الجمـهورية الجزائرية الديمقراطية الشعبيـة

وزارة التعليم العالي والبحث العلمي

جـامعـة سعـيـدة د. مـولاي الطـاهـر

كـلـيـة العلوم

قسم: الإعلام الآلي

Mémoire de Master

Spécialité : Sécurité Informatique et Cryptographie

Thème

Elliptic curves cryptography for lightweight devices in IoT systems

Présenté par :

Toufik Larid

Dirigé par :

Dr. Taleb Fadia

Promotion 2023 - 2024

University of Saïda Dr. Moulay Tahar
Computer Science Department

Elliptic curves cryptography for lightweight devices in IoT systems

by
LARID Toufik

directed by
Dr. Taleb Fadia

2024

Abstract

The rapid rise of the Internet of Things (IoT) has changed several industries by providing a network of connected objects that allow for automation, data collection, and workflow improvement. However, this extension raises serious security concerns, especially for lightweight devices with limited processing power and energy resources. This thesis studies the use of Elliptic Curve Cryptography (ECC) as a secure solution for resource-constrained IoT devices. ECC provides high security with smaller key sizes and fewer computational demands than older cryptographic algorithms such as RSA, making it ideal for IoT applications.

The thesis begins by looking at the principles of IoT, its design, and the security concerns it faces, particularly the restrictions of device memory, computing power, and energy usage. It then digs into the mathematical foundations of ECC, giving an in-depth understanding of the fundamental algebraic structures that provide its cryptographic strength. Various ECC-based cryptographic systems, including Elliptic Curve Diffie-Hellman (ECDH) and Elliptic Curve Digital Signature Algorithm (ECDSA), are thoroughly examined, with a focus on their security and operational efficiency. A comparison of an ECC-based scheme and RSA is carried out to assess their performance in terms of key generation, encryption and decryption, signature verification, and energy consumption.

The findings show that ECC is ideal for protecting lightweight IoT devices, providing a good mix of security and efficiency. This work advances the development of secure cryptographic algorithms for IoT systems and demonstrates ECC's ability to optimize security measures in resource-constrained contexts. The thesis concludes by calling for more widespread use of ECC in IoT applications to provide strong security standards and promote the continuous expansion of IoT technologies.

Acknowledgements

As I complete this thesis, I would like to take this opportunity to express my sincere gratitude to all those who have contributed to the completion of this work.

I want to express my profound gratitude to Dr. Fadia TALEB, my thesis advisor, for her constant support, direction, and inspiration throughout this journey. Her insightful feedback and expertise were invaluable in shaping this research, and her patience and understanding helped me to overcome the obstacles this project presented. I consider myself very fortunate to have worked under her supervision, and I am very grateful for the time and energy she dedicated to supporting me.

I am deeply grateful to my parents for their everlasting love, support, and encouragement, which have been a constant source of strength for me. Their faith in me and sacrifices have made this journey possible. I can't thank them enough for their patience, kindness, and the numerous ways they've helped me during this time. This success would not have been possible without their vision and unwavering support.

Finally, I'd like to express deep appreciation to everyone who contributed to the completion of this thesis, whether directly or indirectly. Your support has been important in helping me accomplish this milestone. This achievement is as much yours as it is mine, and I am grateful to have had you all by my side.

To my beloved Parents.

To the loving memory of my grandmother

Contents

Introduction **9**

1 Internet of Things **12**
1.1 Introduction . . . 12
1.2 Definition . . . 12
1.2.1 Connected Objects . . . 13
1.3 Historical Evolution . . . 13
1.4 Fundamental Concepts and components . . . 14
1.4.1 Sensors . . . 14
1.4.2 Actuators . . . 14
1.4.3 Connectivity . . . 14
1.5 Architecture of IoT . . . 15
1.5.1 Perception Layer . . . 15
1.5.2 Network layer . . . 16
1.5.3 Middleware Layer . . . 16
1.5.4 Application Layer . . . 16
1.5.5 Business Layer . . . 16
1.6 The limitations of IoT . . . 17
1.6.1 Security . . . 17
1.6.2 Memory Constraints . . . 17
1.6.3 Energy Consumption . . . 18
1.6.4 Limited Processing Power . . . 18
1.6.5 Bandwidth Limitations . . . 18
1.7 Security in IoT . . . 19
1.7.1 Symmetric Encryption . . . 20
1.7.2 Asymmetric Encryption . . . 21
1.7.3 Elliptic Curve Cryptography . . . 21
1.7.4 Hashing . . . 22
1.8 Conclusion . . . 24

2 Mathematical Background **25**
2.1 Introduction . . . 25
2.2 Elementary algebraic structures . . . 25
2.2.1 Groups . . . 25
2.2.2 Rings . . . 26
2.2.3 Fields . . . 27
2.2.4 Prime Field . . . 28
2.2.5 Binary Field . . . 29
2.3 Elliptic Curves . . . 29
2.3.1 Composition Law . . . 31
2.3.2 Cardinality of an Elliptic Curve . . . 32
2.3.3 Order and Cofactor of Elliptic Curve . . . 33
2.3.4 The Generator Point . . . 34
2.4 Binary Edwards Curves . . . 34
2.4.1 Definitions and Properties . . . 34
2.4.2 Group Law . . . 35
2.5 Conclusion . . . 36

3 Cryptographic Schemes **37**
3.1 Elliptic Curves Cryptography . . . 37
3.1.1 ECC Keys . . . 37
3.1.2 Private Key, Public Key and the Generator Point in ECC . . . 38
3.1.3 Elliptic Curve Discrete Logarithm Problem ECDLP . . . 39
3.2 Almajed et al. ECC Encryption Scheme . . . 39
3.2.1 CPA and CCA attacks . . . 40
3.2.2 Key Generation Phase . . . 41
3.2.3 Encoding and Mapping Phase . . . 41
3.2.4 Encryption Phase . . . 43
3.2.5 Decryption Phase . . . 43
3.2.6 Decoding Phase . . . 44
3.2.7 Message Integrity and Authenticity . . . 45
3.3 Elliptic Curve Digital Signature Algorithm ECDSA . . . 45
3.4 Rivest-Shamir-Adleman (RSA) . . . 47
3.4.1 RSA Algorithm . . . 47
3.4.2 Security Considerations . . . 49
3.5 Conclusion . . . 50

4 Comparative Study **51**
4.1 Introduction . . . 51
4.2 Hardware and Software Setup . . . 51
4.3 Algorithms Implementation . . . 52

4.3.1 RSA Implementation 52
4.3.2 ECC Implementation 52
4.4 Performance Metrics 53
4.5 Memory Usage 53
4.6 Key Generation 54
4.7 Encryption and Decryption 55
4.8 Signature Generation and Signature Verification 58
4.9 Results Interpretation 59
4.9.1 Key Size and Computational Efficiency 60
4.9.2 Encryption and Decryption 60
4.9.3 Energy Consumption 60
4.10 Conclusion 60

Conclusion **61**

List of Figures

1.1 Five Layer Architecture [23] 15
1.2 Security Requirements for IoT [23] 20
1.3 Lightweight cryptographic for IoT [11] 23

2.1 Elliptic Curves in $\mathbb{R}$ [10] 30
2.2 Elliptic curve defined over $\mathbb{F}_3 1$ with $E : y^2 = x^3 + x + 3$ and $|E| = 41$ [5] . . 31
2.3 Group law of an elliptic curve over $\mathbb{R}$. [10] 32

3.1 Repeated mapped points to the elliptic curve using the ASCII table [3] . . . 40
3.2 RSA algorithm structure 48

List of Algorithms

1 Converting a plain text into a set of blocks [3] 42
2 Mapping the blocks to the elliptic curve. [3] 43
3 Converting binary values into plain text [3] 44
4 ECDSA Key Generation [39] 45
5 ECDSA Signature Algorithm [14] 46
6 ECDSA Verification Algorithm [14] 46
7 RSA Key Generation [35] 47
8 RSA Encryption [33] 48
9 RSA Decryption [33] 48
10 RSA Signature Generation [5] 49
11 RSA Signature Verification [5] 49

List of Tables

4.1 Memory Usage on Raspberry Pi Zero W 53
4.2 Memory Usage on a PC 54
4.3 Key generation and execution time on Raspberry Pi Zero W 54
4.4 Key generation and execution time on PC 55
4.5 Key generation and Energy consumption on Raspberry Pi Zero W 55
4.6 Encryption/Decryption and Execution Time on Raspberry Pi Zero W . . . 56
4.7 Encryption/Decryption and Execution Time on PC 56
4.8 Encryption/Decryption and Energy consumption on Raspberry Pi Zero W . 57
4.9 Signature and Execution Time on Raspberry Pi Zero W 58
4.10 Signature and Execution Time on PC 59
4.11 Signature and Energy consumption on Raspberry Pi Zero W 59

Introduction

The Internet of Things (IoT) has quickly emerged as an influential technology, transforming a variety of industries by connecting billions of objects to the internet. IoT systems provide unprecedented levels of automation, data collection, and workflow optimization in a variety of applications, including smart homes and cities, industrial automation, and healthcare. However, the growth of IoT devices presents substantial security challenges, particularly given the limited computing and energy resources of lightweight devices common in IoT networks. Ensuring secure communication in such a context is critical for protecting data integrity, privacy, and device authenticity.

Cryptography is critical in ensuring the integrity, confidentiality, and authenticity of data in IoT systems, but typical cryptographic systems developed for powerful servers are frequently too computationally demanding for IoT devices. This makes it difficult to install strong security mechanisms like encryption and authentication. Elliptic Curve Cryptography (ECC) presents a possible response to these issues by delivering strong security while requiring less computing and energy than traditional cryptographic methods such as Rivest-Shamir-Adleman (RSA). ECC's ability to provide the same level of security with smaller key sizes makes it ideal for lightweight IoT devices, where resource efficiency is critical. Despite its benefits, implementing ECC in IoT systems presents challenges, including : hardware constraints, energy consumption, and the requirement for efficient algorithms that can work successfully within these constraints.

This thesis explores the use of elliptic curve cryptography in lightweight IoT devices, focusing on its security properties and operational efficiency. It delves at the mathematical foundations of ECC, assesses its performance in IoT contexts, and compares it to other cryptographic algorithms. Furthermore, this work develops on Almajed et al.'s proposed ECC-based scheme [3], by comparing it to RSA and analyzing the trade-offs involved in deploying ECC in IoT scenarios. For this purpose, we take into account factors such as key generation, encryption and decryption processes, signature verification, and energy consumption, in terms of computing efficiency, memory utilization, and energy consumption. By implementing these cryptographic algorithms on a Raspberry Pi Zero W, an example of a lightweight device, this thesis hopes to provide useful insights into the practical implications of deploying ECC in resource-constrained IoT settings.

By investigating these aspects, this study contributes to the development of secure and efficient cryptographic protocols for IoT systems, paving the way for further ECC optimizations for edge computing and increased adoption of IoT technologies across various domains while maintaining robust security standards.

Organization of the thesis

This thesis is structured into four chapters and organized as follows:

The first chapter introduces the Internet of Things (IoT), including its essential concepts, historical evolution, and architecture. It also examines the constraints and security risks associated with IoT devices, particularly in terms of memory, energy usage, and processing capacity. The chapter finishes with an overview of IoT security procedures and how cryptography might help secure IoT devices.

The second chapter gets into the mathematical foundations, namely the algebraic structures that enable elliptic curve cryptography (ECC). This chapter covers fundamental subjects such as groups, rings, and fields, with a particular emphasis on elliptic curves and their use in cryptography applications.

In the third chapter, the thesis delves into several cryptographic algorithms, with a special emphasis on ECC and RSA. Almajed et al.'s ECC-based encryption [3] method is examined in detail, with emphasis on its resilience to CPA and CCA attacks. The chapter also covers the Elliptic Curve Digital Signature Algorithm (ECDSA) and its application to IoT security.

The fourth chapter compares ECC with RSA. It describes the hardware and software setup, as well as the performance measurements used to test various cryptographic algorithms on lightweight devices. The chapter examines key generation, encryption and decryption, and signature verification, with a particular emphasis on energy usage and computational efficiency.

Finally, the conclusion summarizes the research findings, emphasizing the benefits of employing ECC in resource-constrained IoT devices and advocating for additional optimizations in cryptographic protocols for IoT systems.

Chapter 1

Internet of Things

1.1 Introduction

The internet of things (IoT) is rapidly growing, altering our environment with an ever-increasing network of connected objects. IoT is changing our lives in a variety of ways, from smart thermostats that learn our routines to industrial sensors that optimize production lines. Its significance arises from its potential to automate operations, increase efficiencies, and collect important information, moving us into a future filled with intelligent systems. However, this quick growth is not without problems. Security concerns loom big as massive amounts of data are generated and sent, necessitating strong security measures to protect privacy and prevent cyberattacks. As the internet of things becomes more integrated into our lives, resolving these difficulties and developing strong security standards will be critical to guaranteeing its responsible and long-term growth.

In this chapter, we'll present the internet of things, some basic concepts of IoT, his evolution and then the architecture of it, next we will approach the limitations and the security mechanisms, and lastly we'll finish with a conclusion.

1.2 Definition

The internet of things, commonly known as IoT, is an emerging technology that has the potential of changing several industrial sectors. This innovation offers the possibility of collecting and analyzing massive amounts of data, automating and the ability to perform tasks without human intervention [34] [27].

Although the precise definition of IoT varies, this diversity can indicate that it is a complex evolving notion and capable of receiving different forms of instantiation, the term "Internet of Things" can be considered as a general notion designating a network of interconnected smart objects, allowing them to communicate and share information over the internet [18].

It includes the connecting of people, processes, data, and things, which adds value to the IoT and enhances people's lives [15]. IoT devices are typically digital and may include built-in pre-processing mechanisms to mitigate negative effects [2]. The Internet of Things provides prospects for developing smart services and solutions in fields such as climate change, precision agriculture, smart health, advanced manufacturing, and smart cities.

IoT devices may include a wide range of objects, like appliances, cars, wearable gadgets, and industrial equipment; however, the growing use of IoT raises concerns about privacy, security, and data management [27].

1.2.1 Connected Objects

An object is an independent entity that may communicate with other objects using a predefined message exchange sequence [13]. It is distinguished by the services it provides as well as the coordination protocol [22]. These objects can coordinate their resources to meet the system's overall goals [38]. In terms of data transfer, IoT objects can send various kinds of data, including sensory information gathered from their surroundings [25]. The exchange of data between IoT devices is critical for the creation of new services and applications in the IoT environment. The term "object" refers to a wide range of products, from simple household appliances such as thermostats and light bulbs to sophisticated industrial gear and wearable devices.

1.3 Historical Evolution

The term "Internet of Things" was originally the title of a presentation that Kevin Ashton, co-founder of the Auto-ID Center at the Massachusetts Institute of Technology (MIT) in 1999 [4]. However, the concept of Internet of Things predates the term itself, as it emerged as a result of the convergence of various technological trends and the vision of a highly interconnected world. In the beginning, radio frequency identification (RFID) devices were the first deployed technology for simple IoT applications, allowing objects to connect with other objects or a server without human intervention [20]. In the early 2000s, the concept of IoT gained momentum as wireless communication technologies such as Bluetooth and RFID became more widely available. The first IoT device, a toaster that could be controlled over the internet, was developed in 1990. Since then, IoT has grown rapidly, and today, there are billions of connected devices worldwide, ranging from smart home appliances to industrial machinery. The development of IoT has been driven by advances in technology, including the miniaturization of sensors and the increasing availability of wireless communication networks. The Internet of Things is expected to grow rapidly and have a large economic impact. It is projected that more IoT devices and sensors will be connected to the Internet, as well as the emergence of new IoT applications. Gartner estimates that over 8.4 billion connected devices were in operation globally in 2018, up more than 31% from 2016. By

2020, it is estimated that the number would approach 20.8 billion, and the exponential increase will continue in the future [20].

1.4 Fundamental Concepts and components

The key elements of an IoT system they collect and process data from the real world include sensors, actuators, connectivity

1.4.1 Sensors

Sensors is a component that detects some type of input from the real world by transforming an observed physical quantity in a digital quantity. They are a fundamental component of an IoT system as they can capture any quantifiable quantity, including temperature, humidity, light, motion, pressure, sound, and so on, the generated data can then processed by the IoT system. A few examples of the application of sensors in IoT system include : traffic sensors that provide data to operate traffic lights, leak detectors that can operate water valves, and image sensors with facial recognition capabilities that can then perform access control (e.g., via remote door locks) [30].

1.4.2 Actuators

Actuators are the part of the IoT system that take action based on a signal or a command sent from the control system to produce a mechanical or electrical action. Actuators can take many forms, based on the use and purpose for which they are intended for. Some common examples of actuators in IoT systems include : motors, solenoids, servo motors, valves, relays, Actuated switches [30].

1.4.3 Connectivity

The capacity of objects to communicate with one another over one or more networks, is referred to as connectivity. The objects are being able on one hand to send data such as their state or the captured data, and on the other hand receive information such as commands and data. In the context of IoT, Connectivity refers to the capacity to remotely monitor, operate, and manage numerous objects. Several communication protocols and technologies, including : Wi-Fi, Bluetooth, Zigbee, cellular networks, LoRa and more, are used to establish connectivity in IoT systems [1].
Understanding and combining these important principles and components is critical for creating, implementing, and managing secure IoT systems. The combination of these features allows the Internet of Things' full potential to be realized across multiple domains. These components work together to form a functioning and efficient IoT system that enables the seamless integration of physical objects with the digital world.

1.5 Architecture of IoT

The Internet of Things (IoT), which is now a mix of technologies such as RFID, NFC, wireless sensors and actuators, ultra-band or 3/4G, IPv6, 6lowPAN, and this requires the design of an architecture and standards to enable future development.

There are several suggested architectures for IoT such as RAMI 4.0, IIRA, and IoT-A. Other structures were presented, Despite the effort and researches attempting to create a standard architecture for IoT, there is yet no universally approved architecture [28]. One of the most adopted architecture by numerous authors is the five layers architecture.

According to [23], this architecture splits an IoT system as the name suggests into five layers; perception, network, middleware, application, and business Layers, as shown in Figure 1.1.

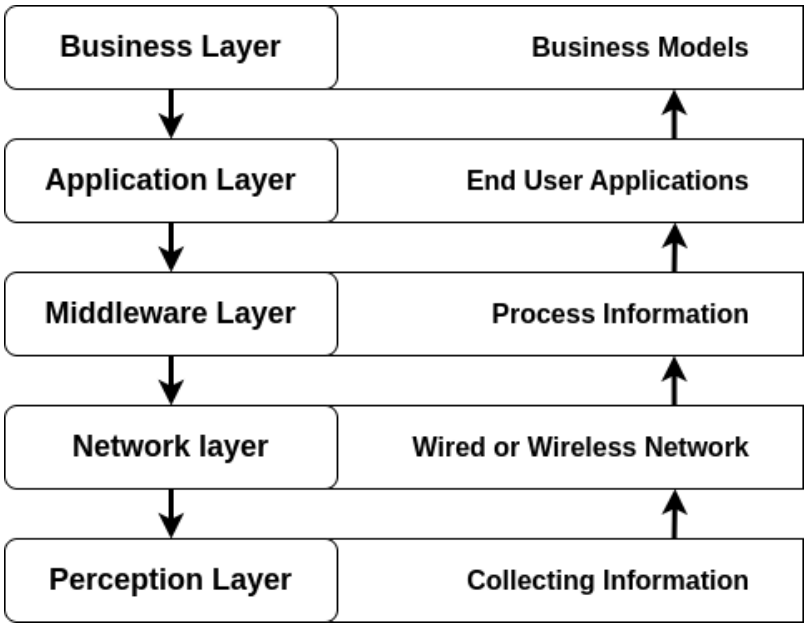


Figure 1.1: Five Layer Architecture [23]

1.5.1 Perception Layer

The perception layer is also known as the recognition layer. The primary function of this layer is to recognize objects and collect information. This layer consists of a collection of physical objects or things. It collects data and provides information to the objects. The layer's job is to ensure the security of data collection, storage, and transmission to the network layer. RFID (Radio-Frequency Identification), sensors, cameras, GPS (Global Positioning System), and other elements are all part of the perception layer, which is determined

by the features of each component, such as the protocols or communication technologies utilized [2].The perception layer is also known as the recognition layer. The primary function of this layer is to recognize objects and collect information. This layer consists of a collection of physical objects or things. It collects data and provides information to the objects. The layer's job is to ensure the security of data collection, storage, and transmission to the network layer. RFID (Radio-Frequency Identification), sensors, cameras, GPS (Global Positioning System), and other elements are all part of the perception layer, which is determined by the features of each component, such as the protocols or communication technologies utilized [23].

1.5.2 Network layer

The network layer is the most advanced component of typical IoT architecture. It is considered to be the brain of the Internet of Things architecture. The network layer is in charge of transferring and processing information provided by the perception layer. It plays an important role in processing data linked to IoT management. The network and communication technologies utilized in this layer, such as wired, wireless, and satellite, are determined by the perception layer's strategies. The perception layer is strongly tied to the network layer because of the communication techniques employed, such as Wi-Fi and Bluetooth [23].

1.5.3 Middleware Layer

The Middleware Layer is often referred to as a processing layer. It is built upon the network layer. IoT systems run on this layer. It offers an API (Application Programming Interface) for implementing applications. Furthermore, it offers a variety of services, including data analysis, data processing, device detection and management, data gathering. The Middleware Layer makes use of conventional protocols such as CoAP, MQTT, XMPP, and HTTP [23].

1.5.4 Application Layer

The application layer provides an application user interface. The collected and analyzed data can be presented to an end user using the Application layer [28]. It is in charge of delivering and offering numerous applications in a wide range of sectors where IoT technology can be deployed and applied, including smart homes, smart cities, smart health, and more. The main purpose of the Application layer is to connect IoT consumers and apps [23].

1.5.5 Business Layer

The business layer manages all aspects of the IoT system, including apps, business models, and data received from the app layer. The business layer creates IoT applications and helps to build viable business models for the promotion of IoT-related technologies. Furthermore,

this layer should monitor and maintain users' privacy, which is critical to the internet of things [23].

1.6 The limitations of IoT

The Internet of Things (IoT) has the potential to bring about significant advancements in several fields, but its successful implementation faces several limitations that can impact the efficiency and effectiveness of IoT systems.

1.6.1 Security

Implementing robust security measures in IoT devices is essential yet challenging due to the computational overhead they need. Encryption, decryption, and other security protocols can significantly affect the performance of IoT systems. This means that achieving a balance between security and performance is critical when developing and implementing security solutions for IoT systems [9].

The vulnerability of IoT stems from its wireless sensor network foundation, making it more susceptible to unauthorized access and malicious attacks compared to wired networks. This vulnerability is exacerbated by the direct control IoT has over physical entities, including critical systems like health monitoring, traffic control, and inventory management [21].

Security is the most critical issue that may face IoT development. Providing security for IoT technology is a huge and real challenge, and failing to address security concerns in IoT can lead to catastrophic consequences, such as factory shutdowns, vehicle accidents, incorrect medical treatments, and traffic congestion caused by malicious intrusions [21] [23].

1.6.2 Memory Constraints

IoT devices tend to have low and limited RAM and flash memory compared to traditional devices, this might affect their capacity to store and retrieve data, and some are unable to transmit or store data at all. This constraint may limit the ability to run programs or save historical data locally, which is a major challenge for IoT hardware manufacturers and software developers to design comprehensive security measures within a low memory. As they need to leave enough space for security software to defend against security threats [32].

Traditional security algorithms are not memory efficient since traditional systems utilize large RAM. As a result, because IoT devices are small, security systems could not have enough memory capacity. Therefore, conventional security approaches cannot be used to secure IoT systems [23].

1.6.3 Energy Consumption

Batteries and other limited sources of energy power a large number of Internet of Things devices. Improving energy efficiency is essential for increasing battery life and lowering the frequency of replacements and recharges. Also, developing energy-efficient IoT devices presents a big problem. Some IoT gadgets use a lot of electricity and aren't rechargeable. Therefore, they use low-bandwidth connections to conserve battery life in devices with limited capacity [32]. Energy supplies are therefore crucial, especially for IoT devices that are battery-powered.

In addition, due to the intermittent and unpredictable nature of renewable energy sources such as solar panels, it poses challenges when relying on renewable energy as the primary source of electricity in IoT networks. Furthermore, factors such as sleep power consumption, harvesting sensitivity power, and desired sensor measurement rate all have an impact on IoT device energy usage [19].

1.6.4 Limited Processing Power

Due to compact size and power constraints of IoT devices, they typically rely on low-power CPUs or microcontrollers with limited computational power. This constraint may have a significant impact on how they handle activities that need extensive processing or analysis on their own [21]. For these devices to function effectively, lightweight protocols are therefore necessary [32].

This constraint can have a wide-ranging impact on IoT applications. For example, in industrial environments where sensors collect massive volumes of data from machinery and equipment, local processing is critical for effective monitoring and predictive maintenance. Such devices are unable to do computing tasks on their own with their limited resources, and must collaborate with other devices [21].

1.6.5 Bandwidth Limitations

The volume of data flowing between devices and central servers on the Internet of Things (IoT) might cause a bottleneck. This is especially true for real-time applications, where delays caused by large data transmissions can be critical. In wireless networks, where bandwidth is already limited, transmitting only necessary data becomes critical.

Compared to wired networks, wireless connections offer significantly less bandwidth. For example, mobile devices like smartphones are relatively powerful and generate data quickly, while small personal area networks like Zigbee (based on the IEEE 802.15.4 standard) use low-power radios and have a low data rate. On the other hand, applications generate massive amounts of data, often transmitted through satellites with limited bandwidth. Furthermore, the growing number of IoT devices connecting to the internet further strains this limited resource, as each device continuously collects and transmits data. This creates a significant challenge [23].

1.7 Security in IoT

Security requirements in IoT systems encompass safeguarding two critical facets: the confidentiality of data and the authentication of identity. These requisites are underpinned by five principal tenets in information security, namely data availability, confidentiality, integrity, authenticity, and authorization. Any compromise within these domains could precipitate security breaches or vulnerabilities within the IoT ecosystem. Consequently, each layer of the IoT architecture must adhere to these stringent security standards. As delineated in Figure 1.2, the primary security imperatives for IoT environments are elucidated. Data availability assumes paramount importance, ensuring unfettered access to secure and reliable data. To mitigate risks such as denial-of-service (DoS) and distributed-denial of services (DDoS) attacks, robust data backup mechanisms are imperative. Upholding data confidentiality mandates employing robust encryption methodologies to forestall unauthorized access and disclosure. Similarly, safeguarding data integrity necessitates measures such as cyclic redundancy checks (CRC) to detect and rectify network errors, thereby fortifying against potential cyber threats. Authentication and authorization mechanisms are pivotal in verifying user and device identities, thereby facilitating access solely to bona fide IoT entities or services [23].

The growth of IoT technology has connected lots of devices to the internet, raising security and integration concerns. Security is a significant concern due to the potential exposure of personal data and the risk of compromising entire IoT systems. Ensuring confidentiality, integrity, and availability of transmitted data remains a major concern. Many IoT devices lack sufficient security measures, leaving them vulnerable to attacks [16].

Cryptography is a primary solution to address security concerns. However, selecting the right encryption algorithm is critical due to resource constraints in IoT systems. While stronger encryption enhances security, it also increases computational demands and energy consumption, which is problematic for resource-constrained IoT devices with limited power and computational capacity, to mitigate these challenges, Lightweight cryptographic techniques have been suggested to overcome the issues of conventional encryption and assesses device longevity while taking secure data transfer into account in IoT devices [16] [36].

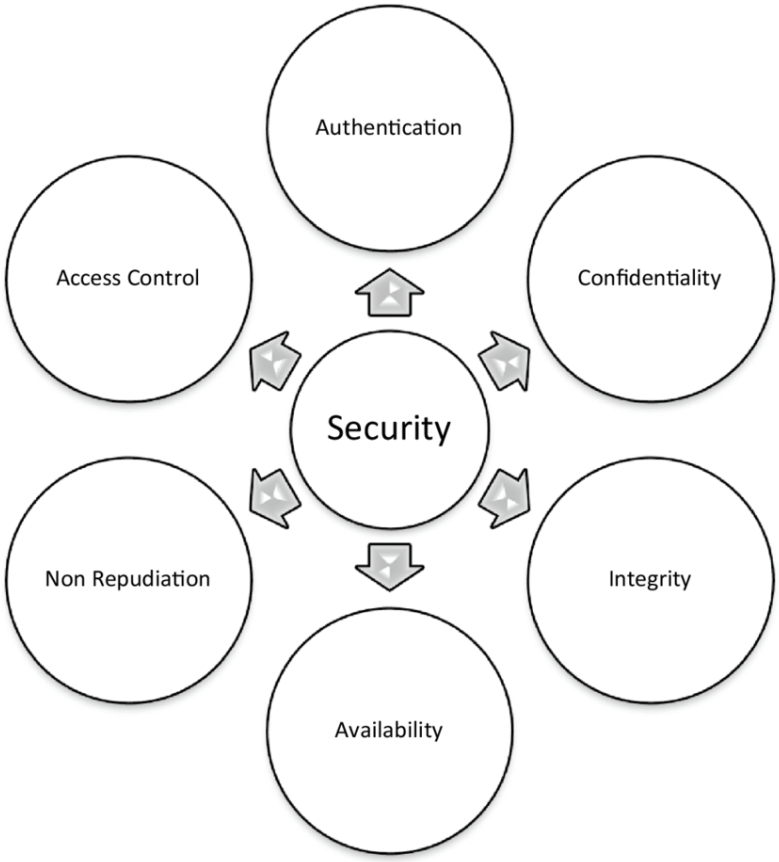


Figure 1.2: Security Requirements for IoT [23]

1.7.1 Symmetric Encryption

in symmetric block ciphers. Such ciphers are based on two types of structure: Substitution Permutation network (SPN) and Feistel [16].

Feistel Structures

The Feistel structure divides a data block into two equal parts and performs rounds for encryption, with each round containing two distinct operations, one for encrypting the plain text and one for substitution. The decryption procedure is similar to encryption, except that the keys are used in reverse order. As one might expect, security is directly related to the number of rounds, but this increase comes at the cost of higher latency associated with the implementation. This structure's downside is slow encryption and decryption, which makes it unsuitable for low-latency networks. The fundamental advantage of the Feistel

structure is that the encryption and decryption procedures use the same software code, which saves memory. This can be used in battery-powered IoT devices with low average power. There are numerous Feistel ciphers available, including DES, TEA, Camellia, SEA, CLEFIACLEFIA, TWINE, LBlock, Piccolo, Blowfish, and HIGHT [16].

Substitution Permutation Network (SPN) Structures

The SPN structure uses a single round function that is applied throughout the whole datablock. It is based on Shannon's principle of confusion, which is realized through substitution, and the principle of diffusion, which is implemented by linear transformation. The security depends on the linear function's complexity. The decryption method is as simple as reversing the encryption, and it delivers faster processing than Feistel ciphers. Its key advantage is low resource implementation, since SPN uses less energy than other structures to provide the same level of security due to a shorter round of execution. The downside is the high level of attacks on SP-based algorithms connected to differential and linear cryptanalysis due to the lack of a key schedule. SPN ciphers include AES, PRESENT, Klein and Serpent [16].

1.7.2 Asymmetric Encryption

Asymmetric ciphers are less common in IoT but very popular for resourceful devices since they are more computationally intensive than symmetric algorithms. Such ciphers can also be used to exchange keys and authenticate IoT devices before encrypting and sending data. Such algorithms include RSA and ECC. ECC, by design, requires a smaller key size than RSA to achieve an identical level of security. As security requirements necessitate larger key sizes, the tendency is to migrate from RSA to ECC, which is a popular choice among IoT security developers and suggested by Symantec. Furthermore, they are included in the majority of cryptographic key management systems used by small and medium-sized organizations, making them appealing to companies. The downside of symmetric ciphers is that they have a big key size, consume more memory, and execute at a slow speed. It is a challenge for developers and researchers to improve ECC by lowering memory needs and computational complexity [16].

1.7.3 Elliptic Curve Cryptography

ECC is an asymmetric cipher that relies on the algebraic structure of elliptic curves over finite fields. The size of an elliptic curve's key is equal to the size of the field over which the curve is defined, thus making it requires less key size and storage than RSA, allowing it to work faster and be deployed on resource-constrained devices which make it an alternative for lightweight cryptography [40]. To optimize the use of low-power devices, ECC utilizes bit-shifting rather than complex multiplication [17].

Here are some of the cryptographic schemes based on elliptic curves frequently used in IoT for lightweight cryptography.

Elliptic Curve Diffie-Hellman

ECDH is a key exchange protocol that enables two parties to establish a shared secret across an unsecured channel. The shared secret can then be used to encrypt future communications. ECDH is efficient and secure, making it ideal for lightweight IoT devices.

Elliptic Curve Integrated Encryption Scheme

ECIES is an encryption method that combines ECC and symmetric encryption. It ensures the secrecy, integrity, and validity of encrypted data. ECIES is ideal for IoT applications where secure data transmission is critical.

Elliptic Curve Digital Signature Algorithm

ECDSA is used for digital signatures, which provide authentication and data integrity. It is commonly used in IoT devices to secure firmware updates and authenticate messages. ECDSA provides the same level of security as RSA but with significantly smaller key sizes, which is beneficial for resource-constrained systems.

In the following chapters, we will dive into a more comprehensive exploration of Elliptic Curve Cryptography (ECC), explaining its concepts in greater detail, as it is the focus of this study.

1.7.4 Hashing

Hashing algorithms play an important role in IoT systems because they provide data integrity and security. In the context of IoT, hashing algorithms are utilized to generate unique fixed-size hash values from raw data. These hash values can be used for a variety of applications, including data verification, authentication, and encryption. One important application of hashing algorithms in IoT systems is to ensure the integrity of data sent between devices. IoT devices can verify that data packets or messages have not been altered during transmission by creating hash values for them. This procedure aids in detecting any illegal changes or tampering with the data. Furthermore, hashing algorithms are utilized for authentication in Internet of Things systems. Devices can securely authenticate one another without communicating sensitive data over the network by generating hash values of sensitive information such as passwords or access tokens. This approach helps to prevent eavesdropping and unauthorized access to sensitive information [31].

Various hashing algorithms are used in IoT systems to improve security and performance. While SHA-256 and MD5 are among the most commonly used, there are other

algorithms that can be used in certain contexts or to meet unique issues in IoT environments. Examples include:

1. **SHA-2 family**: Includes SHA-256, SHA-384, and SHA-512, which provide higher security than prior versions such as SHA-1.
2. **MD5**: Although less secure than modern options, it is nevertheless used in some legacy systems and for compatibility purposes.
3. **Lightweight Cryptography**: Designed for resource-constrained IoT devices, including hashing methods MurmurHash, CityBlock, and PHOTON [37].
4. **Chaotic Sequence-Based Hashing**: Uses chaotic sequences to improve resistance against brute-force and plain-text assaults [6].
5. **Similarity hashing**: Used to detect malware on IoT devices, such as Lempel-Ziv Jaccard Distance (LZJD) [29].
6. **TLSH (Truncated Locality Sensitive Hashing)**: A locality-sensitive hashing algorithm designed for detecting almost identical files [29].

Within the increasingly complex ecosystem of the Internet of Things (IoT), hashing algorithms play a pivotal role in ensuring the integrity and security of massive volumes of data traveling between several devices. These algorithms are chosen based on factors such as security strength, computational complexity, memory requirements, and suitability for IoT devices' constraints. They serve as digital sentinels, ensuring the truthfulness and dependability of communication while protecting sensitive information from malicious actors [31].

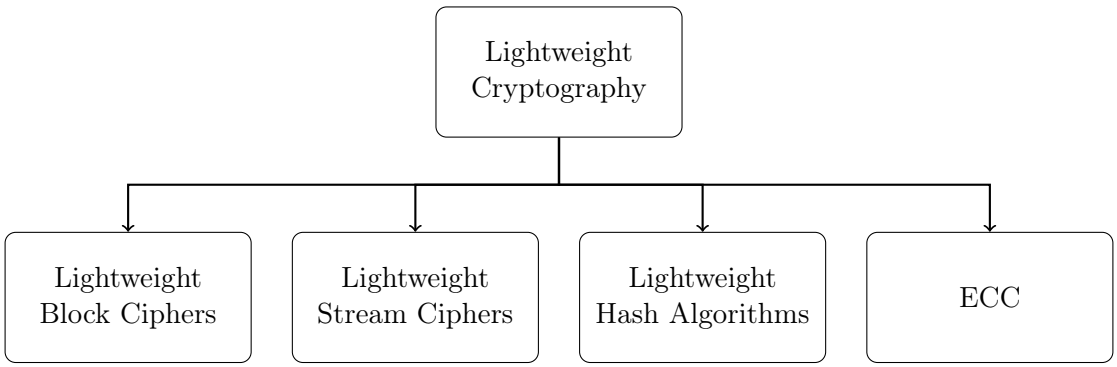


Figure 1.3: Lightweight cryptographic for IoT [11]

1.8 Conclusion

The internet of things (IoT) is on a trajectory of exponential growth, offering a future filled with interconnected devices that effortlessly enhance our lives. However, its expansion is contingent on a fundamental factor: security. Robust encryption and cryptography algorithms, suited to the resource constraints of many IoT devices, form the foundation of a secure and resilient ecosystem.

Implementing these algorithms is more than just a question of technical competence; it is a call to action. Researchers work continuously on lightweight algorithms that provide strong protection without losing performance. Developers must incorporate these algorithms into the core of their designs while prioritizing secure coding approaches.

The rewards are immense. By emphasizing security and adapting cryptography solutions to the IoT's specific needs, we can realize the full potential of this revolutionary technology. We establish a secure environment in which innovation grows, data flows freely without fear of exploitation, and trust underpins all interactions. The future of IoT is promising, but it demands a commitment to security. And with ongoing effort, the interconnected future of the IoT could grow, strong, and safe against the ever-changing world of cyber threats.

Chapter 2

Mathematical Background

2.1 Introduction

Our reminders will center on the concept of Fields. This fundamental algebra structure serves as the foundation for the development of elliptic curves used in cryptography. The group and the ring are two fundamental algebraic objects that serve as the foundation for Fields. This chapter on mathematical background is mostly derived from [10].

2.2 Elementary algebraic structures

2.2.1 Groups

A **group** $(G, +)$ is an ordered pair consisting of a set G and a binary operation $+$ on G satisfying the following properties:

- **Closure** : $\forall a, b \in G \mid a + b \in G$.
- **Associativity** : $\forall a, b, c \in G \mid (a + b) + c = a + (b + c)$.
- **Identity** : $\exists! 0 \in G : \forall a \in G \mid a + 0 = 0 + a = a$.
- **Inverse** : $\forall a \in G$, $\exists! b \in G \mid a + b = b + a = 0$.

  A final important property can be verified by certain groups also called abelian groups.
- **commutativity** : $\forall a, b \in G \mid a + b = b + a$

Definition 2.1. Let G be a group. A subgroup H of G is a subset of G containing the identity and such that :

- $\forall x, y \in H$ one has $x + y \in H$.
- if $x \in H$ then also $x^{-1} \in H$.

Definition 2.2. Let G and G' be two groups with respective laws $\times$ and $\otimes$ and units 0 and $0'$.

- A group homomorphism ψ between G and G' is a map from G to G' such that $\forall x, y \in G \mid \psi(x \times y) = \psi(x) \otimes \psi(y)$.
- The kernel of ψ is $\ker \psi = x \in G \mid \psi(x) = 0'$.

2.2.2 Rings

The **ring** structure is constructed from an abelian group. To this group we add a second binary operation, which will be denoted $\times$. The triplet $(G, +, \times)$ must verify all the previous properties as well as:

- **Associativity** : $\forall a, b, c \in G \mid (a \times b) \times c = a \times (b \times c)$
- **Neutral element** : $\exists! \, 1 \in G, 1 \neq 0 \mid \forall a \in G, a \times 1 = 1 \times a = a$
- **Distributivity** : $\forall a, b, c \in G \mid a \times (b + c) = a \times b + a \times c, (b + c) \times a = b \times a + c \times a$

  As before, certain rings are said to be commutative if they verify:
- **Commutative** : $\forall a, b \in G \mid a \times b = b \times a$

Definition 2.3. Let R and R' be two rings with the respective operations $+, \times$ and $\oplus, \otimes$. A ring homomorphism ψ is an application from R to R' such that for all $x, y \in R$

- $\psi(x + y) = \psi(x) \oplus \psi(y)$
- $\psi(x \times y) = \psi(x) \otimes \psi(y)$
- $\psi(1) = 1$

Definition 2.4. Let R be a ring, I is an ideal of R if it is a nonempty subset of R such that

- I is a subgroup of R with respect to the law $+$
- $\forall x \in R \wedge y \in I \mid xy \in I \wedge yx \in I$

The ideal $I \subsetneq R$ is prime if for all $x, y \in R$ with $xy \in I$ one obtains $x \in I \vee y \in I$. The ideal $I \subsetneq R$ is maximal if for any ideal J of R the inclusion $I \subset J \Rightarrow J = I \vee J = R$. Two ideals I and J of R are coprime if $I + J = \{i + j \mid i \in I \wedge j \in J\}$ is equal to R.

Definition 2.5 (Characteristic)**.** Let R be a ring and ψ the homomorphism defined above. The kernel of ψ is an ideal of $\mathbb{Z}$ generated by a positive integer m. m is then called characteristic of R.

2.2.3 Fields

A field is an algebraic structure built upon a commutative ring. It satisfies one additional property:

- **Inverse** : $\forall a \in G, \exists! \, b \in G \mid a \times b = b \times a = 1$

characteristic of a field

Definition 2.6. Let K and L be fields. A homomorphism of fields is a ring homomorphism between K and L.

We remark that a homomorphism of fields is always injective, for it is immediate that its kernel is reduced to $\{0\}$.

Definition 2.7 (Characteristic)**.** Let R be a ring and ψ the homomorphism defined above. The kernel of ψ is an ideal of $\mathbb{Z}$ generated by a positive integer m. m is then called the characteristic of R.

Proposition 2.1. *The characteristic of a field is either zero or equal to a prime number p.*

Definition 2.8 (Finite Field)**.** A field F is said to be finite if its cardinality is finite.

Theorem 2.1. *For every prime number p and positive integer d, there exists a unique finite field $\mathbb{F}_{p^d}$ defined up to isomorphism.*

Theorem 2.2. *For every element $a \in F$:*

$$a^{|F|-1} = 1.$$

Definition 2.9 (Multiplicative order)**.** The multiplicative order of an element $a \neq 0, 1$ in the finite field F is the smallest integer n such that $a^n = 1$.

Theorem 2.3. *Let F be a finite field. The group F^* is cyclic.*

Definition 2.10 (The Frobenius morphism)**.** Let $\mathbb{F}_{p^d}$ be a finite field, then we define the map Ψ as follows:

$$\Psi = \begin{cases} \mathbb{F}_{p^d} \to \mathbb{F}_{p^d} \\ a \mapsto a^p \end{cases}$$

Field extension

Cryptography on elliptic curves requires the usage of finite fields with a large cardinality. Indeed, this cardinality will play a role in the cryptography's security. There are two instant methods for increasing the size of finite fields. The first is to enhance the characteristic of the field and work, modulo a large prime number (p). The second step involves increasing the dimension d of F as a $\mathbb{Z}/p\mathbb{Z}$-vector space. To describe how to construct a finite field extension, we can begin with the general definition of a field extension:

Definition 2.11. Let K and L be two fields. We say that L is a field extension of K denoted by L/K if there exists a homomorphism from K to L.

Theorem 2.4. *Every finite extension $\mathbb{F}_{p^d}/\mathbb{F}_p$ is a Galois extension, and the group of automorphisms $Gal(\mathbb{F}_{p^d}/\mathbb{F}_p)$ is a cyclic group of order d generated by the Frobenius morphism Ψ_p.*

Definition 2.12 (Trace)**.** The trace of an element a in the finite field F_{p^d} is defined by:

$$\mathrm{Tr}(a) = \sum_{i=1}^{d} a^{p^i}$$

2.2.4 Prime Field

A prime field in algebra is a type of finite (Galois) field not containing proper subfields, denoted by $\mathbb{F}_p$ or $GF(p)$, where p is a given prime number. The field size is determined by the prime number p. The field has exactly p elements[8].

It consists of a set of integers $\{0, 1, 2, \ldots, p-1\}$ with two operations:

- **Addition**: If $a, b \in \mathbb{F}_p$, then $a + b = r$ in $\mathbb{F}_p$, where $r \in [0, p-1]$ is the remainder when the integer $a + b$ is divided by p. This is known as addition modulo p and written $a + b \equiv r \ (mod\ p)$[8].
- **Multiplication**: If $a, b \in \mathbb{F}_p$, then $ab = s$ in $\mathbb{F}_p$, where $s \in [0, p-1]$ is the remainder when the integer ab is divided by p. This is known as multiplication modulo p and written $ab \equiv s \ (mod\ p)$[8].

Theorem 2.5 (prime fields)**.** *For every prime p, the set $R_p = \{0, 1, \ldots, p-1\}$ forms a field (denoted by $\mathbb{F}_p$) under mod-p addition and multiplication.*

Theorem 2.6 (Prime field uniqueness)**.** *Every field $\mathbb{F}$ with a prime number p of elements is isomorphic to $\mathbb{F}_p$ via the correspondence $1 \oplus \cdots \oplus 1 \in \mathbb{F} \leftrightarrow i \in \mathbb{F}_p$.*

Theorem 2.7 ((Existence of finite fields)**.** *For every prime p and positive integer m, there exists a finite field with p^m elements.*

Theorem 2.8 (Existence of finite subfields)**.** *Every finite field with p^m elements has a subfield with p^n elements for each positive integer n that divides m.*

2.2.5 Binary Field

The finite field $\mathbb{F}_{2^m}$ is the characteristic 2 finite field containing 2^m elements. Although there is only one characteristic 2 finite field $\mathbb{F}_{2^m}$ for each power 2^m of 2 with $m \geq 1$, there are many ways to represent the elements of $\mathbb{F}_{2^m}$.

Here the elements of $\mathbb{F}_{2^m}$ should be represented by the set of binary polynomials of degree $m-1$ or less:

$$\{a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \cdots + a_1x + a_0 : a_i \in \{0,1\}\}$$

with addition and multiplication defined in terms of an irreducible binary polynomial $f(x)$ of degree m, known as the reduction polynomial, as follows:

- **Addition**: If $a = a_{m-1}x^{m-1} + \cdots + a_0$, $b = b_{m-1}x^{m-1} + \cdots + b_0 \in \mathbb{F}_{2^m}$, then $a+b = r$ in $\mathbb{F}_{2^m}$, where $r = r_{m-1}x^{m-1} + \cdots + r_0$ with $r_i \equiv a_i + b_i \ (mod\ 2)$.
- **Multiplication**: If $a = a_{m-1}x^{m-1} + \cdots + a_0$, $b = b_{m-1}x^{m-1} + \cdots + b_0 \in \mathbb{F}_{2^m}$, then $ab = s$ in $\mathbb{F}_{2^m}$, where $s = s_{m-1}x^{m-1} + \cdots + s_0$ is the remainder when the polynomial ab is divided by $f(x)$ with all coefficient arithmetic performed modulo 2.

2.3 Elliptic Curves

The world of elliptic curve theory is vast and multifaceted, encompassing both algebraic varieties and the intricacies of Weierstrass equations. For our exploration, we'll adopt an approach that resonates with the realm of cryptography. In essence, an elliptic curve can be defined as follows:

Definition 2.13 (Elliptic curves)**.** Let $\mathbb{K}$ be a field. An elliptic curve E over $\mathbb{K}$ is defined as a smooth, projective curve over $\mathbb{K}$ given by the Weierstrass equation:

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \tag{2.1}$$

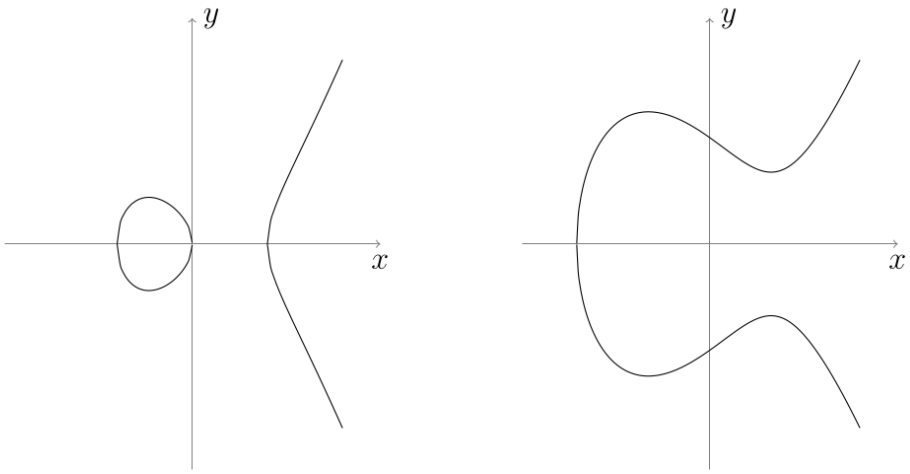
where $a_i \in \mathbb{K}$ and $a_4 \neq 0$.

If the characteristic of K is different from 2, then we can reduce the equation by making the substitution $y \to \frac{1}{2}(y - a_1x - a_3)$, which gives the following equation.

$$E : y^2 = 4x^3 + b_2x^2 + 2b_4x + b_6 \tag{2.2}$$

where $b_2 = a_1^2 + 4a_4$, $b_4 = 2a_4 + a_1a_3$, and $b_6 = a_2^3 + 4a_6$.
If the characteristic of K is equal to 2, we can still reduce the equation by making the following transformation :

$$x \to a_1^2x + \frac{a_3}{a_1}, \quad y \to a_1^3y + \frac{a_3^2 + a_1^2a_4}{a_1^3}$$



(a) $y^2 = x^3 - x$ (b) $y^2 = x^3 - 2x + 2$

Figure 2.1: Elliptic Curves in $\mathbb{R}$ [10]

Definition 2.14 (Elliptic curves (Weierstrass short form))**.** The Weierstrass short form equation of an elliptic curve E over K is given by :

$$\begin{cases} E : y^2 = x^3 + ax + b & \text{if} \quad \text{char}(K) \neq 2, 3 \\ E : y + xy = x^3 + ax + b & \text{if} \quad \text{char}(K) = 2 \end{cases}$$

Where a, b are elements of K.

Definition 2.15 (Discriminant)**.** The discriminant of an elliptic curve E over a field K is defined by:

$$\Delta = \begin{cases} -16(4a^3 + 27a^2) & \text{if char}(K) \neq 2, 3 \\ a^6 & \text{if char}(K) = 2 \end{cases}$$

Definition 2.16. An elliptic curve E over a finite field $\mathbb{F}_q$ is defined as a smooth, projective algebraic curve of genus one equipped with a specified point $O \in E(\mathbb{F}_q)$. It can be described by a Weierstrass equation of the form:

$$y^2 = x^3 + ax^2 + bx + c \tag{2.3}$$

where $a, b, c \in \mathbb{F}_q$ and the discriminant $\Delta = 4a^3 + 27b^2 \neq 0$.

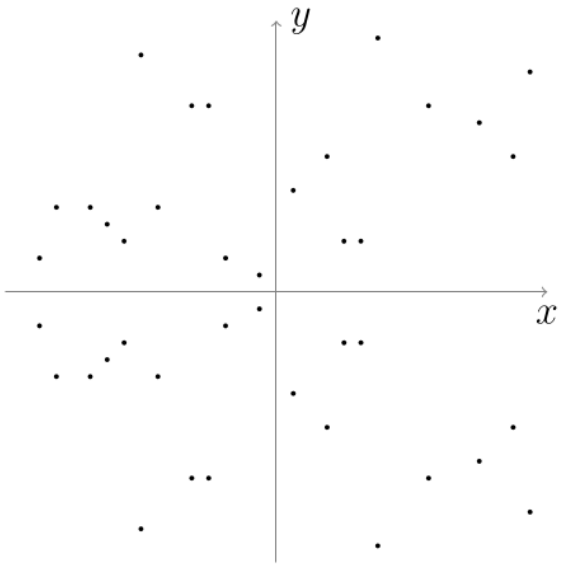


Figure 2.2: Elliptic curve defined over $\mathbb{F}_3 1$ with $E : y^2 = x^3 + x + 3$ and $|E| = 41$ [5]

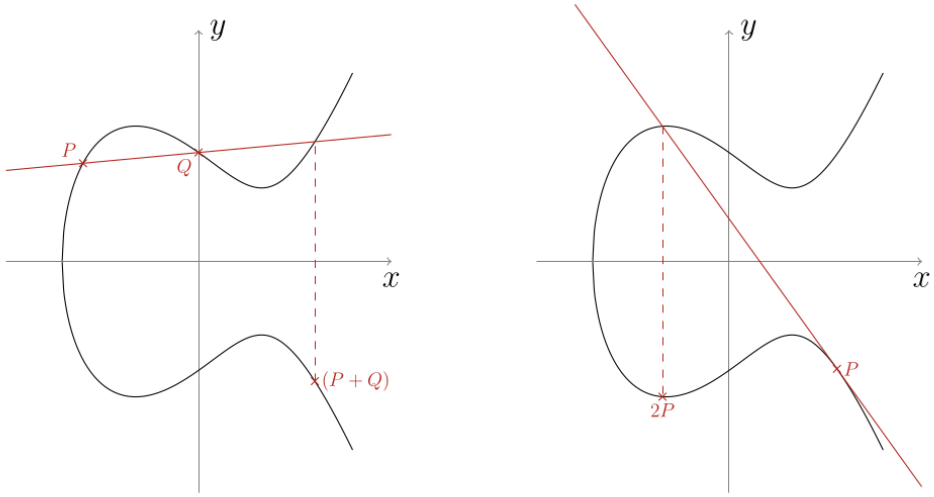
2.3.1 Composition Law

In order to use elliptic curves in cryptographic applications, we define an internal composition law between points of the elliptic curve in such a way as to construct a group. The internal composition law on an elliptic curve will be denoted additively because, as we will see, it is Abelian. Thus, to obtain a group $(E, +)$, the composition law $+$ must satisfy the conditions recalled in section 2.1, namely in this case:

1. **Associativity**: $\forall P, Q, R \in E : (P + Q) + R = P + (Q + R)$
2. **Identity Element**: $\exists! O \in E : \forall P \in E, P + O = O + P$
3. **Inverse**: $\forall P \in E, \exists! Q \in E : P + Q = Q + P = O$
4. **Commutativity**: $\forall P, Q \in E : P + Q = Q + P$

The construction of such a composition law allowing the definition of an Abelian group on the elliptic curve E can be approached in various ways. The first approach is geometric.

Figure 2.3 presents the composition law considered constructing a group on the set of points of curve E, defining the point at infinity O as the neutral element. The second approach, algebraic in nature, will allow us to verify that the group law defined geometrically by Figure 2.3 indeed possesses the properties pertaining to the composition laws of Abelian



(a) Calcul de $P + Q$ (b) Calcul de $2P$

Figure 2.3: Group law of an elliptic curve over $\mathbb{R}$. [10]

groups. To do this, we need to describe algebraically this composition law. For $P = (x_1, y_1)$ and $Q(x_2, y_2)$, two points on curve E, we define $P + Q = (x_3, y_3)$ as follows:

$$x_3 = \lambda^2 - a - x_1 - x_2,$$
$$y_3 = \lambda(x_1 - x_3) - y_1 - x_3$$

where λ is defined differently if $P = Q$ or $P \neq Q$.

$$\lambda = \begin{cases} \frac{y_1 - y_2}{x_1 - x_2} & \text{if } P \neq \pm Q \\ \frac{3x_1^2 + 2ax_1 - y_1}{2y_1 + x_1} & \text{if } P = Q \end{cases}$$

In the particular case where $P = -Q$, the line passing through P and Q is parallel to the y-axis, thus $y_1 = -y_2$, and consequently $P + Q = O$. We have previously seen that to add two points, there are several cases depending on whether $P = Q$, $P = -Q$, or $P = O$. Thus, this group law admits different formulas for adding and doubling points on E. The point at infinity O is a special case to be treated separately. Two essential concepts then qualify a composition law on E.

2.3.2 Cardinality of an Elliptic Curve

In the case where an elliptic curve E is defined over a finite field, the cardinality of E is finite. One way to obtain this cardinality is to search exhaustively for all the points of E.

For curves defined over a finite field of large order, this method is not feasible, as it would require considerable computation time.
The Frobenius endomorphism plays an important role in counting the number of points on an elliptic curve over a finite field. It is defined for any point $P = (x, y)$ of E over the field $\mathbb{F}_{p^d}$ by:

$$\Psi(P) = \begin{cases} (x^{p^d}, y^{p^d}) & \text{if } P \neq O \\ O & \text{if } P = O \end{cases}$$

From this definition of Frobenius follows the following theorem:

Theorem 2.9 (Hasse-Weil)**.** *Let E be an elliptic curve defined over $\mathbb{F}_{p^d}$, then:*

$$|E| = p^d + 1 - t \quad \textit{and} \quad |t| \leq 2\sqrt{p^d}$$

where t is the trace of the Frobenius endomorphism.

Proposition 2.2. *Let E be an elliptic curve over $\mathbb{F}_{p^d}$ and E_{tw} be a twist of E. Then:*

$$|E| + |E_{tw}| = 2p^d + 2$$

2.3.3 Order and Cofactor of Elliptic Curve

An elliptic curve over a finite field can form a finite cyclic algebraic group that includes all the curve's points. In such a group, adding two elliptic curve (EC) points or multiplying an EC point by an integer produces another EC point in the same group. The order of the curve is the total number of EC points on it, which includes the "point at infinity" that results from multiplying a point by zero.

Some curves have a single cyclic group that includes all of their EC points, but others have multiple non-overlapping cyclic subgroups. In the latter instance, the curve's points are separated into h cyclic subgroups, each with order r, so the overall order of the group is $n = h \times r$. The cofactor is the number of subgroups, represented by h.

The cofactor of an elliptic curve is defined by the formula:

$$h = \frac{n}{r}$$

Where n is the total number of points on the curve, h is the number of non-overlapping point subgroups, and r is the number of points in each subgroup, including the infinity point.

Elliptic curve points are arranged into one or more non-overlapping subsets known as cyclic subgroups. The cofactor, h, denotes the number of these subgroups, whereas the total number of points across all subgroups equals the curve's order, n. If the curve contains only one cyclic subgroup, the cofactor is $h = 1$. If there are several subgroups, the cofactor is higher than one. For example, the elliptic curve secp256k1 have a cofactor of 1 and the elliptic curve Curve25519 have a cofactor of 8[39].

2.3.4 The Generator Point

In elliptic curve cryptography (ECC), a particular predefined point on the curve, known as the generator point G, is used to produce any other point in its subgroup by multiplying G by an integer within the range $[0 \dots r]$. The cyclic subgroup's order, indicated by the number r, represents the total number of points in that subgroup.

Curves with a cofactor of 1 have just one subgroup, and the order of the curve n (total points on the curve, including the point at infinity) equals r. Which means that all points on the curve can be generated by multiplying G by integers in the range $[1 \dots n]$. This integer n is known as "order of the curve".

The order r of a subgroup, determined by the generator point G, determines the total number of potential private keys for the curve, given by $r = n/h$, where h is the cofactor. Cryptographers carefully select elliptic curve parameters (curve equation, generator point, and cofactor) to ensure a large enough key space for security.

In ECC, the points on the curve form cyclic groups or cyclic subgroups. This means that that there is a r such that $r \times G = 0 \times G = O$, and all points in the subgroup can be formed by multiplying G by integers in the range $[1 \dots r]$.

While elliptic curve subgroups may contain multiple generating points, cryptographers choose one to produce the full group or a suitable subgroup for optimal performance. Cryptographers prefer subgroups of a prime order r to avoid small-subgroup attacks, because it decreases security. For curves with a cofactor bigger than one, different base points can produce distinct subgroups of points on the curve. Careful generator point selection ensures secure and efficient ECC operations[39].

2.4 Binary Edwards Curves

Binary Edwards Curves (BEC) were introduced by Bernstein[7]. following the work of Edwards[12] in odd characteristic. The arithmetic of these curves has been improved by Kim[24] and Koziel[26]. The remainder of the section presents in detail the mathematical and arithmetic properties specific to this model of elliptic curve.

2.4.1 Definitions and Properties

Definition 2.17. Let K be a field of characteristic 2, and let d_1, d_2 be two elements of K such that $d_1 \neq 0$ and $d_2 \neq d_1^2 + d_1$. We then define the binary Edwards curve E_{d_1,d_2} as the affine curve defined by:

$$E_{d_1,d_2} : d_1(x + y) + d_2(x^2 + y^2) = xy + xy(x + y) + x^2y^2 \tag{2.4}$$

From this definition, we observe that the curve is symmetric in x and y. The theorem below gives an interesting result in the cryptographic application of Binary Edwards Curves.

Theorem 2.10 (Non-singular)**.** *All binary Edwards curves are non-singular.*

An important property of this curve model is its birational equivalence with the Weierstrass model. This property thus allows transitioning from one model to another, ensuring compatibility of cryptographic protocols based on Binary Edwards Curves with those based on the Weierstrass model.

2.4.2 Group Law

The binary Edwards elliptic curve model also admits an internal composition law that allows for a group structure to be given to the set of points on the elliptic curve E_{d_1,d_2}. We can then define addition between two points $P = (x_1, y_1)$ and $Q = (x_2, y_2)$, giving the point $R = (x_3, y_3)$, by:

$$x_3 = \frac{d_1(x_1 + x_2) + d_2(x_1 + y_1)(x_2 + y_2) + (x_1 + x_1^2)(x_2(y_1 + y_2 + 1) + y_1y_2)}{d_1 + (x_1 + x_1^2)(x_2 + y_2)},$$
$$y_3 = \frac{d_1(y_1 + y_2) + d_2(x_1 + y_1)(x_2 + y_2) + (y_1 + y_1^2)(y_2(x_1 + x_2 + 1) + x_1x_2)}{d_1 + (y_1 + y_1^2)(x_2 + y_2)}.$$

Setting $(x_1, y_1) = (0, 0)$ or $(x_2, y_2) = (0, 0)$, it follows that the point $(0, 0)$ is the neutral element of this group law. Similarly, we have $(x_1, y_1) + (1, 1) = (x_1 + 1, y_1 + 1)$, which gives us $(1, 1) + (1, 1) = (0, 0)$. Thus, the point $(1, 1)$ is a point of order two.

In the particular case where $d_1 = d_2$, we have only one parameter d to define the binary Edwards curve, and then the condition for t in K is $d \neq t^2 + t$. The group law formulas give $2(1, 0) = (1, 1)$ and $2(0, 1) = (1, 1)$, so the points $(1, 0)$ and $(0, 1)$ are points of order 4. Thus, regardless of the parameter d of Ed, this subgroup of order 4 will be present, and therefore, the cardinality of a binary Edwards curve with $d1 = d2$ will always be divisible by four. This group law has some interesting properties.

Proposition 2.3. *The group law defined by equations 2.16 and 2.17, on the binary Edwards curve $Ed1, d2$, satisfies the following properties:*

- *If $(x_1, y_1) + (x_2, y_2) = (0, 0)$, then $(x_1, y_1) = (x_2, y_2)$.*
- *If $\varphi(x_1, y_1) = \varphi(x_2, y_2)$, then $(x_1, y_1) = (x_2, y_2)$.*
- *$\varphi(y_1, x_1) = -\varphi(x_1, y_1)$.*
- *If $(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$, then $\varphi(x_1, y_1) + \varphi(x_2, y_2) = \varphi(x_3, y_3)$*

where (x_1, y_1), (x_2, y_2), and (x_3, y_3) are points on E_{d_1,d_2}, and φ is the birational equivalence mapping between the binary Edwards curve model and the Weierstrass model.

Definition 2.18 (Complete Edwards binary curve)**.** Let K be a field of characteristic 2 and let d_1 and d_2 be two elements of this field such that $d_1 \neq 0$ and there does not exist an element t in K satisfying the equation $t^2 + t = d_2$, i.e., the trace of d_2 is equal to 1. We can then define the complete Edwards binary curve, given by the affine equation:

$$\mathrm{Ed}_{d_1,d_2} : d_1(x + y) + d_2(x^2 + y^2) = xy + xy(x + y) + x^2y^2. \tag{2.5}$$

2.5 Conclusion

This chapter established the mathematical foundation required for understanding elliptic curve cryptography (ECC). We started with fundamental algebraic structures, groups, rings, and fields before delving into the unique qualities of finite fields that make them suited for cryptography applications. Our discussion focused on elliptic curves, their definitions, and the group law that allows them to be used in cryptography. We investigated fundamental ideas such as curve cardinality, the Hasse-Weil theorem, and the roles of curve order, cofactor, and generator points in ECC implementations, we also investigated Binary Edwards Curves, which show an approach to elliptic curves in binary fields.

These mathematical foundations are critical to understanding the security and effectiveness of elliptic curve cryptosystems. The distinct qualities of elliptic curves over finite fields, particularly their ability to provide robust security with small keys, make them appealing for modern cryptographic applications, particularly in resource-constrained situations such as IoT devices. As we progress, these ideas will serve as the foundation for developing secure and efficient elliptic curve encryption systems, particularly in resource-constrained situations such as IoT devices.

Chapter 3

Cryptographic Schemes

3.1 Elliptic Curves Cryptography

The introduction of elliptic curve cryptography (ECC) in 1985 changed the way we perform public-key cryptography. Elliptic Curve Cryptography (ECC) is a modern family of public-key cryptosystems that depends on the algebraic structures of elliptic curves over finite fields, as well as the difficulty of the Elliptic Curve Discrete Logarithm Problem [5].

ECC implements all the major asymmetric cryptosystem functions, including encryption, signatures, and key exchange. In this chapter, we will present some cryptosystems based on elliptic curves. But first, we must understand the fundamental principles of elliptic curve cryptography.

3.1.1 ECC Keys

The private keys in the ECC are integers (in the range of the curve's field size). An example of 256-bit ECC private key (hex encoded, 32 bytes, 64 hex digits) is:

0×51897B64E85C3F714BBA707E867914295A1377A7463A9DAE8EA6A8B914246319

The key generation in ECC cryptography is as easy and as securely producing a random integer inside a specific range, hence it is extremely quick. Any integer inside the range represents a valid ECC private key.

The ECC's public keys are EC points, which are pairs of integer coordinates $\{x, y\}$ on a curve. Because of their unique characteristics, EC points can be compressed to a single coordinate plus one bit (odd or even). Thus, the compressed public key, which corresponds to a 256-bit ECC private key, is a 257-bit number.

An example of ECC public key (corresponding to the above private key, as hex with prefix 02 or 03) is:

0×02F54BA86DC1CCB5BED0224D23F01ED87E4A443C47FC690D7797A13D41D2340E1A

In this format, the public key actually takes 33 bytes (66 hex digits), which can be optimized to exactly 257 bits[39].

The choice of elliptic curve impacts the key length, which is typically related to the size of the underlying field. depending on the curve many different ECC key sizes are possible, Here are some curves with their corresponding key sizes:

- 192-bit: secp192r1
- 224-bit: secp224k1
- 256-bit: secp256k1, Curve25519
- 384-bit: p384, secp384r1
- 414-bit: Curve41417
- 521-bit: P-521
- 571-bit: sect571k1

3.1.2 Private Key, Public Key and the Generator Point in ECC

In the ECC, when we multiply a fixed EC point G (the generator point) by certain integer k (k can be considered as private key), we obtain an EC point P (its corresponding public key). Consequently, in ECC we have :

- Elliptic curve (EC) over finite field $\mathbb{F}_p$
- G : generator point (fixed constant, a base point on the EC)
- k : private key (integer)
- P : public key (point)

Elliptic curve cryptography (ECC) relies heavily on the calculation of $P = k \times G$, where k is a scalar and G is an elliptic curve base point. This operation, known as elliptic curve point multiplication, can be completed quickly using well-known algorithms. The 'double-and-add' and 'Montgomery ladder' algorithms are two of the most used for this purpose [39].

The 'double-and-add' algorithm is simple and works by scanning the bits of the scalar k. Each bit doubles the current point and, if the bit is one, adds the point G. This algorithm's time complexity is proportional to $\log_2(k)$, implying that the number of steps required increases logarithmically with k[10].

The 'Montgomery ladder' algorithm, on the other hand, is frequently used because of its resistance to side-channel attacks such as timing and power analysis. This algorithm has

a temporal complexity of $O(\log_2(k))$, but it executes independently of k, offering consistent execution time and better security[10].

For 256-bit elliptic curves, which are widely used in ECC, multiplication will take only a few hundred basic elliptic curve operations. These fundamental operations often include point addition and point doubling on the curve, making the multiplication process fast and computationally possible even for large key sizes[39].

3.1.3 Elliptic Curve Discrete Logarithm Problem ECDLP

The Elliptic Curve Discrete Logarithm Problem (ECDLP) is vital for public key cryptography, especially elliptic curve cryptography (ECC), which provides strong security with smaller key sizes. The difficulty of solving ECDLP ensures the security and confidentiality of digital communications.

The ECDLP can be formally stated as follows: given a point P and a point $Q = kP$ on an elliptic curve E over a finite field $\mathbb{F}_q$, where k is an integer, the goal is to determine k. The problem is considered hard due to the lack of efficient algorithms for solving it in polynomial time. This intractability forms the basis of the security of many cryptographic protocols, including the Elliptic Curve Diffie-Hellman (ECDH) key exchange and the Elliptic Curve Digital Signature Algorithm (ECDSA).

In general, when the field size p is n bits, the security level is around $n/2$. Thus, 256-bit elliptic curves (where the field size p is 256-bit number) typically provide nearly 128-bit security strength. To obtain a comparable level of protection with DLP or RSA, you would need to use several thousands of bits. Due to the usage of smaller numbers, ECC arithmetic is generally faster than RSA or classic Diffie-Hellman[39].

Indeed, ECDLP is a problem of utmost importance to the whole cryptography scheme. In this manner, shorter key lengths may be employed without reducing much from the security sought, due to the complexity derived from the ECDLP equation. For example, in ECC, a 256-bit key will give the same degree of security as a 3072-bit key with RSA. This efficiency leads to faster computations and, in turn, less power consumption and storage requirements, traits especially helpful in constrained environments like mobile devices and smart cards[5][33].

3.2 Almajed et al. ECC Encryption Scheme

This comparison study focuses on Almajed et al. proposed ECC-based scheme [3], where they presented a novel secure encryption strategy using ECC to protect data in IoT and edge computing. The suggested technique is intended to resist a number of encryption attacks, such as the Chosen Plaintext Attack (CPA) and the Chosen Ciphertext Attack (CCA), while also ensuring authenticated encryption (AE).

The study concludes that this new approach not only secures data but also provides effective performance, making it ideal for industrial IoT and urbanization applications where

demand for secure services is increasing [3].

The proposed scheme is divided into nine phases, each of which is crucial to assuring the security and efficiency of data transmission in limited environments such as IoT and edge computing. The primary phases include key generation, encoding, mapping, encryption, decryption, and decoding.

3.2.1 CPA and CCA attacks

The encoding phase is critical because it determines how plaintext is converted into numerical values appropriate for encryption. If this phase is not performed securely, plaintexts may be encoded predictably, leaving the system vulnerable to CPA and CCA.

The CPA occurs when an attacker selects random plain texts and requests the corresponding cipher texts for each text. Thus, the attacker aims to reduce the security of the scheme by analyzing both the plain text and cipher text. The CCA attack is more powerful since the adversary can not only select plaintexts to encrypt, but also retrieve the associated ciphertexts. They can also choose ciphertexts to decrypt and then examine the generated plaintexts. This gives the attacker further flexibility in the attack and may reveal sensitive information.

The security issue exists in the encoding procedure, as figure 3.1 illustrates this problem by displaying how the same letter is encoded using the same value each time. Thus, the matching ciphertext of the encoded value turns into the same encrypted text value each time, allowing the adversary to discriminate between the encrypted texts. And as a result, by evaluating the encrypted texts, the adversary can learn sensitive information about the plaintext [3].

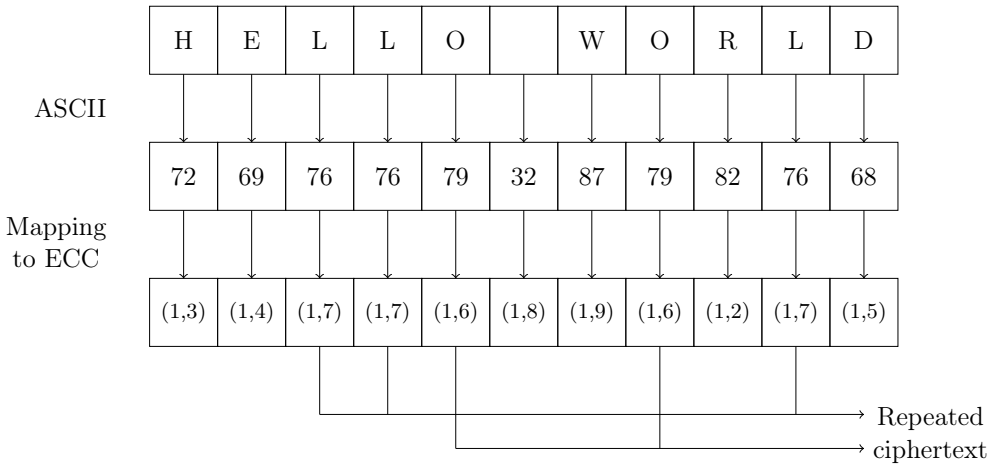


Figure 3.1: Repeated mapped points to the elliptic curve using the ASCII table [3]

3.2.2 Key Generation Phase

During this step, we'll set up a secure communication by creating public and private key pairs for each participant. This operation will also generate a shared group key (gk_{sh}), which will be used to encrypt messages sent within the group.

Shared Group Key K_{sh}

The authors explain how the proposed approach generates the initial cryptographic parameters required to secure communication. This phase is critical since these parameters serve as the foundation for the encryption and decryption operations, which ensure the scheme's security.

For group communication in IoT networks with multiple devices, the approach produces a shared group key k_{sh} to encrypt communications between group members. The group key enables secure communication between everyone in the group. The group key is derived from an initial hash of the device's ID XORed with a private random key PRK used to further secure the group key [3].

$$k_{sh} = H1(\text{id}) \oplus \text{PRK}$$

Where, $H1$ is a hash function applied to the device's ID to produce a secure, unique value.

Shared Group Point gk_{sh}

The group key k_{sh} is updated whenever a new node joins or leaves the group, ensuring that new nodes cannot decrypt previous messages sent before they joined the group. As well as ensuring that nodes leaving the group cannot decrypt future messages. Once the shared group key k_{sh} is generated, it is used to compute a shared group point gk_{sh} on the elliptic curve:

$$gk_{sh} = k_{sh} \times G$$

The point gk_{sh} is utilized during the encryption and decryption operations. It is a common point shared by all members of the group and plays an important function in ensuring communication between the devices. The multiplication of the group key by the base point G ensures that the resultant shared group point is securely linked to the elliptic curve parameters. The shared group point is also computed using the updated group key [3].

3.2.3 Encoding and Mapping Phase

the proposed scheme provides an improved encoding process to overcome the security flaws by dividing the plaintext into manageable blocks that can be securely processed. The size of

each block N is determined by the field size p of the elliptic curve, which is typically a 192-bit prime number for security. To accommodate the padding bits necessary for mapping, the block size is calculated using the formula:

$$N \leq \left\lfloor \frac{p-8}{8} \right\rfloor$$

For a 192-bit prime field, this calculation results in a block size of $N \leq 23$, meaning each block can hold up to 23 characters. The total number of blocks B needed to encode the entire plaintext message is calculated by dividing the message length M by the block size N:

$$B = \left\lceil \frac{M}{N} \right\rceil$$

This ensures that the plaintext is divided into appropriately sized blocks, with the last block potentially smaller if the message length is not a perfect multiple of the block size. Then, after dividing the plaintext into blocks, each block is encoded to prepare it for the mapping phase, where Each character in a block is converted into its corresponding ASCII value, which is an 8-bit numerical representation [3]. The algorithm 3.2.3 summarizes the procedures required in transforming plain text into a group of blocks.

Algorithm 1 Converting a plain text into a set of blocks [3]

Input: The plaintext M and p
Output: Set of blocks
  obtain the plaintext M;
  calculate the size of each block;
  $N \leftarrow \left\lceil \frac{192-8}{p} \right\rceil$;
  calculate the number of blocks;
  $B \leftarrow \left\lceil \frac{|M|}{N} \right\rceil$;
  divide the plaintext into B blocks of size N;
  for $i \leftarrow 1$ to B **do**
    for $j \leftarrow 1$ to N **do**
      Sender: obtain the $B_i = B_i + ASCII(C_{j+((i-1)*N)})$;
    end for
  end for
  Set of blocks $\leftarrow$ the results

To secure the blocks against encryption attacks like Chosen Plaintext Attack (CPA) and Chosen Ciphertext Attack (CCA), the scheme uses Cipher Block Chaining (CBC). In CBC, the first block B_1 is XORed with an initial vector (InV) [3]:

$$B'_1 = B_1 \oplus \text{InV}$$

The subsequent blocks are XORed with the previous block's ciphertext:

$$B'_i = B_i \oplus B'_{i-1}, \text{for } i > 1$$

This chaining process ensures that identical plaintext blocks will produce different encoded outputs, which enhances the security and make them ready to be mapped onto the elliptic curve to prepare them for encryption, where each encoded block B'_i is treated as a numerical value x_i, which needs to be mapped to a point (x_i, y_i) on the elliptic curve. The goal is to find a corresponding y_i such that (x_i, y_i) satisfies the elliptic curve equation. If no such y_i exists for the initial x_i, the value of x_i is incremented by 1 until a valid y_i is found [3]. The algorithm 2 shows the steps involved in mapping.

Algorithm 2 Mapping the blocks to the elliptic curve. [3]

Input: Block B
Output: Mapped points MP
  obtain the decimal value x_i of the secured block
  $x_i \leftarrow x_i \times 16$
  while ${y_i}^2 \not\equiv {x_i}^3 + ax_i + b \pmod{p}$ **do**
    find corresponding y_i such that ${y_i}^2 \equiv {x_i}^3 + ax_i + b \pmod{p}$
    $x_i \leftarrow x_i + 1$
  end while
  return $MP \leftarrow (x_i, y_i)$

3.2.4 Encryption Phase

The encrypted point is generated by adding the mapped point (x_i, y_i) to a shared group point gk_{sh}, which is derived from the shared group key k_{sh}:

$$C_i = (x'_i, y'_i) = (x_i, y_i) + gk_{sh}$$

Here, $gk_{sh} = k_{sh} \cdot G$ is a common point shared among the group members. The addition of the mapped point and the group point is performed using elliptic curve point addition, which ensures that the ciphertext is securely masked by the shared group key [3].

3.2.5 Decryption Phase

Upon receiving the encrypted message, the recipient first verifies the signature to ensure the integrity of the message. If the signature is valid, the recipient decrypts the ciphertext by subtracting the shared group point gk_{sh} from the encrypted point:

$$(x_i, y_i) = (x'_i, y'_i) - gk_{sh}$$

This operation uses the inverse of the point addition performed during encryption. The result is the original mapped point on the elliptic curve [3].

3.2.6 Decoding Phase

The decrypted points are then converted back into their numerical values, and the CBC operation is reversed to reconstruct the original plaintext blocks. The inverse XOR operation is applied using the same initial vector (InV) and the previously decoded ciphertext blocks:

$$B_1 = B'_1 \oplus \text{InV}$$

For subsequent blocks:

$$B_i = B'_i \oplus C_{i-1}, \text{for } i > 1$$

Here, C_{i-1} is the ciphertext of the previous block, These steps undo the chaining effect of CBC, restoring the original numerical values of the plaintext blocks. Then each of these decoded numerical values are converted from its numerical (binary) form into its corresponding ASCII character, this process is repeated for each block until all values are converted back into their corresponding characters.

Finally, the individual characters obtained from the decoding process are combined to reconstruct the original plaintext message. The blocks are concatenated in the order they were processed, ensuring that the original sequence of characters is maintained. This step, results in the complete recovery of the plaintext message [3]. Algorithm 3.2.6 shows the steps required.

Algorithm 3 Converting binary values into plain text [3]

Input: Binary values, B
Output: The plaintext message M
  Get the binary values;
  for $i \leftarrow 0$ to $|B| - 1$ **do**
    convert each 8 bits into its corresponding ASCII code;
    for each N char aggregate to single block;
  end for
  repeat until finish;
  return The message M

3.2.7 Message Integrity and Authenticity

To ensure message integrity and authenticity, the encrypted text is signed with the Elliptic Curve Digital Signature Algorithm (ECDSA), which is commonly used in cryptographic systems to sign and verify messages. This technique ensures that the recipient may verify that the communication has not been tampered with and is actually from the legitimate sender. The process consists of two parts: signing the encrypted message and validating the signature upon reception. This technique provides good security against usual threats while remaining efficient enough for employment in constrained situations such as the Internet of Things [3].

3.3 Elliptic Curve Digital Signature Algorithm ECDSA

The Elliptic Curve Digital Signature Algorithm (ECDSA) is a public-key cryptographic algorithm that is widely used for digital signatures. It leverages the mathematical properties of elliptic curves over finite fields to provide a high level of security with relatively small key sizes compared to other algorithms such as RSA. ECDSA is employed in various applications including secure communications, authentication, and integrity verification[39][5].

The following are the procedures that make up the ECDSA process :

Algorithm 4 ECDSA Key Generation [39]

Input: Domain parameters (p, a, b, G, n, h)
Output: Private key d, Public key Q
Choose a random integer d such that $1 \leq d \leq n-1$, where n is the order of $E(\mathbb{F}_p)$
Compute $Q = dG$ on the elliptic curve $E(F_p)$
return d as the private key and Q as the public key

Signature Generation

Algorithm 5 ECDSA Signature Algorithm [14]

Input: a private key a, a message m
Output: (r, s) the signature associated with m

$H(m) \leftarrow \text{hash}(m)$
repeat
  repeat
    $k \leftarrow$ random integer between 1 and $n-1$
    $(x, y) \leftarrow kP$
  until $x \neq 0$
  $r \leftarrow x \mod n$
  $s \leftarrow k^{-1}(H(m) + ar) \mod n$
until $s \neq 0$

return (r, s)

Signature Verification

Algorithm 6 ECDSA Verification Algorithm [14]

Input: a message m, the signature (r, s), the public key Q
Output: Returns true if the signature is correct, false otherwise

$H(m) \leftarrow \text{hash}(m)$
$u \leftarrow H(m)s^{-1} \mod n$
$v \leftarrow rs^{-1} \mod n$
$(x, y) \leftarrow uP + vQ$
if $x \neq r$ **then**
  return false
end if
return true

ECDSA's security relies on the difficulty of the Elliptic Curve Discrete Logarithm Problem (ECDLP). The key sizes in ECDSA are significantly smaller compared to RSA for a comparable level of security. For instance, a 256-bit key in ECDSA is considered to provide a security level comparable to a 3072-bit key in RSA.

The Elliptic Curve Digital Signature Algorithm provides a robust, efficient method for generating and verifying digital signatures. Its use of elliptic curve cryptography offers a high degree of security with lower computational overhead, making it a preferred choice in many modern cryptographic applications[39].

3.4 Rivest-Shamir-Adleman (RSA)

In this study, we will compare the performance and security features of Elliptic Curve Cryptography (ECC) to RSA. To make this comparison easier, the essential working principles of the RSA algorithm will be covered in the following sections.

The RSA algorithm was developed by Ron Rivest, Adi Shamir, and Len Adleman in 1977 [35]. is a public-key cryptosystem, one of the oldest widely used for secure data transmission. It provides a technique through which encryption keys can be securely conveyed over a non-secure channel between communicating parties without the need to distribute a secret key, hence overcoming one of the major problems in cryptographic communication[35].

The RSA algorithm is used both for public key encryption and for digital signatures. Its security is based on the computational difficulty of factoring large integers, which most would say is infeasible for strong encryption.

To be said in more practical terms, RSA makes it possible for party A to send an encrypted message to party B without having a prior exchange of secret keys. The message will be encrypted with the public key of party B, but it can only be decrypted with his private key; the latter is unknown to anyone else but party B. The other feature also includes digital signatures: Party A, for instance, signs a message with his or her private key, and therefore Party B will be able to check the message by use of the public key from Party A. This kind of double ability is what made RSA so flexible and important in electronic transmissions[5][33].

3.4.1 RSA Algorithm

Key Generation

The first step in RSA is the generation of a key pair, which involves the following steps[33][35]:

Algorithm 7 RSA Key Generation [35]

Input: Key Size k

Output: Public key (n, e), Private key (n, e, d)

Generate two large random prime numbers p and q of size $k/2$ bits each
Compute $n = p \cdot q$
Compute $\phi(n) = (p-1)(q-1)$
Choose a random integer e such that $1 < e < \phi(n)$ and $\gcd(e, \phi(n)) = 1$
Compute d such that $d \equiv e^{-1} \pmod{\phi(n)}$ using the Extended Euclidean Algorithm.
return (n, e) as the public key and (n, e, d) as the private key

Common choices for e include 3, 17, and 65537, as they balance security and computational efficiency. The public key consists of the pair (n, e), and the private key consists of the pair (n, d). The public key can be openly shared, while the private key must remain confidential[39][35].

Encryption

To encrypt a message M using the recipient's public key (n, e) [33][35]:

Algorithm 8 RSA Encryption [33]

Input: Plaintext message M, Public key (n, e)
Output: Ciphertext c
Convert M to an integer m such that $0 \leq m < n$
Compute $c = m^e \mod n$
return c

Decryption

To decrypt the ciphertext c using the private key (n, d) [33][35]:

Algorithm 9 RSA Decryption [33]

Input: Ciphertext c, Private key (n, d)
Output: Plaintext message M
Compute $m = c^d \mod n$
Convert m to the plaintext message M
return M

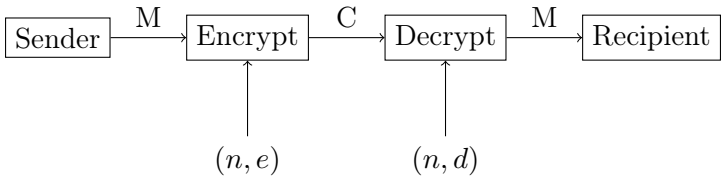


Figure 3.2: RSA algorithm structure

Signature Generation

Creating a digital signature involves using the sender's private key to sign a message. The process is as follows[5][35] :

Algorithm 10 RSA Signature Generation [5]

Input: Message M, Private key (n, d)
Output: Signature S
  Compute $H(M)$ using a cryptographic hash function
  Compute $S = H(M)^d \mod n$
  return S

The digital signature S is sent along with the original message M to the recipient.

Signature Verification

The recipient can verify the digital signature using the sender's public key. The verification process is as follows[5][35] :

Algorithm 11 RSA Signature Verification [5]

Input: Message M, Signature S, Public key (n, e)
Output: Boolean (true if valid, false otherwise)
  Compute $H'(M)$ using the same hash function used for signature generation
  Compute $H(M) = S^e \mod n$
  if $H(M) = H'(M)$ **then**
    return true
  else
    return false
  end if

3.4.2 Security Considerations

The RSA algorithm's security heavily depends on the key length and the secure generation of prime numbers. Longer keys provide greater security but require more computational resources for encryption and decryption. Current standards recommend using key lengths of at least 2048 bits, with 3072 bits or more advised for long-term security, especially in sensitive applications. The primes p and q must be chosen randomly and independently to avoid predictable patterns, as insecure prime generation can lead to the modulus n being susceptible to factorization attacks.

RSA's security also relies on resistance to various cryptographic attacks. Factorization attacks exploit the difficulty of factorizing the modulus n. Continuous advancements in factorization techniques necessitate periodic re-evaluation of key lengths. Timing attacks can occur when attackers gain information from the time taken to perform cryptographic operations, countered by implementing constant-time algorithms and blinding techniques. Chosen ciphertext attacks involve attackers providing selected ciphertexts and analyzing the decrypted outputs; these can be mitigated by using proper padding schemes[33][35].

3.5 Conclusion

In this chapter, we explored the fundamental principles and cryptographic schemes of both Elliptic Curve Cryptography (ECC) and RSA. We began by discussing the theoretical underpinnings of ECC, its efficient key generation, and its application in various cryptographic schemes. Particular attention was given to the Almajed et al. ECC encryption scheme, which addresses security vulnerabilities like Chosen Plaintext Attacks (CPA) and Chosen Ciphertext Attacks (CCA) through innovative encoding and mapping techniques. ECC's strength lies in its ability to provide robust security with smaller key sizes, which translates into faster computations and reduced resource consumption, which makes it suitable for modern, resource-constrained environments.

Following that, we looked at the RSA algorithm, explaining the stages involved in key generation, encryption, decryption, and digital signatures. The RSA algorithm's security is based on the difficulty of factorizing large integers computationally, which supports the security of digital signature and encryption procedures. Even while RSA is still widely used and trusted, maintaining its security requires adhering to best practices in key management and implementation.

This chapter established the foundation for understanding the operational mechanics of ECC and RSA. In the following chapter, we will provide a full comparison of the Almajed et al.'s proposed ECC-based scheme [3] and the RSA system, analyzing their performance, security features, and appropriateness for different uses.

Chapter 4

Comparative Study

4.1 Introduction

ECC and RSA are two popular public-key cryptosystems. ECC provides comparable security to RSA but with smaller key sizes, making it more efficient in contexts with limited resources, such as mobile IoT systems. Evaluating and comparing various algorithms, especially on resource-constrained devices such as the Raspberry Pi Zero W, is critical in determining which gives superior performance and energy efficiency, leading to thesselection of secure, efficient implementations.

The Raspberry Pi Zero W, with its low processing power, memory, and energy resources, belongs to a family of devices commonly employed in IoT and embedded systems where performance is essential. By evaluating how these algorithms function on such a platform, the study emphasizes their applicability for safe communication in low-power, resource-constrained conditions, which can help guide decisions in industries such as IoT security and edge computing.

4.2 Hardware and Software Setup

The Raspberry Pi Zero W was chosen as the test device for analyzing the ECC and RSA algorithms because of its simple hardware specifications, which are similar to those present in many low-power, resource-constrained systems. The Raspberry Pi Zero W has a 1 GHz single-core ARM11 processor, 512 MB of RAM, and integrated wireless networking (Wi-Fi and Bluetooth).

Raspbian Lite, a lightweight version of the Raspberry Pi OS (based on Debian), was used in the evaluation. It is specifically built for headless and minimalistic systems. Raspbian Lite provides a simplified environment with minimum overhead, letting the focus be on the execution of cryptographic algorithms rather than superfluous system activities consuming resources.

However, to provide a comparison, we also evaluated the algorithms on a more powerful system: a PC equipped with an Intel(R) Core(TM) i5-5300U CPU @ 2.30GHz, 3 MiB of L3 cache, and 8 GB of DDR3L SDRAM. The PC ran a modern Linux distribution with optimized performance characteristics with a kernel version of 6.1.99-1.

The UM25C USB power meter was used to measure energy usage while the ECC and RSA algorithms were being executed. This device was connected between the Raspberry Pi Zero W and its power supply, allowing for precise monitoring of energy usage throughout the testing procedure.

4.3 Algorithms Implementation

The Almajed et al. ECC Encryption Proposed Scheme and RSA algorithms were implemented in Python 3, using its standard libraries as well as the gmpy2 package for efficient computation of modular exponentiation, an essential function in both cryptosystems.

4.3.1 RSA Implementation

The RSA algorithm generates keys by selecting two large prime numbers and multiplying them to get the modulus. The public and private keys were generated with Python 3. The encryption and decryption procedures were carried out using modular exponentiation, which is critical to RSA's security and efficiency. To optimize these calculations, the gmpy2 library was used, notably its powmod function. This function efficiently computes the power of an integer modulo another integer, which enabled for quicker execution.

4.3.2 ECC Implementation

The ECC algorithm used is Almajed et al. Proposed Scheme also implemented using the Python 3. Key generation required choosing a random private key and producing the associated public key via elliptic curve point multiplication. The cryptographic procedures used elliptic curve point arithmetic. ECC's modular arithmetic requires efficient handling of large integer operations, hence the gmpy2 library was once again used.

Both algorithms were developed in packages, allowing for easy testing and comparison. The usage of Python's standard libraries, together with gmpy2, meant that the implementations were simple and optimized for the Raspberry Pi Zero W's low processing resources. The Python code remained straightforward and manageable, while the gmpy2 library improved cryptographic operations' efficiency, particularly when dealing with large integers and modular arithmetic.

4.4 Performance Metrics

The ECC and RSA algorithms on the Raspberry Pi Zero W were evaluated on a well-defined set of parameters to ensure an accurate assessment of their performance. The key sizes used for RSA were 1024, 2048, 3072, and 7680 bits, representing a range of security levels from the bare minimum required in modern applications to highly secure settings. The key sizes assessed for ECC were 192, 224, 256, and 384 bits, which corresponded to identical security levels as RSA key sizes but with much shorter key lengths, demonstrating ECC's efficiency in key management.

Data sizes ranging from one to eight bytes were utilized to evaluate the encryption and decryption performance of each algorithm. This range of data lengths represents common small-scale encryption jobs found in IoT devices and other limited situations. By examining such small data blocks, we can better understand how cryptographic operations affect the performance of limited-resource devices.

The evaluation metrics covered execution time, memory usage, and energy consumption, all of which are crucial in resource-constrained contexts. Each algorithm's key generation, encryption, decryption, signature generation, and signature verification operations were timed in seconds to establish their speed. Memory use was measured in bytes to better understand the memory footprint of these processes, which is especially important for devices with low RAM. Energy consumption was measured in milliwatt-hours (mWh) using the UM25C USB power meter, providing information about the algorithms' power efficiency throughout each cryptographic operation. The study's goal was to discover the best efficient method for usage in low-power, limited contexts by comparing these parameters across various key and data sizes.

4.5 Memory Usage

This section presents the memory required for both algorithms. Tables 4.1 and 4.2 show the obtained results.

Table 4.1: Memory Usage on Raspberry Pi Zero W

Security Level	Key size		Memory Usage (bytes)	
	RSA	ECC	RSA	ECC
80	1024	192	16384	1508
112	2048	224	16406	2109
128	3072	256	17537	2889
192	7680	384	22687	6436

Table 4.2: Memory Usage on a PC

Security Level	Key size		Memory Usage (bytes)	
	RSA	ECC	RSA	ECC
80	1024	192	25664	3280
112	2048	224	25648	3280
128	3072	256	25738	3524
192	7680	384	32328	7160

The memory usage of RSA and ECC at various security levels is compared in these two tables. The data across both tables are consistent, for all security levels, RSA consistently needs a lot significantly more RAM than ECC which is more memory-efficient. The consistent pattern across both tables indicates that ECC is far more efficient in terms of memory usage compared to RSA, making it a better choice for applications where memory resources are limited. In conclusion, ECC is more memory efficient than RSA, using less memory while preserving the same level of security.

4.6 Key Generation

Key generation is a fundamental component of an algorithm. It is used to generate keys which require different times in each algorithm. The time and energy required to create keys varies between RSA and ECC. Table 4.3 highlights the differences between both algorithm's execution time.

Table 4.3: Key generation and execution time on Raspberry Pi Zero W

Security Level	Key size		Key Generation Time (s)	
	RSA	ECC	RSA	ECC
80	1024	192	1.098510	0.150403
112	2048	224	12.771519	0.208352
128	3072	256	61.111414	0.276518
192	7680	384	2987.453059	0.675820

In terms of key generation time, RSA is noticeably slower than ECC. For example, at the 128-bit security level, RSA generates a key in over 61 seconds, whereas ECC does so in just 0.28 seconds. Key generation time for RSA grows drastically as the security level rises. ECC's time increases at a far slower rate, demonstrating its scalability and efficiency.

Table 4.4: Key generation and execution time on PC

Security Level	Key size		Key Generation Time (s)	
	RSA	ECC	RSA	ECC
80	1024	192	0.034294	0.017103
112	2048	224	0.574876	0.023364
128	3072	256	1.810646	0.030961
192	7680	384	63.412192	0.129966

Here's the evaluation, but on a PC with more processing power. RSA and ECC key creation times are significantly less than on the Raspberry Pi. However, the pattern stays consistent: RSA times increase exponentially with security level, whereas ECC remains efficient, with the longest key generation time of approximately 0.13 seconds at the 192-bit level. The PC's results underscores ECC's efficiency, as even on a more powerful system, ECC outperforms RSA in terms of speed.

While the table 4.5 shows the energy consumption of key generation for RSA and ECC at various security levels, providing information on the efficiency of each algorithm.

Table 4.5: Key generation and Energy consumption on Raspberry Pi Zero W

Security Level	Key size		Energy consumption (mWh)	
	RSA	ECC	RSA	ECC
80	1024	192	0.3	0.1
112	2048	224	4.9	0.3
128	3072	256	27.9	0.3
192	7680	384	833.7	0.6

RSA's energy consumption rise significantly as security levels increase. ECC is significantly more energy efficient than RSA. Even at a high security level (192-bit), ECC consumes only 0.6 mWh, just a tiny percentage of RSA's consumption at the same level (833.7 mWh).

The data clearly suggests ECC outperforms RSA in terms of key generation efficiency, particularly as security demands rise. This efficiency, combined with lower key sizes, makes ECC an ideal solution for contexts with limited processing power, time, and energy.

4.7 Encryption and Decryption

This section provides a comparison of RSA and ECC in terms of the encryption and decryption, highlighting the strengths and weaknesses of each algorithm in practical applications.

The encryption and decryption timings for RSA and ECC at different security levels are highlighted in the table 4.6.

Table 4.6: Encryption/Decryption and Execution Time on Raspberry Pi Zero W

Security Level	Key size		Encryption / Decryption (s)			Total Time	
	RSA	ECC	Remark	RSA	ECC	RSA	ECC
80	1024	192	Encryption	0.000300	0.305892	0.013259	0.458244
			Decryption	0.012959	0.152352		
112	2048	224	Encryption	0.000839	0.424339	0.092607	0.632645
			Decryption	0.091768	0.208306		
128	3072	256	Encryption	0.001650	0.660138	0.181830	0.936793
			Decryption	0.180180	0.276655		
192	7680	384	Encryption	0.011106	3.197424	1.786930	4.808451
			Decryption	1.775824	1.611027		

In terms of performance, RSA has remarkably fast encryption times throughout all security levels, beginning at 0.0003 seconds at the 80-bit level and growing only slightly to 0.0111 seconds at the 192-bit level. However, RSA's decryption times are much longer, ranging from 0.0130 seconds at the 80-bit level to 1.7758 seconds at the 192-bit level, demonstrating that, while RSA is designed for quick encryption, it is relatively slow in decryption. On the other hand, ECC has substantially longer encryption periods than RSA, especially at higher security levels, with encryption taking up to 3.1974 seconds at the 192-bit level. ECC's decryption time is also longer than RSA's, but the disparity between encryption and decryption times is less obvious, suggesting more balanced performance between the two operations.

Table 4.7: Encryption/Decryption and Execution Time on PC

Security Level	Key size		Encryption / Decryption (s)			Total Time	
	RSA	ECC	Remark	RSA	ECC	RSA	ECC
80	1024	192	Encryption	3.993419e-05	0.039176	6.1893419e-4	0.058917
			Decryption	0.000579	0.019741		
112	2048	224	Encryption	5.257749e-05	0.053570	0.003682	0.079695
			Decryption	0.003629	0.026125		
128	3072	256	Encryption	0.000106	0.070746	0.011026	0.105319
			Decryption	0.010920	0.034573		
192	7680	384	Encryption	0.000269	0.176526	0.118104	0.268639
			Decryption	0.117835	0.092113		

The second table 4.7 illustrates the performance of these cryptosystems on a PC, where encryption and decryption times are greatly decreased due to increased processing power. ECC remains slower than RSA during encryption. RSA decryption times continue to outperform those of ECC.

The comparative analysis of energy consumption related to RSA and ECC shows a considerable increase in energy requirements as security levels rise. as the table 4.8 indicates.

Table 4.8: Encryption/Decryption and Energy consumption on Raspberry Pi Zero W

Security Level	Key size		Encryption / Decryption (mWh)			Total Energy	
	RSA	ECC	Remark	RSA	ECC	RSA	ECC
80	1024	192	Encryption	0.0	0.3	0.1	0.6
			Decryption	0.1	0.3		
112	2048	224	Encryption	0.0	0.3	0.1	0.6
			Decryption	0.1	0.3		
128	3072	256	Encryption	0.0	0.4	0.1	0.8
			Decryption	0.1	0.4		
192	7680	384	Encryption	0.1	1.0	0.8	1.5
			Decryption	0.7	0.5		

The energy consumption results show that RSA can be more energy-efficient at lower security levels, consuming only 0.1 mWh total for the 80, 112, and 128-bit security levels. However, at a high security level of 192-bit, RSA's energy usage jumps to 0.8 mWh, suggesting a significant increase in energy costs as security level increases.

In comparison, ECC constantly consumes more energy than RSA at all security levels. ECC consumes 0.6 mWh more energy than RSA as shown in table 4.8. This pattern continues at the 128-bit level, when ECC consumes 0.8 mWh, and at the 192-bit level, where consumption of energy rises to 1.5 mWh. ECC's energy use for encryption and decryption is more balanced than RSA's, although the overall higher energy usage may be a disadvantage in energy-sensitive applications.

Overall, the data show that RSA is more effective and demonstrates better energy efficiency for applications that need quick encryption, but at the cost of longer decryption, its scalability is limited due to the significant rise at higher security levels, notably for decryption operations. On the other hand, ECC, while slower overall and less energy-efficient, provides more consistent efficiency between encryption and decryption, which may be useful in cases where balanced execution time is important.

4.8 Signature Generation and Signature Verification

This section discusses the performance distinctions between ECC and RSA in terms of digital signature generation and verification. This comparison examines how each algorithm operates in various cryptographic operations using key parameters such as execution time and energy efficiency. The following tables illustrate these characteristics, providing a clear picture of ECC and RSA's strengths and limitations in the practice of signature-related tasks.

Table 4.9: Signature and Execution Time on Raspberry Pi Zero W

Security Level	Key size		Signature Generation Time (s)		Signature Verification Time (s)	
	RSA	ECC	RSA	ECC	RSA	ECC
80	1024	192	0.013063	0.151251	0.000286	0.303269
112	2048	224	0.091786	0.211913	0.000854	0.420587
128	3072	256	0.280127	0.278196	0.001693	0.553496
192	7680	384	2.787939	0.668610	0.069761	1.355559

The table 4.9 compares the signature generation and verification times of RSA and ECC. At lower security levels (80, 112, and 128 bits), RSA generally outperforms ECC in terms of signature creation. However, at the level (192 bits), RSA takes much longer (2.79 seconds) than ECC (0.67 seconds). This shows that, while RSA is more efficient for producing signatures at lower security levels, ECC scales better as security requirements increase.

RSA consistently outperforms ECC in signature verification across all security levels, with verification times of less than 0.07 seconds. In contrast, the verification time of ECC grows with greater security levels. This suggests that, while ECC performs well in signature generation, particularly at higher security levels, it is less efficient than RSA in signature verification, which may limit its appropriateness for applications requiring speedy verification.

Table 4.10: Signature and Execution Time on PC

Security Level	Key size		Signature Generation Time (s)		Signature Verification Time (s)	
	RSA	ECC	RSA	ECC	RSA	ECC
80	1024	192	0.000575	0.019357	1.750420e-05	0.033030
112	2048	224	0.003595	0.026945	3.790260e-05	0.043912
128	3072	256	0.011580	0.037491	7.701340e-05	0.059208
192	7680	384	0.104037	0.074392	0.000264	0.146009

The same trend continues, and the table 4.10 compares the signature generation and verification times for RSA and ECC algorithms at different security levels on a PC. It shows that RSA is consistently faster than ECC in both signature generation and verification across all security levels.

Table 4.11: Signature and Energy consumption on Raspberry Pi Zero W

Security Level	Key size		Signature Generation Energy (mWh)		Signature Verification Energy (mWh)	
	RSA	ECC	RSA	ECC	RSA	ECC
80	1024	192	0.0	0.3	0.0	0.2
112	2048	224	0.1	0.1	0.1	0.4
128	3072	256	0.5	0.3	0.5	0.4
192	7680	384	1.0	0.5	1.7	1.0

The table 4.11 compares energy consumption for RSA and ECC during signature generation and verification. At lower security levels (80 and 112 bits), both algorithms consume very little energy, with ECC spending slightly more than RSA in some circumstances. However, when the level of security improves, so does the energy consumption of both methods, with RSA demonstrating the greater increase. For example, at the 192-bit security level, RSA uses 1.0 mWh for signature generation and 1.7 mWh for verification, whereas ECC uses 0.5 mWh and 1.0 mWh, respectively. This suggests that ECC is more energy-efficient than RSA at higher security levels, making it a superior option for situations where energy consumption is an important consideration.

4.9 Results Interpretation

The study conducted on ECC and RSA, shows interesting results that highlight the relative advantages and disadvantages of these two cryptosystems across a range of performance

metrics, especially in resource-constrained situations. The outcomes clearly demonstrate that ECC performs better than RSA in several of crucial areas, most notably memory use, key generation time, and energy efficiency. These benefits result from the basic distinctions between the two algorithms' modes of operation.

4.9.1 Key Size and Computational Efficiency

ECC uses substantially smaller key sizes than RSA to achieve the same degree of security. For example, the security of a 3072-bit RSA key is about equal to that of a 256-bit ECC key. Faster computing times, less memory use, and lower energy consumption are the outcomes of this smaller key size. When it comes to the key generation process, ECC is much more efficient than RSA in terms of time and energy required. Which enables more secure cryptographic operations to be performed with less processing power.

4.9.2 Encryption and Decryption

Because RSA was intended to be simple, it frequently performs faster than ECC when it comes to encryption, especially at lower security settings. This is explained by the fact that RSA relies on less complicated arithmetic operations rather than the complex elliptic curve point arithmetic that ECC requires.

4.9.3 Energy Consumption

The findings show that ECC consumes less energy compared to RSA during key generation and signature generation processes, making it more suited for low-power systems. However, during encryption and decryption, RSA uses less energy at lower security levels, demonstrating its efficiency in specific scenarios. The more sophisticated arithmetic operations used in elliptic curve cryptography account for ECC's increased energy usage during encryption and decoding.

4.10 Conclusion

The comparison analysis of ECC and RSA across several performance metrics reveals ECC's advantages in limited situations, particularly in terms of memory utilization, key generation efficiency, and energy consumption. ECC's scalability in resource-constrained situations is partly due to its smaller key sizes and lower consumption of resources, making it particularly efficient on platforms like the Raspberry Pi Zero W.

ECC's scalability makes it a better fit for IoT devices, edge computing, and other applications with constrained resources. While RSA may still be advantageous in certain cases, notably those requiring quick encryption at low security levels, ECC's overall performance,

particularly in high-security uses, puts it as the more versatile and feasible choice in modern cryptography. For applications that require the highest levels of security, ECC provides a distinct benefit. As the desired security level rises, RSA key sizes become impractically huge, resulting in significant costs in terms of memory usage, computation time, and energy consumption. However, ECC scales more smoothly, delivering excellent security without overburdening system resources.

Conclusion

The Internet of Things (IoT) is at the cutting edge of technological progress, primed for revolutionary change. However, achieving this potential is intrinsically related to our ability to confront the key challenges which come with rapid growth. One of the most difficult issues is ensuring that the Internet of Things matures into a strong, secure network.

Security emerges as the most pressing worry in the IoT world. The vulnerability of IoT devices, caused by their wireless nature and direct control over physical entities, highlights the critical necessity for strong security measures. As previously noted, the implementation of appropriate security procedures is challenged by the intrinsic limits of IoT devices, such as memory, processing capacity, and energy consumption.

This thesis has explored the fundamental principles and applications of Elliptic Curve Cryptography (ECC) in comparison to traditional public-key cryptosystems, particularly RSA. Through our comprehensive analysis, we have demonstrated the significant advantages that ECC offers in terms of security, efficiency, and suitability for modern cryptographic needs in the IoT systems.

Our research has shown that ECC provides comparable levels of security to RSA while using significantly smaller key sizes. This advantage translates into faster computations, reduced power consumption, and lower memory requirements — crucial factors in resource-constrained environments such as IoT applications. The efficiency of ECC makes it particularly well-suited for the growing demands of secure communications in our increasingly interconnected world.

We paid special attention to the Almajed et al. ECC encryption scheme, which addresses vulnerabilities in the plaintext mapping phase and incorporates authenticated encryption. This innovative approach offers enhanced protection against chosen-plaintext (CPA) and chosen-ciphertext (CCA) attacks, demonstrating the ongoing evolution and improvement of ECC-based systems.

While RSA remains a trusted and widely-implemented algorithm, our analysis suggests that ECC presents a more future-proof solution for many cryptographic applications. As computational power continues to increase and quantum computing looms on the horizon, the inherent strength of ECC against known attacks positions it as a robust choice for long-term security needs.

In conclusion, this thesis underscores the importance of ECC in modern cryptography. Its ability to provide strong security with smaller key sizes and lower computational overhead makes it an ideal choice for a wide range of applications, from secure messaging to digital signatures and key exchange protocols. As we move towards an increasingly digital future, the role of ECC in ensuring the confidentiality, integrity, and authenticity of our communications will only grow in importance.

Future studies in this field should focus on further optimizing ECC implementations, exploring its applications in emerging technologies, and continuing to develop novel ECC-based schemes that can address evolving security challenges. By building on the strong foundation of ECC, we can work towards creating more secure, efficient, and resilient cryptographic solutions for the IoT ecosystem.

Bibliography

[1] Kumar A, Hussain J, and Chun. Iot connectivity considerations. *Connecting the Internet of Things*, 2023.

[2] Zulfiqar Ali, Azhar Mahmood, Shaheen Khatoon, Wajdi Alhakami, Syed Sajid Ullah, Jawaid Iqbal, and Saddam Hussain. A generic internet of things (iot) middleware for smart city applications. *Sustainability*, 2022.

[3] Hisham AlMajed and Ahmad AlMogren. A secure and efficient ecc-based scheme for edge computing and internet of things. *Sensors*, 20(21), 2020.

[4] K Ashton and al. That 'internet of things' thing. *RFID journal*, 22(7):97–114, 2009.

[5] J.P. Aumasson. *Serious Cryptography, 2nd Edition: A Practical Introduction to Modern Encryption*. No Starch Press, 2018.

[6] Ko´sciug B and Bilski P. Energy saving chaotic sequence based encryption, authentication and hashing for m2m communication of iot devices. *International Journal of Electronics and Telecommunications*, 2023.

[7] Daniel J. Bernstein, Tanja Lange, and Reza Rezaeian Farashahi. Binary edwards curves in cryptographic hardware and embedded systems. *10th International Workshop, Washington, D.C., USA*, page 244–265, 2008.

[8] Daniel RL. Brown. Sec 1: Elliptic curve cryptography. *Certicom Research 2*, 2009.

[9] Chetan Chauhan, Manoj Kumar Ramaiya, Anand Singh Rajawat, S B Goyal, Chaman Verma, and Maria Simona Raboaca. Improving iot security using elliptic curve integrated encryption scheme with primary structure-based block chain technology. *Procedia Computer Science*, 215:488–498, 2022.

[10] H. Cohen, G. Frey, R. Avanzi, C. Doche, T. Lange, K. Nguyen, and F. Vercauteren. *Handbook of Elliptic and Hyperelliptic Curve Cryptography*. Discrete Mathematics and Its Applications. CRC Press, 2005.

[11] Sumit Singh Dhanda, Brahmjit Singh, and Poonam Jindal. Lightweight cryptography: A solution to secure iot. *Wireless Personal Communications*, 112:1947–1980, 2020.

[12] Harold Edwards. A normal form for elliptic curves. *Bulletin of the American Mathematical Society*, 44:393–422, 2007.

[13] Mohammad Hammoudeh, Gregory Epiphaniou, Sana Belguith, Devrim Unal, Bamidele Adebisi, Thar Baker, A. S. M. Kayes, Paul A, and Watters. A service-oriented approach for sensing in the internet of things: Intelligent transportation systems and privacy use cases. *IEEE Sensors Journal*, 2021.

[14] D. Hankerson, A.J. Menezes, and S. Vanstone. *Guide to Elliptic Curve Cryptography*. Springer Professional Computing. Springer New York, 2004.

[15] Cecilie Holländer-Mieritz, Christoffer Johansen, and Helle Pappot. ehealth-mind the gap. *Acta Oncologica*, 2020.

[16] Kuzminykh Ievgeniia, Yevdokymenko Maryna, and Sokolov Volodymyr. Encryption algorithms in iot: Security vs lifetime how long the device will live? 2021.

[17] Ayuso J, Marin L, Jara A, and Skarmeta A. Optimization of public key cryptography (rsa and ecc) for 16-bits devices basedon 6lowpan. *In Proceedings of 1st Int. Work. Secure. Internet Things*, 2010.

[18] B Dorsemaine J, P. Gaulier J, P. Wary, N Kheir, and P. Urien. Internet of things: A definition & taxonomy. *2015 9th International Conference on Next Generation Mobile Applications, Services and Technologies, Cambridge, UK*, pages 72–77, 2015.

[19] Furqan Jameel, Zheng Chang, and Riku Jantti. Secrecy limits of energy harvesting iot networks under channel imperfections. *2020 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 1–6, 2020.

[20] Ding Jie, Nemati Mahyar, Ranaweera Chathurika, and Choi Jinho. Iot connectivity technologies and applications: A survey. *IEEE Access*, pages 1–1, 2020.

[21] Yadav Jyoti, Bhatia Anshul, Sangeeta, Jain Eisha, and Goyal Nidhi. Internet of things (iot): Confronts and applications. *International Journal for Research in Applied Science & Engineering Technology (IJRASET)*, 5:1226–1231, 2017.

[22] Jane K, Hart, and Kirk Martinez. Toward an environmental internet of things. *Earth and Space Science*, 2015.

[23] Tabassum K, Ibrahim A, and El Rahman S.A. Security issues and challenges in iot. *2019 International Conference on Computer and Information Sciences (ICCIS)*, 19(2), 2019.

[24] Kwang Ho Kim, Chol Ok Lee, and Christophe Negre. Binary edwards curves revisited. in progress in cryptology. *15th International Conference on Cryptology in India, New Delhi, India*, page 393–408, 2014.

[25] Andreas Korte, Victor Tiberius, and Alexander Brem. Internet of things (iot) technology research in business and management literature: Results from a co-citation analysis. *Journal of Theoretical and Applied Electronic Commerce Research*, 2021.

[26] Brian Koziel, Reza Azarderakhsh, and Mehran Mozaffari Kermani. Low-resource and fast binary edwards curves cryptography. in progress in cryptology - indocrypt 2015. *16th International Conference on Cryptology in India, Bangalore, India*, page 347–369, 2015.

[27] Sachin Kumar, Prayag Tiwari, and Mikhail L. Zymbler. Internet of things is a revolutionary approach for future technology enhancement: a review. *Journal of Big Data*, 2019.

[28] Antão Liliana, Pinto Rui, Reis João Pedro, and Gonçalves Gil. Requirements for testing and validating the industrial internet of things. *2018 IEEE International Conference on Software Testing, Verification and Validation Workshops*, 2018.

[29] Hughes M.B. Similarity hashing of malware on iot devices. 2019.

[30] S Moyer. Iot sensors and actuators. *IEEE Internet of Things Magazine*, 2(3):10–10, 2019.

[31] Mazieri M.R, Scafuto I.C, and da Costa P.R. Tokenization, blockchain and web 3.0 technologies as research objects in innovation management. *International Journal of Innovation*, 2022.

[32] Bello Oladayo, Zeadally Sherali, and Badra Mohammad. Network layer inter-operation of device-to-device communication technologies in internet of things (iot). *Ad Hoc Networks*, 57, 2016.

[33] C. Paar, J. Pelzl, and T. Güneysu. *Understanding Cryptography: A Textbook for Students and Practitioners*. Information Security and Cryptography Series. Springer, 2010.

[34] Roheen Qamar and Baqar Ali Zardari. An analysis of the internet of everything. *Mesopotamian journal of Cybersecurity*, 2023.

[35] Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[36] Kumar S and Kumar D. A survey of lightweight cryptography for power-constrained iot devices: Security challenges and issues. 2021.

[37] Surendran S, Nassef A, and Beheshti B.D. A survey of cryptographic algorithms for iot devices. *2018 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*, 2018.

[38] Mahyar Shirvanimoghaddam, Kamyar Shirvanimoghaddam, Mohammad Mahdi Abolhasani, Majid Farhangi, Vahid Zahiri Barsari, Hangyue Liu, Mischa Dohler, and Minoo Naebe. Towards a green and self-powered internet of things using piezoelectric energy harvesting. *IEEE Access*, 2019.

[39] Marina Shideroff Svetlin Nakov, Milen Stefanov. *Practical Cryptography for Developers*. 2018.

[40] Vahdati Z, Ghasempour A, Salehi M, and Yasin S.Md. Comparison of ecc and rsa algorithms in iot devices. *J Theor. andAppl.Inf. Tech*, 97(16):4293–4308, 2019.

ملخص

تستكشف هذه الدراسة تطبيق التشفير باستخدام المنحنيات الإهليلجية (ECC) في تأمين الأجهزة خفيفة الوزن ضمن إنترنت الأشياء (IoT). نظرًا للمتطلبات الأمنية المتزايدة في بيئات إنترنت الأشياء، يوفر التشفير باستخدام المنحنيات الإهليلجية حلاً من خلال تقديم تشفير قوي مع تكاليف حسابية واستهلاك طاقة أقل مقارنةً بالخوارزميات التقليدية مثل RSA. تركز الدراسة على مقارنة أداء ECC وRSA، لا سيما من حيث توليد المفاتيح، وعمليات التشفير وفك التشفير، واستهلاك الطاقة باستخدام جهاز Raspberry Pi Zero W. وتظهر الدراسة كفاءة وملاءمة ECC في أنظمة إنترنت الأشياء، مع تسليط الضوء على إمكانياته في تحسين الأمان والأداء في هذه الأنظمة.

الكلمات المفتاحية : إنترنت الأشياء، التشفير غير المتماثل، RSA ،ECC

Resume

Cette étude explore l'application de la cryptographie à courbes elliptiques (ECC) pour sécuriser les dispositifs légers dans l'Internet des objets (IoT). Face aux exigences croissantes en matière de sécurité dans les environnements IoT, l'ECC offre une solution en proposant un chiffrement robuste avec des coûts computationnels et énergétiques inférieurs par rapport aux algorithmes traditionnels comme le RSA. La recherche compare les performances de l'ECC et du RSA, notamment en ce qui concerne la génération de clés, les processus de chiffrement/déchiffrement, ainsi que la consommation d'énergie, en utilisant un Raspberry Pi Zero W. L'étude démontre l'efficacité et l'adéquation de l'ECC dans les systèmes IoT, en soulignant son potentiel à améliorer la sécurité et les performances dans ces environnements.

Mots clés : Internet des objets, Cryptographie asymétrique, ECC, RSA

Abstract

This study explores the application of Elliptic Curve Cryptography (ECC) in securing lightweight devices within the Internet of Things (IoT). Given the increasing security demands in IoT environments, ECC offers a solution by providing strong encryption with lower computational and energy costs compared to traditional algorithms like RSA. The research comparing the performance of ECC and RSA, particularly in terms of key generation, encryption/decryption processes, and energy consumption, using a Raspberry Pi Zero W. The study demonstrates the efficiency and suitability of ECC in IoT systems. Highlighting its potential to improve security and performance in IoT systems.

Keywords: Internet of Things, Asymmetric Cryptography, ECC, RSA