DEPARTMENT OF COMPUTER SCIENCE ENGINEERING,
SHARDA SCHOOL OF ENGINEERING AND TECHNOLOGY,
SHARDA UNIVERSITY, GREATER NOIDA

SECURITY, TRUST AND PRIVACY ISSUES IN IOT

A project submitted
in partial fulfillment of the requirements for the degree of
Bachelor of Technology in Computer Science and Engineering

by

Tanya Lillian Borges (2019639262)
Ronika Shrestha (2019003290)
Kanishka Negi (2019600654)

Supervised by:

Dr. Subrata Sahana, Associate Professor

May, 2023

CERTIFICATE

This is to certify that the report entitled **"Security, Privacy, and Trust Issues in IoT"** submitted by **"TANYA LILLIAN BORGES (2019639262), KANISHKA NEGI (2019600654), RONIKA SHRESTHA (2019003290)"** to Sharda University, towards the fulfilment of requirements of the degree of **"Bachelor of Technology"** is record of bonafide final year Project work carried out by them in the **"Department of Computer Science & Engineering, Sharda School of Engineering and Technology, Sharda University".**

The results/findings contained in this Project have not been submitted in part or full to any other University/Institute forward of any other Degree/Diploma.

Signature of the Guide

Name: Dr. Subrata Sahana

Designation: Associate Professor

Signature of Head of Department

Name: Prof.(Dr.)Nitin Rakesh

Place: Sharda University

Date:

Signature of External Examiner

Date:

ACKNOWLEDGEMENT

I would like to express my deepest appreciation to all those who provided me the possibility to complete this report. Apart from the efforts of myself, the success of any project depends largely on the encouragement and guidelines of many others. I take this opportunity to express my gratitude to the people who have been instrumental in the successful completion of this project. First and foremost, we would like to thank Dr. Nitin Rakesh, HOD, CSE who gave us an opportunity to undertake this project. I would like to show my greatest appreciation to Dr. Subrata Sahana. I can't say thank you enough for his tremendous support and help. I feel motivated and encouraged every time I attend his meeting. Without his encouragement and guidance this project would not have materialized. The guidance and support received from all the members who contributed and who are contributing to this project, was vital for the success of the project. I am grateful for their constant support and help. Besides, I would like to thank the authority of Sharda University for providing us with a good environment and facilities to complete this project. Finally, an honourable mention goes to my family and friends for their understanding and support of us in completing this project. Without the help of the particular that mentioned above, I would face many difficulties while doing this.

Name and signature of Students:

TANYA LILLIAN BORGES (2019639262)

KANISHKA NEGI (2019600654)

RONIKA SHRESTHA (2019003290)

ABSTRACT

The Internet of Things (IoT) has the capability to revolutionize society by connecting smart devices to the internet. However, security concerns remain a key obstacle in the widespread adoption of IoT technology. This paper presents an in-depth analysis of the Advanced Encryption Standard (AES) and Elliptic Curve Diffie-Hellman (ECDH) based security model for IoT devices like ESP 32, highlighting its benefits over other key agreement protocols. The ECDH protocol enables quick key agreement on limited resource devices, while the AES encryption algorithm ensures secure data transmission between IoT devices. To further enhance the security of IoT devices. We propose the use of the ESP32 microprocessor for low-cost and low-power solutions in IoT device development.

This research document provides an in-depth description of the ECDH and AES-based security model, including how they can be deployed on the ESP32 model. The ECDH and AES-based Security Model for IoT Using ESP32 offers a potential option for reliably securing IoT devices and can be further optimized to satisfy the evolving security requirements of IoT devices. Making sure that IoT devices are adequately protected against security threats has become a top priority as their use spreads across a variety of industries. The ECDH and AES-based Security Model can help with this goal.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

IoT has the potentiality to radically transform our civilization by simply linking sensors and smart gadgets to the internet. Many IoT initiatives concentrated on the device's functionality and efficiency [2]. Preventive security techniques such as device identity management, encryption, and cross-control, as well as device auditing and administration, are utilized to safeguard IoT connectivity. This technology may be used in a variety of industries, including infrastructure, agriculture, and a variety of other smart applications. Because of widespread connections, the security of communication between IoT devices is becoming increasingly critical in IoT devices [4]. The size of data sent electronically has grown significantly as electronic communication technology has advanced and the user space has grown. As a result, such encryption approaches create a barrier to speedy information transmissions [7].

One of the most promising security models for IoT devices is the AES and ECDH based approach that uses the ECDH protocol for key agreement and the Advanced Encryption Standard (AES) for encryption [1, 8]. With the use of the ECDH protocol, two parties can generate a shared secret key through an insecure communication channel without releasing the key itself. Each party generates a set of public and private keys based on elliptic curves to accomplish this. Thus every side uses its private key and the other side's public key to create a shared secret key after exchanging public keys. Then, symmetric encryption, message integrity, as well as other cryptographic operations can be performed using the shared secret key [10]. When compared to other key agreement protocols, the ECDH protocol has a number of benefits [9]. First, it is computationally more efficient, enabling quick key agreement on limited resource devices like the Internet of Things (IoT). Second, because it is predicated on the complexity of computing discrete logarithms on elliptic curves, it offers a high level of security. This makes it resilient to quantum computer attacks, which could endanger other crucial agreement procedures [10].

The symmetric encryption process can be used to encrypt data after the shared secret key has been created using the ECDH protocol and the AES encryption algorithm [11]. Data transferred between Internet of Things devices can be securely encrypted using the symmetric encryption method AES [19]. Data is encrypted and decrypted using a shared secret key, making sure that only those who have the key may access it [11].

The ECDH-based security architecture, which employs the ECDH protocol for key negotiation and the AES encryption algorithm for encryption, offers a robust security solution for protecting

IoT devices overall. It permits safe data transmission via an insecure channel between IoT devices, preserving the confidentiality and accuracy of the information sent [9].

IoT technology has a major effect on people's behaviour and lifestyles in both the workplace and at home [2]. This is linked to the advancement of accessible hardware components and CPUs [12]. Strong, affordable, and power-efficient approaches to IoT devices are required to improve the Internet of Things and extend its applications. A minimal form factor is another requirement for an IoT device; the greater the item's utility, the smaller and lighter it must be. Every IoT-based device consists of a module for wireless communication (often WiFi) and a CPU (C), or a combination of both. C and a variety of other modules are already widely utilised in the development and design of IoT devices and are readily accessible on the market. These include different Arduino gadgets, WhizFi, and Xbee. However, the vast majority of modern gadgets are either very pricey or very large in both height and weight. Additionally, only a small number of modules are open-sourced devices that have no operational limitations [13].

The ESP32 microprocessor, released in 2015, is a unique gadget that distinguishes out not just for its inexpensive price but also for its features. The ESP32 system-on-a-chip (SoC) from Espressif Systems has Wi-Fi and dual-mode Bluetooth connectivity, a two-core or single-core Tensilica Xtensa LX6 CPU with a maximum clock of 240 MHz, and a full set of peripherals. The chips in the ESP32 family are the ESP32-D0WDQ6 (and ESP32-D0WD), ESP32-D2WD, ESP32-S0WD, and the ESP32-PICO-D4 system in package (SiP). Antenna controls, an RF balun, a power amplifier, a low-noise receive amplifier, filters, and power management components are all included into the ESP32. Because of its unique architecture, it is perfect for mobile devices, wearable electronics, and Internet of Things applications. The ESP32 uses power-saving technologies like as fine-resolution clock gating, different power modes, and dynamic power scaling to achieve ultra-low power usage. Overall, the ESP32 microprocessor is a low-cost, adaptable technology that can deliver outstanding performance in a variety of applications [15].

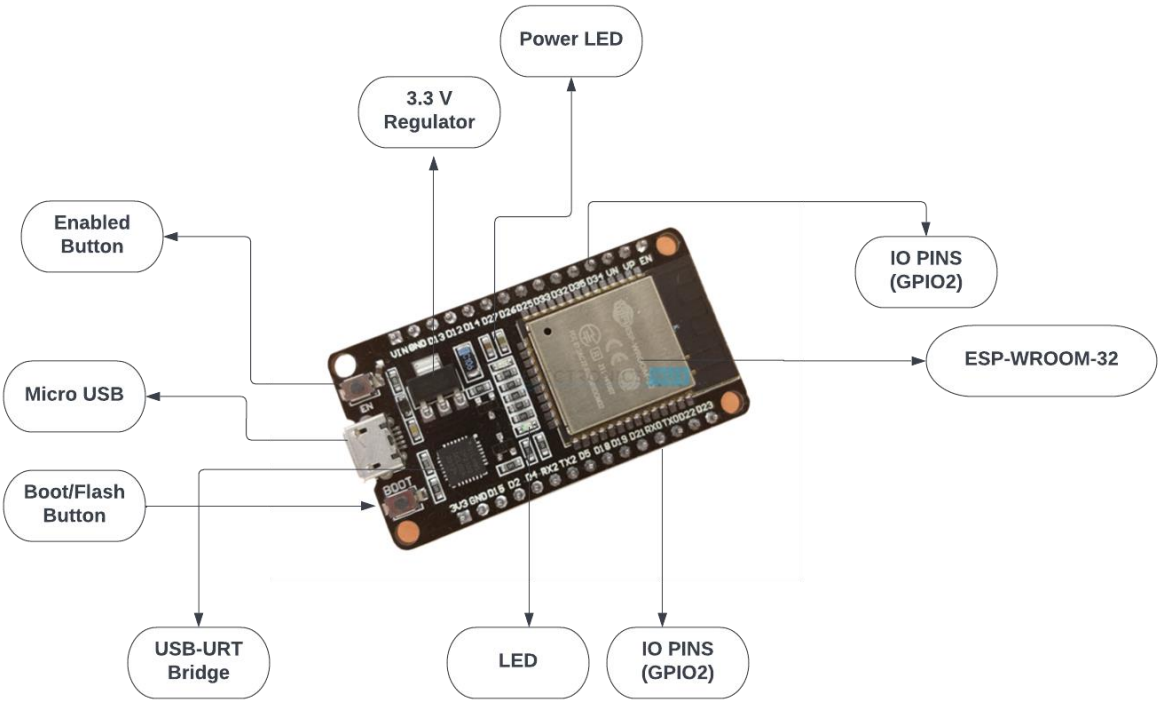

Figure 1.1 ESP 32 BOARD [12]

The ESP32 Board, in figure 1.1, is made up of the following components:

1. Module ESP-WROOM-32
2. Two rows of IO Pins
3. Micro-USB Connector CP2012 USB - UART Bridge IC
4. AMS1117 Regulator IC 3.3V
5. Enable the Reset Button
6. Boot button
7. User LED Power LED

We have given an in-depth description of the ECDH and AES-based security model in this research document, including how the ECC and AES algorithms work, how they can be used together to protect IoT devices, and how they can be deployed on the ESP32 model. Overall, the ECDH and AES-based Security Model for IoT Using ESP32 is a potential option for reliably and efficiently securing IoT devices. With the growing use of IoT devices in a variety of sectors, it is crucial to make sure that these gadgets are properly protected against possible security threats. The ECDH and AES-based Security Model can assist in achieving this

objective and can be further optimised in the future to satisfy the evolving security requirements of IoT devices.

The focus of our other paper was on the increasing importance of cybersecurity in modern business operations due to the growing number of dependencies in network topologies. The use of an Intelligent Intrusion Detection and Prevention System (IIDPS) is recommended to detect and prevent cyber-attacks in computer networks and systems. The architecture of an IIDPS typically includes a sensor, an analyser, and a user interface. Anomaly-based and signature-based systems are the two types of algorithms used in intrusion detection systems to detect and prevent cyber-attacks. Anomaly-based systems create a baseline model of typical system behaviour and compare it with the behaviour that is being tracked, while signature-based systems search for pre-configured attack patterns in data.

IIDPS uses advanced techniques such as machine learning, artificial intelligence, and behavioural analysis to monitor network traffic and identify potential threats. By combining these techniques with threat intelligence and response capabilities, IIDPS provides a comprehensive and intelligent approach to network security. However, both types of intrusion detection systems have their advantages and disadvantages. Anomaly-based IDS can work in real-time and detect malicious attempts that the system was unaware of, but are likely to experience issues with false positives. Signature-based IDS are effective when attack signatures are known in advance but struggle when there is no prior knowledge of the attacks and require frequent database updates.

In summary, cybersecurity is becoming a commercial priority due to the growing number of dependencies in network topologies, and an IIDPS is an advanced security system that uses intelligent algorithms and techniques to detect and prevent cyber threats, providing a higher level of security to computer networks and systems.

1.1 History

Cryptography has a long history dating back to ancient times, where methods such as substitution ciphers and transposition ciphers were used to protect information from being intercepted and understood by unauthorized parties.

In the modern era, cryptography has played a crucial part in the realm of cybersecurity, where it is used to protect sensitive data such as passwords, credit card details, and other private data. The figure 1.2 illustrates the different types of cryptography techniques.

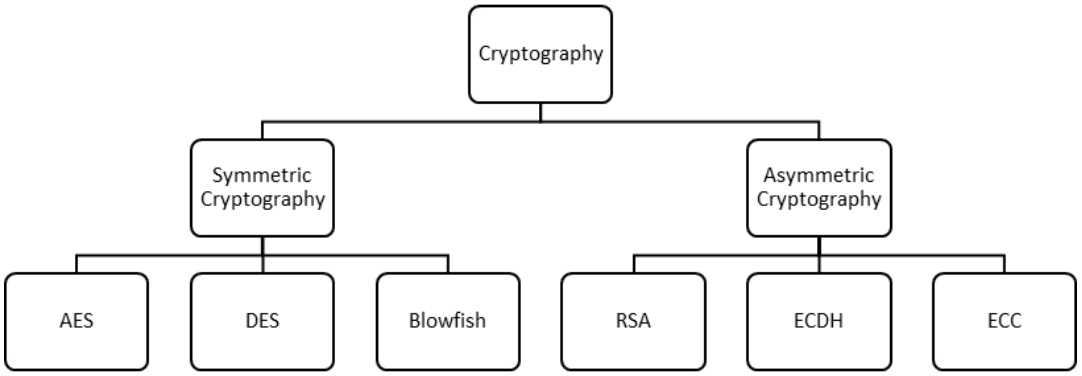

Figure 1.2 Taxonomy of Cryptographic Techniques

Table 1.1 provides an overview of the distinctions between symmetric and asymmetric cryptography, two types of cryptography widely used in the field of information security. The table compares key features such as key distribution, size, and management, authentication, use case, and security, making it easier for readers to understand the differences between the two approaches.

Table 1.1: Difference between Symmetric and Asymmetric Cryptography

Criteria	Symmetric Cryptography	Asymmetric Cryptography
Key distribution	Requires secure exchange of secret key	Public key can be shared freely
Key size	Short key length required (64-256 bits)	Longer key length required (1024-4096 bits)
Key Management	Key management may be complicated and difficult, necessitating secure key storage and dissemination.	Key management is simpler as there is no need to distribute secret keys
Authentication	Does not provide authentication by default	Provides authentication through digital signatures and certificates

Use case	Suitable for encrypting large chunks of data, such as bulk data transfer and storage	Suitable for secure communication and authentication, such as digital signatures and secure email
Security	Less secure due to the need for secret key exchange and potential for key compromise	More secure due to the use of public and private keys, but still exposed to attacks such as man-in-the-middle and brute force

Here are some of the key developments in the history of cryptography related to cybersecurity:

1. **AES (Advanced Encryption Standard):** The AES algorithm is a symmetric key encryption technique that was created in the 1990s to replace the ageing DES (Data Encryption Standard) algorithm [17]. AES is the most frequently used encryption algorithm in the world, and it is utilized for safeguarding internet connections as well as to encrypt data saved on hard drives and other storage devices. A brief description of the working of AES is illustrated in figure 1.3.

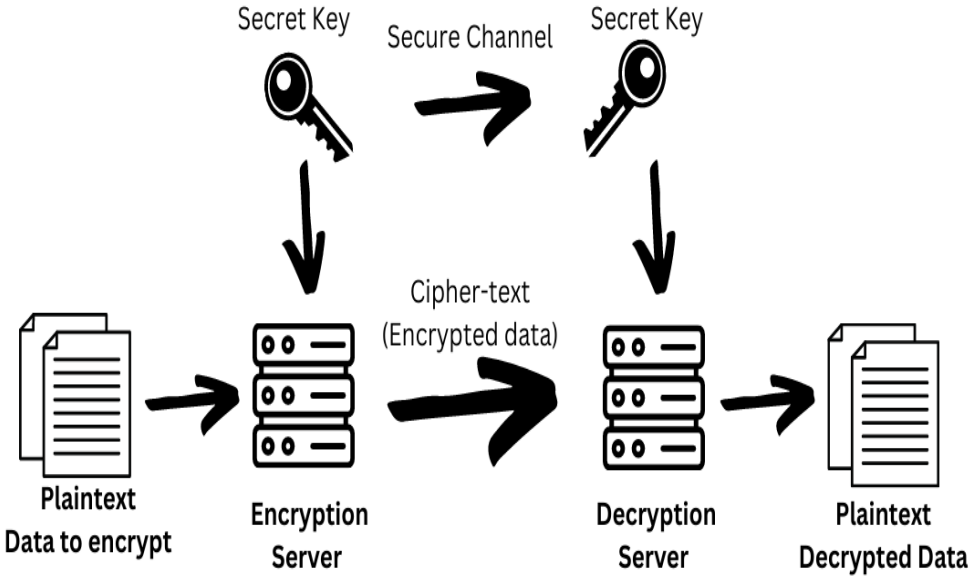

Figure 1.3 Working of AES Cryptography [17]

2. **ECDH (Elliptic Curve Diffie-Hellman):** ECDH is a protocol used for key exchange between two parties over an insecure communication channel, as shown in Figure 1.4. It is based on ECC and Diffie Hellman Key Exchange and is widely regarded as one of the most secure key exchange protocols in use today. The ECDH protocol enables the two parties to establish a shared secret, which can then be used to encrypt subsequent communication between them. Due to its robust security features, ECDH is a popular choice for secure communication in various applications.

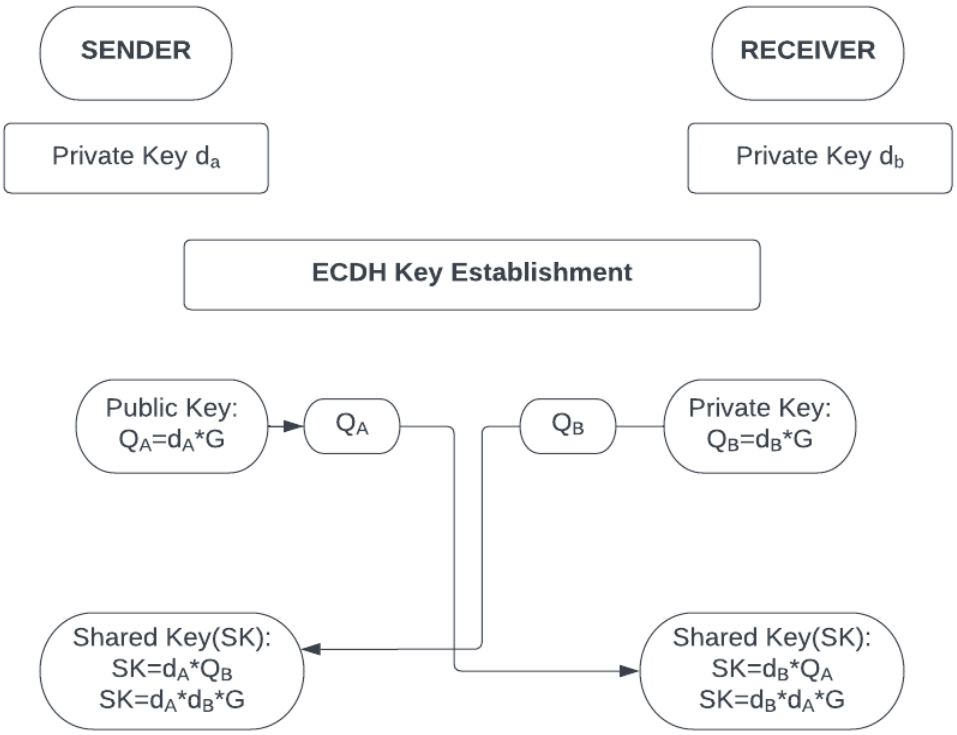

Figure 1.4. Working of ECDH Cryptography [10]

3. **DES (Data Encryption Standard):** DES is a symmetric key encryption method that was created in the 1970s and was widely used for many years to protect sensitive data [28]. However, the algorithm has since AES and other more strong encryption techniques have taken their place. Figure 1.5 focuses on the working of DES cryptography.

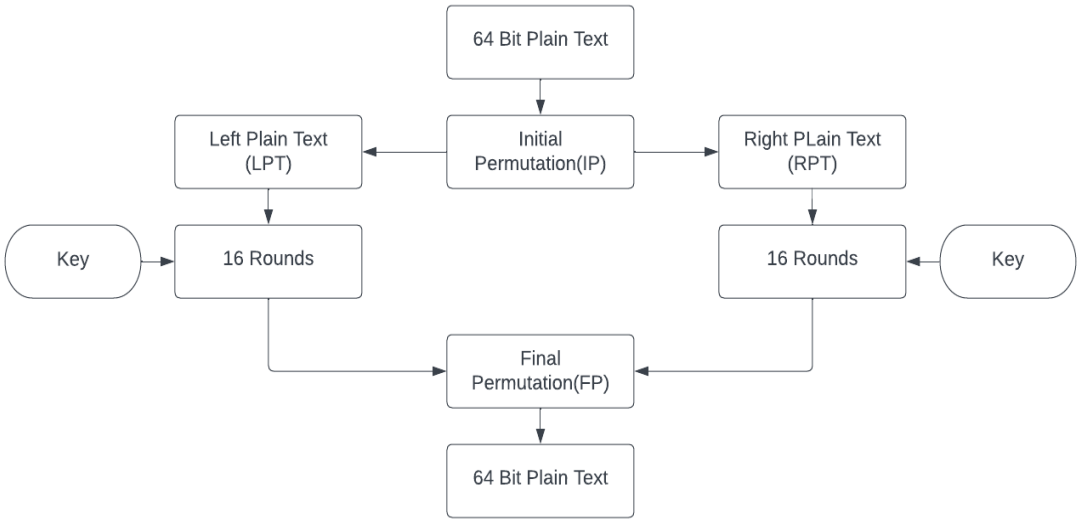

Figure 1.5 Working of DES Cryptography [28]

4. **RSA:** it is a public key encryption algorithm that was developed in the 1970s by Ron Rivest, Adi Shamir, and Leonard Adleman. RSA (Figure 1.6) is widely used for secure communications over the internet, as well as for digital signatures and other cryptographic applications [29].

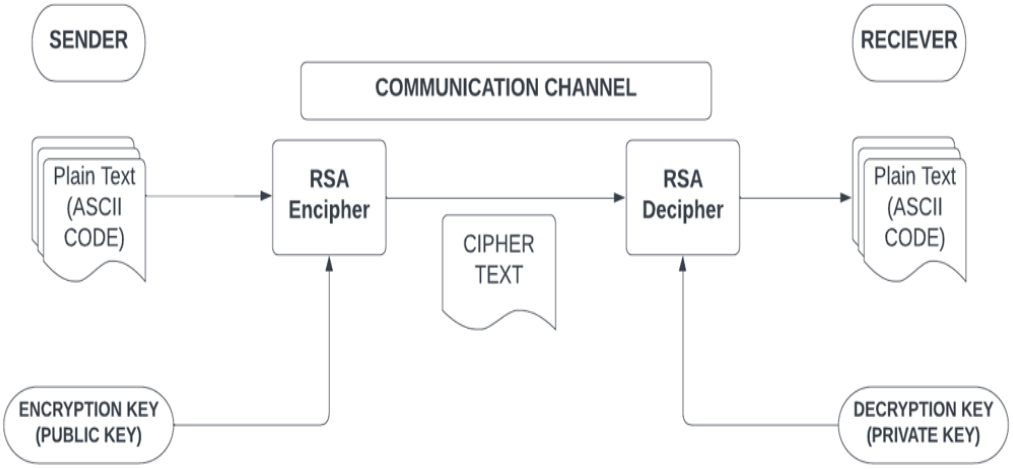

Figure 1.6 Working of RSA Cryptography [29]

5. **Blowfish:** Blowfish Algorithm was developed by Bruce Schneier in 1993 and is one of the most effective algorithms. It is a block cipher (64-bit) symmetric algorithm as depicted in figure 1.7. It is an replacement to DES encryption technique and IDEA algorithm.

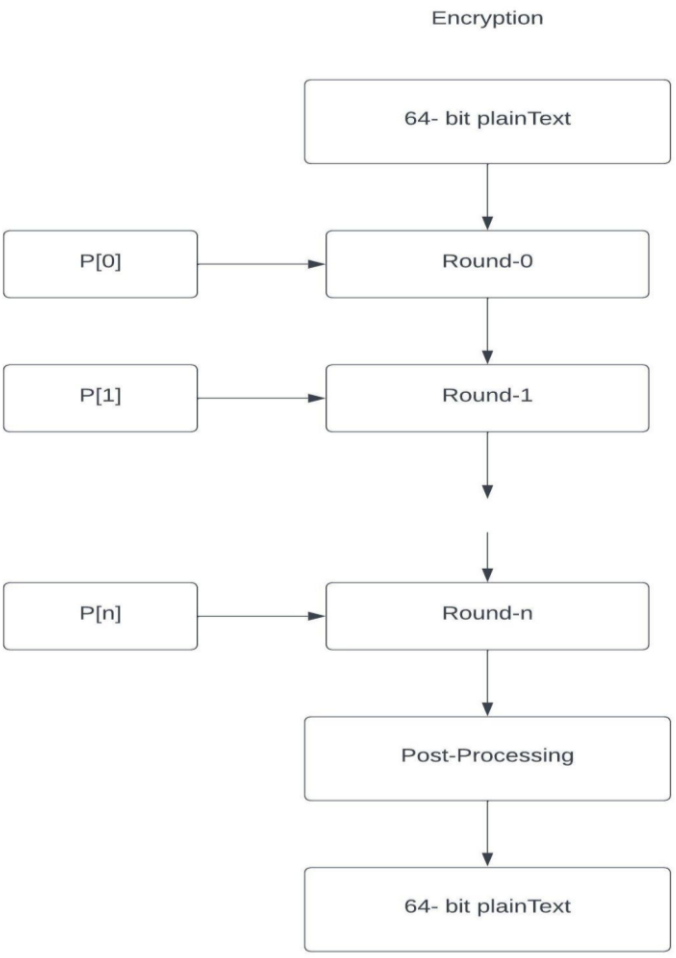

Figure 1.7 Working of Blowfish Cryptography [28]

6. **Elliptic Curve Cryptography:** In 1985, IBM's Victor Miller and the University of Washington's Neil Koblitz made the discovery of ECC [10]. The abbreviation for elliptic curve cryptography is ECC. Although this approach uses a similar methodology to the Diffie-Hellman key exchange protocol and the RSA algorithm, Elliptic-Curve Cryptography is distinguished by the fact that the integers are selected from a limited field specified within an elliptic curve expression**.** Figure 1.8 illustrates the elliptic curve cryptography curve.

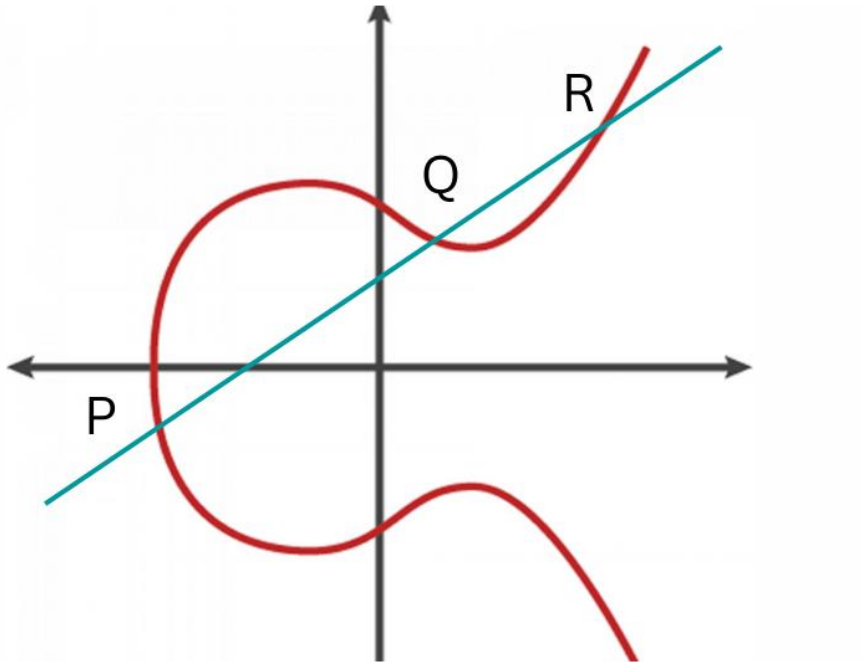

Figure 1.8 ECC Curve [10]

Another important development in the history of cryptography in IoT device cloud storage is the use of Public Key Infrastructure (PKI). PKI is a system that authenticates individuals and devices while also providing secure data transmission and storage. It is widely used in cloud storage services to provide secure access to data.

Overall, cryptography has been essential in the advancement of cybersecurity and the continued evolution of cryptographic algorithms and protocols will be essential in the ongoing fight against cyber threats.

1.2 Problem Statement

IoT device data is increasingly being stored and accessed via cloud storage. Sensitive data transmission over the internet, however, can present substantial security concerns. An ECDH and AES-based security paradigm is suggested for IoT devices deploying ESP32 for cloud storage to alleviate these worries. The suggested architecture intends to offer end-to-end encryption for data exchanged between IoT devices as well as the cloud storage server, guaranteeing that secret and sensitive information is kept safe and secure. ECDH algorithm AES algorithms are both used by the model for secure key exchange and data encryption, respectively. The suggested security model is implemented using the ESP32 microcontroller as

a hardware platform, allowing for secure and effective communications between IoT devices as well as the cloud storage server.

The main goal of this research is to create an effective security architecture that will safeguard sensitive data from hacker assaults and unauthorised access for IoT devices that use ESP32 for cloud storage. The suggested approach is anticipated to enable secure and safe interaction between the internet of things and cloud storage servers, delivering enhanced security for IoT applications.

1.3 Objectives

1. The project will focus on how to cipher or decipher data with AES-256 and ECDH, on the Arduino core operating on the ESP32.
2. The goal is to provide a technique that can render data undetectable by unauthorised users by employing an encryption algorithm. It would subsequently stop valuable data from being taken by hackers via network sniffing.
3. This project aims to provide and simulate an effective solution to face the challenges in storage of IoT data and solve security issues in cloud computation.
4. Implement an Intelligent Intrusion Detection and Prevention System (IIDPS) to determine and alert on any attempts to access or tamper with the encrypted data, in order to strengthen the general security of the IT infrastructure. IIDPS might be based on a variety of approaches such as anomaly detection, signature-based detection, or machine learning and could leverage the sensors and connectivity features of the ESP32 to monitor the local network and external traffic. The goal would be to provide a comprehensive defence against potential attacks and Ensure the security, reliability, and accessibility of IoT data.

1.4 Project Overview

The project includes establishing a security model for IoT devices utilising the ESP32 microcontroller and using both Elliptic Curve Diffie Hellman (ECDH) and Advanced Encryption Standard (AES) algorithms enabling secure data transmission. The project's major objective is to execute ECDH for key exchange and AES for encryption purposes on the ESP32 microcontroller without severely degrading the system's performance. The suggested approach will give IoT devices end-to-end encryption and authentication, making it appropriate for usage in a range of IoT applications.

The creation of the security framework, which will be based on the integration of the ECDH and AES algorithms on the ESP32 microcontroller, is the most crucial stage of the project. The suggested paradigm will be created to be readily integrated with current IoT networks and scalable. To confirm the model's efficacy in guaranteeing safe connectivity for IoT devices, it will also be put to the test in a real-world environment.

In the project's final stage, the performance, scalability, and effectiveness of the suggested security model will be assessed. A number of tests employing various communication situations and network topologies will be performed on the ESP32 microcontroller as the basis for the evaluation. The evaluation's findings will be applied to the proposed security model to help it be improved and modified according to the requirements of the IoT devices in which it will be employed.

1.5 Motivation

The growing popularity of IoT devices for collecting and transmitting sensitive data has raised concerns about security breaches and cyber-attacks. Therefore, it is crucial to adopt a robust security solution that can guarantee safe communication between IoT devices and safeguard the confidentiality, integrity, and authenticity of the transmitted data.

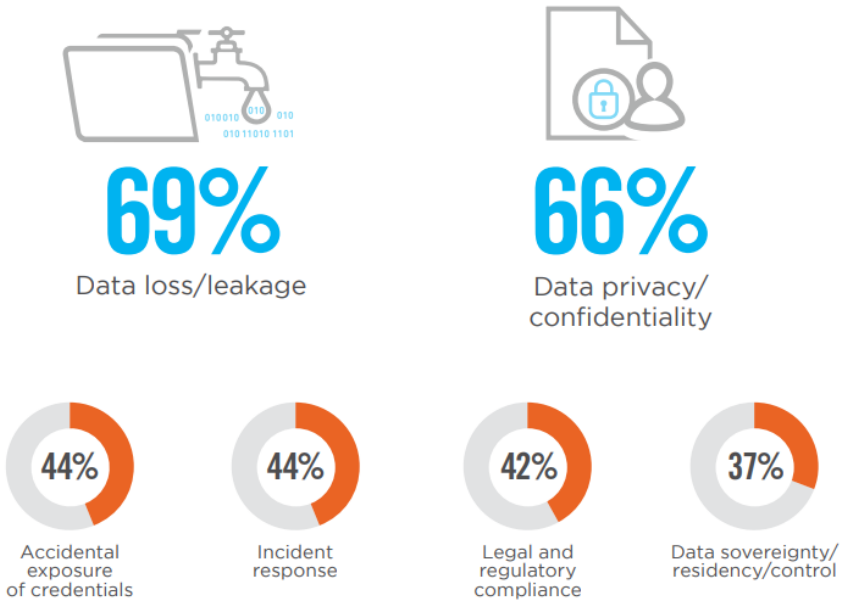

Figure 1.9 Data Breach in Cloud [26]

As depicted in figure 1.9, security configuration remains the top worry in cloud environments with 69% of respondents expressing concern. The second-highest concern was a lack of

visibility into network endpoints, which was acknowledged by 66% of respondents. Threats relating to identity and access management were another big issue, with 61% of CISO professionals saying they were concerned about this threat [26]. In our suggested model, we combine AES and ECDH to provide secure communication and encryption, ensure security of network endpoints, and develop robust access control protocols to address these issues.

In this context, the proposed security model based on ECDH and AES on the ESP32 microprocessor offers several advantages. Firstly, ECDH and AES are well-established cryptographic algorithms that provide strong encryption and decryption capabilities. The ECDH key exchange technique allows two parties to create a shared secret key securely without exchanging the key directly, while AES is a symmetric encryption technique that encrypts and decrypts data using a shared secret key. By using these algorithms together, the proposed security model can provide secure communication between IoT devices.

Secondly, the ESP32 microprocessor is a resource-constrained device that has limited computational and memory capabilities. Therefore, conventional security solutions such as TLS (Transport Layer Security) and SSH (Secure Shell) may not be suitable for such devices. The proposed security model based on ESP 32 is therefore perfect for resource-constrained IoT devices such as the ESP32 as it is lightweight and efficient.

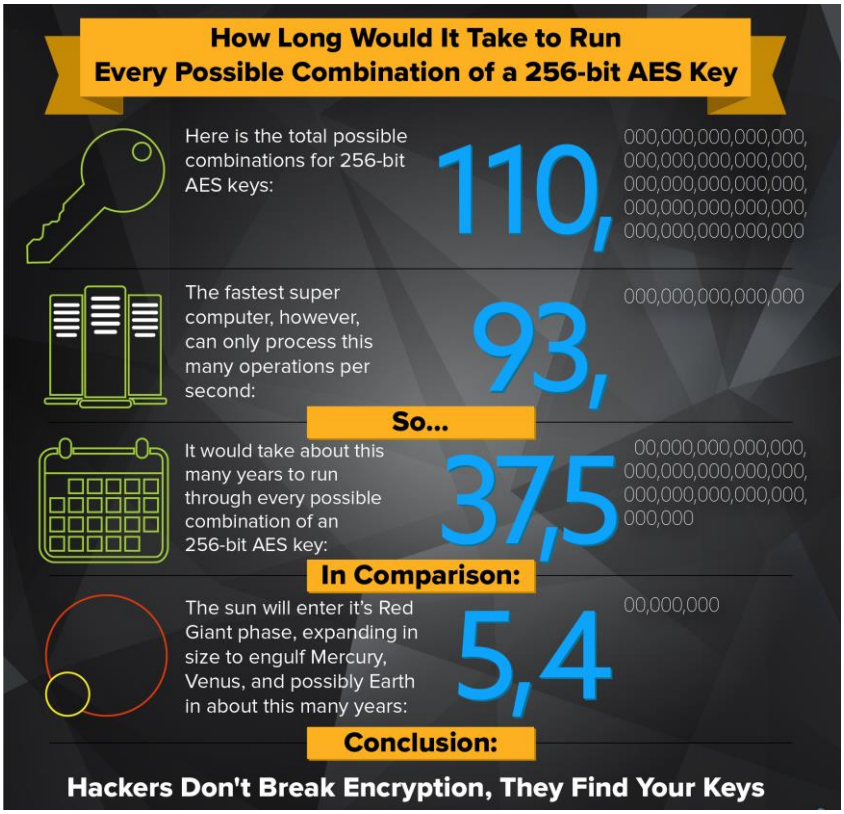

Figure 1.10: Increased safety of 256-bit AES Key Encryption [27]

Thirdly, the suggested security model is scalable, which means it can be easily applied to various IoT applications. As the IoT network expands, the suggested security model can be extended to new devices with ease.

Finally, the suggested security model is compatible with existing ESP32-based IoT networks, making it easy to integrate into current systems. This compatibility also means that the security model can be quickly deployed, making it a viable solution for IoT security.

In conclusion, the proposed IoT security model based on ECDH and AES on the ESP32 microprocessor offers a lightweight, efficient, scalable, and compatible solution to the challenges of IoT security. By adopting this security model, IoT devices and their users can enjoy secure communication and protection against cyber-attacks.

1.6 Expected Outcome

The expected outcome of this project could include: The successful implementation of AES-256 and ECDH encryption and decryption algorithms on the ESP32 microprocessor, providing a mechanism to make IoT data unreadable by unauthorized users and prevent data theft. The development of a comprehensive security solution for IoT devices, including an IDS based on various techniques such as anomaly detection, signature-based detection, or machine learning, to detect and alert on any attempts to access or tamper with the encrypted data.The simulation and testing of the security model on the ESP32 microprocessor to ensure its effectiveness in real-world scenarios. The demonstration of the feasibility of lightweight and efficient security solutions for resource-constrained IoT devices like the ESP32, highlighting their potential for use in various applications. The contribution to the subject of security by giving a viable answer to the issues of IoT data storage and solve security issues in cloud computation, improving the overall security and trustworthiness of IoT systems.

1.7 Hardware and Software specifications

This part of our report focuses on the hardware and software configurations necessary for the successful implementation of our proposed approach.

1.7.1 Hardware Requirements

Given below are the necessary hardware components required for the completion of this project:

1. **Solderless board:** A solderless board is a type of circuit board that enables

component connections without the need of solder. Electronic circuit prototyping and experimentation may become simpler as a result.

2. **Wifi ESP-WROOM-32:** The WiFi ESP-WROOM-32 is a well-liked WiFi and Bluetooth module that is frequently utilised in Internet of Things projects. It has a dual-core processor, 4MB of flash memory, and compatibility for a variety of interfaces and protocols.
3. **Dell Inspiron 5490:** The Dell Inspiron 5490 is a laptop with a 256GB solid-state drive, an Intel Core i5 processor, and 8GB of Memory. A wide range of ports and interfaces are also included, including HDMI, USB Type-A and Type-C, and an SD card reader.
4. **USB to Type B connector:** The cable that joins the laptop and the ESP-WROOM-32 module is USB to Type B. Between computers and other devices, the USB Type B connector is commonly used for data transfer and power delivery.

With the help of these standards, a file uploader system using the ESP-WROOM-32 module can be built and tested. The ESP-WROOM-32 module may be programmed and tested on a laptop, and easy prototyping and experimentation is made possible by the solderless board. Communication between the laptop and the ESP-WROOM-32 module is made easier by the USB to Type B wire. Using the connector we can then encrypt any data that is uploaded to the file system.

1.7.2 Software Requirements

The following software specs are required for the proper implementation of this project:

1. **Integrated Development Environment (IDE):** Arduino IDE version 1.8.19. The Arduino IDE 1.8.19 is a free and open-source software tool that offers a programming environment for generating and uploading code to Arduino boards. It features a syntax-highlighting code editor, a compiler, and a serial monitor for debugging. The IDE is available for a variety of platforms. operating systems, including Windows, macOS, and Linux.
2. **Operating System (OS):** Microsoft Windows Operating System build 22621.1413. The Windows Operating System build 22621.1413 is a version of Microsoft Windows 10 that was released in 2021. It includes updates and security fixes to Boost the operating system's stability and performance.

3. **Programming Language:** C is the programming language used for writing code for the Arduino boards. It is a powerful and efficient language that provides low-level control over hardware resources, making it an ideal choice for embedded systems development. The Arduino IDE comes with a set of pre-built libraries that can be used to simplify the development process and provide easy access to the board's features and functionalities.

1.8 Non-Functional Requirements

Non-functional requirements (NFRs) are parameters used to assess the performance, usability, security, reliability, and other qualities of a system or software application, rather than its functional requirements. Non-functional requirements, as opposed to functional requirements, specify how effectively the system or application should perform. Some examples of non-functional requirements are:

1.8.1 Performance Requirements

a. **Reliability:** The security model should be highly reliable, with minimal downtime or outages. This means that the security model should be able to provide security services continuously, even in the event of hardware or software failures. The security model should be designed to recover from failures automatically, without requiring manual intervention.
b. **Latency:** The security model should minimize latency, which is the delay between when a device sends a message and when it receives a response.This means that the security model should be designed to process messages quickly and efficiently, without causing significant delays.
c. **Efficiency:** The security model should be designed to be efficient, with minimal impact on the performance of the ESP32 microcontroller. This means that the security model should be designed to use minimal system resources, such as memory and processing power.The security model should also be designed to minimize power consumption, to maximize the battery life of IoT devices.
d. **Portability:** The security model should be portable to different platforms and architectures, to enable interoperability and flexibility.

1.8.2 Security Requirements

a. **Integrity:** The security model should ensure that information cannot be altered or tampered with while being sent or stored.
b. **Key Generation:** To maintain the secrecy and integrity of the keys used for decryption and encryption, the security model needs to include a system for maintaining them.
c. **Access Control:** To guarantee that only authorised individuals and systems may access the data, the security model ought to encompass a mechanism to manage the accessibility of Internet of Things devices and cloud storage.
d. **Authentication:** To guarantee that data is only accessed by permitted devices and systems, the security model should include a means for authentication of Connected technologies and cloud storage.
e. **Secure Communication:** The security model should include secure communication protocols that guard against unauthorised parties listening in or intercepting the transmission of data among IoT devices and cloud storage.
f. **Privacy:** The security model should safeguard users' privacy by preventing the disclosure of their sensitive data or personal information to unauthorised parties.

1.8.3 Safety Requirements

a. **Data privacy:** Since the project deals with sensitive data, it is essential to ensure that the data is protected from unauthorized access. This can be achieved by implementing encryption and authentication protocols and following best practices for data handling and storage
b. **User Privacy:** IoT devices often gather sensitive data from their users, such as personal information, location data, and other sensitive data. Therefore, it is essential to ensure that the proposed security model provides sufficient privacy protections for the users of the system. This includes protecting user data with strong encryption and implementing privacy-preserving measures such as data minimization and user consent.
c. **Compliance with Regulations:** It is important to essential that the project

complies with all relevant regulations and standards. For example, if the IoT devices are used in a healthcare setting, they may be subject to HIPAA (Health Insurance Portability and Accountability Act) regulations, which require strict security and privacy controls.

1.9 Report Outline

The report has 5 chapters and will start with:

Chapter 1: Introduction: In this chapter, we will provide an overview of the project, starting with its history. We will then discuss the problem statement, explaining the issues the proposed system aims to solve. Next, we outline the project's objectives, providing a clear vision of what we hope to achieve. The project overview will then describe the system's scope and how it fits into the larger context of the field. The motivation for the project will be explained, highlighting the benefits that the proposed system aims to bring. We will then discuss the expected outcome of the project, including the deliverables and potential impact. The hardware and software specifications, and non-functional requirements will be presented, providing a detailed breakdown of the system's technical requirements. Finally, we will provide a report outline, detailing the structure of the report and the contents of each chapter.

Chapter 2: Literature Survey: This chapter will begin by examining the existing system, including its strengths and weaknesses. We will then conduct a feasibility study to determine the practicality of the proposed system. This will involve analyzing the technical, economic, and operational aspects of the system to determine whether it is viable.

Chapter 3: System Design and Analysis: In this chapter, we will provide a detailed perspective of the project, including its system features. We will then present the proposed system, explaining how it works and how it addresses the problem statement. Methodology will be discussed, including the tools and techniques used in the design and analysis of the system.

Chapter 4: Results and Discussion: This chapter will present the results of the project, including any experiments or testing that was conducted. We will then provide a detailed discussion of the findings, including their implications and potential impact. We will also examine the proposed model outputs, specifically the ESP 32 output, and how they compare to the expected outcomes.

Chapter 5: Conclusion and Scope of Future Work :The final chapter will conclude the report by summarizing the project's main findings and providing a final assessment of its success. We will

then explore the potential for future work, including areas for further research and development. This chapter will also highlight the limitations of the project and provide recommendations for future improvements.

References: Here all sources cited throughout the report will be listed in this section.

CHAPTER 2: LITERATURE SURVEY

2.1 Existing Work:

Kodali et al. [1] proposed an ECDH-based solution to improve IoT device security. The NIST P-192 curve was discussed and used by the authors to implement ECDH key exchange for confidential communication between ESP8266 modules using NodeMCU. The ECDH key exchange was implemented using PlatformIO's BigNumber library and C++. Future uses for their ideas could involve wearable technology, mesh networks, smart electricity metre, home automation, and security ID tags. According to Abdhulla et al. [2], the data transferred by the IoT system over a Wi-Fi network was encrypted using cryptographic techniques. The Caesar Cypher, SHA 256, and AES 128 Bit are three cryptographic algorithms that have been invented and come in various degrees of complexity and computer power. The ability to encrypt Arduino data using the cryptographic algorithms integrated into the Arduino IDE serial monitor has been successfully demonstrated in Arduino Uno. It has introduced an additional layer of security because network sniffers can no longer understand the data. The Arduino Uno's presence of cryptographic capabilities helps the project's objective of securing the data transmitted by the Arduino Uno. To improve the security of RFID data, H.P.T.M. Jayawardana [3] suggests a hybrid encryption technique that makes use of both symmetric and asymmetric cyphers. The asymmetric cypher of the proposed protocol is elliptic curve cryptography (ECC), and the symmetric cypher is the advanced encryption standard (AES). While the AES method is very safe, quick, and well-standardized, the ECC algorithm offers rapid key creation, rapid key agreement, and speedy signatures. The combination of both algorithms is expected to provide the best security results for RFID technology. The proposed protocol is analyzed using the Automation Validation of Internet Security Protocols and Applications (AVISPA) tool. In his paper, Anothay Phimphinith [4] introduced a new shared key agreement and authentication mechanism for Internet of Things (IoT) devices using the inexpensive ESP8266 module. ECDH key exchange on curve 25519 is used in the proposed method to fix the security flaws in a prior Kodali and Naikoti technique. Attacks like replay attacks, modification assaults, and man-in-the-middle attacks are all prevented by the proposed approach. The authors assert that their plan is not only more efficient than 100 other comparable plans, but also more secure. In general, the suggested method works well for secure communication between ESP8266 modules in an Internet of Things environment. AES was the encryption technique utilised by Tulip Dutta [5], and the system was constructed in Java utilising a keystore data structure.In key aggregation, many data files are scrambled using various keys, and then a single combined key is used for decryption. They described the distribution of a secret key to users to whom accessibility must be permitted. Bhale Pradeep kumar Gajendra [6] explains how to increase the security of IBE

encryption methods by adding additional security codes by employing a Third Party Auditor to improve cloud security. Since encryption is essential for information sharing, the IBE algorithm's encryption section has been improved by fusing it with the digital abstract method MD5 to produce a hybrid algorithm that improves security. A dependable Third Party Auditor can be used to guarantee the integrity and safety of data, assuring both cloud customers and service providers that their data is secure.

Bharathi et al. [16] utilized DES, RSA, and AES, to separate data and encrypt it. Files and keys are kept in the cloud and LSB steganography, respectively. The combination of this hybrid approach results in enhanced security. They mentioned that the one drawback of their approach is that active internet connection is necessary in order to connect to the cloud server. Hodowo et al. [17] provided a paradigm and a hybrid cryptographic approach to improve data security in cloud computing. Their model utilized both AES and ECC to increase data security against outsiders or hackers, prevent them from accessing the actual information and enabling confidentiality, integrity of the data, time required for performing cryptographic techniques, and increasing level of speed by applying relatively smaller keys of ECC in the encryption and decryption process. Goyal et al. [18] proposed an effort to identify secure approaches to data sharing and communication in the cloud. In light of this, a DNA-based cryptographic method for use in a cloud-based cryptographic system is proposed. The existing systems require the use of the same randomly generated key for both encoding and decoding of data. A mapping for decryption is thus required for subsequent decryption. This procedure is required to improve the security of the system. Rehman et al. [19] outlined a safe and efficient plan for cloud-based data sharing while maintaining the security and integrity of data. The ECC and AES methods are primarily combined in the proposed system to provide authentication and data integrity. When compared to existing approaches, the experimental results displayed that the proposed approach is more efficient and produces better results. Pavithra et al. [20] developed a secure cloud-based data evaluation tool. The blinding process begins after the user uploads the data. The current Pyycon Python module is used to encrypt data. While the files are being cleaned up, RSA encryption generates a key and password. Anuj et al. [21] recommended using a sample plaintext text string for the implementation and simulation of the suggested approach. The RSA algorithm is used to implement the first security step, that is, data encryption. For the second layer of security, data from the first stage is encrypted using the DES Algorithm. RSA and DES are combined to provide a highly secure method for storing data in the cloud. Machine learning techniques were introduced by Repalle et al. [24], who also demonstrated how to use them in an

intrusion detection system. Finding quality labelled datasets to test the machine learning techniques and algorithms was the major issue that was identified during the thesis. According to their findings, K-Nearest Neighbors provides the greatest outcomes in both the assessment and the real-world scenario. In order to focus their paper on hybrid intelligent systems, Bhumgara et al. [25] contrasted the various kinds of classifiers used for network intrusions. The study is carried out by conducting an empirical assessment of their publications and establishing a foundation for more investigation.

The table given below summarizes the research papers reviewed to help in the implementation of this project.

Table 2.1: Literature Review

S.No.	**Author**	**Year**	**Contributions**
1.	Kodali et al. [1]	2016	1. Application of ECDH on the NIST P-192 curve on ESP8266 modules. 2. Wearable devices, mesh networks, security ID tags, home automation, smart power metres, and sensor networks are just a few of the applications.
2.	Abdullah et al. [2]	2022	1. Encrypt data sent by device network using cryptographic functions. 2. Three cryptographic algorithms have been devised: AES 128 bit key, Caesar Cipher, and SHA 256. 3. The contents of Arduino are encoded using cryptographic techniques programmed into the Arduino Uno, as demonstrated by the Arduino IDE serial monitor.
3.	H.P.T.M. Jayawardana [3]	2020	1. RFID (Radio Frequency Identification) technology is a newer technology in our modern age. The majority of previously utilised RFID technology have privacy concerns. The advent of security and confidentiality problems for RFID has resulted in a trend in research.

			2. Offering a hybrid encryption solution that combines the AES encryption algorithm and ECC key generation technique is the best and most cost-effective way to increase the security of RFID data.
4.	Anothay Phimphinith[4]	2019	1.They presented an effective mutual authentication and key agreement system for IoT devices in this research with the low-cost module ESP8266 2.They have proposed ECDH key exchange on curve25519 for IoT using ESP8266
5.	Tulip Dutta [5]	2016	1. Explains the distribution of a secret key to individuals to whom accessibility must be permitted. 2. In key aggregation, many data files are scrambled using various keys, and then a single combined key is used for decryption. 3. Using a keystore data structure and the encryption method AES, Java is utilised to create the system.
6.	Gajendra et al. [6]	2016	1. Minimises the privacy tradeoff, boosts communication efficiency, and enhances security
7.	Bharathi et al. [16]	2021	1. Utilising DES, RSA, and AES, data is separated and encrypted. Files and keys are kept in the cloud and LSB steganography, respectively. Enhanced security as a result of the hybrid approach. 2. Active internet connection is necessary in order to connect to the cloud server.
8.	Hodowu et al. [17]	2019	1. This research provided a paradigm and a two-level cryptographic approach for improving data security concern in cloud computing. 2. Both asymmetric and symmetric cryptography techniques are used in the model to increase data security

			against hackers, preventing them from accessing the actual information.
9.	Goyal et al. [18]	2019	1. The goal of the proposed effort is to identify approaches for data sharing and communication that are secure in a cloud context. In light of this, a DNA-based cryptographic method is suggested for use in a cloud-based cryptographic system. 2. In the existing technique, the same randomly generated key is needed for both encoding and decoding. A mapping for the decryption is therefore necessary for subsequent decryption. This procedure is required to improve the system's security.
10.	Rehman et al. [19]	2015	1. Proposed a safe and effective method for cloud data exchange while preserving data security and integrity. 2. In order to assure authentication and data integrity, the suggested system primarily works by fusing the ECC and the AES approach. 3. The experimental results show that the proposed approach works and is successful and generates better outcomes than current approaches.
11.	Pavithra et al.[20]	2021	1. Created a safe cloud-based data evaluation tool. 2. After the user uploads the data, the process of blinding begins.The current Pyycon Python package is used to encrypt data. 3. A key and password are generated by RSA encryption while the files are being cleaned up.
12.	Anuj et al.[21]	2020	1. A sample plaintext text string is used to implement and simulate the suggested approach. 2. The RSA algorithm is used to apply the 1st step of security.

			3. RSA algorithm is used for data encryption. 4. For the next layer of security, data that was the result of the first step is encrypted using the DES Algorithm. 5. RSA and DES are combined to provide a highly secure method that can be used to store data on the cloud.
13	Syam Akhil Repalle[24]	2017	1. This paper provides an overview of ML algorithms and their use in IDS. 2. The paper highlights the importance of using good quality labeled datasets to train machine learning algorithms, as this can greatly affect their performance. 3. The K-Nearest Neighbours algorithm was found to be the most effective in both testing and real-world circumstances, however care must be taken in selecting the value of k and proximity measure.
14	A. Bhumgara[25]	2019	1. Using the dataset generated by the NSL-KDD as a benchmark, this paper's contribution is the examination and assessment of several IDS for developing an effective network IDS. 2. The merging of Random Forest with nested divisions and END, which produced a highly successful IDS with an accurate detection rate of 99.5% and an error rate for false alarms of just 0.1%, is the subject of this research. 3. The study also identifies the drawbacks of other approaches, such as the filtering of data by decision tree (J48) with categorization by a radial basis function neural network (RBF), which performed

			poorly in terms of false alarm and intrusion detection rates.

2.2 Existing System Disadvantages:

Existing encryption solutions for ESP32 and ESP8266 microcontrollers have a number of drawbacks that can be problematic for developers and affect the security of IoT devices. Some negative aspects include:

1. **Impact on performance:**

   Some encryption techniques may have a considerably negative effect on the speed and power consumption of ESP32 and ESP8266 microcontrollers. This may have an effect on the IoT system's general effectiveness and responsiveness.

2. **Restricted key sizes:**

   The small key sizes of many encryption techniques used with ESP32 and ESP8266 make them susceptible to brute-force assaults. Sensitive data's security may be jeopardised, and unauthorised access may result.

3. **Complexity:**

   Certain encryption techniques are challenging for developers to integrate into their IoT applications because they involve complicated configuration and implementation. The amount of time and money needed for development, testing, and deployment may rise as a result.

4. **Lack of adaptability:**

   Current encryption solutions might not have the adaptability necessary to keep up with evolving security needs and new security threats. As a result, the security architecture may become less effective over time and may need further development work to address emerging vulnerabilities.

5. **Problems with compatibility:**

   Certain encryption techniques might not work with specific ESP32 and ESP8266 hardware or software combinations, which would restrict their use in some applications. Finding a good encryption solution that fits with particular hardware or software requirements may become challenging as a result.

6. **Cost:**

   Certain encryption techniques can call for extra hardware or software, which would raise the cost of the IoT system as a whole. For developers, especially those working on low-cost or resource-constrained IoT applications, this might be a substantial impediment.

To create a security architecture for ESP32 and ESP8266 microcontrollers that can effectively protect sensitive data from hacker attacks and illegal access, it will be essential to overcome these drawbacks.

2.3 Feasibility Study

This study should focus on the technical, economic, and operational aspects of deploying this security model for IoT devices. To determine whether integrating these encryption techniques into the system is feasible, a feasibility analysis for the ESP32 board's AES and ECDH encryption algorithms can be carried out.

The feasibility study takes into account the following elements:

1. Hardware Capability: In order to support the AES and ECDH encryption algorithms, the ESP32 board must have enough hardware capabilities. The ESP32 is a potent microcontroller with a 32-bit dual-core processor, integrated Wi-Fi, and Bluetooth, making it appropriate for using the encryption technique.

2. Memory Capacity: The ESP32 board needs to have enough memory to store both the encrypted data and the encryption key. For the majority of applications, the ESP32's maximum flash memory capacity of 4 MB and maximum SRAM capacity of 520 KB should be adequate.

3. Encryption Algorithm Speed: The encryption algorithm must be quick enough to allow for real-time encryption and decryption. with a maximum clock frequency of 240 MHz, the ESP32 should have enough computing capability to quickly conduct encryption and decryption.

4. Energy Consumption: Because the ESP32 board is frequently used in battery-powered applications, its energy consumption must be taken into account. Due to their computational complexity, the encryption algorithms AES and ECDH could use more

energy. It is important to assess the ESP32 board's power consumption to make sure it is within reasonable bounds.

5. Compatibility: To ensure that the AES and ECDH encryption algorithm can be effortlessly integrated into the system, it is important to confirm that it is compatible with the ESP32 board and its operating system.

It is possible to utilise the AES and ECDH encryption algorithms on the ESP32 board, according to the evaluation of these criteria. The ESP32 board is equipped with enough hardware, memory, and processing power to enable these algorithms. But it's important to assess the system's energy usage to make sure it stays within reasonable bounds. To ensure that the algorithm can be effortlessly integrated into the system, it must also be confirmed that it is compatible with the ESP32 board and its operating system.

Overall, the ECDH and AES-based security model for IoT using ESP32 has the potential to give a trustworthy and effective security solution to protect IoT devices. However, a feasibility study is necessary to determine the technical, economic, and operational feasibility of implementing this security model in real-world IoT applications.

CHAPTER 3: SYSTEM DESIGN & ANALYSIS

3.1 Project Perspective

We have built an innovative method using the Serial Peripheral Interface Flash File System (SPIFFS) to set up a file uploader system on the ESP32. A compact and effective file system called SPIFFS was created especially for flash-memory microcontrollers like the ESP32. Our method offers a user interface that is comparable to a standard file system on a computer, but with limited access for increased security, making it simple to access the ESP32's flash memory. This method enables us to build a web server, write configuration files, save modest quantities of data without the need for a microSD card, and even save CSS and HTML files.

Additionally, our solution offers a perfect and effective way to upload files to the file system of the ESP32. To do this, we've created a new folder called "data" in the same repository as our sketch folder, and put the files we want to submit there. The "ESP32 Sketch Data Upload" feature is then accessed in the Arduino IDE's "Tools" menu. The files are imported into the ESP32 file system with the push of a button after choosing the ESP32 board and port.

Our solution offers a straightforward and efficient way to post files to the ESP32's file system without the use of extra hardware or complicated setups. Due to this, it serves as an innovative and ideal solution for IoT applications.

A hybrid model that combines symmetric key method with elliptical curve cryptography is put forth. ECC is used to complete the user verification process and prolong the security of the private data. The AES algorithm is employed, allowing users to securely record and retrieve their file in the cloud by data encryption on the user side and decoding it after obtaining it from the cloud. Although the hacker can access the information through some methods, no one can decrypt it because the private key belongs to the user of the data. Furthermore, when logging in to the cloud server, the client will securely authenticate themselves by using various input parameters.

Users can feel confident in the security of their cloud-stored data thanks to this method. Here, we'll use the ECDH algorithm, which offers the same security level as existing public key cryptosystems while requiring a smaller key size, strengthening the algorithm's security. By permitting a correct access mechanism to prevent unauthorized information from being accessed and a secure storage to permit access to data through the cloud network, the entire prototype of the suggested solution would benefit.

The suggested method clearly demonstrates how successfully AES and ECDH protect cloud-stored data. The suggested method uses AES encryption to turn the input file into encrypted text

after the user uploads it, assuring that the data is completely encrypted. If an attacker wishes to attack the client side of the network to get the user's private data or for any other reason, they can do so by using this. Due to the encryption of the data at the moment of upload, even if an attacker is successful in their attack and is able to retrieve the file uploaded by the user, it is rendered useless. The attacker will also be unable to decode the file if such an attack is launched from the other end.

3.2 System Features

The precise capabilities and functionalities of a software or hardware system are referred to as system features. Some of the important characteristics of the ESP32 file uploading system could be:

1. File management: Users should be able to rename, delete, download and upload files on the ESP32 board via the system.

2. Support for the SPIFFS filesystem, which enables users to access flash memory in the same way as a typical file system on a computer, is a need for the system.

3. Security: To guarantee that sensitive data is shielded from unauthorised access, the system should include secure file transmission and storage capabilities.

4. A wide range of users should have access to the system thanks to its compatibility with various ESP32 boards and operating systems.

5. Error handling: To minimize the risk of data loss or corruption in the event of system problems or crashes, the system should have strong error-handling capabilities.

The file management function on the ESP32 board should also allow for the creation of new folders and file transfers across directories. A file explorer or directory tree, for example, should be able to display the files and folders on the ESP32 board in a way that is both obvious and organised.

To ensure that only authorised users are granted access to the files on the ESP32 board, the system should incorporate authentication and access control procedures. This could involve managing user accounts, encrypting sensitive data, or protecting password

The system should offer error logs and notifications to inform users of any problems that develop during file management activities in order to improve the system's error-handling capabilities.

In the event of problems or crashes, it should also provide a recovery mechanism to return the system to a stable condition.

To ensure peak performance and efficiency, the system should be built to minimise the usage of system resources like memory and computing power. To do this, it could be necessary to install caching systems, compression algorithms, or other optimisation procedures in order to minimise the volume of data transmitted and stored on the ESP32 board.

Overall, the ESP32 file uploader system should offer a full range of functions and functionalities that let users securely and effectively manage files on their ESP32 board.

3.3 Proposed Model

In the implementation of this project, C and various libraries are used in the Arduino IDE to create a filesystem upload and encrypt the files uploaded. Arduino IDE is an integrated development environment used to develop and program microcontroller boards, including the ESP32 board. For creating, assembling, and posting code onto an Arduino board, it offers a simple user interface.

The ESP32 board, which is based on the ESP32 microprocessor, is a potent microcontroller. It features dual-core processors, Wi-Fi, Bluetooth, and a range of peripheral interfaces, making it a popular choice for IoT projects. To utilize the Arduino IDE with ESP32 , you will first need to install the ESP32 board in the Arduino IDE. This can be done by adding the ESP32 board to the board's manager in the Arduino IDE. The following actions are necessary for the project to be completed successfully:

1. **File uploader system:**

   To set up a file uploader system on the ESP32, we must first download and set up the Serial Peripheral Interface Flash File System (SPIFFS). A lightweight file system called SPIFFS was created for flash-memory microcontrollers like the ESP32. It enables us to access the flash memory in the same way that a normal file system on a computer would but in an easier and more restricted manner. We can utilise SPIFFS with the ESP32 board to build configuration files, save info permanently, store modest quantities of data rather than an SD card, save files containing HTML and CSS to create a web server, and do a lot more. The figure 2.2 shows the design and structure of ESP 32.

Figure 3.1: ESP 32 [22]

After downloading SPIFFS, we must customize the configurations for uploading the data to the ESP32 board. To accomplish this, we set up an additional folder titled "data" within the same repository as our sketch folder and place the documents we would like to upload in it. Then, from the "Tools" menu, we access the "ESP32 Sketch Data Upload" tool, select the ESP32 board and port, and select "Upload" to import the files to the ESP32 file system.

Using the ESP32 Filesystem Uploader tool in the Arduino IDE, we can create a filesystem uploader for ESP32 by going through the following procedures.

2. ECDH to generate Key:

A few key steps must be taken in the creation of a secure secret key that is shared using ECDH key exchange. First, the elliptic curve constraints, such as the curve equation and generator point, must be agreed upon by both parties. It is critical to choose a widely accepted curve, for instance, the secp256r1 elliptic curve, which is strongly advised by NIST and used in a variety of cryptographic applications. This curve has a 256-bit key size and is based on a prime field, offering a secure connection.

Following that, each side needs to produce a private key, which consists of a random value ranging from 1 and the order of the curve that is kept secret. Then, each party generates a public key by multiplying the curve's generator point by their private key. This public key is shared with the other party.

The shared secret key is then determined by multiplying the opposite party's public key by their private key. The resulting curve point is hashed to produce a binary shared secret

key. Using symmetric-key encryption algorithms such as AES, this shared secret key is utilized to encode and decode messages.

Although ECDH also creates a shared secret key, it is unable to protect the groups from man-in-the-middle attacks or secure them. To solve these problems and ensure secure communication, additional protocols such as digital signatures and certificates are required. ECDH may serve as an efficient and safe way to generate shared secret keys if these steps are followed and additional security measures are implemented.

3. AES to encrypt file

One of the key pairs generated by the elliptic curve defined over a field is used by the AES to encode the text file after the ECDH key exchange procedure is finished. AES uses a key obtained from the secret key that was shared and produced during the ECDH key exchange to perform the encryption. One key, say "d," should be kept a secret from the server, while another, say "e," would be kept hidden from the client. These keys will be combined to form the combined key which is employed for both decryption and encryption. A brief description of the design is illustrated in figure 2.3.

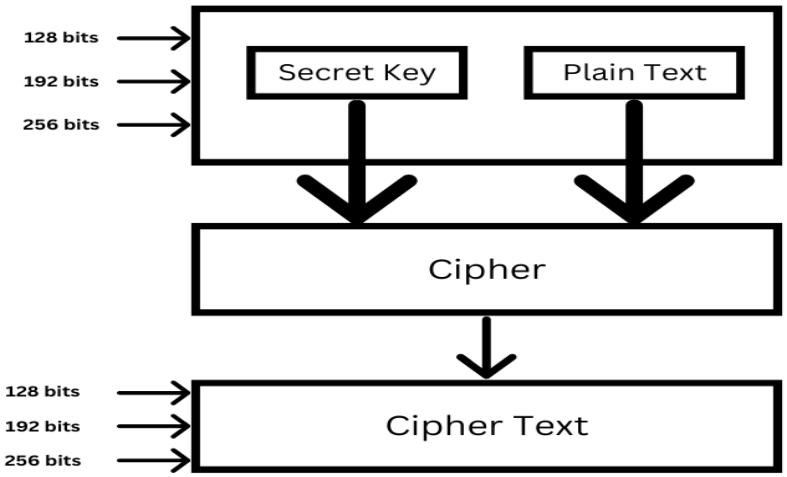

Figure 3.2: AES Design [23]

As a shared secret, AES will encrypt the input text file using the merged key generated by the ECDH. The encrypted content that results is then posted to the server. The customer will install the encoded document from the server and decipher it using the same merged key created by the ECDH after effectively setting up a shared secret between the client and server. The user gets the original file on such satisfactory

implementation of the decryption. AES, therefore, fulfills the gap left by ECDH, which only generates a shared secret key but does not perform encryption.

Overall, the combination of AES and ECDH algorithms provides a highly secure communication system for the filesystem uploader in the ESP32 device. ECDH generates a shared secret key that is used for encrypting and decrypting data, while AES ensures the confidentiality and integrity of the transmitted data. This approach ensures that sensitive data is protected from potential attacks and unauthorized access, making it an ideal solution for various IoT applications. The use of these advanced encryption techniques is a significant step towards enhancing the security of IoT devices and networks, ensuring that data remains secure and confidential throughout transmission.

3.4 System Architecture

In our suggested method, a secure communication system for the filesystem uploader in the ESP32 device is provided by combining AES and ECDH. The algorithm outlined below uses these two encryption techniques to create a shared secret key that may be used to encrypt and decrypt data and protect sensitive data.

Our method assures that the data transmission between the ESP32 and the server is secure, guarding it from unauthorized access and guaranteeing that any sensitive information is kept confidential by combining AES and ECDH. Given below are the steps required to implement our proposed algorithm.

Step 1: The client will ask the server for a file, and the server will choose that file for encryption after receiving the client's request.

Step 2: Elliptic curve will produce several private and public key pairs after receiving a file as input. AES will use one of the key pairs produced by an elliptic curve specified over a field to encrypt the text file. AES performs the encryption. One key will be kept secret with the server, say "d," and one key will be kept hidden with the client, say "e," from among the several key pairs.

Step 3: By achieving a successful key agreement between the two communicating parties, ECDH will establish a shared **s**ecret between the client and server. Therefore, the shared secret will only be between the server and client for that specific session, and a third party must discover a way to the discrete logarithm problem to discover the shared secret. Diffie-Hellman will handle the key agreement between the client and server while

AES encrypts the input text file on the other side. The user will only be able to decode the encrypted text file if the agreement is successful.

Step 4: The input text file will be encoded with AES using the ECC-generated key. Following encryption, the encrypted content is uploaded to the server using the combined key, or another key that has been received using DHKE as a shared secret. After both the client and the server have successfully established a secret key, the client will obtain the encoded document from the server and decrypt it using the combined key created by ECC and DHKE.

Step 5: Once the encrypted file is transmitted to the server, it is decrypted using the shared secret key generated by the ECDH algorithm and then decrypted using the AES algorithm. The decrypted file is then sent back to the client. This process ensures the confidentiality and integrity of the transmitted data, making it highly secure against unauthorized access and tampering attempts. Upon successful decryption of the file, the client receives the original file, ensuring the smooth and efficient transfer of data between the client and server. This secure file transfer mechanism provides a highly reliable and robust approach for transferring sensitive data, which is critical in various industries and applications where data security and privacy are of utmost importance.

The figure 3.1 depicts the flow chart for the proposed hybrid decryption and encryption technique. Elliptical Curve Cryptography (ECC), an asymmetric key method, produces a key that is used to create a connection. ECC and AES are used in a symmetric hybrid technique to encrypt images. The sender receives the encrypted image in a secure manner. The receiver uses this symmetric hybrid technique to decrypt the image, and the original image obtained is compressed.

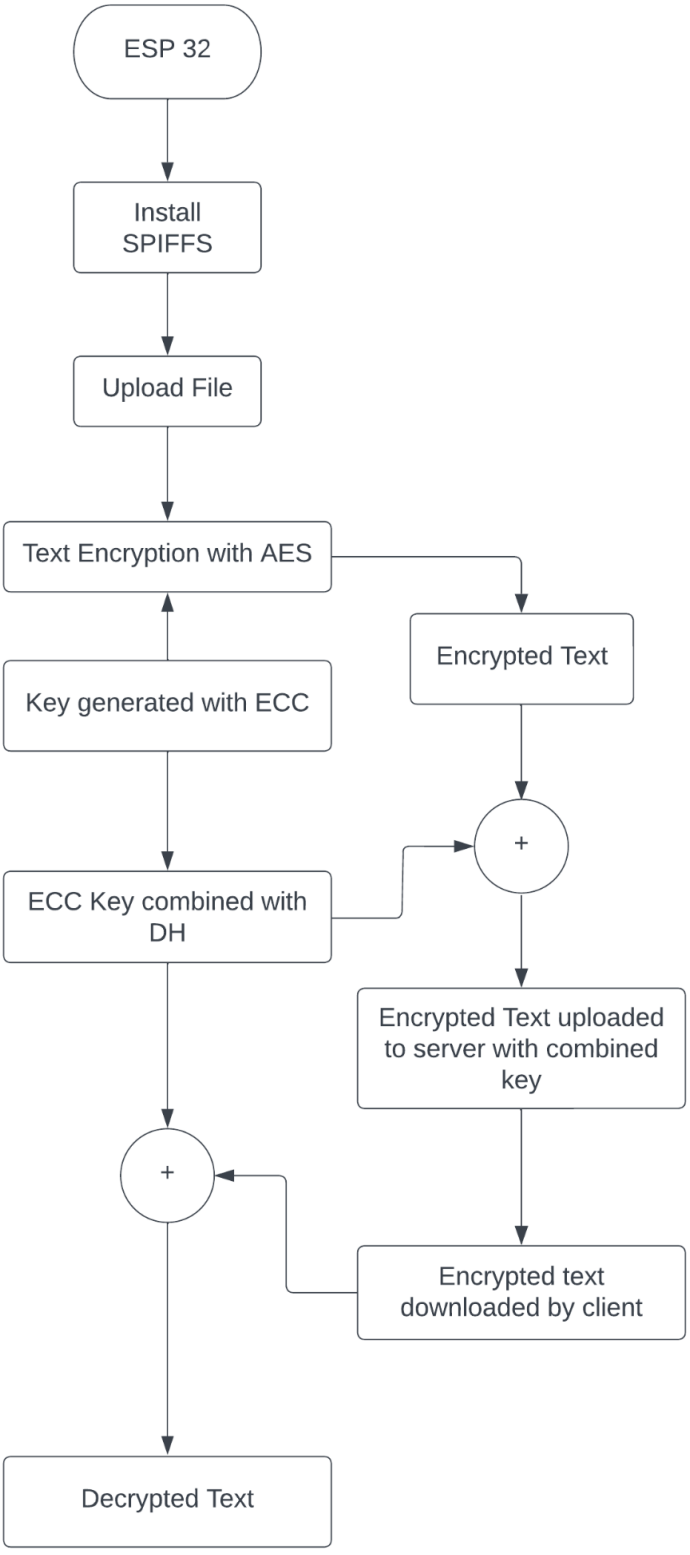

Figure 3.3: Representation of ECDH and AES on ESP 32

ALGORITHM: Hybrid Cryptography Encryption using AES and ECDH for ESP 32

INPUT:

- Files that are stored in a "data" folder within the sketch repository.
- The Elliptic Curve parameters need to be defined
  - ECC represented by an equation in the form $y^2 = x^3 + ax + b$
  - Generator point G on that curve
  - The order of the curve
- Other party public key needs to be received for the key exchange process

OUTPUT:

- Decrypted file saved to the ESP32 filesystem

Procedure :

Step 1 - Initialize the ESP32 board and SPIFFS filesystem. (SPIFFS (Serial Peripheral Interface Flash File System) is a file system designed for use with small embedded systems that use flash memory as their primary storage medium)

Step 2 - Create a folder named "data" within the sketch repository and place the files to be uploaded in it.

Step 3 - Use the ESP32 Sketch Data Upload tool to upload the files to the ESP32 filesystem.

Step 4 - Define the elliptic curve parameters, including the curve equation and generator point. Let E be the elliptic curve, G be a point on the curve, and n be the order of the curve.

Step 5 - Generate a private key (d) for the ESP32 device.

Step 6 - Generate a public key(q) by multiplying the generator point(G) by the private key (d)

$$Q = G * d$$

Step 7 - Share the public key with the other party(q).

Step 8 - Receive the other party's public key(q`).

Step 9 - Generate a shared secret key (K) by multiplying the other party's public key (q`) by the ESP32 device's private key(d`). K=q`*d`

Step 10 - Hash the shared secret key(K) to produce a binary key(K`).

Step 11 - Use the binary key(K`) to generate a combined key(C) for AES encryption and decryption.

- C = H(K')--> AES encryption
- C = H(K')--> AES decryption

where H() represents a cryptographic hash function used to convert the binary key K' into a fixed-size hash value that can be used as the combined key for AES

Step 12 - Open the file to be encrypted.

Step 13 - Use AES encryption with the combined key to encrypt the file.

Step 14 - Upload the encrypted file to the server.

Step 15 - Download the encrypted file from the server.

Step 16 - Generate the combined key again using the shared secret key.

Step 17 - Use AES decryption with the combined key to decrypt the file.

Step 18 - Save the decrypted file to the ESP32 filesystem.

3.5 Methodology

We have proposed a hybrid model that combines symmetric key algorithms and Elliptical Curve Diffie Hellman key exchange in an ESP 32 board. ECC is utilized to complete the user verification procedure and maintain the privacy and integrity of the sensitive data. Data is encoded on the client end and only decoded after being transferred from the cloud. Users may

safely maintain and retrieve their data in the cloud thanks to the AES algorithm. No one is able to decode the data since the end user has the secret key. Hackers may be able to obtain the data through some means, but they would not be able to browse it. Furthermore, while logging in to the cloud server, the client will safely verify their identity by utilising various input specifications. Users may feel confident in the security of their cloud-stored data thanks to this method. Here, we'll use the ECC and ECDH algorithm, which boosts the security of the algorithm while offering the security equivalent as other cryptography technologies with smaller key lengths. Benefits include a suitable access control system to prevent unauthorised access to a system secure data access and storage

3.5.1 Espressif Systems Wi-Fi and Bluetooth System on Chip (ESP 32):

A microcontroller board called the ESP32 comes with a dual-core processor, wireless connection, and a variety of input/output (I/O) options. It’s essential characteristics include:

1. CPU with two cores: The ESP32 is a dual-core CPU that enables high-performance computing.
2. Wi-Fi and Bluetooth are supported by the ESP32, making it simple to connect to other devices and networks.
3. I/O capabilities include digital and analogue inputs and outputs, Pulse Width Modulation (PWM) outputs, and serial communication ports. The ESP32 also includes a wide range of I/O capabilities.
4. Low power consumption: The ESP32 is made with many low-power modes to aid prolong battery life. It is designed to be energy-efficient.
5. A development board, such as the ESP-WROOM-32, and an integrated development environment are required to use the ESP32.

In general, the ESP32 offers a strong and adaptable platform for creating a variety of embedded systems, from straightforward IoT devices to more intricate robotics applications. It is the perfect option for many applications thanks to its mix of high-performance computation, wireless connection, and low power consumption.

3.5.2 ESP 32 and File Uploader System:

Building a file uploader system, which enables users to manage files on the ESP32 board through a web interface, is one of the most common use cases for the ESP32.

The Serial Peripheral Interface Flash File System (SPIFFS) filesystem, which is intended for use with flash-memory microcontrollers like the ESP32, can be used to build a file uploader system on the ESP32. Similar to a conventional file system on a computer, the ESP32 board's SPIFFS file system offers a quick and effective way to manage files.

You must first download and configure the SPIFFS filesystem on the ESP32 board before configuring the settings for data uploading. The files you want to upload can be done by making a "data" folder in the same repository as your sketch folder. The files can then be uploaded to the ESP32 file system using the ESP32 Sketch Data Upload function in the Arduino IDE.

Using the ESP32's built-in Wi-Fi capabilities, you may make a web interface to handle the files after they have been uploaded. You can create a web server on the ESP32 using the ESPAsyncWebServer package, which offers a quick and easy method of doing so. On the ESP32 board, files can be uploaded, downloaded, deleted, and renamed using the web interface.

Overall, the ESP32's dual-core processor, integrated flash memory, and Wi-Fi capabilities make it a potent platform for creating file uploader systems. A reliable and effective file uploader system that can be utilised for a variety of IoT projects may be built utilising the SPIFFS filesystem.

3.5.3 Elliptic Curve Cryptography (ECC):

Victor Miller and Neal Koblitz initially presented elliptic curve systems in 1985. A set of locus (Xi, Yi) on an elliptic curve over the field K are located in the plane. The finite set is represented by the symbol E. One of its most secure algorithms is this one. Public key cryptography uses the ECC algorithm, in which every user has a unique set of public and private keys. The group operator, represented by the symbol "+," is a crucial one in ECC. For certain fixed values for the parameters p and q,

$$y^2 = x^3 + px + q \qquad \text{Eqn(1)}$$

gives the standardised format of ECC. The capacity to compute additional points on the curvature and then encrypt those points before information is transmitted between end users determines how secure the ECC Algorithm is. P, a locus on the curve, may be found using the group operator , which makes it exceedingly challenging for the attacker to get the private information.

3.5.4 Key Agreement Using the ECDH Algorithm:

Cloud A and Cloud B will both agree to accept the data being transported. There won't be a legally enforceable agreement between the parties unless both keys are the same.

1) As his or her private key, A will choose an integer XA = K1. The public key for A will be Y if the secret key is an arbitrary integer then assume the public key is a point like P.

$$A = XA * P \qquad \text{Eqn(2)}$$

2) B chooses a number XB = K2 as his /her secret key, with YB =XB x P serving as B's public key. Both actions are identical. Then the public keys are exchanged between the two parties.

3) A determines the session key using the formula

$$KA = XA * YB = K1 * K2 * p \qquad \text{Eqn(3)}$$

4) B uses Eqn 2 to calculate the session key. Clearly, KA = K this demonstrates the agreement between the data communication between two parties and creation of the public and secret keys.

3.5.5 Key Generation: The process of creating both a public key and a private key is known as key generation. To decode the communication, the recipient will use its secret key. Once it has been encrypted by the sender using the receiver's public key. The next step is to choose a number "d" from the available range of "n". We may create the public key using the equation:

$$Q = d * p \qquad \text{Eqn(4)}$$

d = The arbitrary value we've chosen between the ranges of (1 to n-1).

p is the curve's pivot locus.

The public key is "Q,"

and the secret key is "d."

Public and secret keys are generated using an algorithm. In this instance, the data will be encrypted using Sender, and B, the receiver, will use his or her own secret key to decrypt it.

3.5.6 Encryption:

Assume that originator A sent receiver B the message "m". Only the encryption will actually occur during the message's transmission from Sender A to the receiver, and it will only take a few nanoseconds for the data to get there.

3.5.7 File encryption using the Advanced Encryption Standard (AES):

US government agencies routinely employ AES. For converting plaintext into ciphertext, often known as encrypting and decrypting data. The suggested approach secures the submitted files by using AES encryption and a secret key. The user must choose the file to encrypt and enter the private key. After the file has been encrypted, the user must enter the AES key that was developed during encryption in order to decode it.

3.6 Testing Process

While testing software is a key step in ensuring the effectiveness and accuracy of programmers, it is especially important when using encryption techniques like AES and ECDH on ESP32 devices. These techniques generate a shared secret key that is used to encrypt and decrypt data in order to establish a secure communication system for the ESP32 filesystem uploader. To maintain the security of sensitive data, it is crucial to thoroughly examine these procedures.

We will go over the test cases created and executed to assess the efficiency of the AES and ECDH encryption algorithms on ESP32 devices in this part. These test cases are intended to find any implementation flaws and make the necessary adjustments to confirm the system's appropriateness for real-world scenarios.

3.6.1 Unit Testing

Unit testing is the practice of testing individual parts or units of code in isolation to make sure they work as intended. This is relevant to the proposed approach using AES and ECDH for secure communication in the ESP32 device. This entails creating tests for each function or module of the code, running them to see if they work as intended, and fixing any flaws or errors before integrating the code with the rest of the system. Unit testing is a crucial step in guaranteeing the overall quality and dependability of the system since it enables early issue discovery and resolution, which ultimately saves time and resources.

The modules we will be testing here are:

1. File Storage Uploader on ESP 32: the test case for this module is defined in table 3.1. The table has verified that our model for storage of files is working efficiently and without any bugs.
2. Shared key generation using ECDH: the description for this particular model is in table 3.2, this table helps justify that the shared key is generated without any problem, thus making our key more secure.

3. Encryption of text using AES: table 3.3 helps describe the test case here, and validate the working of text encryption using AES in our approach.

Test Case Descriptions:

Table 3.1: Test Case 1

Test Case ID	1
Test Case Name	File storage uploader using ESP 32
Test Case Description	Check if file is being uploaded using ESP 32
Steps	1. Prepare a text file 2. Connect ESP 32 3. Run the program 4. Print the contents of the file
Expected Results	Contents of file
Actual Results	File Content: Test spiffs My name is tanya. I am b.tech computer science student from sharda university. Im in my final year. This is my final year project.

Table 3.2: Test case 2

Test Case ID	2
Test Case Name	Shared Key Generation using ECDH
Test Case Description	Check if a shared secret key is generated
Steps	1. Using a given curve create a pair of elliptic curve keys for P. 2. Create a pair of keys with the same elliptic curve for Q. 3. Using P's private key and Q's public key, create the shared secret key for P and Q using the ECDH technique.
Expected Results	Shared secret key is successfully generated
Actual Results	Shared secret: BA 46 B9 81 69 52 27 49 8F B7 14 D0 66 CD 76 D1 8

Table 3.3: Test Case 3

Test Case ID	3
Test Case Name	Text encryption using AES
Test Case Description	To verify if our text is encrypted
Steps	1. Choose a message 2. Generate a random 256 bit encryption key 3. Encrypt the message
Expected Results	The message should be displayed as a cipher-text
Actual Results	Let's encrypt: IV b64: AAAAAAAAAAAAAAAAAAAAAA== Message: testing of module 2 that is AES encryption of this message Message in B64: dGVzdGluZyBvZiBtb2R1bGUgMiB0aGF0IGlzIEFFUyBlbmNyeXB0 The lenght is: 80 Encryption done! Cipher size: 80 Encrypted data in base64: E/twfjOyDi+gBdGBIIbQof0ChIwit3uCLu7rdSj9mg0 Done...

3.6.2 Integration Testing

The suggested method integrates two crucial elements, ECDH and AES, to produce a secure communication system for the filesystem uploader in the ESP32 device. To ensure their flawless connection and functionality, these components must be carefully planned during the integration process.

As unit-tested components, like data, are grouped into bigger aggregates and tested as a whole, integration testing is a crucial step in the development of any software model. The integrated testing framework is created by applying the integration test plan tests to these aggregates.

To ensure the secure production and usage of the shared secret key for encrypting and decrypting uploaded data in the case of the proposed approach, integration testing will examine the interaction between the ECDH and AES components. Through this testing, any potential flaws in the integration of these parts will be found, and it will be made sure that they operate well together to create a secure communication system. Table 3.4 has defined the test case, the steps and results for the successful integration of AES and ECDH.

Test Case Description

Table 3.4: Test Case 4

Test Case ID	4
Test Case Name	Integration of AES and ECDH
Test Case Description	Verify if a secret key is generated using ECDH and text is encrypted using AES
Steps	1. Generate secret key 2. Input message 3. Integrate ECDH secret key with AES 4. Encrypt message
Expected Results	Message encrypted
Actual Results	Shared secret key produced using ECDH: 2C4F8B60F6191EB9FDC8E5C3D316B3 Text to be encrypted: hello, this is the integration testing for AES and ECDH encryption Message in B64: aGVsbG8sIHRoaXMgaXMgdGhlIGludGVncmF0aW9uIHRlc3RpbmcgZ Encryption done! Cipher size: 96 Encrypted data: M14XDSVpO+g7QCTX2D+rY7fKk1TXb1/ZgaOonRKg8CuR0yTWLHfe1 Done...

3.6.3 System Testing

System testing is a subset of software testing that assesses a system's overall functionality. System testing would entail checking the complete system, including all interconnected components, to make sure it complies with the functional and non-functional requirements in the context of our suggested model for secure file transfer.

The ESP32 device and the cloud storage system would both be tested as part of the system testing procedure in a real-world setting. To make sure the data is secure and the system operates as expected, it would be necessary to verify the full end-to-end data transfer flow, from file upload to decryption. We can make sure that our proposed model is reliable, secure, and satisfies the requirements of the real-world by thoroughly conducting system testing. Table 3.5 has defined the system testing for our proposed system.

Test Case Description

Table 3.5: Test case 5

Test Case ID	5
Test Case Name	Test the entire approach

Test Case Description	To verify if our file is uploaded and encrypted
Steps	1. Prepare text file 2. Connect ESP 32 3. Upload text file 4. Generate secret key using ECDH 5. Integrate key with AES algorithm 6. Encrypt text file 7. Print contents of the encrypted text file.
Expected Results	Contents of file that is printed are encrypted
Actual Results	Shared secret key produced using ECDH: 2C4F8B60F6191EB9FDC8E5C3 File Content: Test spiffs My name is tanya. I am b.tech computer science student from sharda university. Im in my final year. This is my final year project. Message in B64: VGVzdCBzcGlmZnMNCk15IG5hbWUgaXMgdGFueWEuDQpJIGF Encryption done! Cipher size: 208 Encrypted data: 28fMY092HTvKD3TNKmfq9lzMOLIOCARSCNH+evTVab0mVkL Done...

CHAPTER 4: RESULTS AND DISCUSSION

Numerous data on cloud storage are protected and get protected connectivity for the encryption and decryption of the data by applying a combination of AES and ECDH, which in a novel method improves the system's distinctiveness and effectiveness. This suggests that by utilising these two techniques, the user will be able to comprehend the original information with ease. Java is used to implement the algorithms (Eclipse Platform Version: 3.3.1.1). A text-based file will serve as the implementation dataset.

The encryption and execution times of our suggested method are used to evaluate it. The time spent by the system carrying out a process, as well as the time spent by the system performing run-time or system tasks on its behalf, is indicated as the implementation time or computational time of that activity. We have measured the execution duration in this case in milliseconds. The system must be quick and responsive with a shorter execution time. The execution times of AES, ECDH, and AES and ECDH on identical text files are shown in Fig. 4.1.

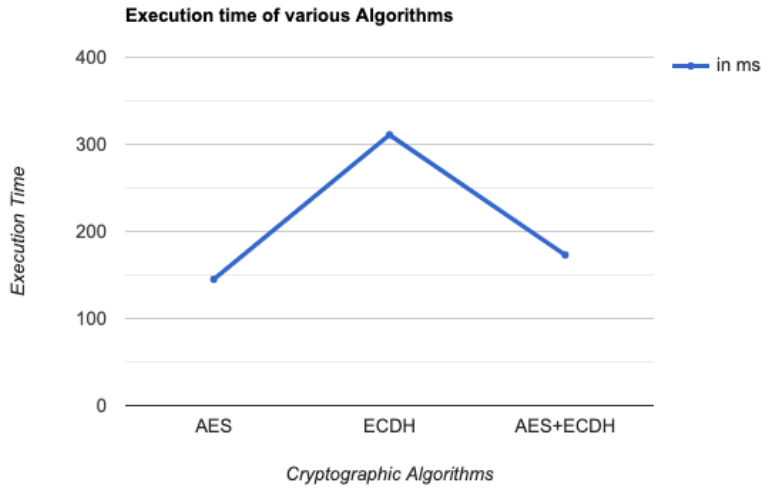

Figure 4.1: Execution time of algorithm

The amount of time required to convert plaintext to ciphertext is referred to as encryption time.

The technique, plaintext block size, and key size all influence the length of the encryption. In our experiment, we measured the encryption time in milliseconds. The system's performance is impacted by the encryption time. Less time must pass for encryption, resulting in a quick and responsive system. Fig. 4.2 shows the encryption time of AES, ECDH, and AES and ECDH on the same text files. The encryption time of the suggested method is the least

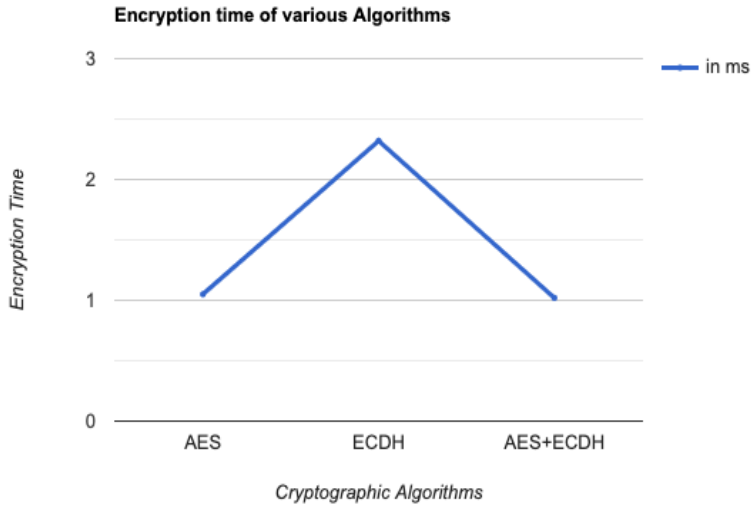

Figure 4.2: Encryption time of algorithm

The table 4.1 below shows the relationship between the execution and encryption time of the algorithms.

Table 4.1: Algorithms with Execution and Encryption time

Algorithms	Execution Time in milliseconds	Encryption Time in milliseconds
AES	145	1.05
ECDH	311	2.32
AES+ECDH	173	1.02

In general, AES is considered to be faster than ECDH, as AES is a symmetric cryptographic algorithm and operates on fixed-length blocks of data, whereas ECDH is an asymmetric encryption algorithm that operates on points on an elliptic curve. However, the speed of AES also depends on the key size, with larger key sizes being slower.

ECDH is generally slower than AES, but it has the advantage of being more secure for a given key size. In comparison to RSA, a popular public-key encryption technique, ECDH uses smaller

key sizes while offering the same security features as RSA with higher key sizes. The speed of ECDH also depends on the specific implementation and the size of the key being used.

The combination of AES and ECDH (i.e., AES+ECDH) is often used to provide both confidentiality and key agreement, and its execution time will depend on the specific implementation and how the two algorithms are combined. It's important to note that the execution time of encryption algorithms is only one factor to consider when choosing an encryption algorithm. Security, compatibility, and other requirements must also be considered when making a decision for choosing an encryption algorithm for a specific use case.

The proposed method demonstrates how well AES and ECDH protect data recorded in the cloud. The new proposed model, which depicts secure user information transmission to the server and subsequent secure storage mechanism owing to encrypted data, reveals how imaginative the proposed technique is. Furthermore, computing time and cost may be utilized to assess innovation. To avoid assaults, the following approach can be used: The input file is transformed to encrypted text once the user uploads it using AES encryption, ensuring that the content is entirely encrypted. This is useful if a hacker wants to target the client to obtain private information or for unethical purposes. As a result of this, even if an attack succeeds and the user-uploaded file is retrieved, the data has already been encrypted when the file was submitted. Similarly, if an assault is conducted from the opposite end, the attacker will be unable to decrypt the encrypted file, therefore protecting the data.

CHAPTER 5: CONCLUSION AND FUTURE SCOPE

5.1 CONCLUSION

If sensitive file data is protected, a cloud-stored file can be transmitted and utilized by others. However, cloud storage also comes with a range of negative challenges, including data security concerns, that have greatly hampered its future adoption. In the context of exchanging files via an insecure network, data security is essential to protect the information being transmitted. Cryptography provides a wide range of techniques to safeguard such communications, allowing information to be securely transferred across the wireless medium while also providing authentication, data integrity, privacy, and non-repudiation. To protect communication from external threats on ESP32 and other microcontrollers, this research suggests a hybrid model that combines the characteristics of the AES algorithm and ECDH. The suggested prototype is implemented on a client-server model, with the user communicating with the server and securely managing all information using the AES-ECDH hybrid model. The AES-ECDH hybrid technique is highly secure and difficult to breach, making it a suitable solution for ESP32-based IoT devices.

Overall, the hybrid model of AES and ECDH for secure file transmission has a promising future scope in various industries where secure file transmission is crucial, particularly on ESP32-based systems. Its low encryption time of 1.02 milliseconds makes it a viable solution for real-time file transmission, and its superior security features ensure the privacy and security of transmitted files.

5.2 FUTURE SCOPE

The proposed model of combining AES with ECDH for cloud security is a strong foundation for securing cloud storage systems. However, there are further improvements and expansions that could be made to enhance its effectiveness. One such improvement is multi-cloud support, which would enable users to store their data across different cloud platforms, increasing data availability and redundancy. Additionally, integrating artificial intelligence (AI) into cloud security could automatically detect and prevent security threats. By analyzing user behavior and

network traffic, AI-based security can identify abnormal activity and take preventive measures. Another potential enhancement is to incorporate a more advanced intrusion detection system that could help detect and prevent unauthorized access attempts to the cloud storage system. Overall, these enhancements would further strengthen cloud security and help to safeguard sensitive data.

REFERENCES

[1] Kodali, Ravi Kishore, and Ashwitha Naikoti. "ECDH based security model for IoT using ESP8266." 2016 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT). IEEE, 2016.

[2] Abdullah, Shapina, et al. "IOT Security: Data Encryption for Arduino-based IOT Devices." *Journal of Positive School Psychology* 6.3 (2022): 8508-8516.

[3] H. P. T. M. Jayawardana and R. L. Dangalla, "Hybrid encryption protocol for RFID Data Security," 2020 International Conference on Decision Aid Sciences and Application (DASA), Sakheer, Bahrain, 2020, pp. 1209-1212, doi: 10.1109/DASA51403.2020.9317034.

[4] Phimphinith, Anothay, et al. "An Enhanced Mutual Authentication Scheme Based on ECDH for IoT Devices Using ESP8266." *2019 IEEE 11th International Conference on Communication Software and Networks (ICCSN)*. IEEE, 2019.

[5] Tulip Dutta , Amarjyoti Pathak , "Secure Data Sharing in Cloud Storage Using Key Aggregation Cryptography," International Journal of Computer Sciences and Engineering, Vol.04, Issue.07, pp.20-23, 2016.

[6] Bhale Pradeep kumar Gajendra, Vinay Kumar Singh, More Sujeet, "Achieving cloud security using third party auditor, MD5 and identity- based encryption", ICCCA2016

[7] Khan, Ayaz & Al-Mouhamed, Mayez & Almousa, A. & Fatayar, A. & Ibrahim, A. & Siddiqui, A.. (2014). AES128 ECB encryption on GPUs and effects of input plaintext patterns on performance. 1-6. 10.1109/SNPD.2014.6888707.

[8] Sharma, S., & Chopra, V. (2017). Data Encryption using Advanced Encryption Standard with Key Generation by Elliptic Curve Diffie-Hellman. International Journal of Security and Its Applications, 11(3), 17–28. https://doi.org/10.14257/IJSIA.2017.11.3.02

[9] A. Abusukhon, Z. Mohammad and A. Al-Thaher, "Efficient and Secure Key Exchange Protocol Based on Elliptic Curve and Security Models," 2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT), Amman, Jordan, 2019, pp. 73-78, doi: 10.1109/JEEIT.2019.8717496.

[10] Lara-Nino, C. A., Diaz-Perez, A., & Morales-Sandoval, M. (2018). Elliptic curve lightweight cryptography: A survey. *IEEE Access*, *6*, 72514-72550.

[11] Patgiri, Ripon. (2021). privateDH: An Enhanced Diffie-Hellman Key-Exchange Protocol using RSA and AES Algorithm. 10.13140/RG.2.2.23938.40647.

[12] Babiuch, Marek & Foltynek, Petr & Smutný, Pavel. (2019). Using the ESP32 Microcontroller for Data Processing. 1-6. 10.1109/CarpathianCC.2019.8765944.

[13] A. Maier, A. Sharp and Y. Vagapov, "Comparative analysis and practical implementation of the ESP32 microcontroller module for the internet of things," 2017 Internet Technologies and Applications (ITA), Wrexham, UK, 2017, pp. 143-148, doi: 10.1109/ITECHA.2017.8101926.

[14] O. Barybin, E. Zaitseva and V. Brazhnyi, "Testing the Security ESP32 Internet of Things Devices," 2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), Kyiv, Ukraine, 2019, pp. 143-146, doi: 10.1109/PICST47496.2019.9061269.

[15] "The Internet of Things with ESP32", Esp32.net, 2019. [Online]. Available: http://esp32.net/.

[16] Bharathi, P., Annam, G., Kandi, J. B., Duggana, V. K., & T., A. (2021). Secure file storage using hybrid cryptography. *2021 6th International Conference on Communication and Electronics Systems (ICCES)*. https://doi.org/10.1109/icces51350.2021.9489026

[17] Hodowu, Dickson & Korda, D. Redeemer & Danso Ansong, Edward. (2020). An Enhancement of Data Security in Cloud Computing with an Implementation of a Two-Level Cryptographic Technique, using AES and ECC Algorithm. International Journal of Engineering and Technical Research. 9. 639-650.

[18] Goyal, S., & Jain, S. (2016). A secure cryptographic cloud communication using DNA cryptographic technique. *2016 International Conference on Inventive Computation Technologies (ICICT)*. https://doi.org/10.1109/inventive.2016.7830158

[19] Rehman, Saba & Bajwa, Nida & Shah, Munam & Aseeri, Ahmad & Anjum, Adeel. (2021). Hybrid AES-ECC Model for the Security of Data over Cloud Storage. Electronics. 10. 2673. 10.3390/electronics10212673.

[20] Pavithra, R., Prathiksha, S., Shruthi, S.G. and Bhanumathi, M., 2021. A New Approach for Security in Cloud Data Storage for IOT Applications Using Hybrid Cryptography Technique. In Advances in Parallel Computing Technologies and Applications (pp. 175-182). IOS Press.

[21] Kumar, A., Jain, V. and Yadav, A., 2020, February. A new approach for security in cloud data storage for IOT applications using hybrid cryptography technique. In *2020*

international conference on power electronics & IoT applications in renewable energy and its control (PARC) (pp. 514-517). IEEE.

[22] https://www.espressif.com/en/products/modules/esp32

[23] Heron, S. (2009). Advanced encryption standard (AES). Network Security, 2009(12), 8-12.

[24] Syam Akhil Repalle, Venkata Ratnam Kolluru, "Intrusion Detection System using AI and Machine Learning Algorithm », International Research Journal of Engineering and Technology (IRJET), Volume: 04 Issue: 12, Dec-2017

[25] A. Bhumgara and A. Pitale, Detection of Network Intrusions using Hybrid Intelligent Systems, 2019 1st International Conference on Advances in Information Technology (ICAIT), Chikmagalur, India, 2019, pp. 500-506, doi: 10.1109/ICAIT47043.2019.8987368.

[26] https://www.nsc42.co.uk/post/cloud-misconfiguration-leads-to-breaches

[27]https://info.townsendsecurity.com/bid/72450/what-are-the-differences-between-des-and-aes-encryption

[28] Atul Kahate “Cryptography and Network Security”, Tata McGraw-Hill Companies, 2008.

[29] Xin Zhou and Xiaofei Tang, "Research and implementation of RSA algorithm for encryption and decryption," Proceedings of 2011 6th International Forum on Strategic Technology, 2011, pp. 1118-1121, doi: 10.1109/IFOST.2011.6021216.

ANNEXURE

1. Kanishka Negi, Ronika Shrestha, Tanya Lillian Borges, Subrata Sahana, Sanjoy Das, “A Hybrid Cryptographic Approach for Secure Cloud-Based File Storage”, IEEE IAS Global Conference on Emerging Technologies **(GlobConET SCOPUS Index)** 2023. **Accepted and Registered**
2. Tanya Lillian Borges, Ronika Shrestha, Kanishka Negi, Subrata Sahana,” An intelligent intrusion detection prevention system using k-means clustering”,Springer Recent Developments in Cyber Security (**ReDCySec-2023 SCOPUS Index**)
   Communicated