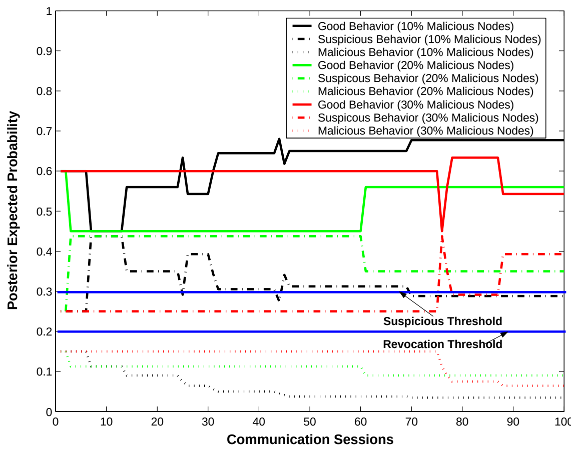
In Figure 4.4(a), we note that false accusations from collusive adversaries cannot affect the good node's opinion about the key status of the victim they select. The posterior expected probability that the victim shows malicious behavior is always less than the revocation threshold. The reason is that good nodes have accumulated good reputation in the early communication sessions and the false accusations from adversaries cannot pass the deviation test set by good nodes. Therefore, the false accusations will be filtered by good nodes and the keys of good nodes will not be wrongly revoked even in the presence of collusive adversaries.

Similar to the case of independent adversaries, Figure 4.4(b) shows that the key of a type-I suspicious node will not be revoked by the good node unless the number of times that it changes its states between good and suspicious amount to t_{max} (see Figure 4.1). Furthermore, if the key of the type-I suspicious node is marked as suspicious due to temporary suspicious behavior, it can be trusted again by a good node after the posterior expected probability $\mathbb{E}\left(p(\theta_s) \mid \vec{\alpha}_k^{i,new}, \vec{a}\right)$ is less than the suspicious threshold. Different from type-I suspicious nodes, malicious behavior of a type-II suspicious node are finally identified by the good node and therefore it will revoke the key of the type-II suspicious node as shown in Figure 4.4(c). Figures 4.4(b) and 4.4(c) demonstrate how a good node responses suspicious behavior in our scheme. While a good node showing suspicious behavior temporarily can get trustworthy again by other good nodes, the real malicious nodes will be evicted from the network. Moreover, false statement attacks from collusive adversaries have no influence on type-I and type-II suspicious nodes since the attack objects of adversaries are good nodes in our simulations.

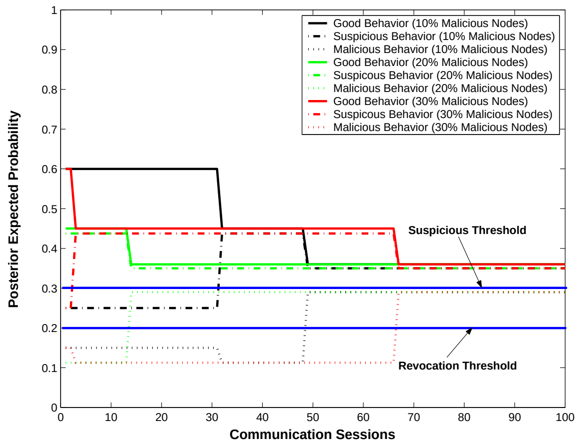
Figure 4.4(d) shows that the key of the malicious node can still be revoked by the good node even if malicious nodes praise each other. The reason is that after the good node have established the bad reputation for the malicious node in the early communication sessions the false praise from the friends of the malicious node is very difficult to pass the deviation test of good nodes. Moreover, even if the false statement can pass the deviation test, this information only has slight influence on the opinion of the good node because of the use



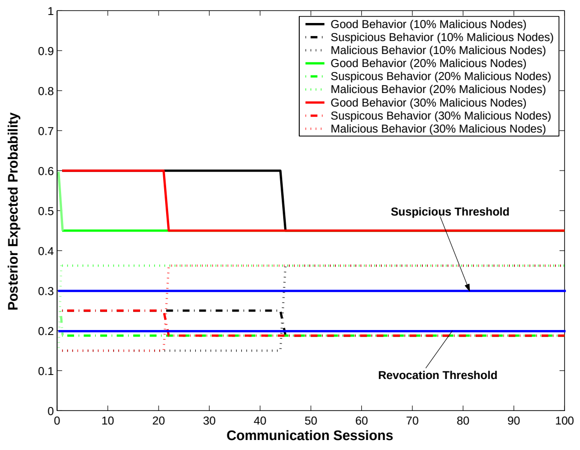
(a) A good node's opinion about the key status of the other good node selected by collusive adversaries



(b) A good node's opinion about the key status of a type-I suspicious node



(c) A good node's opinion about the key status of a type-II suspicious node



(d) A good node's opinion about the key status of a malicious node

Figure 4.4: Simulation Results for False Statement Attacks by Collusive Adversaries

of Dempster-Shafer theory (see Section 4.4.5), which gives less weight to the reports from malicious nodes than those from good nodes.

The simulation results in Figure 4.4 demonstrate that false statements from collusive malicious nodes cannot affect good nodes' opinion about the key status of other nodes. Most false statements are filtered by the deviation tests of good nodes. For those false statements which pass the deviation tests, the information integration technique based on Dempster-Shafer theory guarantees that the false statements only have slight influence on good nodes' opinion. Therefore, our key revocation can still perform well even under the false statement attacks from collusive adversaries.

4.6 Conclusion

MANETs pose formidable challenges on the issue of key revocation due to lack of infrastructure and centralized servers. This work explores a novel self-organized approach to solve the key revocation problem in MANETs. Firmly rooted in statistics, our key revocation scheme provides a theoretically sound basis for nodes analyzing and predicting peers' behavior based on their own observations and other nodes' reports. Furthermore, classifying nodes' behavior into three categories not only provides network designers more flexibility for various application scenarios, but also enables nodes to make multilevel response according to the severity of malicious behavior. In addition, our key revocation scheme is designed to provide strong defense against false statement attacks from independent and collusive adversaries. The effectiveness and attack-resistance properties of our scheme are confirmed by extensive simulation results.

Chapter 5

Accelerating Signature-Based Broadcast Authentication for WSNs

Broadcast authentication is a crucial security mechanism in WSNs, as it allows a multitude of legitimate users to join in and disseminate messages into the networks in a dynamic and authenticated way. During the past few years, several public-key based multi-user broadcast authentication schemes have been proposed in the literature to achieve immediate authentication and address the security vulnerability intrinsic to μTESLA-like schemes. Unfortunately, the relatively slow signature verification in signature-based broadcast authentication has also incurred a series of problems such as high energy consumption and long verification delay. In this chapter we propose an efficient technique to accelerate public-key signature verification in WSNs by exploiting the cooperation among sensor nodes. We first review related work and motivation for using public-key based broadcast authentication schemes for WSNs in Section 5.1, followed by a brief introduction about elliptic curve cryptography and the corresponding digital signature algorithm in Section 5.2. Section 5.3 presents the system model, adversary model and design goal. In Section 5.4, we describe the acceleration technique for signature verification in WSNs and discuss the selection of system parameters. Section 5.5 analyzes the performance of the proposed scheme by a case study. Finally, Section 5.6 concludes this chapter.

5.1 Related Work and Motivation

Considering the security and scalability of symmetric-key based broadcast authentication schemes [134, 145, 146], a couple of public-key based solutions have been proposed during the past few years.

In [145], Ren *et al.* pointed out that μTESLA-like broadcast authentication schemes all suffer from serious denial-of-service (DoS) attacks due to the delayed message authentication. To overcome the security vulnerability inherent to the existing μTESLA-like schemes, the authors proposed several public-key based solutions, including a *certificate-based* authentication scheme, a basic (enhanced) *Merkle hash tree* based authentication scheme, and an *ID-based* authentication scheme (IDS). While the certificated-based scheme provides a straightforward solution, it is inefficient to support user revocation in WSNs. The basic and enhanced Merkle hash tree based schemes improve public key management problem and enable users to make the trade-off between the storage and communication overhead. However, these schemes are communication inefficient when the number of network users become large. To further decrease the communication overhead and provide sound scalability, the authors proposed an ID-based scheme using cryptographic pairings. Unfortunately, this scheme has a very high computation and energy cost due to expensive bilinear pairing computations involved. Therefore, the scheme is only feasible to be implemented on high-end sensor nodes like Imote2 motes from Crossbow Technology [43].

Later on, Ren *et al.* [146] proposed two more advanced public-key based broadcast authentication schemes in order to achieve both storage efficiency and communication efficiency simultaneously. Their schemes are based on the efficient integration of several cryptographic techniques, including the Bloom filter, the partial message recovery signature scheme and the Merkle hash tree. In the *Bloom filter* based authentication scheme (BAS), the authors employed the Bloom filter to address the public key management, and a variant of Elliptic Curve Digital Signature Algorithm (ECDSA) with the partial message recovery property for message signing and authentication. Since the Bloom filter only provides probabilistic membership verification, the authors also discussed the problem of selecting system parameters for minimizing the probability of a false positive. Furthermore, the authors noted that the maximum supported number of users by BAS is usually limited given the storage limit and the probability of a false positive. In order to support more network users, the authors proposed the *hybrid* authentication scheme (HAS) that combines the Bloom filter and Merkle hash tree for public key management. However, using Merkle hash tree only can achieve partial user scalability (i.e., the total number of network users is fixed) and therefore a new network user can be allowed to join the network only when an old user is revoked.

In [29], Cao *et al.* proposed the IMBAS, an identity-based multi-user broadcast authentication scheme with strong security, sound scalability and efficiency for WSNs. Their authentication scheme is based on a variant of BNN-IBS [12], which is a *pairing-free identity-based signature* scheme with reduced signature size. Using the variant of BNN-IBS eliminates the requirement of transmitting public key certificate and provides the source and message authentication simultaneously. By the detailed quantitative performance analysis, the authors show that IMBAS achieves much better scalability as well as lower energy consumption than those of IDS [145] and HAS [146].

Very recently Yamakawa *et al.* [176] presented a lightweight broadcast authentication protocol called McSBA by employing a variant of McEliece signature scheme [39]. Their estimation results show that McSBA protocol can verify much faster than μTESLA and RSA at the same security level. However, McSBA authentication scheme has the following obvious disadvantages when used in WSNs: 1) the size of public key is very large, which incurs a much higher cost for storing and transmitting public keys as compared to other public key cryptosystems; 2) the signature generation of the McSBA protocol is very slow, especially if implemented on various embedded processors. Therefore, the McSBA protocol is more suitable to be used in the case that all sensor nodes store the base station's public key (i.e., the public key does not need to be transmitted) and authenticate messages from the powerful base station (i.e., signatures can be generated within a reasonable time period) rather than a multi-user setting in WSNs.

The main impetus for using public key cryptosystems in WSNs comes from the advances in the manufacturing technology of wireless sensor nodes as well as the efficient implementation of public key cryptographic algorithms on sensor platforms. On the one hand, many high performance and low-power microcontrollers such as 8-bit ATmega128L from Atmel [4], 16-bit MSP430 from Texas Instruments [166], and 32-bit XScale PXA271 from Intel [91], have been widely used in the design of wireless sensor nodes [41–43]. On the other hand, recent studies have showed that even software implementations only of public-key cryptosystems such as elliptic curve cryptosystems (ECC) [76, 111, 121, 154, 170] and pairing-based cryptosystems (PBC) [137, 138, 158, 159, 175] are very viable and efficient on resource-constrained sensor nodes. For example, according to the state-of-the-art software implementation results on an 8-bit microcontroller ATmega128L, the generation and verification of a digital signature on a Koblitz elliptic curve defined over $\mathbb{F}_{2^{163}}$ take $0.36s$ and $0.63s$ [114], respectively, whereas the timing of computing an η_T pairing over $\mathbb{F}_{2^{239}}$ achieves about $1.93s$ [158]. Employing public-key cryptography for implementing broadcast authentication in WSNs provides simple solutions, strong security resilience, good scalability and immediate message authentication, when compared to symmetric-key based solutions. However, public-key based broadcast authentication schemes have a common shortcoming: the signature verification is much slower than the message authentication code verification used in symmetric-key based solutions, which might lead to the case that a large number of packages will wait in a message queue of a senor node for signature verifications when many users broadcast messages. As a result, the message queue will become full quickly and the subsequent messages have to be dropped. Therefore, in order to improve the service quality of broadcast authentication in WSNs, accelerating the signature verifications in public-key based solutions become paramount.

In this chapter we address the issue of speeding up the signature verification for public-key based multi-user broadcast authentication schemes in WSNs by exploiting the *cooper-*

ation among sensor nodes. The basic idea is that some sensor nodes will randomly *release their intermediate computation results* to their neighbors during the signature verification. Then many sensor nodes can use the received intermediate computation results to accelerate their signature verifications. To demonstrate the performance of the proposed technique, we conduct a case study for the broadcast authentication problem in a 4×4 grid-based WSN. The detailed quantitative analysis shows our scheme is greatly superior to the traditional signature verification method for WSNs in terms of energy consumption of the whole network.

5.2 Preliminaries

In this section, we first give a brief introduction about elliptic curve cryptography, followed by the description of ECDSA, which will serve as an example to demonstrate the validity of our proposed acceleration technique for digital signature verification in WSNs.

5.2.1 Elliptic Curve Cryptography

Let $\mathbb{F}_q$ be a finite field with $q = p^m$ elements, where $p > 3$ is a prime and m is a positive integer. An *elliptic curve* $E(\mathbb{F}_q)$ is the set of solutions (x, y) over $\mathbb{F}_q$ satisfying an equation of the form $E : y^2 = x^3 + ax + b$, where $a, b \in \mathbb{F}_q$ and $4a^3 + 27b^2 \in \mathbb{F}_q^*$, together with an additional *point at infinity*, denoted by $\mathcal{O}$. The points on an elliptic curve form an (additive) Abelian group, where $\mathcal{O}$ is the identity element and the group operation is given by the well known chord-and-tangent rule [79]. The order of $E(\mathbb{F}_q)$ is denoted by $\#E(\mathbb{F}_q)$ and the order of a point $P \in E$ is defined as the smallest non-negative integer n such that $nP = \mathcal{O}$, where $n \mid \#E(\mathbb{F}_q)$ and $nP = \underbrace{P + \cdots + P}_{n \text{ times}}$. Let $\mathbb{G}$ be a cyclic subgroup of E generated by the point P, such that the *elliptic curve discrete logarithm problem* (ECDLP) is intractable. For more details about elliptic curve cryptography, the interested reader is referred to [79].

5.2.2 Elliptic Curve Digital Signature Algorithm

The ECDSA [79] is a widely standardized variant of the ElGamal signature scheme, which is described as follows:

1. *System-wide parameters.* Let $\mathbb{G}$ be a cyclic subgroup of $E(\mathbb{F}_q)$ generated by the point P with prime order n and identity element $\mathcal{O}$. Let $H : \{0, 1\}^* \rightarrow \mathbb{Z}_n^*$ be a collision-resistant hash function.

2. *Initial set-up.* Singer A randomly selects an integer $d \in [1, n-1]$ and publishes its public key $Q = dP$. The parameter d is kept secret to A.

3. *Signature generation.* Signer A uses his/her private key d to generate a signature (r, s) for a message $M \in \{0,1\}^*$.

   (a) Select a random integer $k \in [1, n-1]$, compute $R = kP$ and set r to be the x-coordinate of R.

   (b) Compute $s = k^{-1}(e + dr) \bmod n$, where $e = H(M)$.

   (c) If $r, s \in [1, n-1]$, return (r, s); otherwise, go to Step (3-a).

4. *Signature verification.* Upon receiving the message $M \in \{0,1\}^*$ and the signature (r, s) from A, a verifier B verifies the signature using A's public key Q.

   (a) Check that $r, s \in [1, n-1]$. If any verification fails, return 'reject signature'.

   (b) Compute $R' = s^{-1}(eP + rQ)$, where $e = H(M)$.

   (c) Check that the x-coordinate of R' is equal to r. If verification succeeds, return 'accept signature'; otherwise, return 'reject signature'.

Note that like most ElGamal signature schemes, the signature verification of ECDSA is about twice as slow as signature generation, which is an undesirable property when using ECDSA for multi-user broadcast authentication in WSNs.

5.3 System and Adversary Models

In this section, we present system and adversary modes as well as our design goals.

5.3.1 System Model

We consider a large-scale WSN that consists of a base station and a large number of sensor nodes. While the base station is powerful enough to execute various complicated operations, the sensor nodes usually have constrained resources in terms of computational capabilities, memory, bandwidth, and power supply. Typical applications of such large-scale WSNs include habitat and environmental monitoring, in which sensor nodes collect localized measurements and detailed information that is hard, if not impossible, to obtain through traditional instrumentation. By analyzing valuable data collected from WSNs, people can precisely estimate changes of ecological environment and take corresponding measures. In such application scenario, network users usually obtain the latest environmental information through broadcasting queries and commands into WSNs. We assume that users will join and leave the network dynamically, and be evicted from the network in case of misbehaving. Moreover, users need to register with the WSN and obtain necessary credentials for using the broadcast service. We also assume that the base station is always trustworthy and the sensor nodes can be completely captured and manipulated by adversaries. Furthermore, both the base station and users may send a broadcast message into the network. We further assume that one of public-key based multi-user broadcast authentication schemes such as those in [29,145,146] is employed in the WSN and the sensor nodes are capable of executing the corresponding digital signature verification algorithms. In addition, we assume that a single-chip 2.4GHz IEEE 802.15.4 compliant RF transceiver [167] is used as the wireless transmission module in sensor nodes, which supports up to 102 bytes payload and thus provides enough space to contain both the broadcast message and its digital signature in a package. Finally, we assume that the loose clock synchronization is available in the WSN.

5.3.2 Adversary Model

We assume that an adversary can launch a wide range of attacks against the signature-based broadcast authentication schemes. For example, he can simply mount the DoS attacks by injecting bogus messages into the network, aiming at exhausting the limited storage and energy of the sensor nodes. We assume that some efficient pre-authentication techniques like those in [48,134] have been employed in the network to mitigate DoS attacks. Moreover, it is also possible that an adversary attempts to impersonate other legitimate sensor nodes and obtain valuable information through eavesdropping, modifying, deleting, replaying, forging or blocking any network traffic. We also assume that a small fraction of user devices and sensor nodes can be compromised by an adversary and therefore the attacker can manipulate compromised devices to disseminate messages into the network. Some efficient private-key protection mechanism and user revocation scheme [29, 146] can be utilized to thwart the potential node compromise attack in WSNs.

5.3.3 Design Goal

The main design goal in this work is to *accelerate the signature verification for public-key based broadcast authentication schemes in WSNs.* To achieve this goal we will fully exploit the *cooperation* among sensor nodes in WSNs, which will be detailed in the next section.

5.4 Faster Signature Verification in Wireless Sensor Networks

In this section, we first describe the broadcast authentication problem in WSNs and then we use the ECDSA as an example to show how to accelerate the signature verification through the *cooperation* among sensor nodes.

5.4.1 Problem Statement

When WSNs are deployed in hostile environments, broadcast authentication (i.e., verifiability of the authenticity of broadcast packages) is a crucial security mechanism to ensure the trustworthiness of network applications. After registering with the WSN a user first contacts with several sensor nodes in the vicinity and sends a request for the broadcast service. Then the user and the sensor nodes conduct a mutual authentication procedure[1], which grants the access to the WSN only to a legitimate user and, at the same time, guarantees the user of the trustworthiness of the sensor nodes.

As illustrated in the following Figure 5.1, once the user and the sensor nodes establish an authenticated channel, the user will sign a query or command and forward it to the sensor nodes (e.g., nodes A, B and C). Nodes A, B and C then verify the signature of the user[2], respectively. If the verification succeeds, they will locally broadcast the user's query/command (within their communication range). When some node, say D, receives the broadcast package for the first time, it will execute the same signature verification and determine whether the received package should be forwarded to other nodes (e.g., node E). This broadcast and authentication procedure continues until all reachable nodes receive the user's broadcast package. If any verification fails during the broadcast, sensor nodes will drop the package and report to the base station.

5.4.2 A Faster Signature Verification Scheme

In the broadcast authentication procedure as shown in Figure 5.1, all sensor nodes execute the same signature verification after receiving a broadcast package. Assuming that the ECDSA is employed in WSNs, we show how to speed up the ECDSA signature verification below. Although the ECDSA is used as an example, we would like to point out that the

[1]The discussion of user authentication in WSNs is out of the scope of this chapter and the interested reader is referred to the references [13, 44, 45].

[2]For simplicity of exposition, we also assume that the keying materials (i.e., the user's public key) needed for signature verification have been distributed to all nodes in WSNs.

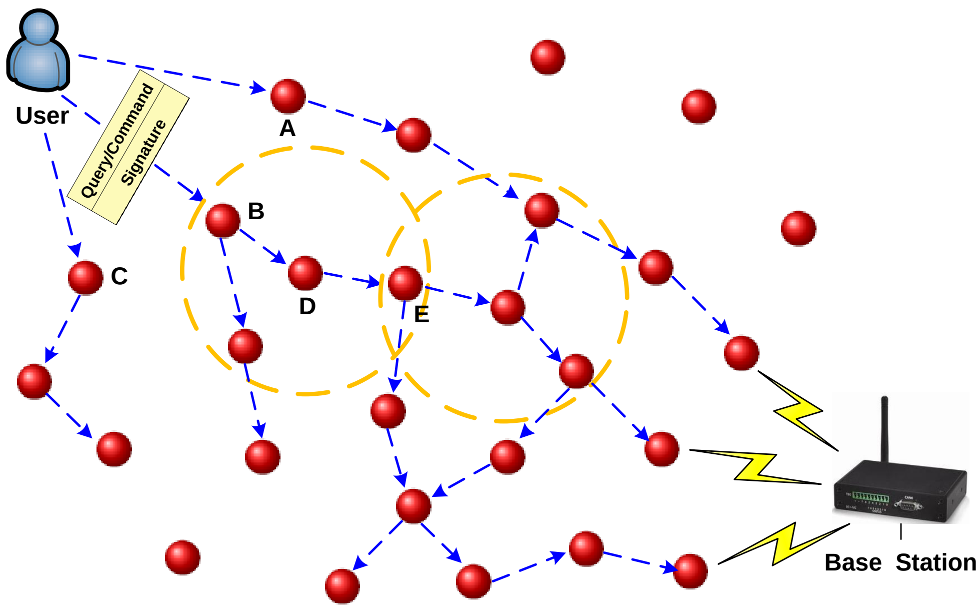


Figure 5.1: User broadcast in wireless sensor networks. A broadcast package is usually forwarded multiple times through multi-hop communication.

proposed acceleration technique of signature verification is also applicable to other public-key based multi-user broadcast authentication schemes such as those in [29, 145, 146].

For verifying an ECDSA signature, each node needs to calculate $R' = s^{-1}(eP + rQ) = l_1P + l_2Q$, where $e = H(M), l_1 = s^{-1}e$ and $l_2 = s^{-1}r$ (see Section 5.2.2). In other words, two scalar multiplications l_1P and l_2Q have to be computed on each node. Our acceleration technique comes from the following key observation: all sensor nodes independently execute the same signature verification procedure during the broadcast authentication. Therefore, if some sensor nodes would like to consume their energy to release some intermediate results, the signature verification of their neighbors can be accelerated significantly. Moreover, the energy consumption of the whole network will be decreased as well. Our idea is clearly illustrated in the following Figure 5.2.

In Figure 5.2, a user's broadcast package $\langle M, r, s\rangle$ will be received by nodes A, B and C, where M denotes a user's query or command and (r, s) is the corresponding ECDSA signature of M. When all these three nodes finish the signature verification successfully, nodes A (the green node) and B (the yellow node) decide to release (i.e., locally broadcast) their intermediate computation results l_1P and l_2Q, respectively, whereas node C would like to keep silent. By this means, nodes D and E (the orange nodes), which are the neighbors of node A, can fast verify the digital signature by performing an elliptic curve point addition $l_1P + l_2Q$, where l_2Q is computed by nodes D and E themselves and l_1P comes from the contribution of node A. Moreover, nodes F and G can also perform fast

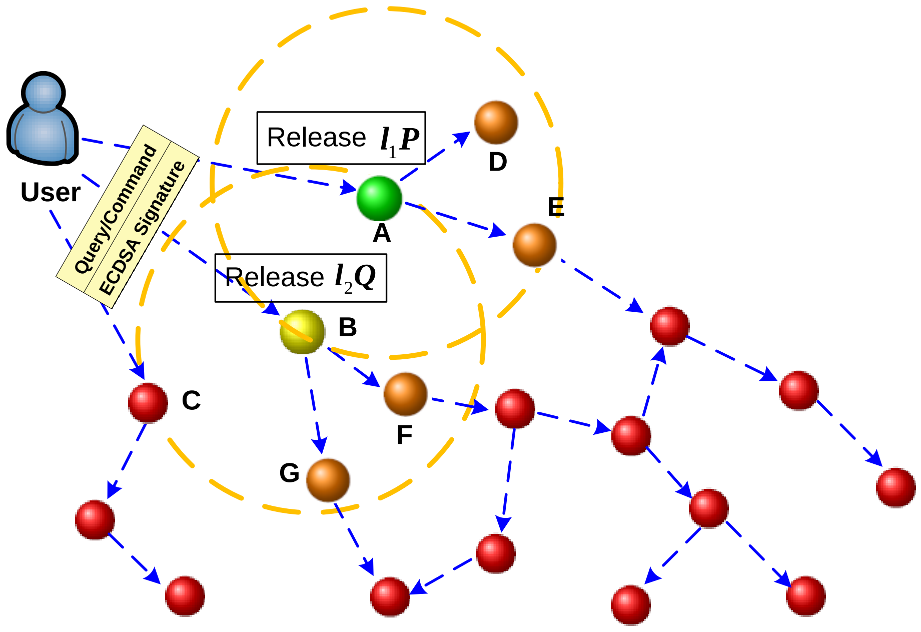


Figure 5.2: Faster ECDSA digital signature verification through nodes cooperation. Nodes A and B release l_1P and l_2Q, respectively, which will significantly accelerate the signature verification of nodes D to G as a result.

signature verification in a similar way. Hence, if some node in WSN releases its intermediate computation result, all its neighbors can fast verify the digital signature by calculating one scalar multiplication and one elliptic curve point addition, which can achieve about 50% performance improvement as compared to the traditional signature verification procedure. For the scenario described in Figure 5.2, the acceleration of signature verification on nodes D to G benefits from the release of the intermediate computation results from nodes A and B. Note that some sensor nodes might receive both intermediate computation results l_1P and l_2Q from their neighboring nodes. However, the sensor nodes cannot use both received l_1P and l_2Q to fast verify the signature with one elliptic curve point addition. The reason is that an adversary can capture a sensor node and easily launch the following attack:

Step 1. The attacker generates a bogus message $\widehat{M}$;

Step 2. The attacker randomly chooses two integers $l'_1, l'_2 \in [1, n-1]$ and calculates $\widehat{R} = l'_1P + l'_2Q$;

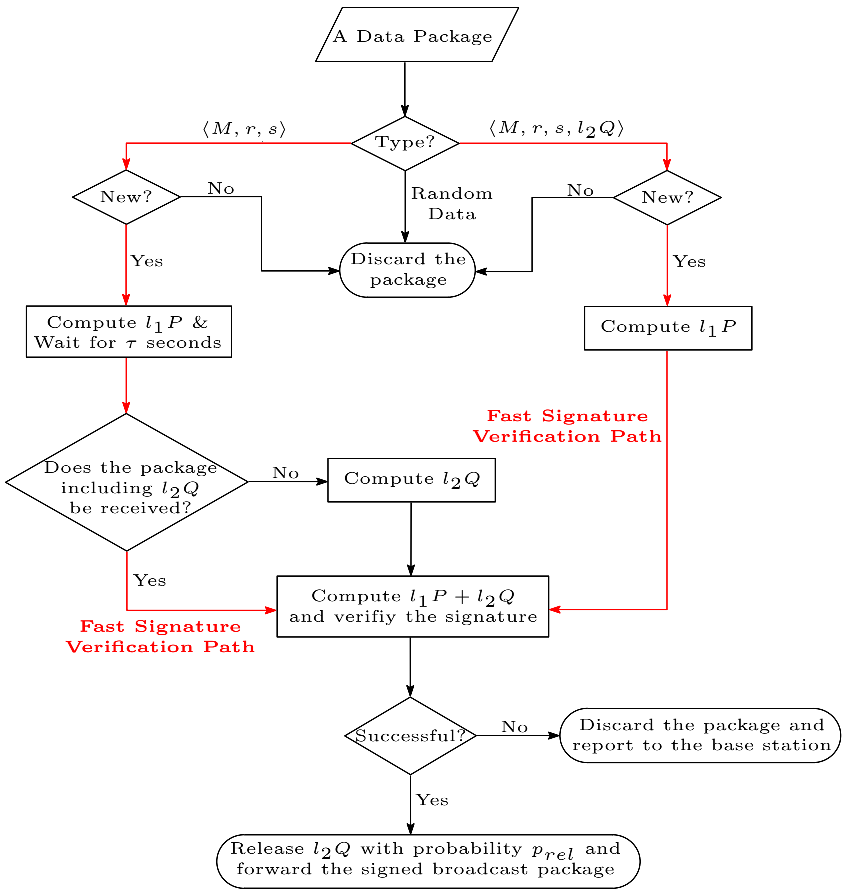
Step 3. The attacker takes the x-coordinate $\hat{r}$ of $\widehat{R}$ together with some random integer $\hat{s} \in [1, n-1]$ to form the fake signature pair $(\hat{r}, \hat{s})$;

Step 4. The attacker successively releases two bogus broadcast packages $\langle \widehat{M}, \hat{r}, \hat{s}, l_1'P \rangle$ and $\langle \widehat{M}, \hat{r}, \hat{s}, l_2'Q \rangle$ to its neighbors;

Step 5. The victims compute $l_1'P + l_2'Q$ and compare the x-coordinate of the result with the received $\hat{r}$. As a result, the victims accept $\widehat{M}$ as a valid message.

Hence, if sensor nodes use two received intermediate computation results to verify a signature, they might accept any bogus broadcast messages from an attacker.

To avoid the above attack, we only allow sensor nodes to use at most one intermediate result (i.e., l_1P or l_2Q) from their neighboring nodes for signature verification. Moreover, for the sake of simplicity of presentation, we further assume that if some sensor nodes release their intermediate computation results they will release l_2Q in the rest of this chapter. We first illustrate a basic scheme of our faster ECDSA signature verification procedure for a sensor node in the following Figure 5.3.



A Data Package
Type?
$\langle M, r, s \rangle$
$\langle M, r, s, l_2Q \rangle$
Random Data
New?
No
Yes
Discard the package
New?
No
Yes
Compute l_1P & Wait for τ seconds
Compute l_1P
Fast Signature Verification Path
Does the package including l_2Q be received?
No
Compute l_2Q
Yes
Compute $l_1P + l_2Q$ and verifiy the signature
Fast Signature Verification Path
Successful?
No
Discard the package and report to the base station
Yes
Release l_2Q with probability p_{rel} and forward the signed broadcast package

Figure 5.3: Faster ECDSA Signature Verification for WSNs (Basic Scheme)

Let SCA and ADD denote the elliptic curve scalar multiplication and the elliptic curve point addition, respectively. In the basic scheme, a sensor node may receive a data package $\langle M, r, s\rangle$ or $\langle M, r, s, l_2Q\rangle$. If a fresh package $\langle M, r, s\rangle$ is received[3], the sensor node will first compute l_1P and then wait for a very short time period τ to see whether it can obtain useful information from its neighbors for accelerating the signature verification. If it is, the node can finish the signature verification with 1 SCA + 1 ADD. Otherwise, the node will complete the verification itself with 2 SCA + 1 ADD after the time period τ. On the other hand, if a fresh package $\langle M, r, s, l_2Q\rangle$ is received, the sensor node will first calculate l_1P and then perform a fast signature verification with 1 SCA + 1 ADD. For the above two cases, if the signature is verified successfully, the sensor node will continue forwarding the broadcast package to its neighbors. Moreover, the intermediate computation result l_2Q will also be released with probability p_{rel}. Otherwise, if the signature verification is failed[4], the sensor node will send a signed report to the base station. Once the base station receives enough reports from the network, it will perform appropriate security mechanisms (see [180] for an example) to identify comprised nodes in WSN. Although the above basic scheme is simple and efficient, it is still vulnerable to the following attack:

Step 1. The attacker generates a bogus message $\widehat{M}$;

Step 2. The attacker randomly chooses an integer $k' \in [1, n-1]$, computes $R' = k'P$ and sets r' to be the x-coordinate of R';

Step 3. The attacker randomly chooses an integer $s' \in [1, n-1]$ and computes $l_2'Q = R' - s'^{-1}e'P$, where $e' = H(\widehat{M})$;

Step 4. The attacker uses (r', s') as the signature of the message $\widehat{M}$ and releases the bogus broadcast package $\langle \widehat{M}, r', s', l_2'Q\rangle$ to its neighbors;

Step 5. The victims compute $l_1'P + l_2'Q$ and compare the x-coordinate of the result with the received $\hat{r}$. As a result, the victims accept $\widehat{M}$ as a valid message.

The above attack works because the attacker knows that all its neighbors will first calculate $l_1P = s^{-1}eP$ and then use the released value l_2Q to accelerate their signature verifications. Consequently, the attacker chooses random r', s' and the bogus message $\widehat{M}$, and forges the broadcast package $\langle \widehat{M}, r', s', l_2'Q\rangle$ that can pass the signature verification equation. Note that if sensor nodes use traditional ECDSA signature verification procedure (see Section 5.2.2) the bogus broadcast package $\langle \widehat{M}, r', s', l_2'Q\rangle$ cannot pass the verification

[3]The sensor node will check the time stamp, which is included in the message M, for the freshness of the received data package.

[4]Both the transmission errors or attacks from compromised nodes can result in the failure of signature verification in WSN.

since the released $l_2'Q$ is equal to $s'^{-1}r'Q$ with negligible probability. To thwart the above attack for the basic scheme, we propose an enhanced scheme as shown in the following Figure 5.4, which takes advantage of the redundancy of broadcast packages in WSN.

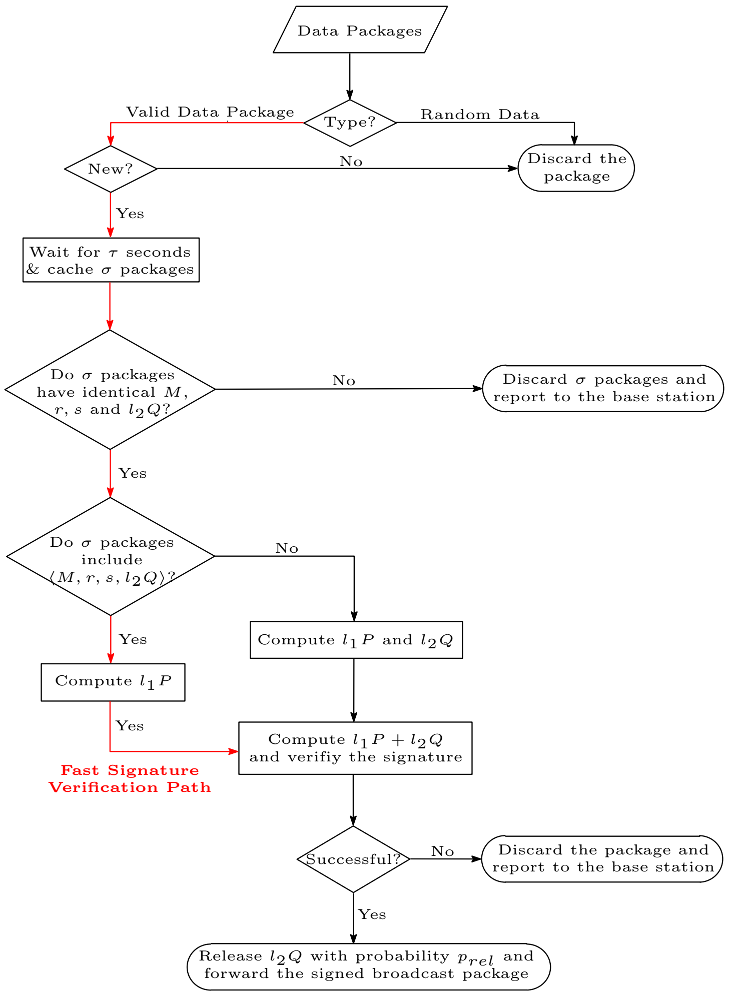

Data Packages
Type?
Valid Data Package
Random Data
New?
No
Discard the package
Yes
Wait for τ seconds & cache σ packages
Do σ packages have identical M, r, s and l_2Q?
No
Discard σ packages and report to the base station
Yes
Do σ packages include $\langle M, r, s, l_2Q \rangle$?
No
Compute l_1P and l_2Q
Yes
Compute l_1P
Yes
Compute $l_1P + l_2Q$ and verifiy the signature
Fast Signature Verification Path
Successful?
No
Discard the package and report to the base station
Yes
Release l_2Q with probability p_{rel} and forward the signed broadcast package

Figure 5.4: Faster ECDSA Signature Verification for WSNs (Enhanced Scheme)

In the enhanced scheme, each sensor node first waits for τ seconds and caches σ data packages (i.e., $\langle M, r, s\rangle$ or $\langle M, r, s, l_2Q\rangle$) received from its neighbors, where τ and σ are selected such that the sensor node can receive at least one data package from an honest neighbor (see Section 5.4.3). The sensor node then checks whether the cached σ data packages have identical M, r, s and l_2Q. Note that the main goal of adversaries is to broadcast fake data packages into WSN. While all honest nodes forward a correct data package $\langle M, r, s\rangle$ or $\langle M, r, s, l_2Q\rangle$ to their neighboring nodes, adversaries try to deceive their neighbors by broadcasting a bogus data package $\langle M', r', s', l_2'Q\rangle$ as described in the basic scheme. Hence, if the senor node finds that the received data packages have different M, r, s or l_2Q, it will report the potential attack to the base station immediately. On the other hand, if all the cached σ data packages have identical M, r, s and l_2Q, the sensor node will further check whether it has received useful data packages $\langle M, r, s, l_2Q\rangle$ for accelerating signature verification. If it is, the sensor node will calculate l_1P and then complete the signature verification with 1 SCA + 1 ADD. Otherwise, the sensor node will perform the traditional ECDSA signature verification with 2 SCA + 1 ADD. The remaining steps after the signature verification are the same as those in the basic scheme.

5.4.3 Selection of System Parameters

In this subsection, we provide guidance for selecting the system-wide parameters τ, σ and p_{rel}.

Selection of the Delay τ and the Threshold σ

In the enhanced scheme, a sensor node needs to wait for τ seconds and cache σ data packages. We assume that on average a sensor node A has λ neighbors and half of them will broadcast data packages to A at a certain communication round[5]. We further assume that among A's $\lambda/2$ neighbors μ nodes can be compromised by adversaries and each of them can send at most ν bogus data packages to A during that communication round. Note that all compromised nodes must collude to send *identical* bogus data packages to A. Otherwise, A will discard all cached data packages and report to the base station. To thwart the collusive attacks from adversaries, the threshold σ should satisfy the following condition:

$$\lambda/2 \geq \sigma \geq \mu \cdot \nu + 1.$$

The above condition guarantees that the sensor node A can receive at least one correct broadcast package from an honest neighbor. As a result, A will not accept the bogus messages for collusive adversaries since all the catched data packages are not identical.

After the threshold σ is determined, we can choose the delay τ such that σ data packages can be received by the sensor node A. If a CC2420 IEEE 802.15.4 radio transceiver from Texas Instruments [167] is used in sensor nodes, the data transmission rate can achieve 250Kbps. We also assume that the signed broadcast package fits in the maximum allowable transmission limit (i.e., 128 bytes) of the CC2420 radio transceiver. So a 128-byte broadcast

[5]The procedure of broadcast authentication can be divided into a series of communication rounds. Although a sensor node A has λ neighbors, only half of them will broadcast packages to A on average at certain communication round and the other half will receive the broadcast package from A in the following communication round.

package will be transmitted in the physical layer and the transmission delay is about 4.096ms. Moreover, we also need to consider the CC2420 radio backoff that is a period of time where the radio pauses before attempting to transmit. Two backoff periods, namely initial backoff and congestion backoff, can be chosen for the CC2420 radio [167]. The initial backoff is $1 \sim 32$ backoff units[6] (i.e., 300μs - 10ms), whereas the congestion backoff is $1 \sim 8$ backoff units (i.e., 300μs - 2.5ms). Therefore, in the worst case a 128-byte broadcast package can be received by the sensor node A within around 17ms. Taking into account all these factors, we the delay τ should satisfy the following condition:

$$\tau \geq 17 \cdot 10^{-3} \cdot \sigma.$$

Selection of the Release Probability p_{rel}

In the enhanced scheme, p_{rel} is the probability that a node will release its intermediate computation result, which is a predefined system parameter characterizing the trade-off between the verification speed of digital signature and the energy consumption of WSNs. Generally speaking, if a large value of p_{rel} is used, more sensor nodes will consume additional energy to broadcast their intermediate computation results and the signature verification for a large group of sensor nodes will be accelerated as a result, and vice versa. The selection of the release probability p_{rel} is closely related to the topology and the deployment of a WSN. Once the WSN is deployed, a network administrator first needs to analyze the network topology and estimates the average number of sensor nodes that might work on the signature verification during the transmission of a broadcast package. The administrator then determines a release probability p_{rel} that can achieve the optimal trade-off between the energy consumption of the whole network and the efficiency of the signature verification. More specifically, assuming that there are on average N sensor nodes working on the signature verification at each communication round, the probability that T sensor nodes will release their intermediate computation results is

$$p_T = \binom{N}{T} \cdot p_{rel}^T \cdot (1 - p_{rel})^{(N-T)}.$$

Let E_s, E_r, and E_{sca} be the energy consumption of sending and receiving one packet, and calculating one elliptic curve scalar multiplication on sensor nodes, respectively. Recall that we assume that each node has λ neighbors on average. Then we can roughly estimate the additional energy consumption/saving due to the use of our fast signature verification technique as follows:

- T sensor nodes will locally broadcast their intermediate computation results, the energy consumption of which is $T \cdot E_s$;

[6]The units of backoff are 10 jiffies (32KHz ticks).

- About $\frac{\lambda T}{2}$ sensor nodes will receive the intermediate computation results, the energy consumption of which is $\frac{\lambda T}{2} \cdot E_r$;
- About $\frac{\lambda T}{2}$ sensor nodes will accelerate their signature verifications using the received intermediate computation results, the energy saving of which is $\frac{\lambda T}{2} \cdot E_{sca}$.

Therefore, the expected additional energy consumption/saving will be

$$\overline{E} = \sum_{T=1}^{N} p_T \cdot (T \cdot E_s + \frac{\lambda T}{2} \cdot E_r - \frac{\lambda T}{2} \cdot E_{sca}). \tag{5.1}$$

Note that the value in the round bracket of Equation (5.1) might be positive or negative, which depends on the energy consumption of the microcontroller and the radio component on different sensor motes. If the above value is positive, we need to choose a release probability p_{rel} that can minimize the energy consumption $\overline{E}$. Otherwise, we select a p_{rel} that can maximize the energy saving $\overline{E}$. Here, we only provide the guidance about the selection of the release probability p_{rel} and omit the details of the specific applications.

5.5 Security and Performance Analysis

In this section, we analyze the performance of the proposed acceleration technique for signature verification with respect to communication and computation overheads (in terms of energy consumption). The performance analysis focuses on a simple 4×4 grid-based WSN. We also compare our scheme with the traditional ECDSA signature verification when applied to the broadcast authentication in the 4×4 grid-based WSNs.

5.5.1 Case Study

Note that the performance of our faster signature verification technique is closely related to the deployment of WSNs and the distribution of attackers in the network. To analyze the performance of our scheme, we conduct a case study for the broadcast authentication problem in a 4×4 grid-based WSN, as illustrated in the following Figure 5.5.

In the above sensor network, each node only can directly communicate with its one-hop neighbors. A user sends its signed broadcast package to the node 1 at Round 0. After six communication rounds, the broadcast package will be received and verified by all sensor nodes. Furthermore, in our faster signature verification scheme we assume that one sensor node will release the intermediate computation result l_2Q in each communication round (see the green nodes $1, 2, 6, 7, 11$ and 12 in Figure 5.5).

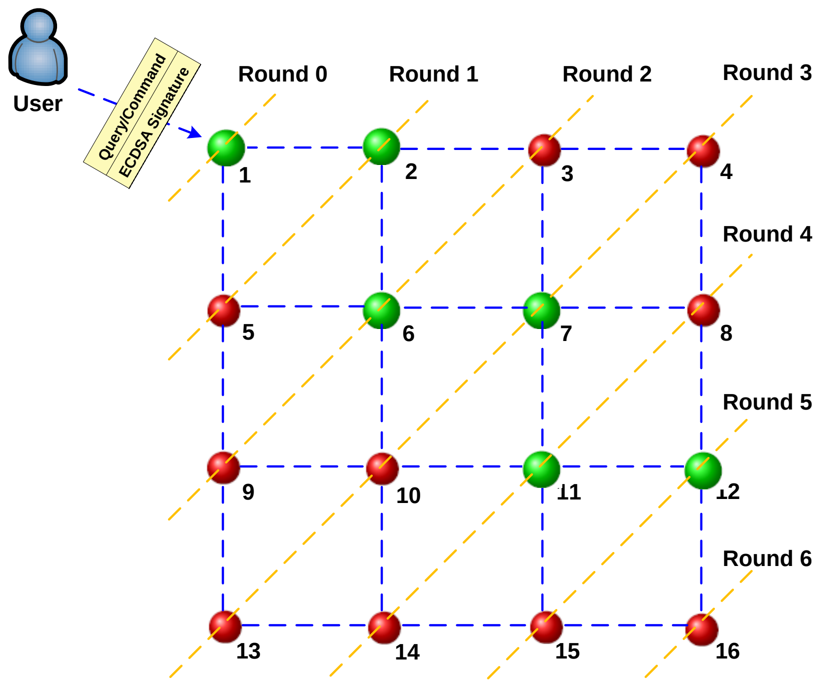


Figure 5.5: Broadcast Authentication in a 4 × 4 grid-based WSN.

To give a detailed quantitative analysis, we further assume that MICAz motes are used in the WSN. Under a typical configuration such as a 3V supply voltage and a 7.37MHz clock frequency, the MICAz mote draws a current of 12mA in an active mode (i.e., CPU is operating) [41]. Based on the formulae of calculating the energy consumption on MICAz motes [51], we obtain the following basic facts:

- A Chipcon CC2420 radio used in MICAz motes consumes $E_s = 83.6\mu$J and $E_r = 90.4\mu$J to transmit and receive l_2Q with 40 bytes[7], respectively;
- An Atmega128L microcontroller used in MICAz motes consumes about $E_{ver} = 22.68$mJ and $E_{sca} = 11.52$mJ to verify an ECDSA signature and compute a scalar multiplication on a 163-bit Koblitz curve.

Note that we will not count the energy consumption of sensor nodes when sending and receiving the broadcast package throughout the whole network since it is the same for both faster and traditional signature verification schemes. We only compare the difference of both schemes in terms of communication and computation overhead in the next subsection.

[7]Assuming that a 160-bit elliptic curve cryptosystem is employed in WSNs, the size of l_2Q is around 40 bytes.

5.5.2 Performance in the Ideal Case

In this subsection we analyze the performance of our faster signature verification technique in the ideal case (i.e., no adversaries exist), which can serve as the upper bound of performance improvement when applying our approach to broadcast authentication in the 4×4 grid-based WSN. In our scheme, some sensor nodes need to release their intermediate computation results in order to accelerate the signature verification for their neighboring nodes. Hence, our scheme consume more energy for transmitting the intermediate computation result l_2Q, when compared to using the traditional ECDSA signature verification in WSNs. In the 4×4 grid-based WSN (see Figure 5.5), the six green nodes (i.e., Nodes 1, 2, 6, 7, 11, 12) will locally broadcast their intermediate computation results to their one-hop neighbors, which causes an extra energy consumption of $6 \times 83.6\mu\text{J} = 501.6\mu\text{J}$ in the network. Furthermore, we list the sensor nodes that will receive and use the intermediate computation results during the broadcast authentication in the following Table 5.1. Note

Table 5.1: Senders and Receivers of the Intermediate Computation Results in Faster ECDSA Verification

Round	Senders	Receivers
0	node 1	nodes 2, 5
1	node 2	nodes 3, 6
2	node 6	nodes 7, 10
3	node 7	nodes 8, 11
4	node 11	nodes 12, 15
5	node 12	node 16
6	—	—

that although some sensor nodes (e.g., node 6) has four one-hop neighbors, only two of them (i.e., nodes 7 and 10) will receive the intermediate computation results since the other two (i.e., nodes 2 and 5) have finished the signature verification in the previous round (i.e., Round 1) and gone into the power-saving sleep mode. Therefore, there are totally 11 sensor nodes receiving the intermediate computation results, which causes an extra energy consumption of $11 \times 90.4\mu\text{J} = 994.4\mu\text{J}$ in the WSN. In brief, our faster signature verification incurs an extra energy consumption of $501.6 + 994.4 = 1496\mu\text{J} \approx 1.5\text{mJ}$ for transmitting (i.e., sending and receiving) the intermediate computation results for the WSN in question.

With respect to the computation aspect, it is not difficult to find that the signature verification on Nodes 2, 3, 5, 6, 7, 8, 10, 11, 12, 15 and 16 (see Table 5.1) will be accelerated by 50% (i.e., saving one elliptic curve scalar multiplication) due to the use of the intermediate computation results from their neighboring nodes, which leads to a significant energy saving of $11 \times 11.52\text{mJ} = 126.72\text{mJ}$ in the WSN, as compared to the traditional ECDSA signature verification technique. Moreover, the signed broadcast package will be forwarded to other sensor nodes only if the signature verification is successful. Therefore, the performance improvement on the signature verification will also reduce the transmission delay of the broadcast package by 50% in each round accordingly. Consequently, the service quality of the broadcast authentication in WSNs has been improved remarkably by using our faster signature verification technique.

To sum up, for the broadcast authentication in the target 4×4 grid-based WSN, our faster signature verification can save the energy consumption of $126.72\text{mJ} - 1.5\text{mJ} = 125.22\text{mJ}$ in total, considering both the communication and computation overheads. Therefore, using the proposed signature verification technique, one can save up to

$$\frac{125.22\text{mJ}}{16 \times 22.68\text{mJ}} \times 100\% = 34.5\%$$

energy consumption for the grid-based WSN in question.

5.5.3 Security and Performance under Attacks from Independent Adversaries

In this subsection, we analyze the security and performance of our scheme when there exist independent adversaries in the 4×4 grid-based WSN. To this end, we assume that two independent adversaries (i.e., 12.5% nodes are compromised and become malicious nodes.) are deployed in the WSN and each of them will broadcast a bogus package to its neighbors. Note that in the grid-based WSN each bogus package will be received by two sensor nodes in the following communication round. To maximize the influence of independent adversaries, we select Node 3 and Node 9 as two independent adversaries in the 4×4 grid-based WSN. Moreover, for those nodes (i.e., Nodes 2, 3, 4, 5, 9, 11) that can only receive one data package from its neighbors in certain communication rounds, they will verify the signature themselves in order to avoid the attack descried in the basic scheme. Other nodes (i.e., Nodes 6, 7, 8, 10, 11, 12, 15, 16) will cache two received data packages from their neighbors and decide whether they can perform faster signature verifications. Like the ideal case, we also assume that the six green nodes (i.e., Nodes 1, 2, 6, 7, 11, 12) will locally broadcast the intermediate computation results l_2Q to their neighbors.

Under the above settings, our faster signature verification still incurs an extra energy consumption of 1.5mJ for transmitting the intermediate computation results l_2Q like the ideal case. However, due to the existence of independent adversaries Node 3 and Node 9, Node 7 and Node 10 will receive two different data packages from their neighbors. Therefore, Node 7 and Node 10 have to verify the signature themselves instead of using the released l_2Q from Node 6 for faster verification. Consequently, the signature verification will be accelerated only for Nodes 6, 8, 11, 12, 15, 16 in this case, which can save the energy of $6 \times 11.52\text{mJ} = 69.12\text{mJ}$ in the WSN. Combining both communication and computation overheads, one can obtain that in the case of two independent adversaries the total energy saving is

$$\frac{69.12 - 1.5\text{mJ}}{16 \times 22.68\text{mJ}} \times 100\% = 18.7\%$$

for the 4×4 grid-based WSN. In addition, attacks from two independent adversaries have no effect on the security of our scheme and all the bogus data packages from independent adversaries are also discarded by legitimate nodes.

5.5.4 Security and Performance under Attacks from Collusive Adversaries

In this subsection, we analyze the security and performance of our scheme under the existence of collusive adversaries in the 4×4 grid-based WSN. Again, we assume that two collusive adversaries are deployed in the WSN and they will broadcast *identical* bogus data packages to their neighbors. To maximize the influence of collusive adversaries, we choose Node 2 and Node 5 as two collusive adversaries in the 4×4 grid-based WSN. Furthermore, we use the same assumptions as the case of independent adversaries for other nodes. Note that Node 6 will be cheated by collusive adversaries because of receiving two *identical* bogus data packages from Node 2 and Node 5. Although Node 6 continues broadcasting the bogus data package, Node 7 and Node 10 will discard it and verify the signature themselves because they receive two different data packages. As a result, bogus data packages from two collusive adversaries cannot be injected into the WSN successfully. Moreover, if Node 6 listens to the channel for one more communication round after broadcasting the bogus package, it is also possible for Node 6 to find the potential attacks itself. More specifically, after Node 7 and Node 10 verify the signature successfully at Round 3, they will broadcast the correct data package to their neighbors. Node 6 will find that all the data packages received from its neighbors (i.e., Nodes 2, 5, 7, 10) are different and therefore some attacks have happened.

Like other cases, our faster signature verification needs to consume an extra energy of 1.5mJ for transmitting the intermediate computation results l_2Q. However, due to the existence of collusive adversaries Node 2 and Node 5, Node 6 will be fooled and then broadcast bogus data packages to Node 7 and Node 10. Note that both Node 7 and Node 10 will receive two different data packages from their neighbors and therefore verify the signature themselves. As a result, the signature verification will be accelerated only for Nodes 8, 11, 12, 15, 16 in this case, which can reduce the energy consumption of the WSN by $5 \times 11.52\text{mJ} = 57.6\text{mJ}$. Taking both communication and computation overheads into consideration, one can save around

$$\frac{57.6 - 1.5\text{mJ}}{16 \times 22.68\text{mJ}} \times 100\% = 15.5\%$$

energy consumption for the 4×4 grid-based WSN. In particular, attacks from two collusive adversaries have very limited effect on the security of our scheme and those attacks can also be detected in the certain communication round.

5.6 Conclusions

Signature-based broadcast authentication schemes for WSNs have attracted a lot of attention in recent years due to their desirable features such as strong security resilience, good scalability and immediate message authentication. However, the relatively slow signature verification in public-key cryptosystems causes high energy consumption and long verification delay for broadcast authentication in WSNs. In this chapter, we propose a novel and efficient acceleration technique for signature verification in WSNs. Our scheme fully exploits the *cooperation* among sensor nodes and enables the significant energy consumption saving for the whole network. As a case study, we apply our technique to the broadcast authentication in a 4×4 grid-based WSN and analyze the security and performance of our scheme under the existence of independent and collusive adversaries. While independent adversaries do not have any influence on the security of our scheme, collusive adversaries have very limited effect. Particularly, bogus data packages from adversaries cannot be disseminated successfully through the entire network in both cases. Moreover, a quantitative performance analysis shows that our scheme can save about $15.5\% \sim 34.5\%$ energy consumption and run 50% faster than traditional signature verification method.

As our future work, we are particularly interested in implementing our scheme on real sensor platforms. Moreover, we will further analyze the effect when applying our scheme to other WSNs with more complicated topologies and deployments. Comparing our method with other faster signature verification techniques (see [2] for an example) as well as conducting a large-scale experiment to evaluate our scheme in real world applications are also highly desirable.

Chapter 6

Conclusions and Future Research

This chapter summarizes the research contributions of this thesis. A summary of the main results as well as recommendations for future research will be provided.

6.1 Conclusions

In this thesis we deal with security solutions for mobile ad hoc networks, with emphasis on efficient cryptographic algorithms and protocols that are targeted for resource-constrained mobile devices.

Motivated by the design of the well-known Enigma machine, we first suggest a novel ultra-lightweight cryptographic algorithm, Hummingbird, for resource-constrained devices. Hummingbird is an elegant combination of block cipher and stream cipher with 16-bit block size, 256-bit key size, and 80-bit internal state. In particular, Hummingbird was developed with both lightweight software and lightweight hardware implementations for constrained devices in mind. Using a hybrid structure enables Hummingbird to provide the designed security with small block size which is expected to meet the stringent response time and power consumption requirements for various embedded applications. A preliminary security analysis shows that Hummingbird seems to be resistant to the most common attacks to block ciphers and stream ciphers including birthday attacks, differential and linear cryptanalysis, structure attacks, algebraic attacks, cube attacks, and so on. Efficient software implementations of Hummingbird across a range of low-cost and low-power microcontrollers obtain the following favorable results, when compared to the state-of-the-art ultra-lightweight block cipher PRESENT:

- On the 4-bit microcontroller ATAM893-D from Atmel, Hummingbird can achieve up to 2.14 times faster throughput than PRESENT, after a system initialization process.

- On the 8-bit microcontroller ATmega128L from Atmel, the throughput of Hummingbird is about 40 and 0.7 times faster than that of PRESENT (implemented on a similar platform) for a size-optimized and a speed-optimized implementation, respectively, after a system initialization process.

- On the 16-bit microcontroller MSP430 from Texas Instruments, Hummingbird can even achieve up to 147 and 4.7 times faster throughput than PRESENT (implemented on a similar platform) for a size-optimized and a speed-optimized implementation, respectively, after a system initialization process.

In addition, efficient hardware implementations of Hummingbird cores on the low-cost FPGAs also demonstrate good performance of Hummingbird. More specifically, on the Spartan-3 XC3S200 FPGA device, the speed optimized encryption core can achieve a throughput of 160.4 Mbps at the cost of 273 slices, whereas the area optimized encryption core can be implemented in 253 slices and operate at 66.1 Mbps.

Next, we propose new variants of Miller's algorithm for computing the Tate pairing on two families of non-supersingular genus 2 hyperelliptic curves over prime fields with efficiently computable automorphisms. We describe how to exploit the automorphism in lieu of the order of the torsion group to construct the rational functions required in Miller's algorithm, and shorten the length of the main loop in Miller's algorithm as a result. In the best case, the length of the loop in our new variants can be up to 4 times shorter than that of the original Miller's algorithm. Furthermore, we also generalize Chatterjee *et al.*'s idea [33] of encapsulating the computation of the line function with the group operations to genus 2 hyperelliptic curves, and derive new explicit formulae for the group operations in projective and new coordinates in the context of pairing computations. When compared to Lange's formulae [105], our mixed-addition formulae can save $5M$ and $3M$ in projective and new coordinates, respectively, whereas our doubling formulae can save $2M$ in both projective and new coordinates. As a case study, we combine our new algorithm with various optimization techniques in the literature to efficiently implement the Tate pairing on a non-supersingular genus 2 curve $y^2 = x^5 + 9x$ over $\mathbb{F}_p$ with an embedding degree of $k = 4$. When compared with pairing computations on supersingular genus 2 curves at the same security level, our new algorithm can obtain 55.8% performance improvements algebraically. Furthermore, favorable experimental results have been obtained for the implementation of the Tate pairing on both a supersingular and a non-supersingular genus 2 curve with embedding degree 4.

Revoking the cryptographic keys of malicious nodes is crucial for the security and robustness of MANETs. We propose a fully self-organized key revocation scheme for MANETs based on the Dirichlet multinomial model and identity-based cryptography. Having noted the interactive behavior of nodes during the procedure of key revocation, we borrowed ideas from reputation systems to design our key revocation mechanism. Firmly rooted in statistics, our key revocation scheme provides a theoretically sound basis for nodes analyzing and predicting peers' behavior based on their own observations and other nodes' reports. In our scheme, each node keeps track of three categories of behavior defined and classified by an external TTP, and updates its knowledge about other nodes' behavior with 3-dimension Dirichlet distribution. Differentiating between suspicious behavior and malicious behavior enables nodes to make different responses by either revoking keys of nodes showing malicious behavior or ceasing the communication with nodes showing suspicious behavior for some time. Extensive simulations show that our key revocation scheme is effective against the false accusation attacks from independent and collusive adversaries. Moreover, our key revocation mechanism can also be adapted to PKI schemes in MANETs with off-line or on-line CAs.

Finally, we address the issue of speeding up the signature verification for public-key based multi-user broadcast authentication schemes in WSNs. We exploit the cooperation among sensor nodes and allow some sensor nodes to randomly release their intermediate computation results to their neighbors during the signature verification. In this way, many sensor nodes can use the received intermediate computation results to accelerate their signature verification. We demonstrate the performance of the proposed technique by conducting a case study for the broadcast authentication in a 4×4 grid-based WSN. A quantitative performance analysis shows that our scheme needs $15.5\% \sim 34.5\%$ less energy and runs about 50% faster than the traditional signature verification method.

6.2 Future Work

Providing security solutions for MANETs is an interesting and challenging research area. While many problems have been addressed successfully, there are many others that still need further study. Even the problems that have been addressed might have to be revisited in terms of improvements in wireless communication technology and computational capability of mobile devices. While the former makes possible the use of higher bandwidths, the latter enables the execution of more complicated cryptographic mechanisms. This section will provide the reader with an overview of several interesting research areas in which further work could be pursued.

Side Channel Attacks on Lightweight Cryptographic Primitives

Since the first introduction by Kocher in 1996 [100], side channel attacks have become an important area of cryptanalytic research. Instead of performing a mathematical attack on a cryptographic algorithm, side channel attacks exploit the information like execution time, power consumption or electromagnetic radiation gained from the physical implementation of a cryptosystem to deduce the secret key. Although quite a few lightweight cryptographic primitives have been proposed during the past few years, the evaluation whether side channel attacks are applicable to those algorithms did not been extensively investigated. Note that lightweight cryptographic primitives will be finally implemented on resource-constrained mobile devices to provide security functionalities. Considering that many resource-constrained devices in MANETs might be deployed in unattended and even hostile environments, it is possible that adversaries will capture those devices and launch various side channel attacks to extract the secret key. Hence, evaluating the security of lightweight cryptographic primitives against various side channel attacks is mandatory to guarantee the long-term security of MANETs.

Security and Privacy in RFID Systems

Radio Frequency Identification (RFID) is a rapidly developing technology enabling automatic objects identification. In an RFID system, each object is labeled with a small transponder, called an RFID *tag*, which receives and responds to radio-frequency queries from a transceiver, called an RFID *reader*. An RFID tag is composed of a tiny integrated circuit for storing and processing identification information, as well as a radio antenna for wireless data transmissions. RFID tags usually have constrained capabilities in every aspect of computation, communication and storage due to the extremely low production cost. Despite the low cost of RFID systems and their convenience in identifying an object without physical contact, the radio communications between RFID tags and readers also raise a number of security issues. For example, today's RFID systems do not conduct the mutual authentication between RFID readers and tags, so it is easy for an adversary to impersonate a reader or a tag to obtain sensitive information, and even launch denial-of-service (DoS) attacks. Moreover, RFID tags automatically emit their unique identifiers upon reader interrogation without alerting their users. Consequently, an adversary equipped with commodity RFID readers can effectively trace a person carrying a tagged item by linking two different sightings of the same RFID tag, which potentially violate the owner's privacy. In addition, many possible security threats arise from unprotected wireless communications between RFID readers and tags. To solve the aforementioned security and privacy issues, a privacy-preserving mutual authentication protocol is required for a reader and a tag to authenticate each other. Although numerous authentication protocols have been proposed for RFID systems, most of them are not secure and far from practical. Therefore, how to design a lightweight mutual authentication protocol, which meets all stringent requirements of RFID system, needs to be further investigated.

APPENDICES

Appendix A

Criteria for Selection of S-Boxes in Hummingbird

Let $\mathbb{F}_2 = \{0, 1\}$ and $\mathbb{F}_2^m = \{(x_0, x_1, \cdots, x_{m-1}) \mid x_i \in \mathbb{F}_2\}$. If $F(x)$ is a function from $\mathbb{F}_2^m$ to $\mathbb{F}_2^m$, i.e., a vectorial boolean function, then it is also called a S-box from $\mathbb{F}_2^m$ to $\mathbb{F}_2^m$. The Walsh transform of $F(x)$ is defined by

$$\widehat{F}(a, b) = \sum_{x \in \mathbb{F}_2^m} (-1)^{\langle a, F(x) \rangle + \langle b, x \rangle},\ a, b \in \mathbb{F}_2^m,$$

where $\langle y, z \rangle = \sum_{i=0}^{m-1} y_i z_i$ is the inner product of two binary vectors $y = (y_0, y_1, \cdots, y_{m-1})$ and $z = (z_0, z_1, \cdots, z_{m-1})$ with $y_i, z_i \in \mathbb{F}_2$.

A.1 Serpent-type S-boxes

Let $S(x)$ be an S-box from $\mathbb{F}_2^4$ to $\mathbb{F}_2^4$. Then the Walsh transform of $S(x)$ is given by

$$\widehat{S}(a, b) = \sum_{x \in \mathbb{F}_2^4} (-1)^{\langle a, S(x) \rangle + \langle b, x \rangle},$$

for any $a, b \in \mathbb{F}_2^4$. $S(x)$ is called a Serpent-type S-box if it satisfies the following properties:

C-1. For any nonzero $a, b \in \mathbb{F}_2^4$, it holds that $|\widehat{S}(a, b)| \leq 8$.

C-2. For any $a, b \in \mathbb{F}_2^4$ with $wt(a) = wt(b) = 1$, we require that $|\widehat{S}(a, b)| = 4$, where $wt(x)$ represents the Hamming weight of a binary string $x \in \mathbb{F}_2^m$.

C-3. For any nonzero $a, b \in \mathbb{F}_2^4$, it holds that $|\{x \in \mathbb{F}_2^4 \mid S(x) + S(x+a) = b\}| \leq 4$.

C-4. For any $a, b \in \mathbb{F}_2^4$ with $wt(a) = wt(b) = 1$, we have $|\{x \in \mathbb{F}_2^4 \mid S(x)+S(x+a) = b\}| = 0$.

The first two properties guarantee that the 16-bit block cipher is resistant to linear cryptanalysis, whereas the last two address differential cryptanalysis. In [107], all Serpent-type S-boxes are classified and presented by a representative in each class.

A.2 Additional Requirements for Serpent-type S-boxes

Let $x = (x_3 \| x_2 \| x_1 \| x_0)$ be the 4-bit input to the S-box $S(x)$. We can write $S(x)$ in terms of its boolean form as follows:

$$S(x) = (S^{(3)}(x) \| S^{(2)}(x) \| S^{(1)}(x) \| S^{(0)}(x)).$$

In order to resist the algebraic attack, we select S-boxes such that the degree of $S^{(i)}(x)$ is 3 for $0 \leq i \leq 3$, i.e., all components of $S(x)$ should have maximum degree. For efficient hardware implementation, the number of terms of $S^{(i)}$ should be small.

On the other hand, $S(x)$ can be viewed as a mapping from $\mathbb{F}_{2^4}$ to $\mathbb{F}_{2^4}$ where $\mathbb{F}_{2^4}$ is a finite field with 2^4 elements. Hence $S(x)$ has a polynomial representation over $\mathbb{F}_{2^4}$. Consequently, any component of $S(x)$ has a polynomial form over $\mathbb{F}_{2^4}$. In order to resist the interpolation attack, $S(x)$ and $S^{(i)}(x)$ should have as many monomial terms as possible under a polynomial representation [92, 177].

Based on the above considerations, in addition to the Serpent-type properties, we list the additional criteria for selection of S-boxes as follows:

C-5. The algebraic degrees of all four component functions in boolean representation are maximized, i.e., 3.

C-6. The number of monomial terms of each component function in boolean representation is small.

C-7. All component functions should have an approximately same number of monomial terms.

C-8. The number of monomial terms of each component function in a polynomial representation should be large under all defining polynomials for $\mathbb{F}_{2^4}$.

C-9. Select one S-box from each equivalent class.

The four S-boxes are selected according to the above nine criteria. Table A.1 lists the number of monomial terms in both boolean and polynomial representations of the selected S-boxes. Note that maximum number of monomial terms for a S-box from $\mathbb{F}_2^4$ to $\mathbb{F}_2^4$ or its component function is 15 whenever it is represented in a boolean form or a polynomial form. In Table A.1 (a), the number under $S^{(i)}$ is the number of monomial terms in $S^{(i)}$. For example, for the S-box $S_1(x)$, both $S_1^{(0)}$ and $S_1^{(1)}$ have 7 monomial terms, and both $S_1^{(2)}$ and $S_1^{(3)}$ have 6 monomial terms. In Table A.1 (b), the number under each defining polynomial of $\mathbb{F}_{2^4}$ denotes the number of monomial terms in each corresponding function. For example, under the defining polynomial x^4+x+1 for $\mathbb{F}_{2^4}$, the S-box $S_1(x)$ has 15 terms and the components $S_1^{(i)}, i = 0, 1, 2, 3$ have 14, 14, 14, and 15 monomial terms respectively.

Table A.1: The Boolean and Polynomial Forms of S-Boxes

(a) The Boolean Form

S-boxes	$S^{(0)}(x)$	$S^{(1)}(x)$	$S^{(2)}(x)$	$S^{(3)}(x)$	Total Monomial Terms	Least Degree
$S_1(x)$	7	7	6	6	26	3
$S_2(x)$	7	8	6	7	28	3
$S_3(x)$	7	6	7	7	27	3
$S_4(x)$	7	6	7	6	26	3

(b) The Polynomial Form

S-boxes and Their Components	x^4+x+1	x^4+x^3+1	$x^4+x^3+x^2+x+1$
$S_1(x)$	15	15	14
$S^{(0)}(x)$	14	14	14
$S^{(1)}(x)$	14	14	14
$S^{(2)}(x)$	14	14	14
$S^{(3)}(x)$	15	13	15
$S_2(x)$	13	13	14
$S^{(0)}(x)$	14	14	12
$S^{(1)}(x)$	14	14	14
$S^{(2)}(x)$	14	14	14
$S^{(3)}(x)$	14	12	12
$S_3(x)$	15	14	14
$S^{(0)}(x)$	14	12	14
$S^{(1)}(x)$	13	15	15
$S^{(2)}(x)$	14	14	14
$S^{(3)}(x)$	14	10	14
$S_4(x)$	14	14	14
$S^{(0)}(x)$	10	12	12
$S^{(1)}(x)$	14	14	14
$S^{(2)}(x)$	12	14	12
$S^{(3)}(x)$	14	14	14

Appendix B

Explicit Formulae for Genus 2 Curves over $\mathbb{F}_p$

In this appendix, we give efficient explicit formulae for group operations on genus 2 curves over $\mathbb{F}_p$ in projective coordinates in the context of pairing computations. Table B.1 and Table B.2 address the cases of projective coordinates. Given two divisor classes $\overline{E}_1$ and $\overline{E}_2$, Table B.1 computes the divisor class $\overline{E}_3 = [u_3(x), v_3(x)]$ and the rational function $l(x)$ such that $E_1 + E_2 = E_3 + \operatorname{div}\left(\frac{y-l(x)}{u_3(x)}\right)$ in the projective coordinate system, where $l(x) = \frac{s_1'}{r}x^3 + \frac{l_2}{rZ_2}x^2 + \frac{l_1}{rZ_2}x + \frac{l_0}{rZ_2}$. For doubling a reduced divisor class E_1, Table B.2 calculates the divisor class $\overline{E}_3 = [u_3(x), v_3(x)]$ and the rational function $l(x)$ such that $2E_1 = E_3 + \operatorname{div}\left(\frac{y-l(x)}{u_3(x)}\right)$ in projective coordinates, where $l(x) = \frac{s_1}{R'}x^3 + \frac{l_2}{R'Z_1}x^2 + \frac{l_1}{R'Z_1}x + \frac{l_0}{R'Z_1}$.

Table B.1: Mixed-Addition Formula on a Genus 2 curve over $\mathbb{F}_p$ (Projective Coordinates)

Input	Genus 2 HEC $C : y^2 = x^5 + f_3x^3 + f_2x^2 + f_1x + f_0$ $\overline{E}_1 = [U_{11}, U_{10}, V_{11}, V_{10}, 1]$ and $\overline{E}_2 = [U_{21}, U_{20}, V_{21}, V_{20}, Z_2]$	
Output	$\overline{E}_3 = [U_{31}, U_{30}, V_{31}, V_{30}, Z_3] = \overline{E}_1 \oplus \overline{E}_2$ $l(x)$ such that $E_1 + E_2 = E_3 + \text{div}\left(\frac{y-l(x)}{u_3(x)}\right)$	

Step	Expression	Cost
1	**Compute resultant $\boldsymbol{r = \text{Res}(u_1, u_2)}$:** $\tilde{U}_{11} = U_{11}Z_2, \tilde{U}_{10} = U_{10}Z_2, z_1 = \tilde{U}_{11} - U_{21}, z_2 = U_{20} - \tilde{U}_{10}$ $z_3 = U_{11}z_1, z_4 = z_2 + z_3, r = z_2z_4 + z_1^2U_{10}$	$5M, 1S$
2	**Compute almost inverse of $\boldsymbol{u_2}$ mod $\boldsymbol{u_1}$:** $inv_1 = z_1, inv_0 = z_4$	–
3	**Compute $\boldsymbol{s'}$:** $w_0 = V_{10}Z_2 - V_{20}, w_1 = V_{11}Z_2 - V_{21}, w_2 = inv_0w_0$ $w_3 = inv_1w_1, s'_1 = z_1w_0 + z_2w_1, s'_0 = w_2 - U_{10}w_3$	$7M$
4	**Precomputations:** $R = r^2, \tilde{s}'_0 = s'_0Z_2, \tilde{s}'_1 = s'_1Z_2, S = s'_1\tilde{s}'_1, \tilde{r} = r\tilde{s}'_1$	$4M, 1S$
5	**Compute $\boldsymbol{l}$:** $l_2 = s'_1U_{21} + \tilde{s}'_0, l_0 = s'_0U_{20} + rV_{20}$ $l_1 = (s'_1 + s'_0)(U_{21} + U_{20}) - s'_1U_{21} - s'_0U_{20} + rV_{21}$	$5M$
6	**Compute $\boldsymbol{U_3}$:** $w_1 = \tilde{U}_{11} + U_{21}, U_{31} = s'_1(2\tilde{s}'_0 - s'_1z_1) - RZ_2, l'_1 = l_1s'_1$ $U_{30} = \tilde{s}'_0(s'_0 - 2s'_1U_{11}) + s'^2_1(z_3 - \tilde{U}_{10} - U_{20}) + 2l'_1 + Rw_1$	$8M, 1S$
7	**Compute $\boldsymbol{V_3}$:** $w_1 = l_2s'_1 - U_{31}, V_{30} = U_{30}w_1 - S(l_0s'_1)$ $V_{31} = U_{31}w_1 + S(U_{30} - l'_1)$	$6M$
8	**Adjust:** $Z_3 = \tilde{r}S, U_{31} = \tilde{r}U_{31}, U_{30} = \tilde{r}U_{30}$	$3M$
Sum		$38M, 3S$

Table B.2: Doubling Formula on a Genus 2 Curve over $\mathbb{F}_p$ (Projective Coordinates)

Input	Genus 2 HEC $C : y^2 = x^5 + f_3x^3 + f_2x^2 + f_1x + f_0$ $\overline{E}_1 = [U_{11}, U_{10}, V_{11}, V_{10}, Z_1]$	
Output	$\overline{E}_3 = [U_{31}, U_{30}, V_{31}, V_{30}, Z_3] = [2]\overline{E}_1$ $l(x)$ such that $2E_1 = E_3 + \text{div}\left(\frac{y-l(x)}{u_3(x)}\right)$	

Step	Expression	Cost
1	**Compute resultant and precomputations:** $Z_2 = Z_1^2, \tilde{V}_{11} = 2V_{11}, \tilde{V}_{10} = 2V_{10}, w_0 = V_{11}^2, w_1 = U_{11}^2, w_2 = \tilde{V}_{10}Z_1$ $w_3 = 4w_0, w_4 = w_2 - U_{11}\tilde{V}_{11}, r = U_{10}w_3 + \tilde{V}_{10}w_4$	$4M, 3S$
2	**Compute almost inverse:** $inv_1' = -\tilde{V}_{11}, inv_0' = w_4$	–
3	**Compute $\boldsymbol{k'}$:** $w_3 = f_3Z_2 + w_1, w_4 = 2U_{10}, \tilde{w}_4 = w_4Z_1, k_1' = 2w_1 + w_3 - \tilde{w}_4$ $k_0' = U_{11}(2\tilde{w}_4 - w_3) + Z_1(f_2Z_2 - w_0)$	$5M$
4	**Compute $\boldsymbol{s'}$:** $w_0 = k_0'inv_0', w_1 = k_1'inv_1', s_2 = w_2k_1' - \tilde{V}_{11}k_0'$ $s_1' = s_2Z_1, s_0' = w_0 - Z_1U_{10}w_1$	$7M$
5	**Precomputations:** $R = rZ_2, \tilde{R} = Rs_1', R' = Rs_2, S_0 = s_0'^2$ $S_1 = s_1'^2, S = s_0's_1', s_0 = s_0's_2, s_1 = s_1's_2$	$6M, 2S$
6	**Compute $\boldsymbol{l}$:** $l_2 = s_1U_{11} + s_0Z_1, l_0 = s_0U_{10} + R'V_{10}$ $l_1 = (s_1 + s_0)(U_{11} + U_{10}) - s_1U_{11} - s_0U_{10} + R'V_{11}$	$6M$
7	**Compute $\boldsymbol{U_3}$:** $U_{30} = S_0 + R(s_2\tilde{V}_{11} + 2rZ_1U_{11}), U_{31} = 2S - R^2$	$4M, 1S$
8	**Compute $\boldsymbol{V_3}$:** $w_1 = l_2 - U_{31}, w_2 = U_{30}w_1, w_3 = U_{31}w_1$ $V_{31} = w_3 + S_1(U_{30} - l_1), V_{30} = w_2 - S_1l_0$	$4M$
9	**Adjust:** $Z_3 = S_1\tilde{R}, U_{31} = U_{31}\tilde{R}, U_{30} = U_{30}\tilde{R}$	$3M$
Sum		$39M, 6S$

Bibliography

[1] R. Anderson, E. Biham, and L. Knudsen, "Serpent: A Proposal for the Advanced Encryption Standard", available at `http://www.cl.cam.ac.uk/~rja14/Papers/serpent.pdf`.

[2] A. Antipa, D. Brown, R. Gallant, R. Lambert, R. Struik, and S. Vanstone, "Accelerated Verificaiton of ECDSA Signatures", *Selected Areas in Cryptography - SAC 2005*, LNCS 3897, B. Preneel, S. Tavares (eds.), Berlin, Germany: Springer-Verlag, pp. 307-318, 2006.

[3] G. Arboit, C. Crépeau, C. R. Davis, and M. Maheswaran, "A Localized Certificate Revocation Scheme for Mobile Ad Hoc Networks", *Ad Hoc Network*, vol.6, no.1, pp. 17-31, 2008.

[4] Atmel Corporation. "8-bit AVR Microcontroller with 128K Bytes In-System Programmable Flash – ATmega128/ATmega128L". Available at `http://www.atmel.com/dyn/resources/prod_documents/doc2467.pdf`.

[5] Atmel Corporation. AVR Studio 4.17. Available at `http://www.atmel.com/dyn/Products/tools_card.asp?tool_id=2725`.

[6] Atmel Corporation, "MARC4 4-bit Microcontrollers Programmer's Guide", available at `http://www.atmel.com/dyn/resources/prod_documents/doc4747.pdf`.

[7] Atmel Corporation, "ATAM893-D: Flash Version for ATAR080, ATAR090/890 and ATAR092/892", available at `http://www.atmel.com/dyn/resources/prod_documents/doc4680.pdf`.

[8] R. M. Avanzi, H. Cohen, C. Doche, G. Frey, T. Lange, K. Nguyen, and F. Vercauteren, *Handbook of Elliptic and Hyperelliptic Curve Cryptography*, Boca Raton, Florida, USA: Chapman & Hall/CRC, 2006.

[9] S. Babbage and M. Dodd, "The Stream Cipher MICKEY 2.0", ECRYPT Stream Cipher, available at `http://www.ecrypt.eu.org/stream/p3ciphers/mickey/mickey_p3.pdf`, 2006.

[10] P.L.S.M. Barreto, S. Galbraith, C. Ó hÉigeartaigh, and M. Scott, "Efficient Pairing Computation on Supersingular Abelian Varieties", *Design, Codes and Cryptography*, 42:239-271, 2007.

[11] P.L.S.M. Barreto, H. Y. Kim, B. Lynn, and M. Scott, "Efficient Algorithm for Pairing-Based Cryptosystems", *Advance in Cryptology - CRYPTO'2002*, LNCS 2442, M. Yung (ed.), Berlin, Germany: Springer-Verlag, pp. 354-368, 2002.

[12] M. Bellare, C. Namprempre, and G. Neven, "Security Proofs for Identity-Based Identification and Signature Schemes", *Advances in Cryptology-EUROCRYPT'2004*, LNCS 3027, C. Cachin and J. Camenisch (eds.), Berlin, Germany: Springer-Verlag, pp. 268-286, 2004.

[13] Z. Benenson, N. Gedicke, and O. Raivio, "Realizing Robust User Authentication in Sensor Networks", *Proceedings of the First Workshop on Real-World Wireless Sensor Networks (REALWSN'05)*, June 2005.

[14] E. Biham, "New Types of Cryptanalytic Attacks Using Related Keys", *Journal of Cryptology*, Vol. 7, pp. 229-246, 1994.

[15] E. Biham, "Cryptanalysis of Multiple Modes of Operation", *Journal of Cryptology*, 11(1), pp. 45-58, 1998.

[16] E. Biham, "Cryptanalysis of Triple Modes of Operation", *Journal of Cryptology*, 12(3), pp. 161-184, 1999.

[17] E. Biham and L. R. Knudsen, "Cryptanalysis of the ANSI X9.52 CBCM Mode", *Journal of Cryptology*, 15(1), pp. 47-59, 2002.

[18] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe, "PRESENT: An Ultra-Lightweight Block Cipher", *The 9th International Workshop on Cryptographic Hardware and Embedded Systems - CHES 2007*, LNCS 4727, P. Paillier, I. Verbauwhede (eds.), Berlin, Germany: Springer-Verlag, pp. 450-466, 2007.

[19] A. Biryukov and D. Wagner, "Slide Attacks", *The 6th Annual Fast Software Encryption Workshop - FSE 1999*, LNCS 1636, L. R. Knudsen (ed.), Berlin, Germany: Springer-Verlag, pp. 245-259, 1999.

[20] A. Biryukov and D. Wagner, "Advanced Slide Attacks", *Advances in Cryptology - EUROCRYPT 2000*, LNCS 1807, B. Preneel (ed.), Berlin, Germany: Springer-Verlag, pp. 589-606, 2000.

[21] J. Broch, D. Maltz, D. Johnson, Y. Hu, and J. Jetcheva, “A Performance Comparison of Multi-hop Wireless Ad Hoc Network Routing Protocols”, *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom’98)*, pp. 85-97, 1998.

[22] S. Buchegger and J.-Y. Le Boudec, “Performance Analysis of the CONFIDANT Protocol: Cooperation of Nodes – Fairness In Dynamic Ad-hoc Networks”, *Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc’02)*, pp. 226-236, 2002.

[23] S. Buchegger and J.-Y. Le Boudec, “The Effect of Rumor Spreading in Reputation Systems for Mobile Ad-hoc Networks”, *Proceedings of WiOpt’03: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, 2003.

[24] S. Buchegger and J.-Y. Le Boudec, “A Robust Reputation System for Peer-to-Peer and Mobile Ad Hoc Networks”, *Proceedings of the 2nd Workshop on the Economics of Peer-to-Peer Systems*, 2004.

[25] P. Bulens, F.-X. Standaert, J.-J. Quisquater, and P. Pellegrin, “Implementation of the AES-128 on Virtex-5 FPGAs”, *Progress in Cryptology - AFRICACRYPT 2008*, LNCS 5023, S. Vaudenay (ed.), Berlin, Germany: Springer-Verlag, pp. 16-26, 2008.

[26] C. De Cannière, O. Dunkelman, and M. Knežević, “KATAN and KTANTAN – A Family of Small and Efficient Hardware-Oriented Block Ciphers”, *The 11th International Workshop on Cryptographic Hardware and Embedded Systems - CHES 2009*, LNCS 5747, C. Clavier, K. Gaj (eds.), Berlin, Germany: Springer-Verlag, pp. 272-288, 2009.

[27] C. De Cannière and B. Preneel, “Trivium – A Stream Cipher Construction Inspired by Block Cipher Design Principles”, ECRYPT Stream Cipher, Available at `http://www.ecrypt.eu.org/stream/papersdir/2006/021.pdf`, 2005.

[28] D. Cantor, “Computing in Jacobian of a Hyperelliptic Curve”, *Mathematics of Computation*, vol. 48 (177), pp. 95-101, January 1987.

[29] X. Cao, W. Kou, L. Dang, and B. Zhao, “IMBAS: Identity-Based Multi-User Broadcast Authentication in Wireless Sensor Networks”, *Computer Communications*, vol. 31, no. 4, pp. 659-667, 2008.

[30] S. Capkun, L. Buttyán, and J.-P. Hubaux, “Self-Organized Public-Key Management for Mobile Ad Hoc Networks”, *IEEE Transactions on Mobile Computing*, Vol. 2, No. 1, pp. 52-64, 2003.

[31] S. Čapkun, and J.-P. Hubaux, “Secure Positioning in Wireless Networks”, *IEEE Journal on Selected Areas in Communications*, 24(2): 221-232, 2006.

[32] S. Chang, S. Shieh, W. Lin, and C.-M. Hsieh, ”An Efficient Broadcast Authentication Scheme”, *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security (ASIACCS 2006)*, pp. 311-320, 2006.

[33] S. Chatterjee, P. Sarkar, and R. Barua, “Efficient Computation of Tate Pairing in Projective Coordinate over General Characteristic Fields”, *Information Security and Cryptology - ICISC 2004*, ser. LNCS 3506, C. Park, S. Chee (eds.), Berlin, Germany: Springer-Verlag, pp. 168-181, 2005.

[34] P. Chodowiec and K. Gaj, “Very Compact FPGA Implementation of the AES Algorithm”, *The 5th International Workshop on Cryptographic Hardware and Embedded Systems - CHES 2003*, LNCS 2779, C. D. Walter, Ç. K. Koç, C. Paar (eds.), Berlin, Germany: Springer-Verlag, pp. 319-333, 2003.

[35] Y. Choie and E. Lee, “Implementation of Tate Pairing on Hyperelliptic Curve of Genus 2”, *Information Security and Cryptology - ICISC’2003*, LNCS 2971, J. I. Lim and D. H. Lee (eds.), Berlin, Germany: Springer-Verlag, pp. 97-111, 2004.

[36] Y. Choie, E. Jeong, and E. Lee, “Supersingular Hyperelliptic Curves of Genus 2 over Finite Fields”, *Journal of Applied Mathematics and Computation*, 163(2):565-576, 2005.

[37] J. Clulow and T. Moore, “Suicide for the Common Good: A New Strategy for Credential Revocation in Self-Organizing Systems”, *SIGOPS Operating System Reviews*, vol. 40, no. 3, pp. 18-21, 2006.

[38] C. Cocks and R. G. E. Pinch, “Identity-based Cryptosystems Based on the Weil Pairing”, Unpublished manuscript, 2001.

[39] N. Courtois, M. Finiasz, and N. Sendrier, “How to Achieve a McEliece-Based Digital Signature Scheme”, *Advances in Cryptology-ASIACRYPT’2001*, LNCS 2248, C. Boyd (ed.), Berlin, Germany: Springer-Verlag, pp. 157-174, 2001.

[40] C. Crépeau and C. R. Davis, “A Certificate Revocation Scheme for Wireless Ad Hoc Networks”, *Proceedings of the 1st ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN’03)*, pp. 54-61, 2003.

[41] Crossbow Technology Inc. “MICAz – Wireless Measurement System”. Available at `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAz_Datasheet.pdf`.

[42] Crossbow Technology Inc. “TELOSB – TELOSB Mote Platform”. Available at http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/TelosB_Datasheet.pdf.

[43] Crossbow Technology Inc. “Imote2 – High-Performance Wireless Sensor Network Node”. Available at http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/Imote2_Datasheet.pdf.

[44] M. L. Das, “Efficient User Authentication and Secure Data Transmission in Wireless Sensor Networks”, *Proceedings of the 16th IEEE International Conference on Networks (ICON’08)*, 2008.

[45] M. L. Das, “Two-Factor User Authentication in Wireless Sensor Networks”, *IEEE Transactions on Wireless Communications*, vol. 8, no. 3, pp. 1086-1090, March 2009.

[46] H. Deng, A. Mukherjee, and D. Agrawal, “Threshold and Identity-based Key Management and Authentication for Wireless Ad Hoc Networks,” *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC’04)*, pp. 107-111, 2004.

[47] I. Dinur and A. Shamir, “Cube Attacks on Tweakable Black Box Polynomials”, *Advances in Cryptology - EUROCRYPT 2009*, LNCS 5479, A. Joux (ed.), Berlin, Germany: Springer-Verlag, pp. 278-299, 2009.

[48] Q. Dong, D. Liu, and P. Ning, “Pre-Authentication Filters: Providing DoS Resistance for Signature-Based Broadcast Authentication in Wireless Sensor Networks”, *Proceedings of the First ACM Conference on Wireless Network Security (WiSec’08)*, pp. 2-12, 2008.

[49] J. Douceur, “The Sybil Attack”, *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS’02)*, pp. 251-260, 2002.

[50] J. Drissi and Q. Gu, “Localized Broadcast Authentication in Large Sensor Networks”, *Proceedings of the International conference on Networking and Services (ICNS’06)*, pp. 25-31, 2006.

[51] B. Driessen, A. Poschmann, and C. Paar, “Comparison of Innovative Signature Algorithms for WSNs”, *Proceedings of the First ACM Conference on Wireless Network Security (WiSec’08)*, pp. 30-35, 2008.

[52] I. Duursma, P. Gaudry, and F. Morain, “Speeding up the Discrete Log Computation on Curves with Automorphisms”, *Advances in Cryptology — ASIACRYPT’1999*, LNCS 1716, K. Y. Lam, E. Okamoto, C. Xing (eds.), Berlin, Germany: Springer-Verlag, pp. 103-121, 1999.

[53] I. Duursma and H. S. Lee, "Tate Pairing Implementation for Hyperelliptic Curves $y^2 = x^p - x + d$," *Advances in Cryptology - ASIACRYPT'2003*, LNCS 2894, C. S. Laih (ed.), Berlin, Germany: Springer-Verlag, pp. 111-123, 2003.

[54] ECRYPT Network of Excellence. The Stream Cipher Project: eSTREAM. Available at `http://www.ecrypt.eu.org/stream/`.

[55] T. Eisenbarth, S. Kumar, C. Paar, A. Poschmann, and L. Uhsadel, "A Survey of Lightweight-Cryptography Implementations", *IEEE Design & Test of Computers*, vol. 24, no. 6, pp. 522-533, 2007.

[56] D. Engels, X. Fan, G. Gong, H. Hu, and E. M. Smith, "Ultra-Lightweight Cryptography for Low-Cost RFID Tags: Hummingbird Algorithm and Protocol", *Centre for Applied Cryptographic Research (CACR) Technical Reports*, CACR 2009-29, available at `http://www.cacr.math.uwaterloo.ca/techreports/2009/cacr2009-29.pdf`.

[57] D. Engels, X. Fan, G. Gong, H. Hu, and E. M. Smith, "Hummingbird: Ultra-Lightweight Cryptography for Resource- Constrained Devices", to appear in the Proceedings of *The 14th International Conference on Financial Cryptography and Data Security - FC 2010*, Berlin, Germany: Springer-Verlag, 2010.

[58] N. N. Espresso. Available at `http://embedded.eecs.berkeley.edu/pubs/downloads/espresso/index.htm`, November 1994.

[59] X. Fan and G. Gong, "Key Revocation Based on Dirichlet Multinomial Model for Mobile Ad Hoc Networks", *The Fourth IEEE LCN Workshop on Network Security (WNS 2008)*, pp. 958-965, 2008.

[60] X. Fan, H. Hu, G. Gong, E. M. Smith and D. Engels, "Lightweight Implementation of Hummingbird Cryptographic Algorithm on 4-Bit Microcontrollers", *The 1st International Workshop on RFID Security and Cryptography 2009 (RISC'09)*, pp. 838-844, 2009.

[61] X. Fan, G. Gong, and D. Jao, "Speeding Up Pairing Computations on Genus 2 Hyperelliptic Curves with Efficiently Computable Automorphisms", *Pairing-Based Cryptography - Pairing 2008*, LNCS 5209, S.D. Galbraith, K.G. Paterson (eds.), Berlin, Germany: Springer-Verlag, pp. 243-264, 2008.

[62] X. Fan, G. Gong, and D. Jao, "Efficient Pairing Computation on Genus 2 Curves in Projective Coordinates", *Selected Areas in Cryptography - SAC 2008*, LNCS 5381, R. Avanzi, L. Keliher, F. Sica (eds.), Berlin, Germany: Springer-Verlag, pp. 18-34, 2009.

[63] M. Feldhofer, S. Dominikus, and J. Wolkerstorfer, "Strong Authentication for RFID Systems Using the AES Algorithm", *The 6th International Workshop on Cryptographic Hardware and Embedded Systems-CHES 2004*, LNCS 3156, M. Joye, J.-J. Quisquater (eds.), Berlin, Germany: Springer-Verlag, pp. 357-370, 2004.

[64] M. Feldhofer, J. Wolkerstorfer, and V. Rijmen, "AES Implementation on a Grain of Sand", *IEE Proceedings Information Security*, vol. 15, no. 1, pp. 13-20, 2005.

[65] D. Freeman, "Constructing Pairing-Friendly Genus 2 Curves over Prime Fields with Ordinary Jacobians", *Pairing-Based Cryptography - Pairing 2007*, LNCS 4575, T. Takagi, T. Okamoto, E. Okamoto, T. Okamoto (eds.), Berlin, Germany: Springer-Verlag, pp. 152-176, 2007.

[66] G. Frey and T. Lange, "Fast Bilinear Maps from The Tate-Lichtenbaum Pairing on Hyperelliptic Curves", *Algorithmic Number Theory Symposium - ANTS VII*, LNCS 4076, F. Hess, S. Pauli, M. Pohst (eds.), Berlin, Germany: Springer-Verlag, pp. 466-479, 2006.

[67] G. Frey and H.-G. Rück, "A Remark Concerning m-Divisibility and the Discrete Logarithm Problem in the Divisor Class Group of Curves", *Mathematics of Computation*, 62(206):865-874, 1994.

[68] E. Furukawa, M. Kawazoe, and T. Takahashi, "Counting Points for Hyperelliptic Curves of Type $y^2 = x^5 + ax$ over Finite Prime Fields", *Selected Areas in Cryptography - SAC 2003*, LNCS 3006, M. Matsui, R. Zuccherato (eds.), Berlin, Germany: Springer-Verlag, pp. 26-41, 2004.

[69] S. D. Galbraith, J. F. McKee, and P. C. Valença, "Ordinary Abelian Varieties Having Small Embedding Degree", *Finite Fields and Their Applications*, vol. 13, iss. 4, pp. 800-814, 2007.

[70] P. Gaudry, "An Algorithm for Solving the Discrete Log Problem on Hyperelliptic Curves", *Advances in Cryptology - EUROCRYPT'2000*, LNCS 1807, B. Preneel (ed.), Berlin, Germany: Springer-Verlag, pp. 19-34, 2000.

[71] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin, *Bayesian Data Analysis, Second Edition*, Boca Raton, Florida, USA: Chapman & Hall/CRC, 2004.

[72] T. Good and M. Benaissa, "AES on FPGA from the Fastest to the Smallest", *The 7th International Workshop on Cryptographic Hardware and Embedded Systems - CHES 2005*, LNCS 3659, J. R. Rao, B. Sunar (eds.), Berlin, Germany: Springer-Verlag, pp. 427-440, 2005.

[73] T. Good and M. Benaissa, "ASIC Hardware Performance", *New Stream Cipher Design - the eStream Finalists*, LNCS 4986, M. Robshaw and O. Bilet (eds.), Berlin, Germany: Springer-Verlag, pp. 267-293, 2008.

[74] X. Guo, Z. Chen, and P. Schaumont, "Energy and Performance Evaluation of an FPGA-Based SoC Platform with AES and PRESENT Coprocessors", *Embedded Computer Systems: Architectures, Modeling, and Simulation - SAMOS'2008*, LNCS 5114, M. Berekovic, N. Dimopoulos, and S. Wong (eds.), Berlin, Germany: Springer-Verlag, pp. 106-115, 2008.

[75] R. Granger, F. Hess, R. Oyono, N. Thériault, and F. Vercauteren, "Ate Pairing on Hyperelliptic Curves", *Advance in Cryptology - EUROCRYPT'2007*, LNCS 4515, M. Naor (ed.), Berlin, Germany: Springer-Verlag, pp. 430-447, 2007.

[76] N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz, "Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs", *The 6th International Workshop on Cryptographic Hardware and Embedded Systems-CHES 2004*, LNCS 3156, M. Joye and J.-J. Quisquater (eds.), Berlin, Germany: Springer-Verlag, pp. 119-132, 2004.

[77] M. Haneda, M. Kawazoe, and T. Takahashi, "Suitable Curves for Genus-4 HEC over Prime Fields: Point Counting Formulae for Hyperelliptic Curves of Type $y^2 = x^{2k+1} + ax$", *The 32nd International Colloquium on Automata, Languages and Programming - ICALP 2005*, LNCS 3580, L. Caires, G. F. Italiano, L. Monteiro, C. Palamidessi, M. Yung (eds.), Berlin, Germany: Springer-Verlag, pp. 539-550, 2005.

[78] P. Hämäläinen, T. Alho, M. Hännikäinen, and T. D. Hämäläinen, "Design and Implementation of Low-Area and Low-Power AES Encryption Hardware Core", *The 9th EUROMICRO Conference on Digital System Design: Architectures, Methods and Tools - DSD 2006*, pp. 577-583, IEEE Computer Society, 2006.

[79] D. Hankerson, A. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography*, New York, USA: Springer-Verlag, 2004.

[80] C. Ó. hÉigeartaigh, and M. Scott, "Pairing Calculation on Supersingular Genus 2 Curves", *Selected Areas in Cryptography - SAC 2006*, LNCS 4356, E. Biham, A. M. Youssef (eds.), Berlin, Germany: Springer-Verlag, pp. 302-316, 2007.

[81] F. Hess, N. P. Smart, and F. Vercauteren, "The Eta Pairing Revisited", *IEEE Transactions on Information Theory*, 52(10):4595-4602, 2006.

[82] M. Hell, T. Johansson, and W. Meier, "Grain: A Stream Cipher for Constrained Environments", *International Journal of Wireless and Mobile Computing*, vol. 2, no. 1, pp. 86-93, 2007.

[83] L. Hitt, "Families of Genus 2 Curves with Small Embedding Degree", Cryptology ePrint Archive, Report 2007/001, 2007, `http://eprint.iacr.org/2007/001`.

[84] K. Hoeper, "Authentication and Key Exchange in Mobile Ad Hoc Networks", Ph.D. thesis, Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Canada, 2007.

[85] K. Hoeper and G. Gong, "Identity-Based Key Exchange Protocols for Ad Hoc Networks", *Proceedings of the Canadian Workshop on Information Theory (CWIT'05)*, pp. 127-130, 2005.

[86] K. Hoeper and G. Gong, "Key Revocation for Identity-Based Schemes in Mobile Ad Hoc Networks", *Proceedings of the 5th International Conference on Ad-Hoc, Mobile, and Wireless Networks (ADHOC-NOW 2006)*, LNCS 4104, Thomas Kunz, S. S. Ravi (eds.), Berlin, Germany: Springer-Verlag, pp. 224-237, 2006.

[87] D. Hong, J. Sung, S. Hong, J. Lim, S. Lee, B. S. Koo, C. Lee, D. Chang, J. Lee, K. Jeong, H. Kim, and S. Chee, "HIGHT: A New Block Cipher Suitable for Low-Resource Device", *The 8th International Workshop on Cryptographic Hardware and Embedded Systems-CHES 2006*, LNCS 4249, L. Goubin and M. Matsui (eds.), Berlin, Germany: Springer-Verlag, pp. 46-59, 2006.

[88] R. Housley, W. Polk, W. Ford, and D. Solo, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, RFC 3280, 2002.

[89] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Rushing Attacks and Defense in Wireless Ad Hoc Network Routing Protocols", *Proceeding of the 2003 ACM Workshop on Wireless Security (WiSe 2003)*, pp. 30-40, 2003.

[90] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Wormhole Attacks in Wireless Networks", *IEEE Journal on Selected Areas in Communications*, 24(2): 318-328, 2006.

[91] Intel Corporation. "Intel PXA27x Processor Family". Available at `http://int.xscale-freak.com/XSDoc/PXA27X/28000304.pdf`.

[92] T. Jakobsen and L. Knudsen, "The Interpolation Attack on Block Ciphers", *The 4th Annual Fast Software Encryption Workshop - FSE 1997*, LNCS 1267, E. Biham (ed.), Berlin, Germany: Springer-Verlag, pp. 28-40, 1997.

[93] D. B. Johnson and D. A. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks", *Mobile Computing*, vol. 353, Chapter 5, pp. 153-181, Kluwer Academic Publishers, 1996.

[94] A. Jøsang, "Probabilistic Logic Under Uncertainty", *Proceedings of the thirteenth Australasian symposium on Theory of computing - Volume 65*, pp. 101-110, 2007.

[95] A. Jøsang and J. Haller, "Dirichlet Reputation Systems", *Proceedings of the 2nd International Conference on Availability, Reliability and Security (ARES 2007)*, pp. 112-119, 2007.

[96] A. Jøsang and R. Ismail, "The Beta Reputation Systems", *Proceedings of the 15th Bled Electronic Commerce Conference - eReality: Constructing the eEconomy*, pp. 324-337, 2002.

[97] A. Jøsang, R. Ismail, and C. Boyd, "A Survey of Trust and Reputation Systems for Online Service Provision", *Decision Support Systems*, vol. 43, no. 2, pp. 618-644, 2007.

[98] J.-P. Kaps, "Chai-Tea, Cryptographic Hardware Implemenations of xTEA", *The 9th International Conference on Cryptology in India - INDOCRYPT 2008*, LNCS 5356, D.R. Chowdhury, V. Rijmen, and A. Das (eds.), Berlin, Germany: Springer-Verlag, pp. 363-375, 2008.

[99] M. Kawazoe and T. Takahashi, "Pairing-friendly Hyperelliptic Curves with Ordinary Jacobians of Type $y^2 = x^5 + ax$", *Pairing-Based Cryptography - Pairing 2008*, LNCS 5209, S.D. Galbraith, K.G. Paterson (eds.), Berlin, Germany: Springer-Verlag, pp. 164-177, 2008.

[100] P. C. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems", *Advances in Cryptology - CRYPTO'96*, LNCS 1109, N. Koblitz (ed.), Berlin, Germany: Springer-Verlag, pp. 104-113, 1996.

[101] P. C. Kocher, "On Certificate Revocation and Validation", *Proceedings of the 2nd International Conference on Financial Cryptography - FC'98)*, LNCS 1465, R. Hirschfeld (ed.), Berlin, Germany: Springer-Verlag, pp. 172-177, 1998.

[102] J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang, "Providing Robust and Ubiquitous Security Support for Mobile Ad Hoc Networks", *Proceedings of the 9th International Conference on Network Protocols (ICNP'01)*, pp. 251-260, 2001.

[103] S. Kozaki, K. Matsuo, and Y. Shimbara, "Skew-Frobenius Maps on Hyperelliptic Curves", *The 2007 Symposium on Cryptography and Information Security - SCIS 2007*, IEICE Japan, 1D2-4, January 2007.

[104] X. Lai, "Higher Order Derivatives and Differential Cryptanalysis", *Proceedings of Symposium on Communication, Coding and Cryptography*, in honor of James L. Massey on the occasion of his 60'th birthday, 1994.

[105] T. Lange, “Formulae for Arithmetic on Genus 2 Hyperelliptic Curves”, *Applicable Algebra in Engineering, Communication and Computing*, vol.15, No.5, pp. 295-328, 2005.

[106] G. Leander, C. Paar, A. Poschmann, and K. Schramm, “New Lightweight DES Variants”, *The 14th Annual Fast Software Encryption Workshop-FSE 2007*, LNCS 4593, A. Biryukov (ed.), Berlin, Germany: Springer-Verlag, pp. 196-210, 2007.

[107] G. Leander and A. Poschmann, “On the Classification of 4 Bit S-Boxes”, *The 1st International Workshop on the Arithmetic of Finite Fields - WAIFI 2007*, LNCS 4547, C. Carlet and B. Sunar (eds.), Berlin, Germany: Springer-Verlag, pp. 159-176, 2007.

[108] E. Lee, H.-S. Lee, and Y. Lee, “Eta Pairing Computation on General Divisors over Hyperelliptic Curves $y^2 = x^7 - x \pm 1$”, *Pairing-Based Cryptography - Pairing 2007*, LNCS 4575, T. Takagi, T. Okamoto, E. Okamoto, and T. Okamoto (eds.), Berlin, Germany: Springer-Verlag, pp. 349-366, 2007.

[109] C. Lim and T. Korkishko, “mCrypton - A Lightweight Block Cipher for Security of Low-cost RFID Tags and Sensors”, *Workshop on Information Security Applications-WISA 2005*, LNCS 3786, J. Song, T. Kwon, and M. Yung (eds.), Berlin, Germany: Springer-Verlag, pp. 243-258, 2005.

[110] D. Liu and P. Ning, “Multi-Level μTESLA: Broadcast Authentication for Distributed Sensor Networks”, *ACM Transactions on Embedded Computing Systems*, vol. 3, no. 4, pp. 800-836, 2004.

[111] A. Liu and P. Ning, ”TinyECC: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks,” *Proceedings of the 7th International Conference on Information Processing in Sensor Networks (IPSN 2008)*, SPOTS Track, pp. 245-256, 2008.

[112] D. Liu, P. Ning, S. Zhu, and S. Jajodia, “Practical Broadcast Authentication in Sensor Networks”, *Proceedings of the Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous 2005)*, pp. 118-129, 2005.

[113] D. Liu, Y. Yang, J. Wang, and H. Min, “A Mutual Authentication Protocol for RFID Using IDEA”, *Auto-ID Labs White Paper*, WP-HARDWARE-048, March 2009, available at `http://www.autoidlabs.org/uploads/media/AUTOIDLABS-WP-HARDWARE-048.pdf`.

[114] J. López, D. Aranha, D. Câmara, R. Dahab, L. Oliveira, and C. Lopes, "Fast Implementation of Elliptic Curve Cryptography and Pairing Computation for Sensor Networks", *The 13th Workshop on Elliptic Curve Cryptography (ECC 2009)*, Available at `http://ecc.math.ucalgary.ca/sites/ecc.math.ucalgary.ca/files/u5/Lopez_ECC2009.pdf`.

[115] M. Luk, A. Perrig, and B. Whillock, "Seven Cardinal Properties of Sensor Network Broadcast Authentication", *Proceedings of the Fourth ACM workshop on Security of Ad Hoc and Sensor Networks (SASN'06)*, pp. 147-156, 2006.

[116] J. Luo, J.-P. Hubaux, and P.T. Eugster, "DICTATE: Distributed CerTification Authority with probabilisTic frEshness for Ad Hoc Networks", *IEEE Transactions on Dependable and Secure Computing*, Vol. 2, No. 4, pp. 311-323, 2005.

[117] H. Luo, J. Kong, P. Zerfos, S. Lu, and L. Zhang, "URSA: Ubiquitous and Roubust Access Control for Mobile Ad Hoc Networks", *IEEE/ACM Transactions on Networking*, vol.12, no.6, pp. 1049-1063, 2004.

[118] F. Mace, F.-X. Standaert, and J.-J. Quisquater, "ASIC Implemenations of the Block Cipher SEA for Constrained Applications", *RFID Security - RFIDsec 2007*, Workshop Record, pp. 103-114, 2007.

[119] F. Mace, F.-X. Standaert, and J.-J. Quisquater, "FPGA Implemenation(s) of a Scalable Encryption Algorithm", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 16, no. 2, pp. 212-216, 2008.

[120] A. Malpani, S. Galperin, M. Myers, R. Ankney, and C. Adams, *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP*, RFC 2560, 1999.

[121] D. J. Malan, M. Welsh, and M. D. Smith, "Implementing Public-Key Infrastructure for Sensor Networks", *ACM Transactions on Sensor Networks*, vol. 4, no. 4, pp. 22:1-22:23, 2008.

[122] A. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, Boca Raton, Florida, USA: Chapman & Hall/CRC, 1997.

[123] S. Micali, "Efficient Certificate Revocation", Technical Memo MIT/LCS/TM-542b, Massachusetts Institute of Technology, Laboratory for Computer Science, March 1996.

[124] P. Michiardi and R. Molva, "CORE: A Collaborative Reputation Mechanism to Enforce Node Cooperation in Mobile Ad Hoc Networks", *Proceedings of the IFIP TC6/TC11 Sixth Joint Working Conference on Communication and Multimedia Security*, pp. 107-121, 2002.

[125] V. S. Miller, “Short Programs for Functions on Curves”, Unpublished manuscript, 1986, available at `http://crypto.stanford.edu/miller/miller.pdf`.

[126] V. S. Miller, “The Weil Pairing and Its Efficient Calculation”, *Journal of Cryptology*, vol. 17, no. 4, pp. 235-261, 2004.

[127] A. Mishra, K. Nadkarni, and A. Patcha, “Intrusion Detection in Wireless Ad Hoc Networks”, *IEEE Wireless Communication*, vol. 11, no. 1, pp. 48-60, Feb. 2004.

[128] Y. Miyamoto, H. Doi, K. Matsuo, J. Chao and S. Tsujii, “A Fast Addition Algorithm of Genus Two Hyperelliptic Curve”, *The 2002 Symposium on Cryptography and Information Security - SCIS 2002*, pp. 497-502, 2002, in Japanese.

[129] T. Moore, J. Clulow, R. Anderson, and S. Nagaraja, “New Strategies for Revocation in Ad Hoc Networks”, *Proceedings of the Fourth European Workshop on Security and Privacy in Ad Hoc and Sensor Networks (ESAS 2007)*, LNCS 4572, F. Stajano, C. Meadows, S. Capkun, T. Moore (eds.), Berlin, Germany: Springer-Verlag, pp. 232-246, 2007.

[130] D. Mumford, “Tata Lectures on Theta II”, *Prog. Math.*, vol. 43. Birkhäuser, 1984.

[131] J. Mundinger, and J.-Y. Le Boudec, “Analysis of A Robust Reputation System for Self-Organized Networks,” *European Transactions on Telecommunications, Special Issue on Self-Organisation in Mobile Networking*, vol.16, no.5, pp. 375-384, 2005.

[132] J. Mundinger, and J.-Y. Le Boudec, “Analysis of A Reputation System for Mobile Ad-hoc Networks with Liars,” *Proceedings of WiOpt 2005: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, pp. 41-46, 2005.

[133] J. Mundinger, and J.-Y. Le Boudec, “The Impact of Liars on Reputation in Social Networks,” *Proceedings of Social Network Analysis: Advances and Empirical Applications Forum*, Oxford, UK, July 2005.

[134] P. Ning, A. Liu, and W. Du, “Mitigating DoS Attacks against Broadcast Authentication in Wireless Sensor Networks”, *ACM Transactions on Sensor Networks*, vol. 4, no. 1, pp. 1-35, January 2008.

[135] National Institute of Standards and Technology (NIST). Wireless Ad Hoc Network Projects. Available at `http://www.antd.nist.gov/wahn_home.shtml`.

[136] M. Naor and K. Nissim, “Certificate Revocation and Certificate Update”, *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 4, pp. 561-570, 2000.

[137] L. B. Oliveira, D. F. Aranha, E. Morais, F. Daguano, J. López, and R. Dahab, "TinyTate: Computing the Tate Pairing in Resource-Constrained Sensor Nodes", *Proceedings of the Sixth IEEE International Symposium on Network Computing and Applications (NCA 2007)*, pp. 318-323, 2007.

[138] L. B. Oliveira, M. Scott, J. López, and R. Dahab, "TinyPBC: Pairings for Authenticated Identity-Based Non-Interactive Key Distribution in Sensor Networks", *Proceedings of the 5th International Conference on Networked Sensing Systems (INSS 2008)*, pp. 173-180, 2008.

[139] Y-H. Park, S. Jeong, and J. Lim, "Speeding Up Point Multiplication on Hyperelliptic Curves with Efficiently-Computable Endomorphisms", *Advance in Cryptology - EUROCRYPT'2002*, LNCS 2332, L.R. Knudsen (ed.), Berlin, Germany: Springer-Verlag, pp. 197-208, 2002.

[140] C. E. Perkins, *Ad Hoc Networking*, Addison-Wesley, Boston, MA, 2001.

[141] C. E. Perkins, E. M. Royer, and S. R. Das, "Ad Hoc On Demand Vector (AODV) Routing", *IETF Internet Draft*, Internet Draft (draft-ietf-manet-aodv-09.txt), November 2001, Work in Progress.

[142] A. Perrig, "The BiBa One-time Signature and Broadcast Authentication", *Proceedings of the ACM Conference on Computer and Communications Security*, pp. 28-37, 2001.

[143] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler, "SPINS: Security Protocols for Sensor Networks", *ACM Wireless Networks*, vol. 8, no. 5, pp. 521-534, 2002.

[144] A. Poschmann, "Lightweight Cryptography - Cryptographic Engineering for a Pervasive World", Ph.D. Thesis, Department of Electrical Engineering and Information Sciences, Ruhr-Universitäet Bochum, Bochum, Germany, 2009.

[145] K. Ren, W. Lou, K. Zeng, and P. J. Moran, "On Broadcast Authentication in Wireless Sensor Networks", *IEEE Transactions on Wireless Communications*, vol. 6, no. 11, pp. 4136-4144, 2007.

[146] K. Ren, S. Yu, W. Lou, and Y. Zhang, "Multi-user Broadcast Authentication in Wireless Sensor Networks", to appear in *IEEE Transactions on Vehicular Technology*, 2009.

[147] C. Rolfes, A. Poschmann, G. Leander, and C. Paar, "Ultra-Lightweight Implementations for Smart Devices-Security for 1000 Gate Equivalents", *The 8th Smart Card*

Research and Advanced Application IFIP Conference - CARDIS 2008, LNCS 5189, G. Grimaud and F.-X. Standaert (eds.), Berlin, Germany: Springer-Verlag, pp. 89-103, 2008.

[148] G. Rouvroy, F.-X. Standaert, J.-J. Quisquater, and J.-D. Legat, "Compact and Efficient Encryption/Decryption Module for FPGA Implementation of the AES Rijndael VeryWell Suited for Small Embedded Applications", *International Conference on Information Technology: Coding and Computing - ITCC 2004*, pp. 583-587, 2004.

[149] Rowley Associates. CrossWorks for MSP430. Available at `http://www.rowley.co.uk/msp430/index.htm`.

[150] K. Rubin and A. Silverberg, "Supersingular Abelian Varieties in Cryptography", *Advance in Cryptology - CRYPTO'2002*, LNCS 2442, M. Yung (ed.), Berlin, Germany: Springer-Verlag, pp. 336-353, 2002.

[151] N. Saxena, G. Tsudik, and J.H. Yi, "Identity-Based Access Control for Ad Hoc Groups", *The 7th International Conference on Information Security and Cryptology - ICISC 2004)*, LNCS 3506, C. Park, S. Chee (eds.), Berlin, Germany: Springer-Verlag, pp. 362-379, 2004.

[152] M. Scott, "Faster Pairings Using an Elliptic Curve with an Efficient Endomorphism", *Progress in Cryptology - INDOCRYPT'2005*, LNCS 3797, S. Maitra, C.E. Veni Madhavan and R. Venkatesan (eds.), Berlin, Germany: Springer-Verlag, pp. 258-269, 2005.

[153] M. Scott and P.L.S.M. Barreto, "Compressed Pairings", *Advance in Cryptology - CRYPTO'2004*, LNCS 3152, M. Franklin (ed.), Berlin, Germany: Springer-Verlag, pp. 140-156, 2004.

[154] S. C. Seo, D.-G. Han, and S. Song, "TinyECCK: Efficient Elliptic Curve Cryptography Implemenation over $GF(2^m)$ on 8-bit Micaz Mote", *IEICE Transactions on Information and Systems*, E91-D(5):1338-1347, 2008.

[155] G. Shafer, *A Mathematical Theory of Evidence*, Princeton University, 1976.

[156] A. Shamir, "How to Share a Secret", *Communications of the ACM*, vol. 22, no. 11, pp. 612-613, 1979.

[157] A. Shamir, "Identity Based Cryptosystems and Signature Schemes", *Proceedings of Advances in Cryptology - CRYPTO 1984*, LNCS 196, G. R. Blakley, D. Chaum (eds.), Berlin, Germany: Springer-Verlag, pp. 47-53, 1984.

[158] M. Shirase, Y. Miyazaki, T. Takagi, D.-G. Han, and D. Choi, "Efficient Implementation of Pairing Based Cryptography on a Sensor Node", *IEICE Transactions on Information and Systems*, E92-D(5):909-917, 2009.

[159] P. Szczechowiak, A. Kargl, M. Scott, and M. Collier, "On the Application of Pairing Based Cryptography to Wireless Sensor Networks", *Proceedings of the Second ACM Conference on Wireless Network Security (WiSec'09)*, pp. 1-12, 2009.

[160] J. H. Silverman, *The Arithmetic of Elliptic Curves*, Graduate Texts in Mathematics 106. Springer-Verlag, 1986.

[161] J. Solinas, "Generalized Mersenne Primes", *Centre for Applied Cryptographic Research (CACR) Technical Reports*, CORR 99-39, available at `http://www.cacr.math.uwaterloo.ca/techreprots/1999/corr99-39.pdf`.

[162] F. Stajano and R. Anderson, "The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks", *Proceedings of 7th International Workshop on Security Protocols*, ser. LNCS 1796, B. Christianson, B. Crispo, and M. Roe (eds.), Berlin, Germany: Springer-Verlag, pp. 172-194, 1999.

[163] F.-X. Standaert, G. Piret, N. Gershenfeld, and J.-J. Quisquater, "SEA: A Scalable Encryption Algorithm for Small Embedded Applications", *The 7th IFIP WG 8.8/11.2 International Conference on Smart Card Research and Advanced Applications-CARDIS 2006*, LNCS 3928, J. Domingo-Ferrer, J. Posegga, and D. Schreckling (eds.), Berlin, Germany: Springer-Verlag, pp. 222-236, 2006.

[164] F.-X. Standaert, G. Piret, G. Rouvroy, and J.-J. Quisquater, "FPGA Implementations of the ICEBERG Block Cipher", *Integration, the VLSI Journal*, vol. 40, iss. 1, pp. 20-27, 2007.

[165] K. Takashima, "Scaling Security of Elliptic Curves with Fast Pairing Using Efficient Endomorphism", *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Science*, vol. E90-A NO.1, pp. 152-159, January 2007.

[166] Texas Instrument Inc. "MSP430 16-bit Ultra-Low Power MCUs". Available at `http://focus.ti.com/mcu/docs/mcuprodoverview.tsp?sectionId=95&tabId=140&familyId=342`.

[167] Texas Instrument Inc. "2.4 GHz IEEE 802.15.4/ZigBee-ready RF Transceiver". Available at `http://focus.ti.com/lit/ds/symlink/cc2420.pdf`.

[168] M. Vogt, A. Poschmann, and C. Paar, "Cryptography is Feasible on 4-Bit Microcontrollers - A Proof of Concept", *2009 IEEE International Conference on RFID*, pp. 241-248, 2009.

[169] D. Wagner, "Cryptanalysis of Some Recently-Proposed Multiple Modes of Operation", *The 5th Annual Fast Software Encryption Workshop - FSE 1998*, LNCS 1372, S. Vaudenay (ed.), Berlin, Germany: Springer-Verlag, pp. 254-269, 1998.

[170] H. Wang and Q. Li, "Efficient Implementation of Public Key Cryptosystems on MICAz Motes", *The 8th International Conference on Information and Communications Security-ICICS 2006*, LNCS 4307, P. Ning, S. Qing, and N. Li (eds.), Berlin, Germany: Springer-Verlag, pp. 519-528, 2006.

[171] WinAVR. Suite of Executable, Open Source Software Development Tools for the Atmel AVR Series of RISC Microprocessors Hosted on the Windows Platform. Available at `http://winavr.sourceforge.net/`.

[172] A. Withby, A. Jøsang, and J. Indulska, "Filtering Out Unfair Ratings in Bayesian Reputation Systems", *The Icfain Journal of Management Research*, vol. 4, no. 2, pp. 48-64, 2005.

[173] T. Wollinger, J. Pelzl, V. Wittelsberger, C. Paar, G. Saldamli, and Ç. K. Koç, "Elliptic & Hyperelliptic Curves on Embedded μP," *ACM Transactions in Embedded Computing Systems (TECS)*, vol. 3, no. 3, pp. 509-533, 2004.

[174] Xilinx Inc., "Spartan-3 FPGA Family Data Sheet", DS099, December 4, 2009, available at `http://www.xilinx.com/support/documentation/data_sheets/ds099.pdf`.

[175] X. Xiong, D. C. Wong, and X. Deng, "TinyPairing: Computing Tate Pairing on Sensor Nodes with Higher Speed and Less Memory", *Proceedings of the Eighth IEEE International Symposium on Network Computing and Applications (NCA'09)*, pp. 187-194, 2009.

[176] S. Yamakawa, Y. Cui, K. Kobara, and H. Imai, "Lightweight Broadcast Authentication Protocols Reconsidered", *Proceedings of the IEEE Wireless Communications & Networking Conference (WCNC 2009)*, 2009.

[177] A. Youssef and G. Gong, "On the Interpolation Attacks on Block Ciphers", *The 7th Annual Fast Software Encryption Workshop - FSE 2000*, LNCS 1978, B. Schneier (ed.), pp. 109-120, Berlin, Germany: Springer-Verlag, 2001.

[178] Y. Zhang, W. Liu, W. Lou, and Y. Fang, "Location-Based Compromise-Tolerant Security Mechanisms for Wireless Sensor Networks", *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 2, pp. 247-260, 2006.

[179] Y. Zhang, W. Liu, W. Lou, and Y. Fang, “Securing Mobile Ad Hoc Networks with Certificateless Public Keys”, *IEEE Transactions on Dependable and Secure Computing*, vol. 3, no. 4, pp. 386-399, 2006.

[180] Q. Zhang, T. Yu, and P. Ning, “A Framework for Identifying Compromised Nodes in Wireless Sensor Networks”, *ACM Transactions in Information and Systems Security (TISSEC)*, vol. 11, no. 3, pp. 1-37, 2008.

[181] C. Zhao, F. Zhang, and J. Huang, “Speeding Up the Bilinear Pairings Computation on Curves with Automorphisms”, Cryptology ePrint Archive, Report 2006/474, 2006, `http://eprint.iacr.org/2006/474`.

[182] L. Zhou and Z.J. Hass, “Securing Ad Hoc Networks”, *IEEE Networks Special Issue on Network Security*, vol. 13, no. 6, pp. 24-30, 1999.

[183] P. Zimmermann, *The Official PGP User’s Guide*, MIT Press, 1995.