REPUBLIC OF TÜRKİYE

ALTINBAŞ UNIVERSITY

Institute of Graduate Studies

Electrical and Computer Engineering

IMPLEMENTATION OF ELLIPTIC CURVE CRYPTOGRAPHY FOR INTERNET OF THINGS (IOT) SECURITY

Saif Maad Khaleefah KHALEEFAH

Master's Thesis

Supervisor

Assoc. Prof. Dr. Sefer KURNAZ

İstanbul, 2024

IMPLEMENTATION OF ELLIPTIC CURVE CRYPTOGRAPHY FOR INTERNET OF THINGS (IOT) SECURITY

Saif Maad Khaleefah KHALEEFAH

Electrical and Computer Engineering

Master's Thesis

ALTINBAŞ UNIVERSITY

2024

The thesis titled IMPLEMENTATION OF ELLIPTIC CURVE CRYPTOGRAPHY FOR INTERNET OF THINGS (IOT) SECURITY prepared by SAIF MAAD KHALEEFAH KHALEEFAH and submitted on 15/01/2024 has been **accepted unanimously** for the degree of Master of Science in Electrical and Computer Engineering.

Assoc. Prof. Dr. Sefer KURNAZ

the Supervisor

Thesis Defense Committee Members:

Assoc. Prof. Dr. Sefer KURNAZ	Department of Computer Engineering, Altınbaş University	_____
Asst. Prof. Dr. Oğuz KARAN	Department of Software Engineering, Altınbaş University	_____
Asst. Prof. Dr. Serdar KARGIN	Department of Biomedical, Istanbul Arel University	_____

I hereby declare that this thesis meets all format and submission requirements of a Master's thesis.

Submission date of the thesis to Institute of Graduate Studies: ___/___/___

I hereby declare that all information data presented in this graduation project has been obtained in full accordance with academic rules and ethical conduct. I also declare all unoriginal materials and conclusions have been cited in the text and all references mentioned in the Reference List have been cited in the text, and vice versa as required by the abovementioned rules and conduct.

Saif Maad Khaleefah KHALEEFAH

Signature

DEDICATION

I devote and pledge this research work to my supervisor who is salient for guiding me through whole research work as well as my family for always assisting me in my hard time.

PREFACE

First and foremost, I would like to thank my supervisor Assoc. Prof. Dr. Sefer KURNAZ for guiding and helping me along the way in writing this dissertation. Discussing my progress, problems, and ideas with my supervisor Assoc. Prof. Dr. Sefer KURNAZ a couple of times every week helped me tremendously in understanding the logic behind the research. It made me better realize the technical need for this research work.

ABSTRACT

IMPLEMENTATION OF ELLIPTIC CURVE CRYPTOGRAPHY FOR INTERNET OF THINGS (IOT) SECURITY

KHALEEFAH, Saif Maad Khaleefah

M.Sc., Electrical and Computer Engineering, Altınbaş University,

Supervisor: Assoc. Prof. Dr. Sefer KURNAZ

Date: January /2024

Pages: 64

The Internet of Things, or IoT, is essential to the fields of industry, healthcare, and information technology, among others. It is made up of numerous interconnected things that are in communication with one another. Because approved objects, in addition to users, may access data regarding the Internet of Things. For IoT applications and technology to be widely adopted, security is a necessary component. To improve the security of the IoT data, this study suggests an Elliptic Curve Cryptography approach. Two keys are used in Elliptic Curve Cryptography (ECC): a public key and a private key. The user uses the private key for encryption, and the public key is used for user identification during authentication. Similar to this, using the private key, the sender encrypts; if secrecy is desired, using the public key, one can decrypt the communication. Selecting the private key is a problem with every public key. Random selection of small values raises concerns about the overall algorithm's security. considering the values. This study suggests using the Cuckoo Search Algorithm to select values at random.

Keyword: IOT, ECC, ENCODING, CRYPTOGRAPHY, SECURITY.

TABLE OF CONTENTS

Pages

LIST OF TABLES

Pages

LIST OF FIGURES

Pages

ABBREVIATIONS

IoT	:	Internet of Things
ECC	:	Elliptic Curve Cryptography
ECDSA	:	Elliptic Curve Digital Signature Algorithm
ECDLP	:	Elliptic Curve Discrete Logarithm Problem
ECDH	:	Elliptic Curve Diffie Hellman
DSA	:	Digital Signature Algorithm
ACS	:	Asymmetric Crypto-System
GPUs	:	Graphics Processing Units
ANSI	:	American National Standards Institute

1. INTRODUCTION

The networked devices that make up the Internet of Things (IoT) individuals, services, items, and gadgets that may exchange data and information and communicate in order to accomplish shared objectives across various domains and applications. "Internet of Things is a dynamic global network infrastructure with self-configuring capabilities based on standard and interoperable communication protocols where physical and virtual things have identities, physical attributes, and virtual personalities and use intelligent interface and are seamlessly integrated into the information network," according to O. Vermesan and P. Friess (2013), the European Internet of Things Research Cluster (IERC) [1].

IoT can be described as the actual intelligent world items with constrained processing and storage capacity [R. Roman, P. Najera, & J. Lopez, 2011]. [2]. IoT can be used in a variety of fields, including energy production and distribution, transportation, healthcare, agriculture, and many more. These fields require internet-connected devices to interact in order for activities to be completed intelligently with no assistance from others. A Management of Identity (IM) strategy to be used with a group of comparable and diverse devices taking part in Internet of Things activities. Every entity in an IoT region needs a unique identity, and the IP address can serve as the region's definition.

By enabling the intelligent gadgets in our environment to carry out routine jobs and activities with little assistance from humans, the promise of the Internet of Things is to completely change the way we live. Smart cities, smart homes, smart infrastructure, and smart transportation are some of the phrases that are used in relation to the Internet of Things.

Home and commercial environments are just two of the many application domains for IoT [M. Abomhara and G. M. Koien, 2014 [3]. The transportation sector also uses IoT to provide appropriate and safe transportation options through the use of various smart traffic lights, smart roads, and smart automobiles. Applications used in banking, marketing, finance, and other industries to support various organizational tasks are included in the IoT's enterprise and industries area. Farming, breeding, energy conservation, and recycling and other related fields are included in the last application domain, which is the service and utility monitoring sector [M. Abomhara and G. M. Koien, 2014 [3].

Data, gadgets, apps, and consumers linked to the internet have multiplied rapidly. The internet can now be connected to anything under the sun. including wearables, medical

gadgets, mobile devices, and a vast range of other goods and services. Since so many businesses provide internet-connected goods and services, it is imperative that similar effort be put into preventing illegal access and data breaches via Internet of Things devices

Put simply, there is a chance that hostile actors will acquire access of internet-connected controls while using an IoT device. Ensuring the security of IoT devices is just as crucial as enabling their use, as there is a chance of customer data or private system access being compromised.

The information security community is well aware that hackers are always evolving and creating new, inventive methods to take advantage of security flaws. Moreover, in certain instances, the cost of exploiting security flaws is declining.

Businesses are getting increasingly involved in IoT-related activities as hackers become more proficient, which opens up new avenues for exploitation. Today's market is full of internet-facing gadgets that only have rudimentary security features, such one password authentication, but connect and share important data. One password is no longer sufficient adequate for authentication given the dangerous state of the Internet of Things today. To effectively reduce risk and stop breaches, several levels of identity authentication must be implemented across a user, device application, and data.

The physical object network, or "things," is known as the Internet of Things (IoT). Through the Internet, these gadgets can converse and engage with one another, as well as be monitored and controlled remotely, thanks to their embedded hardware, Internet access, and other components (like sensors) [4][5][6].

Regarding the Internet of Things, a "Thing" is any device that has sensors built into it that can automatically gather and send data across a network. They can interact with their internal states and the outside world thanks to the embedded technology in the object, which facilitates decision-making.

Through the use of current network infrastructure, IoT makes it possible to perceive and control objects from a distance, opening up possibilities for more direct integration between computer-based systems and the physical world and improving efficiency, accuracy, and financial gain.

In addition, a variety of everyday tasks are supported by IoT applications, including elderly and child supervision, traffic and healthcare monitoring, route planning, navigation, and transportation decisions [7].

As with the Internet's future, encouraging individuals to utilize modern technology and gain safe access a variety of tools for IoT requires the offering of security services, such as identity verification. Without protection schemes to thwart any malicious behavior, users would find it inconvenient to exchange and share their personal information and data. Thus, effective security and authentication methods are required for the quick and widespread implementation of IoT.

The relationship between tangible items, such as structures, automobiles and other objects with sensors, actuators, software, and electronics as well as network access to gather and exchange information, referred to as the Things Network.

As the field of study known as the Things Network, security focuses on protecting networks and connected devices. According to [8], the Internet of Things is the growing a variety of items which are equipped possess distinctive identification numbers and the capacity to transmit data automatically across a network. The use of sensors embedded in computers and embedded devices in intelligent energy networks, wearable computing, home building automation, industrial M2M (machine-to-machine) connectivity, and communication between vehicles accounts for much of the growth in IoT communication.

The primary the problem is that security hasn't always been taken into the product design because equipment for networking and other items are still a relatively new concept. Older embedded operating systems and unpatched software are frequently included with IoT products.

Consumers frequently forget to modify the passwords set by default on their smart devices, or when they do, they do not choose strong enough passwords. This project outlines several security techniques that may be used in upcoming Internet of Things products.

One of the information technology industry's most exciting developments is the Internet of Things (IoT) paradigm. By 2020, 50 billion gadgets should be online, according to predictions [9].

Numerous ubiquitous apps have found an increasingly wide deployment in a variety of daily-operated services as a result of the Internet of Things' information and communication technologies rapid expansion and universality. These applications are being used to look for new business prospects or greater individual advantage.

For instance, custom and on-demand entertainment services could be offered by a smart home with connected smart gadgets to improve people's quality of life. Another illustration

is how people are gradually switching from traditional credit card payment methods to innovative ones like online payments through wearable technology.

However, the growing number of potential security vulnerabilities over the past few decades has impacted the development of the Internet of Things. Because of its open deployment environment and absence with regard to security safeguards, the Internet of Things is particularly susceptible to assault [10]. Numerous IoT attacks have occurred [11], resulting in billions of dollars being lost.

The foundation of IoT system security is endpoint and authentication security. However, because of their limited resources, IoT devices are a tempting target for assaults. Many device hardware and system limitations, such as constrained authentication and endpoint device security are becoming increasingly difficult due to factors like energy, memory, and computational capacity, as well as real-time operation, message size, and communication latency [12].

Conventional security method implementation may not work since when time is of the essence, the device's ability to manage its core function may be compromised by the time to execute these techniques, which may not be acceptable. The development of efficient security we, need to concentrate on solutions for the Internet of Things (IoT) and other technical settings that can protect systems despite constraints on power consumption, computer capacity, and memory footprint if we are to meet users' and operators' security needs for applications.

Endpoint security solutions can be integrated with other cutting-edge technologies, like fog and edge computing. By relocating processing, closer to the network edge for storage and management of an IoT network, the fog computing paradigm seeks to provide quicker sensor data actuation and real-time analysis [13].

People are currently dealing with the issue of enormous IoT data maintenance because to the widespread use of smartphones and wearable devices.

To manage the enormous amount of data in the Internet of Things, the cloud-based architecture was created, wherein the data was outsourced and managed by the cloud. This partially resolves the issue of the Internet of Things' low processing power. But for the cloud-based IoT architecture, reliable access management and safe data retrieval present new security challenges. For the cloud-based Internet of Things architecture, authentication is essential for access control and data retrieval.

However, the foundation of the Internet of Things' security protocols is made up of cryptographic primitives. One of the most significant methods for creating asymmetric encryption and signature protocols is elliptic curve cryptography, which has been widely used to safeguard the security of cyber-physical systems.

Since the Internet of Things (IoT) was introduced, elliptic curve cryptography primitives have been extensively used for security purposes. For IoT security, elliptic curve cryptography implementation in devices with limited resources is very important.

Even though a lot of research has been carried out regarding security and privacy issues with IoT, there are not many publications that provide a high-level explanation of IoT endpoint security. But endpoints are an essential component of the Internet of Things architecture.

Many device hardware and system limitations, such as constrained Endpoint device security becomes increasingly difficult due to factors like energy, memory, and processing power, message size, communication latency, and real-time functionality [12].

Despite the availability of a well-developed collection of algorithms, there are still difficulties in implementing cryptographic algorithms efficiently given the numerous IoT restrictions.

These obstacles are comparable to those encountered when implementing digital signal processing functions efficiently. Endpoints demand effective implementation strategies in terms because they typically have a multitude of underlying resources and applications, are heterogeneous, and have different requirements for memory, logical processing, and execution time.

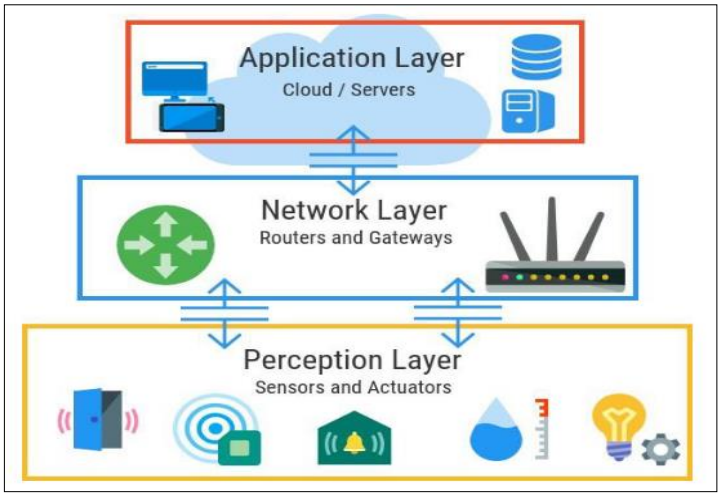

Figure 1.1: The Architecture of IoT [6].

The following five categories fully classify the security strategies conveyed in IoT: updates and patches, farewelling and intrusion prevention, device authentication, access control, and secure booting.

a. Secure booting:

Cryptographically generated digital signatures are employed to verify the integrity of the software and authenticity when the device is first powered on. An electronic signature appended within the software's image and validated via the equipment works in a similar manner to a person signing a check or legal document to guarantee that only software approved to operate on that apparatus and endorsed by the authorized entity is going to load. Although a basis of confidence has been established, the gadget still requires to be protected against malicious intent and a variety of run-time threats.

b. Access control:

The operating system's mandatory or role-based access controls restrict device components' and apps' access to the resources they require performing their functions. Access control makes sure that the compromised component has the least amount of access to different sections of the framework. Mechanisms for access control that is device-based comparable to access control based on the network programs like Active Directory from Microsoft restrict compromised information to only those parts of the network that are permitted by the specific credentials, even in the event that someone were to successfully steal corporate credentials to obtain network access. To reduce the impact of any security breach, only the bare minimum access necessary to carry out a task should be granted, according to the principle of least privilege.

c. Device authentication:

Before receiving or sending data, the network-connected device needs to authenticate itself. Users of deeply embedded devices frequently do not wait to enter the credentials needed to access the network by sitting in front of keyboards. Therefore, how might we make sure the products are accurately identified before authorization? Automated verification enables a network-accessible device using an identical set of qualifications kept in a safe place to be stored, much for user verification enables a user's ability to log in to a corporate network using their login credentials.

d. Firewalling and IPS:

To regulate traffic that is meant to end at the device, it is also necessary for the device to have a firewall or deep packet inspection feature. Why, if network-based appliances are in place, is a host-based firewall or intrusion prevention system necessary? Different from enterprise IT protocols, deeply embedded devices have their own set of procedures. The grid of smart energy, for instance, has a group of guidelines controlling device-to-device communication. To find malicious payloads concealed in non-IT protocols, deep packet inspection capabilities and protocol filtering specific to a given industry are therefore required. The network appliances should handle filtering higher-level, regular Internet traffic, so the device only needs to focus on filtering the specific data that will end up on that device and do so in a way that maximizes the restricted amount of computing power at hand.

e. Patches and updates:

The device will begin receiving software updates and hot patches as soon as it is powered on. Operators must distribute devices and patches must verify their identity in a manner that is not used up capacity or compromise the functional safety of the device. When Windows users receive updates from Microsoft and their laptops become unusable for fifteen minutes, that has one thing. It has a different story when thousands of field-installed devices are relying regarding security updates to keep out the unavoidable weakness that finds its way into the wild while carrying out vital tasks or providing essential services. It is imperative that security patches and software upgrades are distributed in a manner that preserves an embedded device's constrained bandwidth and sporadic connectivity while completely removing any chance of jeopardizing functional safety.

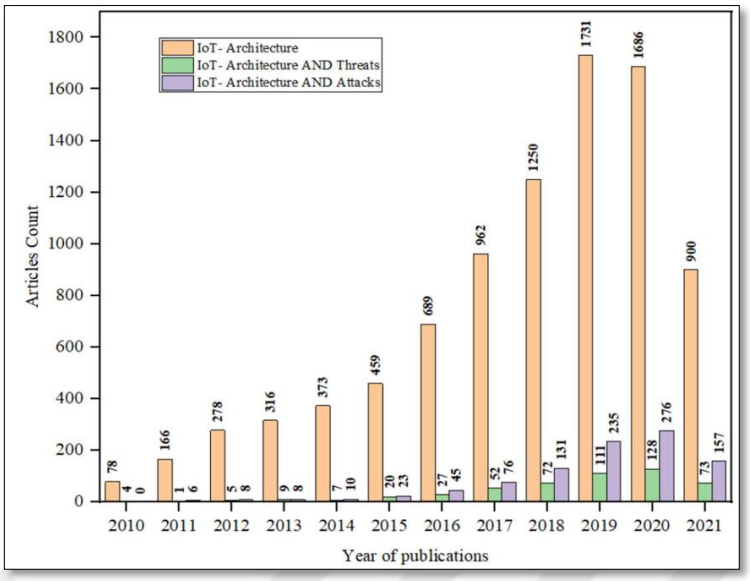

Figure 1.2: IoT Attacks and Architecture, IoT Threats and Architecture, and Literature Statistics [7].

Many bio-authentication tokens, including electrocardiography (ECG) [14–16], iris and eye movement [17, 18], face [19, 20], ear [21], voice [22, 23], Numerous research have examined finger nail [24], handwritten signature [25], keystroke dynamics & on-screen movement [26], text analysis [27, 28], brainwave [29], and gait gesture [30, 31], recent years that have examined continuous authentication (also known as continuous authentication).

Regretfully, the majority of current continuous authentication techniques are inadequate for the majority of end users since they necessitate sophisticated instruments, like those used to harvest resumes.

Cryptography using elliptic curves was discovered in 1985 by Victor Miller and Neal Koblitz [V.S. Miller, 1985] [33], [N. Koblitz, 1987] [32]. The most effective curve theory. Elliptic curve cryptography is frequently used in comparison analyzes with public key encryption methods such as RSA and the Diffie-Hellman key exchange problem.

ECC uses low-power computer hardware while still providing the highest level of protection. Certain public key encryption systems, such as D-H key exchange, RSA, and Digital Signature Algorithm (DSA), are excellent choices for high-power computing; but, if cloud or Internet of Things (IoT) computing is selected, these devices might not be supported by low-power computing devices.

A public key is employed in the cryptosystem of elliptic curve cryptography (ECC). that makes use of the elliptic curve arithmetic model with constrained fields.

Because of how ECC differs from other public key algorithms, smaller keys were needed to provide the same level of safety. Public key cryptosystems are required to detect modern encryption or sophisticated signature techniques. The "decoding key" or "signature creation key" is one secret key, and the "cypher text generation key" or "signature confirmation key" is another, similar key. Even though data can be stolen by unapproved parties, encrypted source content is difficult to discover without the correct key.

Encryption keys and decryption to perform encryption and other tasks, keys are required. decryption. In cryptanalysis, upgraded systems like simulated annealing, genetic algorithms, and mimetic algorithms are developed to produce an active and practical method. Some important techniques are developed by combining the advanced cryptanalysis features for upcoming generations.

The discovered factors are used in the selection of the elliptic curve bounds. The created genetic process alteration was assessed using a traditional evaluation approach, despite certain problems that have been fixed through improved aspects. The accessibility guarantee, which calls for the data to be accessible for implementation, is used to prevent data loss.

The ECC is a safe, practical technology that should only be used in specific situations. In order to create a continuous cryptographic curve, generating curves must pass through numerous algorithms and processes. The methods for elliptic curve creation and the Elliptic Curve Digital Signature Algorithm (ECDSA) provided by the ANSI (American National Standards Institute) standard.

Elliptic curves are usually employed because they may be used at the same security level as other popular methods like RSA and have smaller key sizes. The theory behind the security of asymmetric encryption systems is that theoretical problems such as discrete algorithms and integer factorization are strictly limited to certain areas on an elliptic curve. More specifically, provide a precise formula for the vectors of polynomials that represent the best at coupling over a family of elliptic curves that is cyclotomic.

1.1 PROBLEM STATEMENT

An important need for robust security systems has been brought to light by the proliferation of IoT devices. Limited memory, processing power, and energy resources negatively impact a lot of IoT devices. Therefore, traditional cryptographic methods may not be ideal for IoT implementations because they often require significant computational resources. IoT devices

are also becoming more and more attractive targets for cyberattacks as they become increasingly common. Cryptographic arrangements must, at the very least, be able to be reconciled into asset-compelled IoT devices and be safe from potential threats.

1.2 PROJECT OBJECTIVE

a. To evaluate Elliptic Bend Cryptography's (ECC) energy consumption compared to alternative cryptographic techniques when used with asset-compelled Internet of Things devices.
b. To create a comprehensive regulation that ensures consistent safety efforts are kept while integrating the ECC principles into various IoT stages.

1.3 STRUCTURE OF THE THESIS

This thesis has six chapters, the outline of it as follows:

CHAPTER 1:

INTRODUCTION: This chapter provides an introduction as well as the project's objective and motivation.

CHAPTER 2:

BACKGROUND AN Related work: By outlining their findings and identifying any research gaps, the past contributions to the topic are highlighted in this part of the thesis.

CHAPTER 3:

SQLI ATTACK: This chapter will give a broad synopsis of the SQLI attack.

CHAPTER 4:

PROPOSED SYSTEM: This chapter gives practical side.

CHAPTER 5:

RESULT: This chapter we discuss results and analysis.

CHAPTER 6:

Conclusion and future work: The conclusions from the analyze and conversations are highlighted in this chapter, which also points readers toward additional subjects for potential future research.

2. BACKGROUND

2.1 PUBLIC KEY CRYPTOGRAPHY

A cryptosystem using public keys is made up of two families, "{E_K} (K∈{K})" and "{D_K }(K∈{K})", from reversible algorithms that represent transformations, according to Diffie and Hellman in New Cryptographic Directions [35]:

$$E_K: \{M\} \rightarrow \{M\}$$
$$D_K: \{M\} \rightarrow \{M\} \quad (2.1)$$

within a limited message area $\{M\}$, so that

a. With each " $K \in \{K\}, E_K$" is the opposite of D_K.
b. With each "$K \in \{K\}$ and $M \in \{M\}$" $The\ computer\ programs\ E_K$ and D_K are simple to calculate.
c. Regarding nearly all $K \in \{K\}$, every quickly calculated methodology that is comparable to D_K It is not feasible to compute the result from E_K .
d. With each $K \in \{K\}$, Inverse pair computation is possible E_K and D_K from K.

Because of this, public key cryptography has some intriguing features. According to [36], "public-key encryption is a simple and elegant concept with far-reaching implications" within the Applied Cryptography Handbook (HoAC).

For example, the public key strategy can be used to construct a secure communication channel without requiring the transmission of an encryption key across a secure channel, which is different from symmetric-key cryptography.

Because the third exposed property allows the key for encryption to be supplied over an unsecured channel (which may be the same channel used to transport the ciphertext) without risking system security, even though it is mathematically impossible to extract the decryption key from the disclosed data. Moreover, you do not have to make a unique key for every object you wish to connect to using the same key pair, public-key cryptography facilitates communication with any other entity.

Nevertheless, this technique has a serious flaw in that it requires authentication. Impersonation of the receiver, which happens when a possible attacker provides the sender

with access to the public encryption key E_K^', which the sender wrongly thinks is the intended recipient, is one of the main issues with the system. The attacker can intercept and decrypt the messages in this way.

Securely equivalent symmetric systems have been shown to perform faster than public key cryptosystems in general. Because of this, it is standard procedure to create a shared key for a symmetric cryptosystem using public key cryptography, which will then be used to transfer data between two distinct parties.

ECC and RSA are the two public-key encryption techniques. The primary focus of this work, Wireless Sensor Networks [37], has shown these two public-key cryptosystems to be feasible for application.

2.1.1 Rivest, Shamir, and Adleman (RSA)

Rivest, Shamir, and Adleman are the people behind the acronym RSA. They presented the first description of this public-key cryptosystem in 1977. Publication of a Method for Obtaining Public-key Cryptosystems and Digital Signatures one year later [38].

The RSA algorithm derives its basis from the Rivest, Shamir, and Adleman article, which proposed the RSA problem (RSAP), which is predicated on the computational effort and difficulty of factorizing big integers.

By employing two encryption keys: a "secret key (d,n)" and a "public key (e,n)" where "n" and "e,d" are all positive integers, The following protocol was proposed by Rivest, Shamir, and Adleman for message encrypting and decryption:

$$\begin{gathered} "C \equiv E(M) \equiv M^e "(modn), \text{ as a message } M. \\ "D(C) \equiv C^d "(modn), \text{ as a ciphertext } C. \end{gathered} \tag{2.2}$$

The first two large primes are chosen to generate the keys " $n = p * q$". $Next, however, d$ is chosen as a sizable, arbitrary prime that is more prime than "p." That is:

$$Gcd\ (d, (p-1) * (q-1)) = 1 \tag{2.3}$$

Finally, e is computed as the inverse multiplicative of d modulo "$(p-1) * (q-\ 1)$" which is:

$$e * d \equiv 1 mod((p-1) * (q-1)) \tag{2.4}$$

The authors of the study demonstrate that even the most straightforward methods of breaking the system are at least as difficult as factoring n, and that the problem's effective solution is unknown.

The most popular public-key algorithm is RSA, but as processing power and resources expand, longer keys are needed because the algorithm relies on the amount of work needed to factor large prime integers.
For discrete logarithm group, public key encryption techniques, like RSA, NIST, a federal agency independent of the US Technology Administration and the Commerce Department, suggested a key length of 2048 in their article "Recommendation for Key Management" [39].

2.1.2 Elliptic Curve Cryptography (ECC)

Elliptic curve cryptography was independently created in 1985 by Victor S. Miller and Neal Koblitz. It was initially suggested as "the analogue based on elliptic curves based on finite fields of public key cryptosystems which use the multiplicative group of a finite field" in the book by Kobiltz. It was first released by Koblitz in Elliptic Curve Cryptosystems two years later [40], originally appeared in Use of Elliptic Curves in Cryptography, by Miller [41].
According to both authors, elliptic curve-based cryptosystems are faster than the most recent ones and more resistant to known attacks against traditional discrete logarithm-based cryptosystems. Algorithms analogous to the then-current cryptosystems and protocols are presented in both papers. An analog of the Diffie-Hellman protocol, as it is presented in the publication by Diffie and Hellman [33], is put forth in [40].
Analogs a presentation of the Massey-Omura and El-Gammal cryptosystems illustrated in Koblitz work [41], offering an Elliptic Curve-based solution that is comparable to these.
The main reason ECC is a better option than RSA is because it offers heightened protection when compared to RSA for the same key size. The equivalent key dimensions required for the ECC and RSA to provide the same degree of protection as a method of In Table 2.1, symmetric cryptography is presented. Based on an assessment of performance of ECC in Wireless Sensor Networks Compared to RSA [37], ECC outperforms RSA in terms of data transit and storage volume as well as computation time.

Table 2.1: NIST Guidelines [39] Provide the Equivalencies of Necessary Bit Sizes for Keys for a Comparable Degree of Security for Various Encrypting Techniques, as well as the Ratio Between ECC and RSA Sizes.

Bits of security	RSA (bits)	ECC (bits)	Ratio
80	160	1024	1:6
112	224	2048	1:9
128	256	3072	1:12
192	384	7680	1:20
256	512	15360	1:30

2.1.3 Diffie-Hellman Key Exchange Protocol

Martin E. Hellman and Whitfield Diffie, who gave the protocol Diffie-Hellman in their work in 1976, are credited with giving it its name [33]. Instead of using a previously secured channel, as is the case with all classical key distribution systems, the protocol was put forth by the authors as a way to distribute keys to users of a public key distribution system via an unsafe channel.

Their approach used the problem of a discrete logarithm as the foundation for the protocol, which is the basis of public key cryptography. The parties trade public keys, use exponentiation and the multiplicative groups fashion to combine the public key of the other party with their own private key, and establish a shared secret through an unsecure channel. According to Diffie and Hellman's description in their work: Allow user A to choose a random number X_A in the interval "$0 < X_A < q$ ", calculating and keeping it under wraps:

$$Y_A = \alpha^{X_A} mod(q) \tag{2.5}$$

Let user B follow the same procedure, with $0 < X_B < q$:

$$Y_B = \alpha^{X_B} mod(q) \tag{2.6}$$

When users A and B want to communicate, they use:

$$K_{AB} = \alpha^{X_A X_B} mod(q) \tag{2.7}$$

which they acquired by fusing the other party's public information with their own secrets:

$$Y_B^{X_A} mod(q) = \alpha^{X_B X_A} mod(q) = Y_A^{X_B} mod(q) = K_{AB} \quad (2.8)$$

The data released to the public is Y_A and Y_B. It is demonstrated that obtaining the key is a computationally simple task when one of the secrets is known K_{AB} However, using the discrete logarithm problem as a guide, it is not ignorant of X_A or X_B .

This makes it possible to create a secure channel using an insecure channel and make the required information public. The Diffie-Hellman protocol's lack of authentication is one of its well-known issues. It is possible to launch a man-in-the-middle attack, in which an entity C can act as the legitimate party by intercepting the communication, assuming the role of an intermediary in the legitimate parties' communications, and giving them a different public key.

2.2 ELLIPTIC CURVE CRYPTOGRAPGY

Elliptic Curve Cryptography (ECC) was developed concurrently in 1985 by Victor S. Miller [40] and Neal Koblitz [41], as was previously mentioned. The two authors suggest that Discrete Logarithm (DL) based cryptosystems can be analogized to using EC.

The following is the definition of the classical discrete logarithm problem from Elliptic Curves, Number Theory and Cryptography [42]:

Let p be a prime and let a, b be nonzero integers modulo p . The task of finding an integer is the traditional discrete logarithm problem k such as:

$$a^k \equiv b(modp) \quad (2.9)$$

The Elliptic Curve Discrete Logarithm Problem (ECDLP) serves as the foundation for EC cryptosystems when converting to EC groups. Koblitz defined it as "Given an elliptic curve" E defined over $GF(q)$ and two points $P, Q \in E$, find an integer x such that such that $Q = xP$ if such x exists" This, when represented in the usual group additive notation used in elliptic curves, is the EC counterpart of the conventional discrete logarithm issue.

2.2.1 Mathematical Foundations

Elliptic Curves are most commonly expressed using the Weierstrass Equation. An equation of the type is graphed by an elliptic curve E.

$$y^2 = x^3 + Ax + B \tag{2.10}$$

where the constants are A" and B". A, B, x, and y are often components of a field, which could be any of the following: a finite field, the real numbers R, C, and Q, or any combination of these.

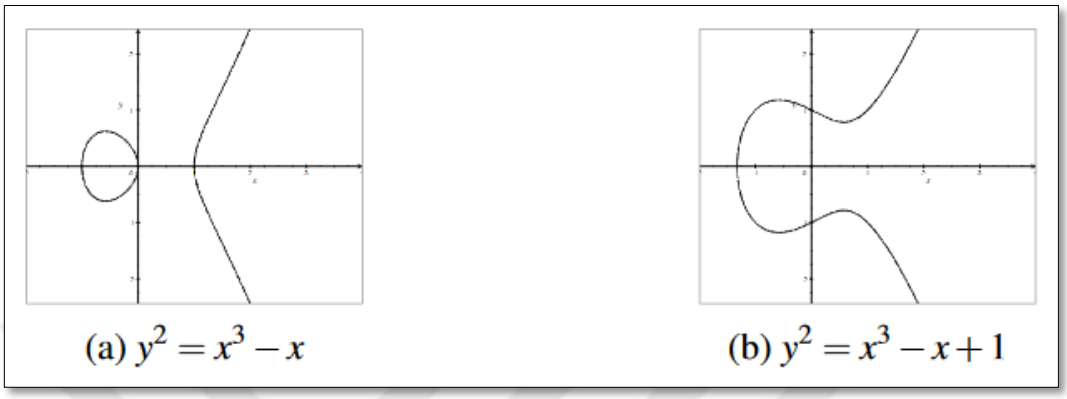

Figure 2.1: Two Distinct Elliptic Curves Over R are Represented, with Positive and Negative (B) Discriminants.

field K. By definition, an elliptic curve is not unique. In order to meet this requirement, the discriminant $\delta = -16(4A^3 + 27B^2)$ has to be less than or equal to 0. This means that A and B must meet the following criteria:

$$4A^3 + 27B^2 \neq 0 \tag{2.11}$$

The form of the discriminant in the actual curve representation is determined by its sign. graphical depiction. Figure 2.1 provides two instances.

The points with coordinates in a field LK are shown as:

$$E(L) = \{\infty\} \cup \{(x, y) \in L \times L \mid y^2 = x^3 + Ax + B\} \tag{2.12}$$

The group law, also referred to as point addition, is defined as follows in [42]: Suppose that the definition of the elliptic curve E is:

$$y^2 = x^3 + Ax + B. \text{ Let } P_1 = (x_1, y_1) \tag{2.13}$$

And:

$$P_2 = (x_2, y_2) \tag{2.14}$$

points on E with $P_1, P_2 \neq \infty$. Define $P_1 + P_2 = P_3 = (x_3, y_3)$ as follows:

a. If $x_1 \neq x_2$ then:

$$x_3 = m^2 - x_1 - x_2, y_3 = m(x_1 - x_3) - y_1, \text{ where } m = \frac{y_2 - y_1}{x_2 - x_1} \quad (2.15)$$

b. If $x_1 = x_2$ but $y_1 \neq y_2$, then $P_1 + P_2 = \infty$.

c. If $P_1 = P_2$ and $y_1 \neq 0$, then:

$$x_3 = m^2 - 2x_1, y_3 = m(x_1 - x_3) - y_1, \text{ where } m = \frac{3x_1^2 + A}{2y_1} \quad (2.16)$$

d. If $P_1 = P_2$ and $y_1 = 0$, then $P_1 + P_2 = \infty$.

Moreover, define:

$$P + \infty = \infty \quad (2.17)$$

or each and every P on E. Moreover, Miller [41] earlier suggested that the elliptic curves used in elliptic curve cryptography be defined over a finite field F, forming an additive Abelian group with as the identity element from the points on E.

Let E be an elliptic curve defined over a finite field F. The set of points on the curve whose coordinates (x,y) are a component of F is called group E(F). The order is the first parameter we can define for the group E(F). Let #E(F) denote both the order of E(F) and the total number of points that comprise the group E(F). Moreover, suppose that G is a point inside E(F) on the curve of elliptical E.

The order of a point G is the lowest positive integer k such that kP=∞.Furthermore, it is known that, as a basic result of group theory and a corollary of Lagrange's theorem, the order of the point always divides the order of the group E(F) [43].

Given this and in compliance with RFC 9060 [44], a generator points of the group and a cyclic subgroup of an elliptic curve constitute an elliptic curve parameter set in cryptographic situations. When the method is used on a prime order finite field whose characteristic exceeds three, characteristics listed below completely:

a. The field F_p's order is indicated by the prime number p.
b. The curve equation's parameter A in its given Weierstrass form.
c. The parameter B in the curve equation's given Weierstrass form.
d. The subgroup's generating point G.
e. The subgroup that G created has order n.

2.2.2 Operations

In the context of cryptography, an elliptic curve's operations are linked to a specific set of parameters. Points in the same group must be used for operations involving separate points. The so-called refine coordinates are used as the fundamental method of representing the points. It is consistent with the Weierstrass equations that have been provided, where Two coordinates (x, y) are used to denote the points. Points on elliptic curves can be represented using a variety of coordinate systems.

Projective coordinate systems are another widely used coordinates. The points on an elliptic curve can be represented in the projective space P_K thanks to the projective coordinates. That these representations make it possible to avoid (multiplicative) inversions in the group law—an expensive operation in many platforms—is one characteristic of great significance for this study. The two projective coordinate systems that are frequently utilized in this work are described in the IEEE Standard Specifications for Public-Key Cryptography [45]:

a. In Homogeneous coordinates, One way to represent an affine point (x,y) is as $(X:Y:Z) = (\theta x, \theta y, \theta)$ for some $\theta \neq 0$. (0:θ:0) represents the neutral element, also known as the point at infinity in EC notation. All three coordinates are considered to have the same weight in homogeneous coordinates. The elliptic curve in this coordinate system is transformed into $Y^2 = X^3 + AXZ^2 + BZ^3$.
b. In Jacobian coordinates, One way to represent an affine point (x,y) is as $(X:Y:Z) = (\lambda^2 x, \lambda^3 y, \lambda)$ for some $\lambda \neq 0$. In the given, the point at infinity is $(\lambda^2:\lambda^3:0)$. In jacobian coordinates, It is said that the weights of the X" and Y coordinates are, in that order, 2 and 3. In this coordinate system, the elliptic curve is converted into $Y^2 = X^3 + AXZ^4 + BZ^6$.

Section 2.2.1 has already addressed the group operation, or point addition, for the coordinates that are affine. Within homogeneous and Jacobian projective coordinate systems, the modular multiplications are used to trade inversions, improving efficiency on systems where the inverse operation (mostly division) is computationally expensive. Several references evaluate and suggest using several coordinate systems, such as Marc Joye in [46]. A table comparing the number of operations required in different coordinate systems for doubling and adding points — that is, adding the point to itself—is shown in Joye's work. A comprehensive table of the coordinate frameworks employed in this investigation is

presented in Table 2.2. The symbols "M" represents costs of multiplication, S represent costs of squaring, I represent costs of modular inverse, with c standing for multiplicative costs.

Table 2.2: Cost in Operations of Point Addition and Point Doubling for Different Coordinate Systems.

Coordinate system	Point addition	Point doubling
Affine	$2M + S + I$	$2M + 2S + I$
Homogeneous	$12M + 2S$	$5M + 6S + 1c$
Jacobian	$10M + 4S$	$1M + 8S + 1c$

The fundamental function of elliptic curve cryptography is scalar multiplication. It is comparable to the additive notation's exponentiation and is based on repeatedly adding points.

The binary exponentiation method, due to the duality inherent in exponentiation, is among the first algorithms to appear for the execution of this operation. By substituting addition operations for multiplication operations and finishing the method with the scalar multiplication result, the duality may be shown.

Like in all other public key schemes, a key pair in the ECC context comprises a private key and a public key. The matching public key of the key pair is created using the private key, which is an integer that is chosen at random.

The process of scalar multiplication is used to produce the public key. The point is the chosen to generate point of the chosen EC group, and the scalar is the chosen secret key. Let p be the prime number signifying the elliptic curve group that has been selected and let G be its generator field's order, and K_(A_priv),K_(A_pub) be the entity "A'"s private and public keys, respectively. The following is how the key pair is created:

a. Private key $K_{A_{priv}}$. A random number x is selected such a $0 < x < p$.

i. $K_{A_{priv}} = x$.

b. Public key $K_{A_{publ}}$. "A" scalar multiplication is carried out with the group's generating point and private key. $K_{A_{publ}} = K_{A_{priv}} * G$.

Victor S. Miller introduced the Elliptic Curve Diffie Hellman (ECDH) protocol in [40]. Miller claims that an elliptic curve's points "have the structure of an Abelian group" and that the protocol of Diffie-Hellman "really only uses the property that we are working in a group

[...]" in that passage. We could develop comparable structures over elliptic curves. The ECDH procedure follows these steps:

a. A and B, the parties concerned, have already decided on a set of EC parameters.
b. The parties swap public keys, which are represented by an EC point.
c. Each party combines its private key with the public key it receives in a scalar multiplication operation. The EC gains one more point as a result of that action. The shared secret that the protocol generated is that result. Typically, a session key is obtained by taking the point's x coordinate and applying a technique that is agreed upon by both protocol participants.

An adaptation of the ElGamal signature system is the Digital Signature Algorithm (DSA) with an elliptic curve [47]. The algorithm makes it possible to create and validate message signatures using ECC. Let r represent EC set. Share the signer's public key, K_(A_publ), with the recipient. In essence, the following algorithm is employed [42]

a. Signature generation for message m.

Choose a random integer k with $1 \leq k < r$ and calculate $R = kG = (x, y)$.

Calculate $s = k^{-1}\left(m + K_{A_{priv}}x\right)(modr)$.

The entire signed copy that was delivered to the recipient is(m, R, s).

b. Signature verification for (m, R, s).

Calculate $u_1 = s^{-1}m(modr)$ and $u_2 = s^{-1}xmod(r)$.

a. Calculate $V = u_1G + u_2K_{A_{publ}}$.

b. Declare the signature valid if $V = R$.

The verification equation holds if the verification is successful:

$$V = u_1G + u_2K_{A_{publ}} = s^{-1}mG + s^{-1}xK_{A_{publ}} = s^{-1}\left(mG + xK_{A_{priv}}G\right) = kG = R \quad (2.18)$$

ElGamal has three scalar multiplication operations to verify the signature, but ECDSA only utilizes two, which is one of the primary distinctions between the two methods. ECDSA outperforms ElGamal because it is the algorithm's most expensive operation.

2.2.3 Comparison/Advantages

In this sense, ECC is far better than RSA. To get the same level of security, a significantly smaller key size is needed. According to Table 2.1 and Hankerson et al.'s Guide to Elliptic

Curve Cryptography [48], for symmetric encryption using RSA, a key length of 3072 bits is needed. whereas ECC only needs 256 bits to achieve the same level of security as a 128-bit key.

This is one major advantage that ECC has over RSA. Until RSA key sizes become unmanageable, ECC will be a better alternative than RSA due to the increasing ratio of needed key sizes for both protocols. This is due to the increasing limited level of protection needs. This feature, moreover, the smaller size of the controlled variables in ECC makes it an excellent choice for memory-intensive systems, as those in Wireless Sensor Networks (WSN).

ECC is dependent on important size-related factors, as was previously indicated. Typically, this size does not drop below 160 bytes. Variables that match the size of the processor architecture are used for arithmetic operations. in the current architectures and programming languages; regarding 32-bit systems, this is 32 bits. and for 64-bit architectures, it is 64 bits. This means that an arithmetic system must be able to manipulate and perform arithmetic operations on variables of this size. As a result, an arbitrary precision arithmetic becomes necessary.

2.3 TRADITIONAL APPLICATION

Over the last ten years, ECC has been more widely used online. Every mainstream operating system and web browser supports ECC in some way. ECC is also supported by popular server software and security libraries. extensive adoption follows extensive backing.

Change of Keys We have tested the top 100 most popular websites as of right now on Alexa, the popular website ranking service, to find out what kind of encryption each one uses, if any, on their landing page.

Of the top 100 pages, we discovered that 26 pages did not provide a certified HTTPS connection, while 2 more sites provided an HTTPS connection but lacked a certificate. Seventy-two websites are still providing an HTTPS connection with a certificate.

Of these 72 websites, key exchange was done via ECC on 69 of them. Elliptic Curve Diffie-Hellman Ephemeral RSA (ECDHE_RSA) was used by 53 websites, whereas Elliptic Curve Diffie-Hellman was used by 16 websites. Key exchange using ephemeral ECDSA (ECDHE_ECDSA).

Temporary keys created instantly that do not require authentication are called ephemeral

keys. Out of the top 100 websites, just 3 did not utilize any form of ECC; instead, they all used RSA. ECDHE_RSA was utilized by both of the websites that provided an HTTPS connection without a certificate.

Validation of DNSSEC Information assurance and security are also concerns for the domain name system (DNS), which converts online inquiries into computer-understandable IP addresses. In order to defend DNS servers and exchanges from assaults such as distributed denial-of-service attacks (DDoS attacks), DNS Security Extensions (DNSSEC) were created.

Scholars propose that several threats stem from DNSSEC using RSA as a signing technique. It is thought that ECC may also be used to strengthen security on the DNS server, however it would be slower at validation.

It has been demonstrated that even the most sophisticated computer demanding ECC techniques do not beyond the capabilities of a contemporary CPU core through Comparing benchmarks for ECC digital signatures and modelling DNS resolution.

ECC DNSSEC makes DNS servers more dependable and secure by defending against amplification attacks and packet fragmentation.

Server for Signatures Graphics processing units (GPUs) with limited instruction sets can often outperform CPUs with bigger instruction sets because of more weight from a bigger instruction set provided by the lightweight mathematical equations used by ECC.

A group took advantage of this and created a GPU-based cryptography accelerator. GPUs may be used to accelerate ECDSA and build a working universal signature server that can also perform key agreement and encryption using ECC, as shown by W. Pan et al.

GUESS stands for GPU-driven Universal Elliptic-curve Signature Server Architecture given to their server. By using several GPU threads, they were able to surpass current prototypes and products with noticeably increased efficiency in regard to throughput. The group intends to test, refine, and add to GUESS in the future, as well as increase the collection of curves available to it.

2.4 RELATED WORK

Numerous open-source operating systems have been created especially for low-power electronics. TinyOS is one of the most widely used. A thin operating system is called TinyOS [49]. with an event-driven architecture that was written in the C dialect nesC. Furthermore,

a low-power device Elliptic Curve Cryptography library called TinyECC already exists. In 2008, A Liu and Peng Ning presented TinyECC [50]. In addition to many optimizations to the conventional ECC processes to increase, it offers an extremely versatile ECC solution by improving them for the TinyOS operating system and low-power devices.

Table 2.3: Related Works on Authentication Techniques.

Author Name	Paper Title	Methods or Steps used	Description
Moghaddam, Faraz Fatemi, et al [51]	A scalable and efficient user authentication scheme for cloud computing environments	Modified Diffie-Hellman Agent (MDHA), ZKP Diffie-Hellman	To verify a user's identification on the client side, a client-based user authentication agent has been introduced in this paper.
Singh, Ashish, and Kakali Chatterjee [52]	An encrypted multi-tier authentication system for cloud computing	Multi-tier authentication	This study suggests a more sophisticated and dependable multi-tier authentication system for cloud service access.
Yang, Jen Ho, and Pei Yu Lin [53]	An ID-based user authentication scheme for cloud computing	One way hash function, XOR operation	This study presents a novel user authentication strategy for cloud computing that is based on identification. The suggested approach provides lower computing costs and higher security levels, as demonstrated by the authors.
Tien-Ho Chen, Hsiu-lien Yeh, Wei-Kuan Shih [54]	An Advanced ECC Dynamic ID-Based Remote Mutual Authentication Scheme for Cloud Computing	Elliptic Curve Cryptosystem (ECC)	This paper proposes an ECC dynamic ID-based remote mutual authentication system for faraway devices in order to overcome the issues.

3. ELLIPTIC CURVE CRYPTOGRAPHY

3.1 OVERVIEW

In public key cryptography, elliptic curves typically serve as a substitute for conventional systems based on finite field discrete logarithms and public-key cryptosystems like RSA. The state of elliptic curve cryptography stems from its capacity to guarantee data transport in a commendable way. Its efficiency is still commendable, even though its key size is much less than several other encryption techniques, such as RSA.

A person's capacity for thought might be one of their greatest gifts. It is astonishing and can be efficiently used when a human thinks differently than any machine, bot, or other human. The optimal time to use this cognitive skill is when the system is authenticating users. The verified user makes use of his cognitive ability to choose specific supplied coordinates that he has to remember during the log-in procedure, which are derived from an elliptic curve equation.

These addresses the issue of user authentication by confirming the user's memory of the coordinates they chose when logging in. Elliptic curve cryptography, whose merits have been extensively covered in this introduction thus far, is used for the encryption procedure that comes after the authentication step.

To produce a shared secret key, the Diffie-Hellman key exchange mechanism is employed. on both sides of the encryption process. The method facilitates communication amongst the users who were a part of the previously formed link. Texts sent over the channel in encrypted form are coordinates, which makes them even harder to decipher. Finally, the cypher text is decoded and received at the opposite end.

Compared to other encryption techniques that employ relatively larger keys, smaller keys are used in Elliptic Curve Cryptography, a public key encryption method for encryption. Elliptic Curve Cryptography's 164-bit key increases data security capabilities to almost that of a 1,204-bit key encryption method.

An electronic method called cryptography is used to safeguard sensitive data during transmission. Information security is the main scientific focus of the science of cryptography. Data security is the main objective of cryptography, which uses a variety of authentication techniques. The cost should be lower than the value of the original information when data authentication is the main factor to be considered. The public key cryptosystem known as elliptic curve cryptography was developed in the 19th century by

Victor Miller and Neil Kobiltz [32] [33]. It is comparable to RSA public key cryptography. The Problem of Discrete Logarithmic.

As a result, the ECC key functions effectively with gadgets that have constrained resources, such smart cards and mobile phones. Another challenging task is choosing a suitable elliptic curve. To achieve an efficient and practical application, ECC must be regulated. It is believed that using the ECC requirements from NIST, the National Centre for Technology and Standards, is safe for use in cryptographic applications. The two main concepts used to define the two methods used in cryptography are encryption and decryption. Encryption techniques are used when sending confidential information across communication channels. The encryption procedure requires a key and an encryption algorithm.

In cryptography, key creation is a crucial procedure that involves generating a private key in addition to a public necessary. The data is encrypted by the sender using the recipient's public key, and it is decrypted by the recipient using their private key. The primary functions of these keys are to encrypt and decode the original data. While public key algorithms—which come with more current cryptographic systems—specify RSA and ECC, symmetric-key methods specify DES and AES.

The original data is protected by a single shared key in the symmetric-key method, and it is necessary to keep this key secret. There are two primary keys in elliptic curve cryptography (ECC), commonly referred to as public key cryptography. In order to perform cryptographic operations, it also includes a set of key-connected activities. ECC's primary benefit is its compact key size.

ECC was able to provide a short encryption key and a value that has to be added to the encryption process in order for the encrypted data to be decrypted. Since the elliptic curve cryptosystem relies on discrete logs above finite fields or integer factorization, it is more secure than other cryptosystems. It is the safest and most inventive public key cryptosystem. The history of elliptic curves is rich and extensive. Elliptic functions that fulfil cubic equations originated from the difficulty of calculating the arc length of an ellipse; for this reason, planar cubic curves are also known as elliptic curves. Elliptic curves have applications in factorization of integers, cryptography, and coding theory. They establish a connection between number theory, algebraic geometry, and complex analysis. Elliptic curves are the subject of numerous well-known open issues and conjectures. They were also essential to Andrew Wile's well-known demonstration of Fermat's last theorem.

Public key cryptography is implemented using elliptic curve cryptography. The foundation of ECC consists of numerical sets related to elliptic curves, which are mathematical entities. Just as with numbers modulo p, there exist rules for multiplying and adding these numbers. The collection of points that fulfil a particular mathematical equation is known as an elliptic curve. They have symmetry.

3.2 WHAT IS AN ELLIPTIC CURVE?

An algebraic plane curve described by an equation of the type is called an elliptic curve:

$$y^2 = x^3 + ax^2 + b \tag{3.1}$$

It has no cusps or self-intersections, making it non-singular elliptical curve the following mathematical principles are covered Via mathematical means:

a. Abelian group
b. Elliptic Curves over Real Numbers
c. Elliptic Curves over Zp
d. Elliptic Curves over GF(2m)

A group in which the outcome of applying a group operation to two group components is independent of the order in which they are written is known as an abelian group, also known as a commutative group. In other words, these are the groups that adhere to the commutativity principle. Properties:

(A1) Closure:	If a and b belong to G, then $a \cdot b$ is also in G.
(A2) Associative:	$a \cdot (b \cdot c) = (a \cdot b) \cdot c$ for all a, b, c in G.
(A3) Identity element:	There is an element e in G such that $a \cdot \mathrm{e} = \mathrm{e} \cdot a = a$ for all a in G.
(A4) Inverse element:	For each a in G there is an element a' in G such that $a \cdot a' = a' \cdot a = \mathrm{e}$.
(A5) Commutative:	$a \cdot b = b \cdot a$ for all a, b in G.

Discrete curves differ from ellipses. They get their name from the fact that they can be defined as cubic equations, which are comparable to those that can be used to find an ellipse's circumference. Generally speaking, elliptic curve cubic equations have the following structure, which is referred to as a Weierstrass equation:

$$y^2 + axy + by = x^3 cx^2 + dx + e \tag{3.2}$$

The real numbers a, b, c, d, and e represent the values of x and y, respectively. It could be divided as:

$$y^2 = x^3 + ax + b \tag{3.3}$$

Since a 3 is the largest exponent in these equations, they are referred to as cubic equations, or equations of degree 3. A single element designated O, sometimes known the definition of an elliptic curve also includes the zero point or the point at infinity. In order to draw such a curve, we must calculate:

$$y = \sqrt{x^3 + ax + b} \tag{3.4}$$

The elliptical curve Elliptic curves are employed in cryptography, where the variables and coefficients are all restricted to elements of a finite field. Two families of elliptic curves are used in cryptography: binary curves over GF(2m) and prime curves across Zp. When dealing with a main curve greater than using a cubic equation, Zp, all the variables and coefficients are having values inside the whole number range of 0 to p - 1, and computations are performed modulo p.

Two million elements and operations on polynomials that can be constructed for addition and multiplication make up a finite field, GF(2m). We employ a cubic formula for elliptic curves larger than GF(2m), in which all the variables and coefficients have values in GF(2m) for any integer m, and computations are performed using the arithmetic rules in GF(2m).

3.3 ELLIPTIC CURVE CRYPTOGRAPHY (ECC)

A set of cryptographic methods and protocols that rely on modified variants of the discrete logarithm issue for their security are collectively referred to as elliptic curve cryptography, or ECC. No numbers modulo p are used in it. Variants of several cryptographic techniques, like ElGamal encryption and the Digital Signature Algorithm, which were first created for modular numbers are included in ECC.

When applied, two points on an elliptic curve are considered to be the discrete logarithm problem. significantly more difficult. This leads to a change in focus from numbers modulo p to elliptic curve points. If elliptic curve-based variations are used, shorter keys can also achieve an equivalent security level.

There are two advantages to the shorter keys:

a. Ease of key management

b. Efficient computation

The following is an example of key exchange using elliptic curves.

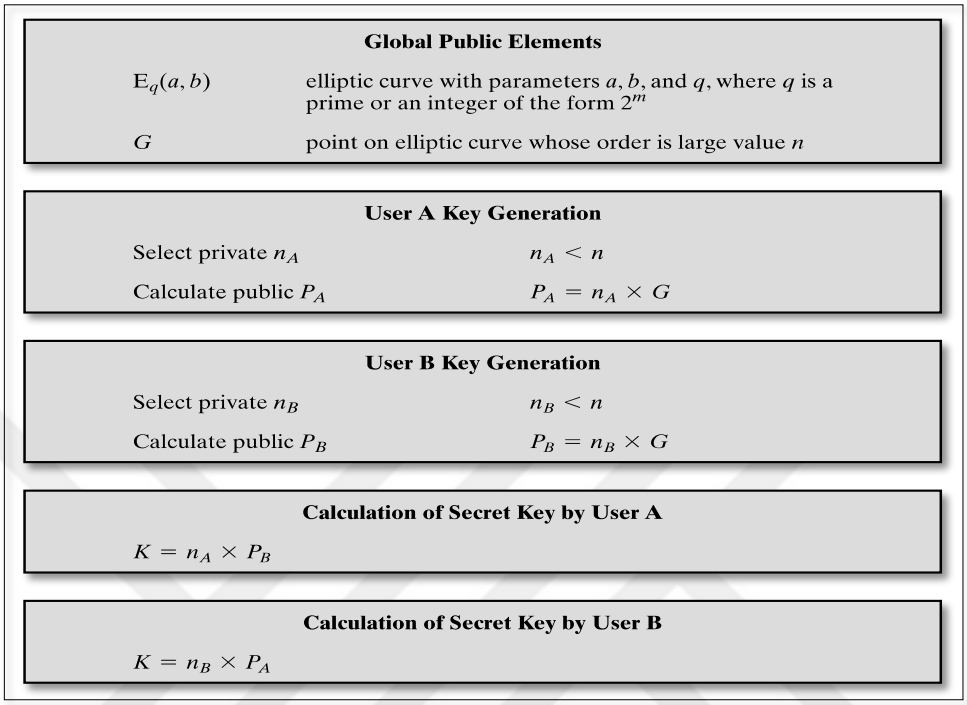

Figure 3.1: Elliptic Curve Encryption/Decryption.

Like key exchange systems, encryption and decryption systems also require a point. As guidelines, consider G and an elliptic group Eq(a, b). Every single A chooses a private key. This produces a key that is unlocked. Equal to nA * G equals PA. When PM-ing B with a message via encryption, A generates the ciphertext Cm, which is made up of the two points, and chooses a positive integer at random, k.

$$C_m = \{kG, P_m + kP_B\} \tag{3.5}$$

Remember that A used the public key of B. PB. To decrypt the ciphertext, B multiplies the first point in the pair by B's private key and subtracts the result from the second point:

$$P_m + kP_B - n_B(kG) = P_m + k(n_B G) - n_B(kG) = P_m \tag{3.6}$$

A has added kPB to the message Pm in order to conceal it. Since only A is aware of the value of k, nobody is able to remove the mask kPB, even if Pb is a public essential. But A also has a "clue," which, if one knows the private key, is enough to take off the mask. B.

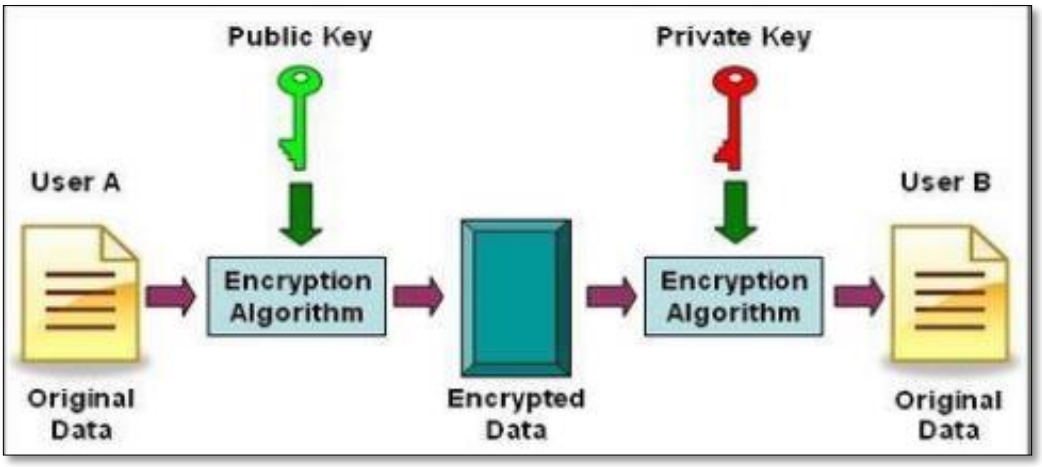

Figure 3.2: The Cryptography.

a. Public key

Public key cryptography is based on ellipse curve field limitations, which are made up of arithmetic functions with finite regions. There are two primary components to public key cryptography: the secret section and the public section [54]. Before a user called "Alice" can send a message to another user, a public key is required named "Bob" [55]. Bob needs to make his public key available to everyone on his homepage if he needs to get in touch with her as well.

Consequently, Alice needs to obtain the public key, encrypt the data with it, and provide it to Bob. Here, Bob can use the code that unlocks the message [56].

As long as the user transmits the fingerprint, The RSA public key approach is linked to the system, indicating that it is the fundamental component of the Asymmetric Crypto-System (ACS). Due to the phase modulation method usage, as long as the security is disturbed, the system can avoid the attack of iterative algorithm.

ECC is an asymmetric cryptosystem that uses public keys. It indicates that the decryption key as well as the encryption key varies. When a secure channel is unavailable for the transmission of the private key, ECC is more suitable than private key cryptography.

b. Private key

The encryption/decryption key, also known as the private or secret key in cryptography, is only known to the persons involved in secret communication exchanges. Prior to the development of secret key cryptography, the communicators could exchange both the encrypted and decrypted message keys.

The primary drawback of this approach is that it becomes inoperable if the key is misplaced or stolen. Later on, this system was modified to combine private and public keys.

For instance, Alice wishes to send Bob a message that requires encryption, and she and Bob both have the same key. Bob will likewise need to XOR Alice's message with her (the same) secret key in order to decode it if Alice shared it with him.

c. Encryption process

By using encryption, it is possible to alter a message or piece of information from its original format so that only individuals with permission can read it. Encryption can be used to hide a message from an interceptor even while it cannot stop data from being altered. The message that is encrypted when employing an encryption technique is known as "cypher text," and it is unintelligible until it is decrypted.

The original text is known as "plain text." The user can hide the original image during the encryption process by adding noise, and they can obtain the original image with less noise during the decryption process. Assume that a randomly generated image with a truly random pixel inside can be interpreted as a onetime pad mechanism that provides unbreakable encryption.

Common authentication procedures are employed to safeguard a lengthy string of numbers commonly referred to as keys. Passwords, tokens, and biometrics (such as fingerprints) are a few of the mechanisms. PEKS is the acronym for public key encryption.

Using the email server, the sender delivered the message to the recipient in this method. The message's content is encrypted using a standard encryption technique. However, the keywords are encrypted using a searchable encryption technique, so even without knowledge of the plain text, the server may use a trapdoor to search for a certain phrase.

To prevent key exchanges between the two parties, public key encryption is employed. Symmetric key encryption is faster than public key encryption because it requires less effort from the computer to decrypt data. Symmetric encryption secures stored data and personal information within the system, but it is not appropriate for external access. It is possible for outsiders to intercept the keys and get access to communications if someone is communicating insecurely.

d. Decryption Process

The process of decoding encrypted information for individuals with authorization is called

decryption. Using the right codes and keys, or manually decrypting the data, is possible with this phrase. Data that is encrypted is more difficult for others to steal. This procedure is known as asymmetric key cryptography because it makes use of separate keys, such as a p

4. METHODOLOGIES

4.1 OVERVIEW

IoT, or the internet of things, has emerged as a crucial aspect of modern life due to the rapid advancement of innovation. Billion devices are linked, transmitting and exchanging data constantly, from wearables and smart homes to connected cars and contemporary gear. Still, as the Internet of Things expands, it becomes a visible target for offensive comedians. The standard cryptographic techniques that ensure data integrity and security in conventional PC architectures might not be feasible for Internet of Things devices due to their resource needs. Elliptic curve cryptography (ECC) provides an alternative that is appropriate for devices with limited computational power, providing comparable security to conventional methods but with shorter key lengths.

4.2 PROPOSED SYSTEM

By leveraging the user's cognitive ability during the authentication process, the suggested method enhances the current one. The first time the user uses the system, he is given a set of points represented on a picture to help him recall the points he has selected by using his cognitive talents. He finds that by keeping these factors in mind, he can authenticate himself the next time, and every other time. When it comes to the encryption procedure, a private key is created for this user based on these chosen criteria.

To do this, an ex-or operation applies to the points. A public key is also produced for this user using the base point and the now-obtained private key. This far, the other verified end has followed suit.

Now, a session key must be generated on both ends and it must be equal for them to be able to connect and essentially validate one another. Both end users are now prepared to speak with one another after creating and confirming the session key. Elliptic curve cryptography is then used to encrypt a message supplied by either user. On the receiving end, this sent communication has now been decoded.

Planning and structuring are done for software, hardware, or an integrated solution during the crucial system design phase of the development process. Determining the architecture, parts, and connections of the system is necessary. Scalability, security, performance maintenance, and the effective achievement of functional objectives are all guaranteed by a

well-executed system design. In the end, it lays the groundwork for effective implementation and operation by considering elements like user experience, maintainability, and adaptability.

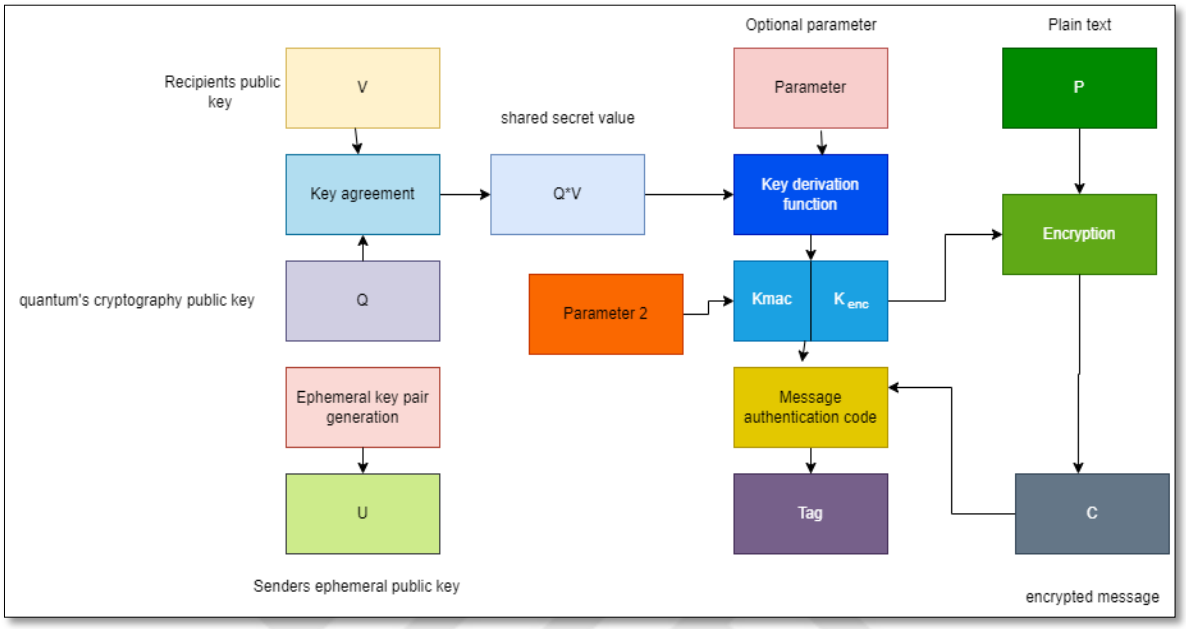

Figure 4.1: System Design (Source: Self-Created).

Installing the library was the first step in this work; it is an essential component of the execution and the framework supporting the ECC library and its features. The study installed the libraries "TensorFlow, NumPy, Hexpytes, Ecies, Matplotlib, Ecdsa, OS Timeit" and used "Google Collab" to carry out the project.

4.3 SYSTEM ARCHITECTUR

4.3.1 Cognitive

The first one focuses on the authentication procedure, where the user is shown with an image that has elliptic curve coordinates scattered across it and is asked to select the desired coordinates. Applying the ex-or function to these chosen coordinates yields the necessary private key.

The user receives his private key and its x coordinate from the ex-or operation, and he uses this information to encrypt data on his end. The receiver is left using his unique private key to decrypt procedure when the same thing occurs on the other end of the transmission

4.3.2 Diffie-Hellman Key Exchange

The Diffie-Hellman key exchange, also known as exponential key exchange, is a digital encryption technique that bases the production of decryption keys on components never readily transmitted. This makes it mathematically impossible for a would-be code breaker to crack the system.

The Diffie-Hellman method, which recommends using a shared secret key on both ends of the transmission, is advantageous to the second module. For user validation on both sides, this key generated on one end must match the key generated on the other. The recipient's public key (PuB) and private key of the sender (PrA) must primarily be multiplied with each other in order to generate the secret key (K). The public key on both sides is produced by multiplying the corresponding private keys (PrA, PrB) by a base point (G) from the provided elliptic curve.

$$\mathrm{PuA} = \mathrm{G}^{*}\mathrm{PrA}\ \ \mathrm{PuB} = \mathrm{G}^{*}\mathrm{PrB} \tag{4.1}$$

The public keys are shared between the sender and the recipient over a channel that is presumed to be either transparent or insecure within the system, hence the name shared-secret key (K). The corresponding private keys and the shared public keys on both ends of the transmission are now used to generate the shared-secret key. The sender side's shared-secret key (K) looks like this:

$$\mathrm{K} = \mathrm{PrA} * \mathrm{PuB} \tag{4.2}$$

The receiver side's shared-secret key (K) looks like this:

$$\mathrm{K} = \mathrm{PrB} * \mathrm{PuA} \tag{4.3}$$

The Diffie-Hellman method is used in this module, and the communication channel formed by These confidential keys on both ends will eventually be identical to one another. Given that they are extremely private and cannot be shared with anybody, the private keys that are crucial to calculating the mutually exclusive key should never be sent over the medium. In terms of protecting the shared secret key, though, it is not nearly as serious of a threat from a hacker because it is most likely a very huge and random number.

4.3.3 Elliptic Curve Encryption and Decryption

Lastly, for the encryption and decryption procedures, elliptic curve cryptography is used. Thinking about an elliptic curve

$$y^2 = (x^3 + ax + b) \mathrm{mod} p \tag{4.4}$$

where p is a prime number that is used to extract the prime field that we intend to work with differ for distinct elliptic curves, and with comes from an infinite field of the elliptic curve.

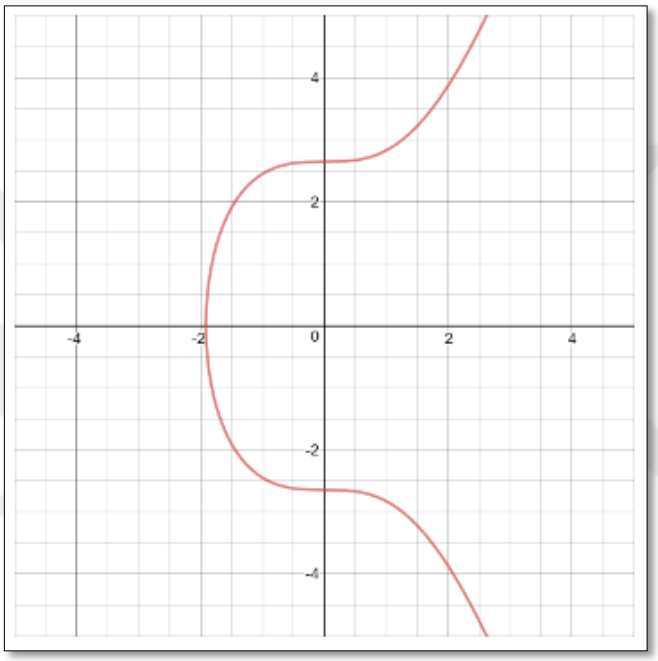

Figure 4.2: Elliptic Curve.

Additionally, the equation must be satisfied by the constants (a) and (b).

$$4a^3 + 27b^2 \neq 0 \tag{4.5}$$

to preserve the non-singularity. The equation describing the encryption process on the sender's end is as follows:

$$Pc = \{KG, Pm + KPuB\} \tag{4.6}$$

where Pm is the point form of the plain text and Pc is the text's cypher form. On the elliptic curve, the plain text is encoded as a point form. Every character in the plain text is taken into consideration and mapped to an elliptic curve point to produce the point form of.

Getting the relevant the character's ASCII value and multiplication it by the curve's base point G is one method to find the stated point for an elliptic curve character. To get a new point in this procedure, addition and doubling techniques are required. Pm is the obtained point.

The decryption procedure occurs after the encryption process, meaning that the encrypted character is immediately decrypted without having to await the encryption of the entire text. This is how the decryption formula works:

$$\mathrm{Pm} = \{(\mathrm{Pm} + \mathrm{KPuB}) - \mathrm{PrBKG}\} \tag{4.7}$$

4.4 IMPLEMENTATION

We have used TensorFlow to create an XOR cypher for text encryption and decoding. It first transforms the input and key strings into binary representations, uses TensorFlow to carry out bitwise XOR operations, and finally transforms the outcome back into a string that can be understood by humans. To ensure that the XOR cypher works properly, the code contains a test case in which plain text is encrypted and successfully decrypted. Using XOR, a key cryptographic technique, this code offers a simple example of text encryption and decryption.

```
Algorithm XOR_Cipher(input_string, key_string):
    input_binary = StringToBinary(input_string)
    key_binary = StringToBinary(key_string)
    input_tensor = ConvertToTensor(input_binary)
    key_tensor = ConvertToTensor(key_binary)
    xor_result_tensor = PerformXOR(input_tensor, key _tensor)
    xor_result = TensorToString(xor_result_tensor)
    return xor_result
Function StringToBinary(input_string):
    binary_list[] =
    for each character in input_string:
        binary_representation = ConvertCharToBinary(character)
        Append binary_representation to binary_list
    return binary_list
```

Figure 4.3: Algorithm XOR Cypher (Source: Self-Created).

Elliptic Curve Cryptography (ECC) is used for encryption and decryption in the Ethereum

ecosystem, as demonstrated by Python code. After encrypting a plaintext message and successfully decrypting it, it generates private and public keys compatible with Ethereum. In order to provide safe data transmission and reception, the code, which depends on the 'ecies' library, shows how ECC-based security can be implemented practically for Ethereum applications.

```
Algorithm ECC_Encryption _Decryption:()
    private_key = GeneratePrivateKey()
    public _key = DerivePublicKeyFrom(private_key)
        Display "Private key:", private   key
        Display "Public key:", public _key
        plaintext = DefinePlaintext()
        encrypted_data = Encrypt(public _key, plaintext)
        Display "Encrypted data:", encrypted_data
        decrypted_data = Decrypt(private_key, encrypted_data)
        Display "Decrypted data:", decrypted_data
Function GeneratePrivateKey:()
    return GenerateETHKey()
Function DerivePublicKeyFrom(private_key):
    return GetPublicKeyFrom(private_key)
Function DefinePlaintext:()
    return "Hello, ECC encryption"!
```

Figure 4.4: ECC Encryption and Decryption Algorithm (Source: Self-Created).

ECC encryption and Ethereum key visualization are demonstrated by Python code. It creates public and private keys that work with Ethereum and encrypts and decrypts textual messages. It can also be used Matplotlib to visualize the public key, private key, and some encrypted data as binary pictures. The use of ECC encryption is illustrated through the graphic representation of keys and encrypted data.

```
Algorithmn ECC_Visualization():
  private_key = GeneratePrivateKey()
    public_key = DerivePublicKeyFrom(private _key)
    Display "Private key:", private_key
    Display "Public key:", public_key
    plaintext = DefinePlaintext()
    encrypted_data = Encrypt(public key, plaintext)
    Display "Encrypted data:", encrypted_data
    DisplayVisualizations(private_ key, public_key, enerypted_data)
Function GeneratePrivateKey():
    return GenerateETHKey()
Function DerivePublicKeyFrom(private_key):
    return GetPublicKeyFrom(private key)
Function DefinePlaintext():
    return "Hello, ECC encryption!"
```

Figure 4.5: Algorithm ECC Visualization (Source: Self-Created).

Elliptic Curve Digital Signature Algorithm (ECDSSA) key generation, message signing, and verification are demonstrated by Python code. In order to sign a message and accurately verify it with the public key, it first generates a public key that is compatible with the ECDSA private key. Validity and communication integrity are guaranteed by the ECDSA technique.

```
Algorithm ECDSA  Signature  Verification():
    Private_key = GenerateSigningKey() # Generates a private key
    public_ key = DeriveVerifyingKeyFrom(private_key) # Derives the corresponding
public key
    message = DefineMessage() # Defines the message to be signed
    signature = SignMessage(private_key, message) # Signs the message using the private
key
    VerifySignature(public_key, signature, message) # Verifies the signature using the
public key
    Display "Message Verified!"
Function GenerateSigningkey():
    return GenerateSigningKeyUsingCurve(NIST384p)
Function DeriveVerifymgReyFrom(private_key):
    return GetVerifyingKeyFrom(private_key)
Function DefineMessage():
    return "Hello, ECC for loT!"
```

Figure 4.6: Algorithm ECDSA Signature Verification (Source: Self-Created).

```
Algorithm ECC_Key_Sign Visualization):
    private_key - GenerateSigningKey) # Generates a private key
    public_key = DeriveVerifyingKeyFrom(private_key) # Derives the corresponding
public key
    message = DefineMessage) # Defines the message to be signed
    signature = SignMessage(private key, message) # Signs the message using the private
key
    VerifySignature(public_key, signature, message) # Verifies the signature using the
public key
    VisualizeKeys AndSignature(private_key, public key, signature) # Visualizes keys and
signature
Function GenerateSigningKey():
    return GeneraleSigningKeyUsingCurve(NIST384p)
Function Derive VerifyingKeyFrom(private key):
    return GetVerifyingKeyFrom(private_key)
Function DefineMessage():
    return "Hello, ECC for loI!"
```

Figure 4.7: Algorithm ECC Key Sign Visualization (Source: Self-Created).

5. RESULTS AND DISSCUSISION

5.1 RESULT ANALYSIS

Using the Elliptic Curve Digital Signature Algorithm, the provided Python code produces keys, signs messages, and displays a mocked-up bar plot visualization of the generated keys and signature values. It generates the public and private keys for the ECDSA, signs a message, and verifies the signature. In the visualization part, Matplotlib bar graphs are used to depict the byte values of the private key, public key, and signature. Each bar graph shows the distribution of the byte values. The byte values are displayed using bar graphs for educational purposes.

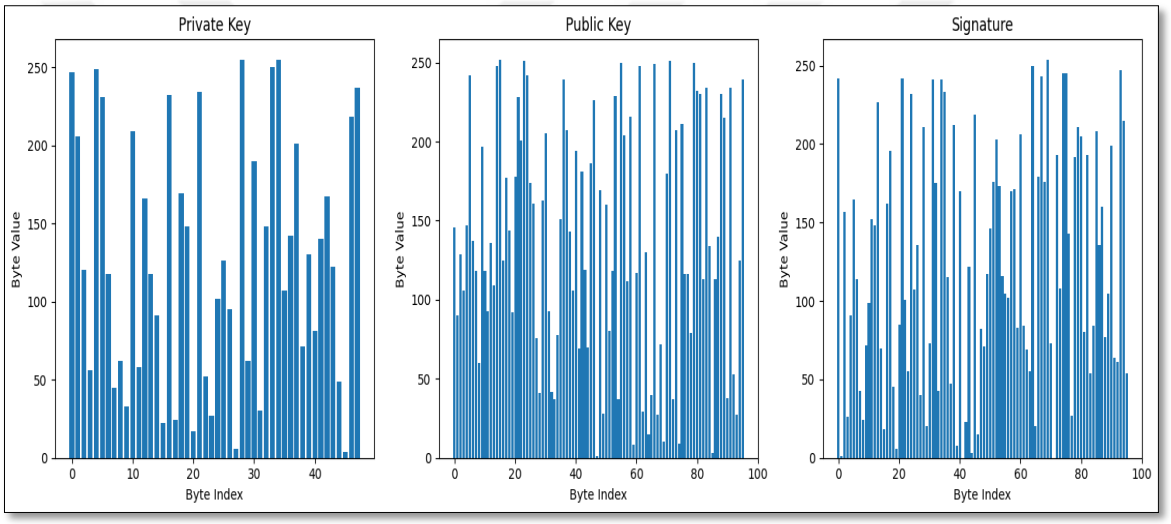

Figure 5.1: ECDSA Key Generation, Signing, and Visualization with Mocked Key Representation (Source: Self-Created in Google Colab).

An extensive example of creating, signing, and validating ECDSA keys using the NIST384p curve It also compares the key sizes of RSA and ECC to show the benefits of RSA over ECC. Moreover, it represents a simple elliptic curve for illustrative purposes, even if it is not the actual curve utilized in ECC cryptography. The elliptic curve plot and bar charts that the code produces with Matplotlib offer helpful details regarding the benefits and practical applications of ECC.

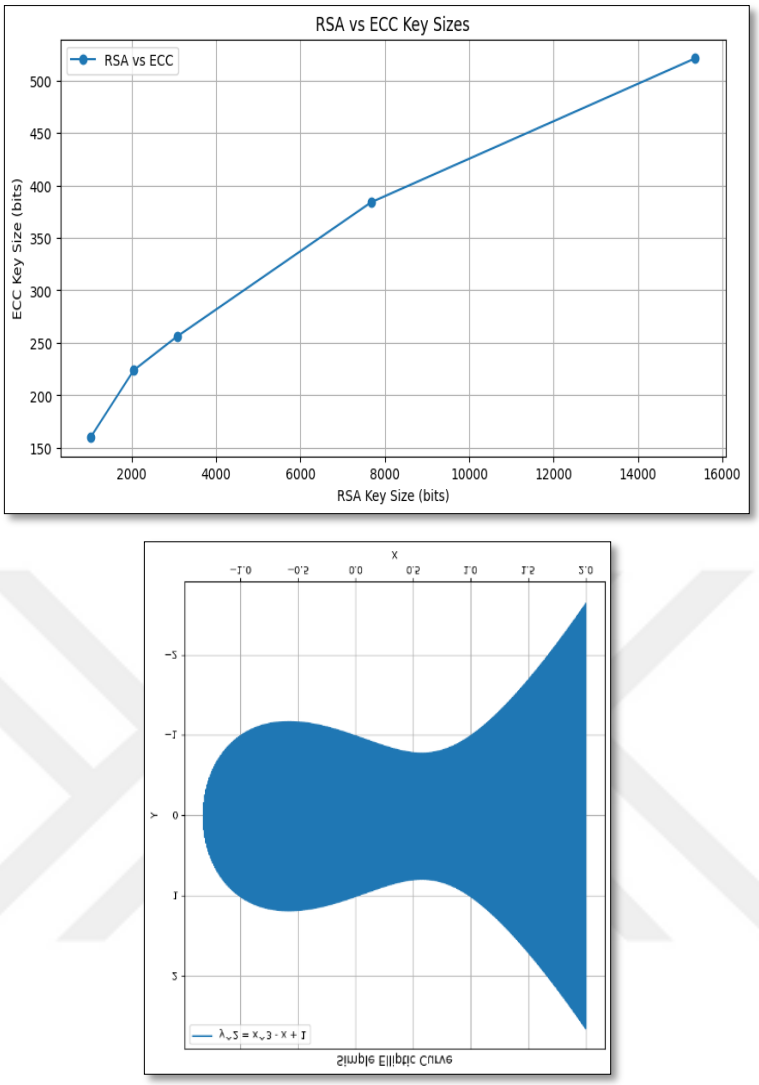

Figure 5.2: Comparison of Key Sizes: RSA vs. ECC and Visualization of a Simple Elliptic Curve (Source: Self-Created in Google Colab).

It is also added illustrates point addition and point doubling operations using elliptical curves created by the formula y^2 = x^3 + axe + b. The operation of addition of points (P + Q) and doubling of points (2P) are shown, and a graphic representation of the curve is provided. By using Matplotlib to plot the curve, and sample points, and visualize the results of these fundamental processes, the code aids in understanding the fundamentals of elliptic curve encryption.

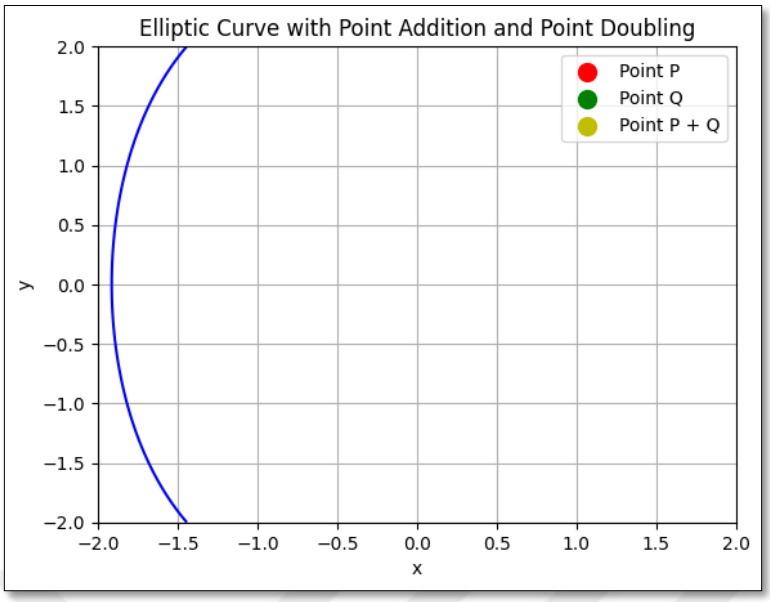

Figure 5.3: Point Addition and Point Doubling on an Elliptic Curve (Source: Self-Created in Google Colab).

Evaluates the effectiveness of various ECC key sizes (256, 384, and 521 bits) for the purposes of signing and verifying messages. For each key size, it performs these actions 1000 times using the {timeit` module, recording the execution times. A bar plot is used to visualize the findings and show how different ECC key sizes work. Understanding the computational expense of ECC operations is made easier by this study.

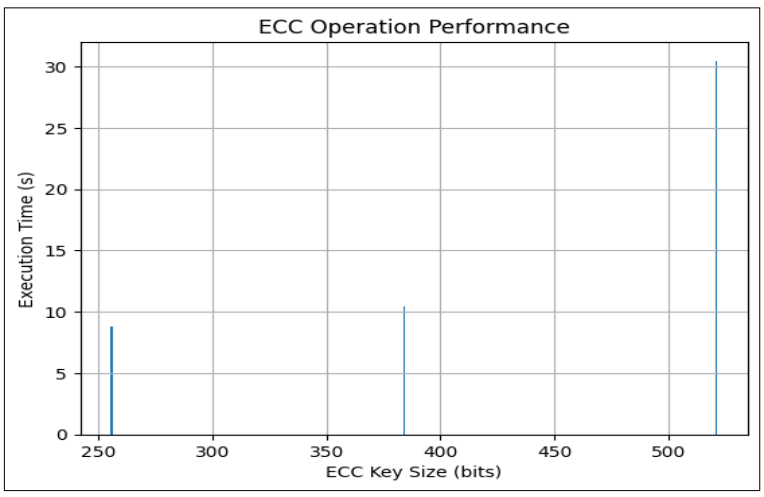

Figure 5.4: Performance Comparison of ECC Operations (Source: Self-Created in Google Colab).

The Elliptic Curve Cryptography (ECC) and classic Rivest-Shamir-Adleman (RSA) key sizes are contrasted for Internet of Things security. It makes use of Matplotlib to plot key sizes, showing how ECC provides comparable or enhanced security at much smaller key sizes, which makes it more appropriate for IoT devices with limited resources.

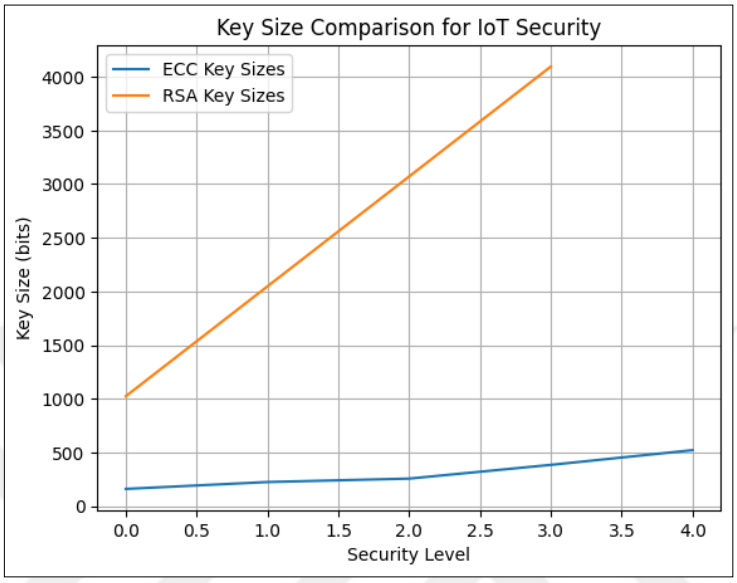

Figure 5.5: Key Size Comparison for IoT Devices (Source: Self-Created in Google Colab).

The private and public keys are transformed into hexadecimal strings, which also creates an ECC (Elliptic Curve Cryptography) key pair. It creates binary graphics as visual representations of these key values using Matplotlib. While this representation is not how ECC keys are normally visualized, these visualizations offer an instructional symbolic picture of the byte values of the private and public keys.

5.2 DISCUSSION

A crucial part of every project, study, or research is the discussion of the results. To provide insights, interpretations, and consequences for the stakeholders, it entails a methodical and critical analysis of the results and conclusions. In a result, discussion, the presentation of the data or observations are usually followed by an exploration of their significance. The main emphasis is on elucidating the findings, their pertinence to the study inquiries or goals, and their conformity with extant literature to create a cohesive story, researchers frequently examine patterns, trends, variances, or unexpected results. The purpose of this step is to discuss the strengths and limits of the research and to identify any biases or uncertainties.

A comparison of the results with those from earlier research may also be included, which could support or refute the findings. The practical consequences of the results can be emphasized through a discussion that includes suggestions for future actions, policy modifications, or additional studies. It serves as a place to consider the work's value in a larger perspective as well. Researchers talk about how their discoveries advance the subject, whether they provide any fresh perspectives, and how they might affect practical applications.

Table 5.1: Compare Between this Project and Another Project.

Aspect	Current Project (ECC-based)	Another Project (Reference Project)
Objective	Secure Smart cards and ECC are used for remote user authentication in a multi-server architecture.	Remote user authentication with an emphasis on biometric authentication and traditional cryptography.
Cryptography Method	Utilizes Elliptic Curve Cryptography (ECC), offering reduced key sizes and robust security.	Relies on traditional cryptography methods, such as RSA, for encryption and authentication (De Rango et al. 2020) [57].
Security Measures	Prevents man-in-the-middle attacks, offline dictionary attacks, and more through multi-factor authentication and secure hash functions.	Focuses on biometric authentication to enhance security. May have fewer measures against specific cryptographic attacks.
Computational Efficiency	Effective in terms of communication and computing expenses, suitable for implementation on smart cards with limited resources.	May have higher computational demands due to traditional cryptography, potentially posing challenges for resource-constrained devices.
Authentication Phases	Consists of registration, precomputation, phases of login, session key agreement, authentication, and password changes.	May have a different set of authentication phases, depending on the specific biometric authentication techniques used.
Protection Against Attacks	Effective protection against man-in-the-middle attacks and offline dictionary attacks.	Protection may be more tailored to the specific security needs of the chosen biometric authentication methods.

6. CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

ECC for IoT security may be implemented, as the project has successfully shown, and it provides effective encryption and decryption. IoT devices with limited resources are a good fit for ECC due to its advantages, which include lower key sizes. Additional investigation and refinement may improve its usefulness.

6.2 FUTURE WORK

a. Performance Optimization: Reduce energy consumption and computational overhead while increasing the efficiency of ECC procedures for IoT devices.
b. Security analysis: Find potential weaknesses and look at adding more levels of protection by doing a thorough security assessment.
c. Key management: Create strong key management protocols to guarantee safe key distribution, creation, and storage in Internet of Things networks.
d. Standardization: Participate in IoT security standardization initiatives to guarantee best practices compliance and interoperability.
e. Integration: Examine how ECC can be seamlessly integrated into IoT frameworks and platforms to promote wider acceptance and implementation.
f. Quantum-Resistant ECC: Research and create quantum-resistant ECC versions to protect against potential threats from quantum computing in the future.

REFERENCES

[1] O. Vermesan and P. Friess, Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems, River Publishers Series in Communications, Aalborg, (2013).

[2] R. Roman, P. Najera and J. Lopez, Securing the Internet of Things, IEEE Computer, Vol.44, 51-58 (2011).

[3] M. Abomhara and G. M. Koien: Security and privacy in the Internet of Things, Current status and open issues, in Int'l Conference on Privacy and Security in Mobile Systems (PRISMS), 1-8, (2014).

[4] "IEEE Standard, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," IEEE Std 802.11-2012 (Revision of IEEE Std 802.112007), pp. 1–2793, March 2012.

[5] Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, ANSI/IEEE Std 802.11, LAN/MAN Standards Committee of the IEEE Computer Society Std., 2012.

[6] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," ACM Special Interest Group on Data Communication Computer Communication Review, vol. 38, no. 2, pp. 69–74, 2008.

[7] R. Riggio, M. K. Marina, J. Schulz-Zander, S. Kuklinski, and T. Rasheed, "Programming abstractions for software-defined wireless networks," IEEE Transactions on Network and Service Management, vol. 12, no. 2, pp. 146–162, 2015.

[8] L. Suresh, J. Schulz-Zander, R. Merz, A. Feldmann, and T. Vazao, "Towards Programmable Enterprise WLANS with Odin," in Proc. of ACM Workshop on Hot Topics in Networks, New York, 2012.

[9] R. H. Weber, "Internet of things–new security and privacy challenges," Computer law & security review, vol. 26, no. 1, pp. 23–30, 2010.

[10] A. Riahi, Y. Challal, E. Natalizio, Z. Chtourou, and A. Bouabdallah, “A systemic approach for iot security,” in Distributed Computing in Sensor Systems (DCOSS), 2013 IEEE International Conference on. IEEE, 2013, pp. 351–355.

[11] E. Ronen and A. Shamir, “Extended functionality attacks on iot devices: The case of smart lights,” in Security and Privacy (EuroS&P), 2016 IEEE European Symposium on. IEEE, 2016, pp. 3–12.

[12] “Industrial Internet Consortium: Industrial Internet of Things Volume G4: Security Framework,” http://www.iiconsortium.org/pdf/IIC PUB G4 V1.00 PB-3.pdf, 2016, accessed on 4 Feb. 2018.

[13] B. Tang, Z. Chen, G. Hefferman, S. Pei, T. Wei, H. He, and Q. Yang, “Incorporating intelligence in fog computing for big data analysis in smart cities,” IEEE Transactions on Industrial Informatics, vol. 13, no. 5, pp. 2140–2150, Oct 2017.

[14] Y. N. Singh, “Human recognition using fisher’s discriminant analysis of heartbeat interval features and ecg morphology,” Neurocomputing, vol. 167, pp. 322–335, 2015.

[15] W. Louis, M. Komeili, and D. Hatzinakos, “Continuous authentication using onedimensional multi-resolution local binary patterns (1dmrlbp) in ecg biometrics,” IEEE Transactions on Information Forensics and Security, vol. 11, no. 12, pp. 2818–2832, 2016.

[16] D. Ramli, M. Hooi, and K. Chee, “Development of heartbeat detection kit for biometric authentication system,” Procedia Computer Science, vol. 96, pp. 305–314, 2016.

[17] V. Yano, A. Zimmer, and L. L. Ling, “Extraction and application of dynamic pupillometry features for biometric authentication,” Measurement, vol. 63, pp. 41–48, 2015.

[18] C. Galdi, M. Nappi, D. Riccio, and H. Wechsler, “Eye movement analysis for human authentication: a critical survey,” Pattern Recognition Letters, vol. 84, pp. 272–283, 2016.

[19] S. Sarkar, V. M. Patel, and R. Chellappa, “Deep feature-based face detection on mobile devices,” arXiv preprint arXiv:1602.04868, 2016.

[20] P. Samangouei, V. M. Patel, and R. Chellappa, “Attribute-based continuous user authentication on mobile devices,” in Biometrics Theory, Applications and Systems (BTAS), 2015 IEEE 7th International Conference on. IEEE, 2015, pp. 1–8.

[21] K. Annapurani, M. Sadiq, and C. Malathy, “Fusion of shape of the ear and tragus– a unique feature extraction method for ear authentication system,” Expert Systems with Applications, vol. 42, no. 1, pp. 649–656, 2015.

[22] H. Crawford, K. Renaud, and T. Storer, “A framework for continuous, transparent mobile device authentication,” Computers & Security, vol. 39, pp. 127–136, 2013.

[23] M. Baloul, E. Cherrier, and C. Rosenberger, “Challenge-based speaker recognition for mobile authentication,” in Biometrics Special Interest Group (BIOSIG), 2012 BIOSIG-Proceedings of the International Conference of the. IEEE, 2012, pp. 1–7.

[24] A. Kumar, S. Garg, and M. Hanmandlu, “Biometric authentication using finger nail plates,” Expert systems with applications, vol. 41, no. 2, pp. 373–386, 2014.

[25] S. H. Khan, M. A. Akbar, F. Shahzad, M. Farooq, and Z. Khan, “Secure biometric template generation for multi-factor authentication,” Pattern Recognition, vol. 48, no. 2, pp. 458–472, 2015.

[26] H. Saevanee, N. Clarke, S. Furnell, and V. Biscione, “Continuous user authentication using multi-modal biometrics,” Computers & Security, vol. 53, pp. 234–246, 2015.

[27] M. L. Brocardo, I. Traore, and I. Woungang, “Authorship verification of e-mail and tweet messages applied for continuous authentication,” Journal of Computer and System Sciences, vol. 81, no. 8, pp. 1429–1440, 2015.

[28] L. Fridman, A. Stolerman, S. Acharya, P. Brennan, P. Juola, R. Greenstadt, and

M. Kam, “Multi-modal decision fusion for continuous authentication,” Computers & Electrical Engineering, vol. 41, pp. 142–156, 2015.

[29] Y. Matsuyama, M. Shozawa, and R. Yokote, “Brain signal’s low-frequency fits the continuous authentication,” Neurocomputing, vol. 164, pp. 137–143, 2015.

[30] C. Ntantogian, S. Malliaros, and C. Xenakis, “Gaithashing: a two-factor authentication scheme based on gait features,” Computers & Security, vol. 52, pp. 17–32, 2015.

[31] S. Choi, I.-H. Youn, R. LeMay, S. Burns, and J.-H. Youn, “Biometric gait recognition based on wireless acceleration sensor using k-nearest neighbor classification,” in Computing, Networking and Communications (ICNC), 2014 International Conference on. IEEE, 2014, pp. 1091–1095.

[32] N. Koblitz, Elliptic Curve Cryptosystems, Mathematics of Computation, Vol.49, 203-209, (1987).

[33] V.S. Miller, Use of Elliptic Curves in Cryptography, Advances in Cryptology CRYPTO85, Lecture Notes in Computer Science, Springer Verlag, Vol.128, 417-426,(1985).

[34] W. Diffie and M. Hellman, “New directions in cryptography,” IEEE Trans. Inf. Theor., vol. 22, no. 6, pp. 644–654, Sep. 2006. doi:10.1109/TIT.1976.1055638. [Online]. Available: http://dx.doi.org/10.1109/TIT.1976.1055638.

[35] A. J. Menezes, S. A. Vanstone, and P. C. V. Oorschot, Handbook of Applied Cryptography, 1st ed. Boca Raton, FL, USA: CRC Press, Inc., 1996. ISBN 0849385237.

[36] A. Wander, N. Gura, H. Eberle, V. Gupta, and S. Shantz, “Energy analysis of public-key cryptography for wireless sensor networks,” in Third IEEE International Conference on Pervasive Computing and Communications, 2005. PerCom 2005, Mar. 2005. doi: 10.1109/PERCOM.2005.18 pp. 324–328.

[37] R. L. Rivest, A. Shamir, and L. Adleman, “A method for obtaining digital signatures and public-key cryptosystems,” Commun. ACM, vol. 21, no. 2, pp. 120–126, Feb. 1978. doi: 10.1145/359340.359342. [Online]. Available:
http://doi.acm.org/10.1145/359340.359342

[38] B. Barker, W. C. Barker, W. E. Burr, W. T. Polk, and M. E. Smid, “SP 800-57. Recommendation for Key Management, Part 1: General (Revision 3),” Gaithersburg, MD, United States, Tech. Rep., 2012, accessed: 2014-09-10. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57 part1 rev3 general.pdf

[39] V. S. Miller, “Use of elliptic curves in cryptography,” in Lecture Notes in Computer Sciences; 218 on Advances in cryptology—CRYPTO 85. New York, NY, USA: Springer-Verlag New York, Inc., 1986. ISBN 0-387-16463-4 pp. 417–426. [Online]. Available: http://dl.acm.org/citation. cfm?id=18262.25413;

[40] N. Koblitz, “Elliptic curve cryptosystems,” Math. Comp., vol. 48,no. 177, pp. 203–209, 1987. doi: 10.1090/S0025-5718-1987-0866109-5. [Online]. Available: http://www.ams.org/mcom/1987-48-177/S0025-5718-1987-0866109-5/

[41] L. C. Washington, Elliptic Curves: Number Theory and Cryptography.Boca Raton, FL, USA: CRC Press, Inc., 2003. ISBN 1584883650.

[42] J. Lagrange, “R´eflexions sur la r´esolution alg´ebrique des ´equations,” in Nouveaux M´emoires de l’Acad´emie Royale des Sciences et Belles-Lettres de Berlin, 1971, pp. 138–254.

[43] K. Igoe, D. McGrew, and M. Salter, “Fundamental elliptic curve cryptography algorithms,” accessed: 2014-01-13. [Online]. Available: http://tools.ietf.org/html/rfc6090.

[44] IEEE standard specifications for public-key cryptography,” IEEE Std 1363- 2000, pp. 1–228, Aug. 2000. doi: 10.1109/IEEESTD.2000.92292.

[45] M. Joye, “Fast point multiplication on elliptic curves without precomputation,” in Proceedings of the 2Nd International Workshop on Arithmetic of Finite Fields, ser. WAIFI ’08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 36–46.

[46] T. El Gamal, “A public key cryptosystem and a signature scheme based on discrete logarithms,” in Proceedings of CRYPTO 84 on Advances in Cryptology. New York, NY, USA: Springer-Verlag New York, Inc., 1985. ISBN 0-387-15658-5 pp. 10–18.

[47] D. Hankerson, A. J. Menezes, and S. Vanstone, Guide to Elliptic Curve Cryptography. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2003. ISBN 038795273X.

[48] T. T. . x. W. Group, “TinyOS 2.0,” in Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems, ser. SenSys ’05. New York, NY, USA: ACM, 2005. doi: 10.1145/1098918.1098985. ISBN 1-59593-054-X pp. 320–320. [Online]. Available: http://doi.acm.org/10. 1145/1098918.1098985.

[49] A. Liu and P. Ning, “TinyECC: A configurable library for elliptic curve cryptography in wireless sensor networks,” in International Conference on Information Processing in Sensor Networks, 2008. IPSN ’08, Apr. 2008. doi: 10.1109/IPSN.2008.47 pp. 245–256.

[50] Moghaddam, Faraz Fatemi, et al. "A scalable and efficient user authentication scheme for cloud computing environments." Region 10 Symposium, 2014 IEEE. IEEE, 2014.

[51] Singh, Ashish, and Kakali Chatterjee. "A secure multi-tier authentication scheme in cloud computing environment." Circuit, Power and Computing Technologies (ICCPCT), 2015 International Conference on. IEEE, 2015.

[52] Yang, Jen Ho, and Pei Yu Lin. "An ID-based user authentication scheme for cloud computing." Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), 2014 Tenth International Conference on. IEEE, 2014.

[53] Chen, Tien-Ho, Hsiu-lien Yeh, and Wei-Kuan Shih. "An advanced ECC dynamic id-based remote mutual authentication scheme for cloud computing." Multimedia and Ubiquitous Engineering (MUE), 2011 5th FTRA International Conference on. IEEE, 2011.

[54] K. Shankar, G. Devika and M. Ilayaraja, "Secure and Efficient Multi-Secret Image Sharing Scheme based on Boolean Operations and Elliptic Curve Cryptography", International Journal of Pure and Applied Mathematics, Volume 116, Issue. 10, page(s): 293-300, October 2017.

[55] M. Ilayaraja, K. Shankar and G. Devika, "A Modified Symmetric Key Cryptography Method for Secure Data Transmission", International Journal of Pure and Applied Mathematics, Volume 116, Issue. 10, page(s): 301-308, October 2017.

[56] G. Devika, M. Ilayaraja and K. Shankar, "Optimal Radial Basis Neural Network (ORBNN) For Effective Classification of Clouds in Satellite Images with Features", International Journal of Pure and Applied Mathematics, Volume 116, Issue. 10, page(s): 309-329, October 2017.

[57] De Rango, F., Potrino, G., Tropea, M., & Fazio, P. (2020). Energy-aware dynamic Internet of Things security system based on Elliptic Curve Cryptography and Message Queue Telemetry Transport protocol for mitigating Replay attacks. Pervasive and Mobile Computing, 61, 101105. https://doi.org/10.1016/j.pmcj.2019.101105

APPENDIX A

```
# Install TensorFlow if necessary
#!pip install tensorflow

import tensorflow as tf
import numpy as np

def string_to_binary(string):
    return [format(ord(char), '08b') for char in string]

def binary_to_string(binary_list):
    return ''.join([chr(int(binary, 2)) for binary in binary_list])

def xor_binary(binary_tensor, key_tensor):
    binary_list = binary_tensor.numpy().tolist()
    key_binary = key_tensor.numpy().tolist()[0]  # Assuming the key length is 1␣
 ↪for simplicity

    xor_result = [''.join([str(int(b) ^ int(k)) for b, k in zip(binary,␣
 ↪key_binary)]) for binary in binary_list]

    return np.array(xor_result)

# Define XOR cipher using TensorFlow
def xor_cipher_tf(input_string, key_string):
    # Convert string to binary
    input_binary = string_to_binary(input_string)
    key_binary = string_to_binary(key_string)

    # Convert to tensors
    input_tensor = tf.convert_to_tensor(input_binary, dtype=tf.string)
    key_tensor = tf.convert_to_tensor(key_binary, dtype=tf.string)
```

```python
import matplotlib.pyplot as plt
from ecies import encrypt, decrypt
from ecies.utils import generate_eth_key
from hexbytes import HexBytes

# Generate private and public keys
private_key = generate_eth_key()
public_key = private_key.public_key

print(f"Private key: {private_key.to_hex()}")
print(f"Public key: {public_key.to_hex()}")

# Original plaintext
plaintext = b'Hello, ECC encryption!'

# Encryption
encrypted = encrypt(public_key.to_hex(), plaintext)
print(f"\nEncrypted data: {HexBytes(encrypted)}")

# Decryption
decrypted = decrypt(private_key.to_hex(), encrypted)
print(f"\nDecrypted data: {decrypted.decode()}")

# Visualization
fig, ax = plt.subplots(1, 3, figsize=(15, 5))
```

APPENDIX B

```
import os
from ecdsa import SigningKey, NIST384p
import matplotlib.pyplot as plt

# 1. Key Generation
sk = SigningKey.generate(curve=NIST384p) # private key
vk = sk.get_verifying_key() # public key

# 2. Message Signing
message = b"Hello, ECC for IoT!"
signature = sk.sign(message)

# 3. Message Verification
assert vk.verify(signature, message)

# 4. Plotting keys and message

# This is a mock visualization. In reality, ECC keys are not represented this␣
 ↪way.
private_key_values = list(sk.to_string())
public_key_values = list(vk.to_string())
signature_values = list(signature)

plt.figure(figsize=(15,5))
```

```
plt.subplot(1, 3, 1)
plt.bar(range(len(private_key_values)), private_key_values)
plt.title('Private Key')
plt.xlabel('Byte Index')
plt.ylabel('Byte Value')

plt.subplot(1, 3, 2)
plt.bar(range(len(public_key_values)), public_key_values)
plt.title('Public Key')
plt.xlabel('Byte Index')
plt.ylabel('Byte Value')

plt.subplot(1, 3, 3)
plt.bar(range(len(signature_values)), signature_values)
plt.title('Signature')
plt.xlabel('Byte Index')
plt.ylabel('Byte Value')

plt.tight_layout()
plt.show()
```

```
import os
from ecdsa import SigningKey, NIST384p, NIST256p
import matplotlib.pyplot as plt
import numpy as np

# Key Generation, Signing, and Verification as previously discussed

sk = SigningKey.generate(curve=NIST384p)
vk = sk.get_verifying_key()
message = b"Hello, ECC for IoT!"
signature = sk.sign(message)
```

APPENDIX C

```
assert vk.verify(signature, message)

# 1. Comparison of Key Sizes

# Simulated key sizes for illustration
rsa_key_sizes = [1024, 2048, 3072, 7680, 15360]
ecc_key_sizes = [160, 224, 256, 384, 521]

plt.figure(figsize=(10,5))
plt.plot(rsa_key_sizes, ecc_key_sizes, 'o-', label='RSA vs ECC')
plt.title('RSA vs ECC Key Sizes')
plt.xlabel('RSA Key Size (bits)')
plt.ylabel('ECC Key Size (bits)')
plt.grid(True)
plt.legend()
plt.show()

# 2. Elliptic Curve Plotting

def elliptic_curve(a, b, range_x):
    """Simple elliptic curve plot y^2 = x^3 + ax + b"""
    y_values = []
    x_values = []
    for x in range_x:
        y2 = x**3 + a*x + b
        y = np.sqrt(y2)
        x_values.append(x)
        y_values.append(y)
        x_values.append(x)
        y_values.append(-y) # negative root
    return x_values, y_values
```

```python
import matplotlib.pyplot as plt
import numpy as np

# Define an elliptic curve
a = 0
b = 7
x = np.linspace(-2, 2, 400)
y = np.linspace(-2, 2, 400)

X, Y = np.meshgrid(x, y)
F = Y**2 - X**3 - a*X - b


# Sample points for addition and doubling
P = (0.5, np.sqrt(0.25**3 - a*0.25 - b))
Q = (1.5, np.sqrt(1.5**3 - a*1.5 - b))

# Calculate the result of point addition P + Q
R = (P[0], -P[1] - Q[1])  # Mirror image of P+Q with respect to the x-axis

# Visualize the elliptic curve
plt.contour(X, Y, F, [0], colors='b')
plt.scatter(*P, color='r', label='Point P', s=100)
plt.scatter(*Q, color='g', label='Point Q', s=100)
plt.scatter(*R, color='y', label='Point P + Q', s=100)
plt.legend()
plt.grid(True)
plt.title('Elliptic Curve with Point Addition and Point Doubling')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

```python
import matplotlib.pyplot as plt

# Key sizes for ECC and traditional methods (RSA) in bits
ecc_key_sizes = [160, 224, 256, 384, 521]
rsa_key_sizes = [1024, 2048, 3072, 4096]

plt.plot(ecc_key_sizes, label='ECC Key Sizes')
plt.plot(rsa_key_sizes, label='RSA Key Sizes')
plt.xlabel('Security Level')
plt.ylabel('Key Size (bits)')
plt.title('Key Size Comparison for IoT Security')
plt.grid(True)
plt.legend()
plt.show()
```