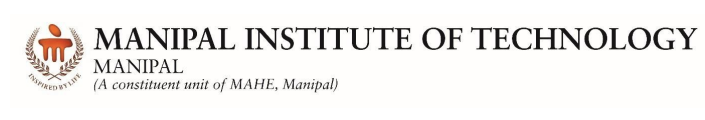
**File Transfer System Using Elliptic Curve Cryptography**

*A project report submitted  
to*  
**MANIPAL ACADEMY OF HIGHER EDUCATION**  
  
*For Partial Fulfillment of the Requirement for the  
Award of the Degree  
of***Bachelor of Technology**  
*in*  
**Information Technology**  
  
  
*by*  
  
**Nilay Yadav 225811382**

**Anvitha Karanth 225811384**  
  
*Under the guidance of*  
  
Dr. Abhijit Das  
Assistant Professor - Senior Scale  
Department of IT  
Manipal Institute of Technology  
Bengaluru, India



**Nov 2024**

**DECLARATION**

I hereby declare that this project work entitled **'File Transfer System Using Elliptic Curve Cryptography'** is original and has been carried out by me in the Department of Information Technology at Manipal Institute of Technology, Manipal, under the guidance of **Dr. Abhijit Das**, Assistant Professor - Senior Scale, Department of I & T, M.I.T., Manipal. No part of this work has been submitted for the award of a degree or diploma either to this University or to any other Universities.  
  
Place: Bengaluru  
Date: 25-10-24

**Nilay Yadav**

**Anvitha Karanth**

**A close up of a sign

Description automatically generated**

**CERTIFICATE**

This is to certify that this project entitled **'File Transfer System Using Elliptic Curve Cryptography'** is a bona fide project work done by **Mr. Nilay Yadav (Reg. No.: 225811382)** and **Ms. Anvitha Karanth (Reg. No.: 225811384)** at Manipal Institute of Technology, Bengaluru, independently under my guidance and supervision for the award of the Degree of Bachelor of Technology in Information Technology.

Dr. Abhijit Das Dr. Dayananda P  
Assistant Professor - Senior Scale HOD (incharge) Department of I & T Department of I & T  
Manipal Institute of Technology Manipal Institute of Technology  
Bengaluru, India Bengaluru, India

**TABLE OF CONTENTS**

|  |  |
| --- | --- |
| Introduction | 5 |
| Literature Review | 7 |
| Methodology | 10 |
| Implementation | 12 |
| Results and Discussion | 16 |
| Conclusion | 17 |
| References | 18 |
| Github link | 18 |

# Chapter 1 Introduction

# 1.1 Background

In today’s digital age, secure data transfer has become essential to protect information integrity, confidentiality, and authenticity. With the exponential increase in data exchange across various platforms—ranging from personal communications to enterprise-level transactions—safeguarding the security of data in transit is critical. Cybersecurity threats, such as data breaches, unauthorized access, and man-in-the-middle attacks, continually jeopardize the privacy and integrity of sensitive data. These risks drive the need for secure communication channels and the development of advanced cryptographic protocols integrated into modern applications.

This project, titled **“File Transfer System Using Elliptic Curve Cryptography (ECC)”**, addresses these challenges by offering a secure method for file transfers in networked environments. Elliptic Curve Cryptography (ECC), a modern form of asymmetric cryptography, is employed in this system due to its efficiency and reduced computational requirements. Unlike traditional cryptographic methods such as RSA (Rivest-Shamir-Adleman), which require large key sizes to ensure strong security, ECC achieves equivalent security with much smaller keys. This efficiency is especially beneficial for environments with limited computational resources, such as IoT devices, mobile applications, and embedded systems, where power and memory resources are at a premium.

**1.2 Objective**

The primary objective of this project is to develop a client-server model that guarantees data security during transfer, preserving the integrity and confidentiality of files in transit. In this model, files are encrypted on the client side and then transmitted securely to the server, where they are decrypted. By leveraging ECC for secure key exchange and AES (Advanced Encryption Standard) for the actual file encryption, the project adopts a hybrid approach that combines the strengths of both asymmetric and symmetric cryptography. This dual approach aims to provide an optimized solution for cure file transfer that is highly resistant to interception or unauthorized access.

**1.3 Scope**

By implementing ECC, the project achieves an **optimized encryption scheme** that is both robust and resource-efficient. ECC’s use of smaller key sizes accelerates computation, reducing both the time and power required for encryption and decryption. This advantage enhances system performance and extends the system’s application to low-power environments, such as mobile and real-time applications. Additionally, ECC’s resilience against cryptographic attacks ensures that the system meets or exceeds current industry security standards, providing a level of data protection that can withstand sophisticated threats.

In summary, the **File Transfer System Using Elliptic Curve Cryptography** project offers a secure, efficient, and scalable method for file transfer across networks. By addressing the need for both high security and low computational overhead, the project demonstrates the practical advantages of ECC in securing data communications, adding a valuable solution to the field of data security.

# Chapter 2

**Literature Review**

**2.1 Overview of Elliptic Curve Cryptography (ECC)**

Elliptic Curve Cryptography (ECC) has emerged as a vital cryptographic method, especially in the context of secure digital communication. Introduced in the mid-1980s, ECC offers a robust level of security using algebraic structures of elliptic curves over finite fields. This efficiency enables ECC to achieve comparable security to traditional encryption algorithms, such as RSA, with significantly smaller key sizes. Due to this advantage, ECC has quickly gained traction for applications requiring high security and low computational power, making it highly suitable for mobile and embedded systems where processing resources are limited.

The mathematical foundation of ECC lies in the difficulty of the Elliptic Curve Discrete Logarithm Problem (ECDLP), which is believed to be computationally infeasible to solve in a reasonable timeframe without the proper private key. This inherent complexity makes ECC an attractive option for cryptographic applications that need to prioritize both security and efficiency.

**2.2 Advantages of ECC over RSA and Other Cryptographic Methods**

ECC has several advantages over traditional cryptographic methods, particularly RSA. RSA has been widely used in public-key cryptography; however, it requires significantly larger key sizes to achieve a comparable level of security, resulting in slower computation and increased power consumption. For instance, a 256-bit key in ECC offers a similar level of security as a 3072-bit key in RSA. This difference means that ECC can perform encryption and decryption operations much faster, reducing the demand on system resources.

These benefits make ECC a preferred choice for cryptographic applications where resources such as processing power, memory, and battery life are constrained. Mobile devices, embedded systems, and Internet of Things (IoT) applications are particularly well-suited to ECC due to their reliance on lightweight security mechanisms. Studies comparing ECC and RSA have consistently shown that ECC's efficiency not only leads to faster computation times but also extends battery life and improves performance in low-power devices.

**2.3 Applications of ECC in Modern Cryptography**

ECC's efficiency and security have led to its adoption in various real-world applications. Notably, ECC is widely used in IoT, where devices often have limited computational capabilities. In IoT environments, ECC enables secure communication between devices without overwhelming their limited resources. Secure messaging platforms, which prioritize the privacy of communication and often operate in constrained environments (like mobile networks), have also benefited from ECC’s compact and efficient encryption.

In addition to IoT and secure messaging, ECC has become a staple in several internet security protocols. For example, the Transport Layer Security (TLS) protocol, which secures most online transactions and communications, now commonly incorporates ECC to ensure that data sent over the internet remains confidential and protected from interception. ECC's role in TLS and SSL (Secure Sockets Layer) further emphasizes its critical contribution to online security standards. The cryptocurrency space also relies heavily on ECC for transaction validation and wallet security, with Bitcoin and other major cryptocurrencies using ECC-based cryptographic techniques to ensure the security and integrity of transactions.

**2.4 ECC and Secure Key Exchange Protocols**

Another significant area where ECC excels is in secure key exchange protocols. Secure key exchange is a foundational aspect of encryption systems, as it allows two parties to establish a shared secret key without transmitting sensitive information over the network. One commonly used protocol, the Elliptic Curve Diffie-Hellman (ECDH), is a key exchange method based on ECC, providing a way to derive a shared secret key using each party’s ECC public and private keys.

ECDH has become widely adopted due to its speed and security. Compared to traditional Diffie-Hellman key exchange, which is based on modular exponentiation, ECDH requires fewer computational resources while still offering strong security guarantees. Research highlights ECDH’s effectiveness in both performance and security, making it ideal for systems that require real-time data transmission, such as secure file transfer systems and encrypted messaging services. By integrating ECDH into a file transfer system, this project leverages ECC's strength in secure key exchange to ensure that data remains confidential and protected against eavesdropping during transfer.

**2.5 Previous Work on ECC-Based File Transfer Systems**

Studies in recent years have explored various methods of implementing ECC for secure file transfer and data encryption, often concluding that ECC is highly suitable for such applications. Existing work on ECC-based encryption systems has focused on its application in client-server models, where data is encrypted on the client side, transferred over a secure channel, and decrypted on the server side. This setup has proven effective in maintaining the confidentiality of sensitive data throughout transmission, especially in scenarios involving remote or cloud-based storage.

Previous research also emphasizes ECC’s adaptability in hybrid encryption systems, where ECC is used for the initial key exchange, followed by symmetric encryption algorithms like AES for the main data encryption. This hybrid approach capitalizes on the speed of symmetric encryption while benefiting from the secure key exchange that ECC provides. This project’s design is informed by these studies, which demonstrate that using ECC in conjunction with secure file transfer protocols significantly improves data transmission security while maintaining efficient performance across networked environments.

# Chapter 3

**Methodology**

**3.1 Overview**

The methodology for this project involves the design and establishment of a secure client-server architecture that uses **Elliptic Curve Cryptography (ECC)** for file transmission. The aim is to create a file transfer system that ensures both confidentiality and integrity throughout the data transmission process, leveraging ECC’s key exchange capabilities in conjunction with **Advanced Encryption Standard (AES)** for encryption.

**3.2 Approach**

The approach centers on implementing a two-party **key exchange system** between the client and the server. Each participant (client and server) generates an ECC key pair consisting of a public and private key. The public keys are then exchanged, allowing both parties to independently derive a **shared secret key**. This shared key is subsequently used as an AES key to encrypt the file content on the client side, ensuring that the file remains unintelligible to anyone intercepting the transmission. AES is employed here due to its speed and reliability in encrypting large data sets, complementing ECC’s secure key exchange mechanism.

**3.3 Tools and Technologies**

The project is implemented using **Python**, with the support of the **Cryptography library** to handle both ECC and AES functionalities. The library provides secure and efficient cryptographic tools, allowing the project to implement ECC for key exchange and AES for symmetric encryption. Python was selected as the programming language for its readability, extensive libraries, and support for rapid prototyping, which streamlined the development and testing phases. The cryptography library specifically offers modules and algorithms optimized for high-security applications, ensuring the system adheres to modern cryptographic standards.

**3.4 Data Collection and Transmission**

The system treats files as raw data, preparing them for transmission by encrypting each file prior to sending. This setup focuses on securely encapsulating the file content, protecting it from unauthorized access if intercepted. During transmission, each file is converted into a ciphertext using AES encryption, with the shared ECC-

derived key as the AES key. This method ensures that the original data remains secure and unreadable without the decryption key, guaranteeing the integrity and confidentiality of the transferred files.

**3.5 Security and Error Handling**

To address potential transmission errors and maintain compatibility across different systems, security mechanisms and error-handling procedures are integrated within the methodology. **PEM formatting** is used for key serialization and deserialization during transmission, ensuring the ECC keys can be accurately transmitted and interpreted by both the client and server. Network disruptions are managed by implementing error-handling procedures, allowing the system to reattempt connections or alert the user if transmission fails. This attention to compatibility and reliability helps ensure that the system operates smoothly even in varied network environments.

# Chapter 4

**Implementation**

**4.1 Key Pair Generation**

The initial stage of implementation is **key pair generation**. Each participant (client and server) generates an ECC key pair, which consists of a public key and a private key. This is achieved using Python’s Cryptography library, which includes modules specifically designed for ECC. The generated keys are essential for establishing a shared secret key, which will be used later for AES encryption. Key pair generation is a critical step as it provides the foundation for a secure key exchange and, consequently, a secure file transfer.

**4.2 Public Key Exchange**

Once the ECC key pairs are generated, the client and server exchange their **public keys** to derive a shared secret. This exchange occurs over a secure channel, and both the client and server use their private keys in conjunction with the other party’s public key to generate a shared secret using the **Elliptic Curve Diffie-Hellman (ECDH)** protocol. The derived shared secret key is unique to the session and accessible only to the client and server, protecting it from external parties. This ECDH protocol provides robust security while ensuring efficient key exchange, creating a secure basis for the file transmission.

**4.3 AES Encryption of File Content**

The client uses the shared ECC-derived key to encrypt the file content with **AES encryption** before initiating the transfer. AES is a symmetric encryption algorithm chosen for its ability to encrypt large files quickly while maintaining security. AES’s block cipher approach divides data into smaller blocks, which are encrypted individually, making it suitable for efficient file encryption. The encrypted file content, or ciphertext, is then ready for transmission. By using AES, the system achieves the speed and reliability required for real-time file transfer, especially when handling large files.

**4.4 File Transmission and Decryption**

During this stage, the **encrypted file** is transmitted from the client to the server. Upon receiving the encrypted file, the server uses the shared ECC-derived key to

decrypt the ciphertext, restoring the file to its original, readable state. The server then saves the decrypted file, completing the secure transmission process. This method ensures that files transferred across the network are protected end-to-end, preserving the confidentiality and integrity of the data throughout the transmission.

**4.5 Addressing Challenges**

The implementation faced several challenges, including **network disruptions** and ensuring **key compatibility** across devices. To address these, the system was equipped with error-handling mechanisms to handle intermittent network connectivity and automatically reattempt failed transmissions. Additionally, **PEM formatting** for key serialization and deserialization facilitated compatibility across different platforms, making it easier for the client and server to exchange and process the ECC keys accurately.

## **Code Screenshots**

**A screen shot of a computer program

Description automatically generated**

A screenshot of a computer program

Description automatically generated

**A screenshot of a computer program

Description automatically generated**

A screen shot of a computer program

Description automatically generated

A screen shot of a computer screen

Description automatically generated

# Chapter 5

**Results and Discussion**

**5.1 System Testing in Varied Network Conditions**

The system was rigorously tested under various network conditions to evaluate its reliability, efficiency, and ability to maintain confidentiality. Simulated tests included high-latency and low-bandwidth scenarios to observe the system’s response and stability under constrained conditions. Results indicated that the system could securely and reliably transmit files even in less-than-ideal network environments, confirming its robustness.

**5.2 Achievement of Project Objectives**

The project successfully met its objectives. Each file decrypted on the server matched the original file on the client, validating the encryption and decryption processes. The system’s reliance on ECC proved to be advantageous, as it minimized computational overhead compared to RSA-based alternatives. This demonstrates ECC’s suitability for real-time applications where computational resources and speed are critical.

**5.3 Performance Analysis**

**Performance testing** showed that the system could handle file transfers with high efficiency, even on devices with limited resources. ECC’s smaller key size resulted in faster computation times, particularly during the key exchange phase. The combination of ECC for key exchange and AES for data encryption ensured a balance between security and performance, allowing the system to provide secure file transfer without significant delay or resource consumption.

**5.4 Discussion on ECC’s Applicability**

These results underscore ECC’s effectiveness for secure data transfer, particularly for applications that prioritize both security and performance. The project demonstrated ECC’s potential for real-world applications where resource constraints are a concern, supporting its integration into devices like smartphones, IoT devices, and embedded systems.

# Chapter 6 Conclusion

**6.1 Summary of Findings**

The **File Transfer System Using Elliptic Curve Cryptography (ECC)** project demonstrated that ECC can provide a secure and efficient means of data encryption for file transfers. The project’s use of ECC for secure key exchange, combined with AES for fast file encryption, achieved the dual goals of confidentiality and efficiency. Through the encryption process, ECC ensured data integrity and confidentiality, successfully addressing common security concerns in file transfer.

**6.2 Implications for Future Applications**

This project’s successful implementation illustrates that ECC, due to its lightweight nature and strong security, is ideal for applications that demand both high security and low computational overhead. The system’s reliance on ECC allows it to serve a range of environments, from mobile devices and IoT applications to secure messaging platforms, where traditional cryptographic methods may be too resource-intensive.

**6.3 Potential for Future Enhancements**

Future enhancements to this system may include the integration of additional security protocols, such as multi-factor authentication or end-to-end encryption. Moreover, adapting the system for deployment in **cloud environments** or further customizing it for **IoT devices** could expand its applicability. These improvements would extend the system’s use cases, enhancing its ability to provide secure file transfer across an even wider array of networked environments.

# References

[1] William Stallings, 'Cryptography and Network Security,' Pearson.  
[2] Neal Koblitz, 'Elliptic Curve Cryptosystems.'

# Github Link

<https://github.com/Nilayyyy/Ecc_file_transfer>