Fortified File Exchange: A Secure Data Transfer Technique

A major project report submitted in partial fulfilment of the requirement for the award of degree of

Bachelor of Technology

in

Computer Science & Engineering / Information Technology

Submitted by

Basu Narayan (201352) & **Manan Gupta (201287)**

Under the guidance & supervision of

Dr. Shubham Goel

to

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology, Waknaghat, Solan - 173234 (India)

Certificate

This is to certify that the work which is being presented in the project report titled "**Fortified File Exchange: A Secure Data Transfer Technique**" in partial fulfilment of the requirements for the award of the degree of B.Tech in Computer Science and Engineering and submitted to the Department of Computer Science And Engineering, Jaypee University of Information Technology, Waknaghat is an authentic record of work carried out by **Basu Narayan** (201352) and **Manan Gupta** (201287) during the period from August 2023 to May 2024 under the supervision of **Dr. Shubham Goel**, Assistant Professor (SG) Department of Computer Science and Engineering, Jaypee University of Information Technology, Waknaghat.

Basu Narayan
201352

Manan Gupta
201287

The above statement made is correct to the best of my knowledge.

Dr. Shubham Goel
Assistant Professor (SG)
Department of Computer Science & Engineering
Jaypee University of Information Technology, Waknaghat

Declaration

We hereby declare that the work presented in this report entitled "**Fortified File Exchange: A Secure Data Transfer Technique"** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science & Engineering / Information Technology** submitted in the Department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology, Waknaghat is an authentic record of our own work carried out over a period from August 2023 to May 2024 under the supervision of **Dr. Shubham Goel** (Assistant Professor (SG), Department of Computer Science & Engineering and Information Technology).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Basu Narayan
201352

Manan Gupta
201287

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Dr. Shubham Goel
Assistant Professor (SG)
Department of Computer Science & Engineering
Jaypee University of Information Technology, Waknaghat

Acknowledgement

Firstly, we express our heartiest thanks and gratefulness to almighty God for His divine blessing makes us possible to complete the project work successfully.

We are really grateful and wish our profound indebtedness to **Dr. Shubham Goel, Assistant Professor (SG)**, Department of CSE Jaypee University of Information Technology, Waknaghat. Deep Knowledge & keen interest of our supervisor in the field of "**Cyber Security**" has helped us to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism and valuable advice along with reading many inferior drafts and correcting them at all stage have made it possible for us to complete this project.

We would also generously welcome each one of those individuals who have helped us straight forwardly or in a roundabout way in making this project a win. In this unique situation, we might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated our undertaking.

Finally, we must acknowledge with due respect the constant support and patience of our parents.

Basu Narayan
201352

Manan Gupta
201287

Table of Contents

List of Tables

List of Figures

List of Abbreviations and Symbols

Abbreviations and Symbols	Meaning
FFE	Fortified File Exchange
UI/UX	User Interface/ User Experience
AES	Advanced Encryption Standard
DES	Data Encryption Standard
FTP	File Transfer Protocol
SFTP	Simple File Transfer Protocol
FTPS	File Transfer Protocol Secure
2FA	Two-Factor Authentication
OTP	One-Time Password
RAM	Random Access Memory
ROM	Read-only Memory
SSD	Solid State Drive
Gbps	Gigabits per second
ELK	Elastic-Logstash-Kibana
RSA	Rivest-Shamir-Adleman
MRS	MapReduce Service
HDFS	Hadoop Distributed File System
OMS	Order Management System
ER	Entity-Relationship
TCP	Transmission Control Protocol
P2P	Peer-to-Peer
ECC	Elliptic Curve Cryptography
DFD	Data Flow Diagram
RC6	Rivest Cipher 6

Abstract

Enhancing Security and Efficiency in File Exchange with a Fortified Solution

In today's generation the exchange of sensitive and confidential documents among users or people, businesses and industries are a yet increasingly challenging task.

Today's File Exchange Systems often lack the security measures that are important to safeguard the data from harmful threats such as data manipulation. On the other hand, it does have performance and susceptibility issues. So, after observing all these cases, we led to a pressing need of a Fortified File Exchange System that enables the user to perform the data transmission with ease and security that would decrease the chances of critical security issues or any malicious activities such as data leakage or malware infiltration. When we looked into the conventional data transmission methods such as sending files as e-mail attachments or sharing them over insecure cloud-based systems or storage devices, the sender and the receiver face problems such as data leakage and slow transfer speeds and unavailability of real-time location of the data packets over the internet. Our project's emphasis is on the security of user data and user- credentials while he/she is using our Fortified File Exchange system with having a good efficiency and a better User-Experience (UX) with enhanced data transmission speeds and secure data transfer protocols. Using data encryption standards such as AES, DES, Triple-Des, RC6 and ECC with a combination of Secure File Transfer Protocols (SFTPs) would help us in solving the ever-existing problems in the digital world. Since, Data is the new gold, our project Fortified File Exchange (FFE) aims to overcome the drawbacks that exist in the today's file exchange systems. Starting from the roots of an organization's digital infrastructure to the pinnacle of its cyberthreats we aim to develop UI as per the user requirement.

Chapter 1
Introduction

1.1 Introduction

In today's interconnected world, the emphasis on data security is very important. Organizations of all scales try their best to keep their sensitive information away from the reach of hackers and exploiters in a safe and secure environment. When data security is considered, data sharing is also considered as an eternal part of the whole process and for which individuals and organizations have been using Traditional data transfer methods, such as email, which usually fail to safeguard sensitive data. This is where our idea of Fortified File Exchange will emerge as a much secure and reliable option and solution for data sharing over networks.

Fortified File Exchange, which we abbreviated as FFE is a data transfer technique which promises to provide multiple layers of security to ensure safe and secure data transmission. We tried to use and embed various security protocols and encryption standards to protect data being transmitted, throughout its journey from the sender to receiver. We tend to provide a comprehensive and secure approach for file sharing, so that the individuals and organizations share their sensitive information freely, without worrying about its leakage. By using our technique, both the individuals and the organizations will be able to safeguard their valuable data because of the encryption standards, authentication, data integration and logging mechanisms our software uses.

1.2 Problem Statement

Irrespective of the size of an organization, Secure and efficient file exchange has been an area of constant concern. Whenever individuals share their personal or organizational data through insecure channels such as emails or third-party file sharing services, they are unaware of how vulnerable the data is to unauthorized access. If a hacker is able to intercept a file before it reaches the receiver, and makes modifications or destruction to some of the most important fragments of the data, it will no longer be authentic and the receiver wouldn't even notice.

In evolving cyber threats, such as ransomware and phishing attacks, attackers use the same approach and intercept the passage between the sender and the receiver to send modified data.

Apart from the security concerns, individuals and organizations also look into the UI/UX of the application or platform they are using for file sharing, where our platform will try to inherit according to their systems UI/UX and develop a world-class experience for the company's employees.

Considering the requirements of the present-day cyber world, our system tends to use strong encryption standards such as AES, DES and Triple-DES(3DES) or ECC and RC6 as per requirements, implement end-to-end encryption, which will ensure the data is encrypted when it leaves the sender's device and is decrypted only when it reaches the recipient's device. For all transfers, we shall use FTP protocols such as SFTP and FTPS that allow a secure transfer of files over a network. In the near future, we plan to implement two-factor authentication(2FA) which will provide an extra layer of security as it will require the users to enter an OTP sent to their mobile numbers in addition to entering the password while logging in.

1.3 Objectives

In order to get a better understanding of the FFE, we have divided our objectives into a hierarchy of 5 and those are listed as follows:

1) Evaluating existing safe and secure file exchange technologies to look for vulnerabilities in them and areas where there is a scope of improvement.
2) When the evaluation is done, we will work on the development and implementation of algorithms and mechanisms which will help us in identifying vulnerabilities in our system and by improving them we would be able to enhance the file exchange security.
3) Afterwards, we will develop a system to verify the authenticity and safety of both the users and the files.
4) Integrating both the algorithms and mechanisms to the file transfer system.
5) Generating our own performance and security report.

1.4 Significance and Motivation of the Project Work

Being students and learners of Information Technology there has always been curiosity amongst us to develop a system that protects our personal sensitive data and whenever we used to transfer something from one of our devices to another using wired technology, it would disconnect often because of lose connections or some wear and tear, use of a technology that is wireless and doesn't depend on factors like these was thought to be more beneficial and user-friendly and that is when the idea of creating such file transfer technique developed in us, which we call our very own Fortified File Exchange(FFE).

Apart from our own problems, we were sure that not only the users with same background as that of ours but also many others are in an ocean of such problems. Therefore, it is an effort not only for ourselves but for more like us.

1.5 Organization of Project Report

Our project report is divided into 6 Chapters and brief on each one of them is duly mentioned below:

Chapter 1: Introduction

The introduction covers all sections like brief of our project, our problem statement, our objectives, significance and motivation of the project report and this section in particular where we showcase the hierarchy of our chapters.

Chapter 2: Literature Survey

The Literature Survey includes the existing information available in standard books, journals, popular websites, top organizations' technical/white papers with a clear emphasis on work done in recent times, i.e., five years.

Chapter 3: System Development

This chapter includes sections which highlight the requirements and analysis, our project design and architecture, data preparation, implementation (including code snippets, algorithms used, tools and techniques, etc.). Moreover, here we also highlight the challenges faced during the development process and the way they need to be addressed.

Chapter 4: Testing

In this chapter, we discuss about our Testing Strategy and the Test cases and Outcomes of the process as a whole.

Chapter 5: Results and Evaluation

This section of our report includes presentation of our findings and interpretation of the results, etc. We also try to compare our system with the existing systems.

Chapter 6: Conclusions and Future Scope

In this section, we summarize key findings, limitations and our contribution to this ever-growing field.

Chapter 2

Literature Survey

2.1 Overview of Relevant Literature

2.1.1 A Summary of Relevant papers

R.Mathwale and R.Ramisetty[1],The author purposed a blockchain inter-organizational secure file transfer system that handle smart covenant that guarantees data integrity and privacy. It is robust and secure method for transferring confidential or sensitive files or data between two systems while maintaining record of the data. In this, we studied how blockchain maintains a tamper-proof of file transfers and also get to know about File Transfer Nodes, i.e., which helps the system to initiate and receiving the file transfers by having those authorized entities. It works on a decentralized system cause of which risk of failure is negligible and also prevents unauthorized interference of file transfer system and by having a access control mechanism, which prosecute access control over files which leads to avoid unauthorized manipulation. It had some disadvantages such as scalability and complexity.

K. S Gupta, M.jadon[2], In this authors had implemented a peer to peer secure data communication, which have hybrid cryptographic approach combining Twine-based Advanced Encryption Standard (T-AES) and Elliptic Curve Cryptography(ECC). T-AES is used for symmetric encryption which provides data security and ECC is used for public key encryption which offers efficient key management and authentication capabilities. This avoids the problem such as interference attacks and message manipulation and key reusability. But this also comes up with some limitations such as key management overhead, computational complexity and limited scalability due to which performance decreases because this imposes overhead on resource constrained systems especially in case of large data transfers.

C.Susmitha, K.S.Laasya, S.K.Kannaiah and S.Bulla[3],The authors performed a hybrid cryptography system which consist of Advanced Encryption Standard (AES) and (RSA) the Rivest-Shamir-Adleman algorithms which is mixture of both symmetric and asymmetric encryption algorithms that increases security and performance. The hierarchical key

management enables efficient key issuing and repeal within the file storage system. AES provides strong encryption and RSA conducts key management and validation of file ownership. Due to simple design, there are some many disadvantages such as complexity occurring while key management and susceptibility to known attacks and storage requirements.

S. Nimgade and V. Ghode[4],The main focus of the author was to create a system that have increased privacy, security and scalability i.e. Decentralized File Sharing(DFS) have a central server system which enables the users to have more control over their data and is more confidential. Due to its architecture, it is less vulnerable to attacks and it can handle a large number of the users and files because there is no limit to resources. The disadvantages of using a DFS system as the users has to directly connect with each other so performance of the system decreases, users can also use this system to send illegal or harmful content and all this leads to high complexity.

V.S. Mhatre and S.N.S atel[5]- The objective of this research was to have convenient and affordable place to store data i.e. Cloud server but authors were also concerned about the security of the system while storing sensitive data in the cloud. So, they thought of using RSA but due to its vulnerabilities and cost, a new algorithm was introduced Elliptic curve cryptography, this a public key encryption algorithm that works on properties of elliptic curves. It is more secure and affordable than RSA because of having smaller key size and lesser steps which performs data sharing really quick with cloud server. Generally, this method secures data from many attacks but lacks in detecting quantum computing attacks.

R. Ghosh and Z. Khan [6], Authors performed a DFS which utilizes blockchain technology instead of cloud sever storing system which may makes the system more complex but increases the chances of the data to be more secured and confidential by major attacks that could lead to data manipulation. As per the blockchain technology which causes a increment in complexity and scalability issues, also is not immune to misuse.

S.S Ranadive and H. Sawant [8], The authors researched and came up with a new method of securing files on cloud computing servers which is a combination of two cryptography

algorithms i.e., AES and ECC, AES was imposed to secure the data by encrypting the files and ECC is used to generate and manage the keys to enhance security. This method or system was mostly emphasizing the security of the files in cloud storage to avoid any case of data breaching or unauthorized access or interference, it also empowers users to maintain control over the data. The limitations to this system that due large no. of features the overhead is generated and also creates challenge to manage the keys.

M. Schnieder [12], In this author purposed a secure automated file exchange system that consists of a mixture of cryptographic algorithms to guarantee the integrity and confidentiality of files. The automated nature allows users to exchange the files without any intervention with the help of the asymmetric public key along with symmetric encryption for file protection. This method rule outs the issue of scalability and increases the performance obviously and cause of having a automated system it eliminate the need of manual work such as file handling and labor cost that comes with manual file transfers. This increase in the complexity of the system and sensitivity.

J. Shimati,G. Honjo[13], The author purposed a secure file transfer on 100GB on a single server due to which there is a decreased complexity and ready for any kind of attacks cause it has a enhanced security system which is decentralized . The concept the author used was successful and had a high performance by allowing a single server to handle large volume of thefiles without using any other hardware or software.

S. Haung and K. Zhang [15], The author came up with a new tool which is a Bluetooth stack that provides various beneficial features, especially with FTP support i.e., Blue Solei. The authors implemented this tool with FTP to create a system or server that supports transferring large files via Bluetooth and is capable of running over any system without any issue as it's the most convenient and easy to implement. But due to the use of the Bluetooth tool there is a drawback of limited range, susceptibility and power consumption. Security point of view it is well secured but major attacks can lead to a data manipulation and in case of large no. of connections the complexity of the system increases which effects the performance.

A. Srivastav and A. Bhartee[17],the author purposed a SFT system that comprises of the a cryptographic algorithms and secure communication protocols to safeguard the integrity and authenticity during the transmission as it has AES to encrypt files before transmission and create a digital signature and uses SSL/TLS protocols are used to create a secure communication. Drawbacks it brings is computational overhead and implementation errors and using only one encryption algorithm it is less secure to harmful attacks.

2.1.2 Comparison of Fortified File Exchange Techniques

S. No.	Paper Title [Cite]	Journal/ Conference (Year)	Tools/ Techniques/ Dataset	Results	Limitations
1.	"Blockchain Based Inter-Organizational Secure File Sharing System"	INOCON (2023)	Blockchain-based File sharing, Smart contract-based access control & Synthetic Datasets.	Files shared without central authority & resistant to tampering.	Slow to share large files & Expensive to operate.
2.	"T-AES- and ECC-Based Secure Data Communication in P2P Networks"	WIDECOM (2023)	T-AES Encryption, ECC Key Exchange, ECC Digital Signatures & Synthetic Datasets.	Secure communication in P2P networks & resistant to eavesdropping attack.	Computationally Expensive for Large Files.
3.	"Hybrid Cryptography for Secure File Storage"	ICCMC (2023)	Hybrid encryption(AES & DES) & Synthetic Datasets.	Secure File Storage & resistant to MITM attacks.	Hybrid cryptography is complex & costly.

Figure 1: Comparison between techniques unable to perform large file transfer

S. No.	Paper Title [Cite]	Journal/ Conference (Year)	Tools/ Techniques/ Dataset	Results	Limitations
4.	"Decentralized File Sharing"	SCEECS (2023)	Peer-to-Peer(P2P) network for file sharing.	Securely share of files between peers on a P2P network & it is resistant to tampering.	Slow to share large files & P2P file sharing systems are vulnerable to Sybil attacks.
5.	"Secure File Storage On Cloud Using Elliptic Curve Cryptography (ECC) Algorithm"	IRJMETS (2023)	ECC-based file encryption, AES-based file authentication & RSA-based key management.	Authors believe the system to provide efficient file authentication and key management.	Expensive for storage of large files & ECC implementations are vulnerable to side-channel attacks.
6.	"Decentralized file storing and sharing system"	IJRMETS (2023)	P2P Network & Blockchain	Secure and decentralized file storing & sharing.	Vulnerable to Sybil attacks.

Figure 2: Comparison between techniques vulnerable to attacks

S. No.	Paper Title [Cite]	Journal/ Conference (Year)	Tools/ Techniques/ Dataset	Results	Limitations
7.	"A Review Of Block-chain Based Secure Sharing Of Healthcare Data"	MDPI (2022)	- Electronic Health Records(EHRs) - Medical Images (such as X-rays, MRI Scans) -Genomic Data	Block-Chain based healthcare data can improve Security, privacy & efficiency.	Lack of Standardization & Need for Robust Security & User – Friendliness.
8.	"Secure File Storage On Cloud Using Cryptographic Algorithm"	IJRASET (2022)	- A set of 1000 text files of various sizes of 1 KB to 1 MB. - A set of 1000 image files of various sizes from 1 KB to 1 MB.	Secure transfer of data was possible because of the use of RSA & AES.	Vulnerability to side-channel attacks & Limited Flexibility.
9.	"Secure Automated File Exchange (Safe) – Enabling More Efficient Transfers Of Sensitive Data"	IJPDS (2020)	Data encryption, Data authentication & Data auditing.	SAFE successfully transfers sensitive data b/w different organizations and helps in reduction of time and effort required to transfer sensitive data.	Not designed to transfer large datasets & does not protect against all types of data breaches, such as insider attacks.

Figure 3: Comparison between techniques that are highly insecure with regards to current landscape

S. No.	Paper Title [Cite]	Journal/ Conference (Year)	Tools/ Techniques/ Dataset	Results	Limitations
10.	"Secure file transfer on 100 Gbps network using single server"	IEICE Technical Report (2020)	-Linux server with SSD and 100G NIC -A set of large files (100 MB, 500 MB, and 1 GB)	The authors achieved a basic performance of encrypted data transfer over 80 Gbps, which exceeds the performance of existing methods.	Need for additional hardware resources & the potential for performance degradation over long distances.
11.	"A Method of Bluetooth Large File Transfer Based on BlueSoleil+ FTP"	JCSA (2019)	-A BlueSoleil SDK -A set of large files (100 MB, 500 MB, and 1 GB)	The method achieved a transmission speed of 141.51 kb/s, which was 2X the methods looked upon.	Requires both devices to be connected on a same Bluetooth network.
12.	"Design Of Secure File Transfer Over Internet"	IJARCCE (2016)	DES, AES, RSA & Triple DES.	AES is the most efficient, less complex amongst all the ones used.	Limited Flexibility

Figure 4: Comparison between techniques that are limited for personal usage only

2.2 Key Gaps in the Literature

After carefully going through the hard work of the authors, we identified the following gaps in the literature:

- Traditional file transfer methods like FTP are simple and convenient to use but as a reason of which they lack security and performance wise.
- FTP uses old-age encryption methods that are volatile and heavily depend upon TCP connections, and thus they tend to have slow transfer speeds for large files.
- P2P based systems support fast file transfer but the aspect of security is compromised. Such systems are easily violated by Sybil attacks, where multiple identities are used to manipulate network and spread malware.
- Techniques like Hybrid Cryptography are secure but costly.
- ECC is efficient but is exploited by side-channel attacks, compromising sensitive information present on the system.
- Cloud Services are very convenient to use but it is costly to store large files.

Chapter 3

System Development

3.1 Requirements and Analysis

In order to design a secure file transfer technique for today's data-driven world, we need to take into consideration various requirements for the same. Below are some key requirements and analysis that we need to consider:

3.1.1 Functional Requirements

Our model FFE requires a range of functional requirements to securely transfer files:

1. **Secure Encryption Algorithms**: The users should be able to send files without any fear of losing their data to hackers because it remains unreadable to unauthorized personnel because of encryption algorithms used.
2. **End-to-End Encryption**: FFE tries to bring in end-to-end encryption to the picture, which portrays the importance of data encryption when the sender sends the file and data decryption when the receiver receives the file.
3. **SFTP**: The users would love to have a file transfer protocol governing their data transmission and SFTP or FTPS serve the purpose.
4. **User Authentication**: In the near future, we are planning to inherit techniques like 2FA.
5. **Access Control**: If the model is a boon in its initial phase, we would tend to introduce techniques to restrict access to file transfers based on user permissions etc., which will only allow a certain group of authorized users to access sensitive data.
6. **Audit Logging**: Logging our events in a app-based UI is not possible as of now and any movement of data from one user to another is governed only by the packets visible in tools like Wireshark. Introducing our very own Logging system would provide us the option to take accountability and tracing of security incidents, if any occur.

Functional Requirement	Description
Secure Encryption Algorithms	These encryption standards enable sending files without the fear of losing them and keeping them unreadable to unauthorized users
End-to-End Encryption	Through FFE we try to embed end-to-end encryption into file transfers
SFTP	When the user wants to share a file, there should be a file transfer technique/protocol governing the data transfer.
User Authentication	In the near future, we will try to inherit techniques like 2FA
Access Control	If we are able to develop a model that boons in the initial phase, we would try to embed enhanced security features like restricted access.
Audit Logging	Just like normal windows logs, we plan to log our events which is not currently possible in an app-based UI.

Table 1: Functional Requirements

3.1.2 Non-Functional Requirements

In addition to the above-mentioned functional requirements, FFE also relies on some non-functional requirements for smooth and proper functioning:

1. **Performance**: Every user requires high-speed and secure file transfer and our software aims to minimize time taken for transfer and ensuring timely availability of data from sources
2. **Scalability**: With gradually increasing scope in the digital landscape, FFE needs to be scaled significantly to meet the expectations of users for high volume of file transfers.
3. **Usability**: Creating a User-friendly Interface will always be our topmost priority, so that it is easy to use for non-technical users.
4. **Security**: To meet all available and developing security standards, providing data protection to users in contrast to the latter is very important.
5. **Reliability**: Ensuring minimum crashes and uninterrupted data transfer.
6. **Maintainability**: Timely updates and security maintenance is very important.
7. **Compatibility**: Windows is the only OS, in which FFE is proficient. If everything goes as planned, we will try to acquire the user database from other operating systems as well.

Non-Functional Requirement	Description
Performance	User preference of high-speed transfer is something that our software aims to cater in the future
Scalability	To meet user expectations for high volume transfers.
Usability	User friendly UI, that is easier to use for non-technical users
Security	Data protection to both current and evolving security threats
Reliability	Minimal crashes and uninterrupted data transfers
Maintainability	If we keep working and integrate AI to our software, timely updates will become a necessity.
Compatibility	Making FFE compatible to other operating systems also (Currently compatible with Windows only)

Table 2: Non-Functional Requirements

3.1.3 Hardware Requirements

The hardware requirements are as follows:

1. **Processing Power**: FFE requires sufficient processing power to handle both encryption as well as decryption. A modern server with multi-core processors would serve the purpose.
2. **Memory**: FFE requires sufficient RAM and ROM to smoothly govern the whole data transfer process.
3. **Storage**: High-Performance storage systems such as SSDs are recommended.
4. **Network Connectivity**: A dedicated network connection with at least 1 Gbps bandwidth is advisable.

3.1.4 Software Requirements

1. **Development Tools**
    - Code Editor: Visual Studio Code
    - ELK Stack (Preferably Elastic Security)
    - Wireshark
2. **Programming Languages**
    - Python – Backend Development, JavaScript – Frontend Development
3. **Encryption Algorithms**
    - AES
    - DES
    - Triple-DES
    - RSA

3.2 Project Design and Architecture

The project design and architecture of FFE are structural representations of our project plan and learning approach. In this section we will talk about the Structure of FTP Server and the process of FTP-Server data access process.

Figure 5: Project Design

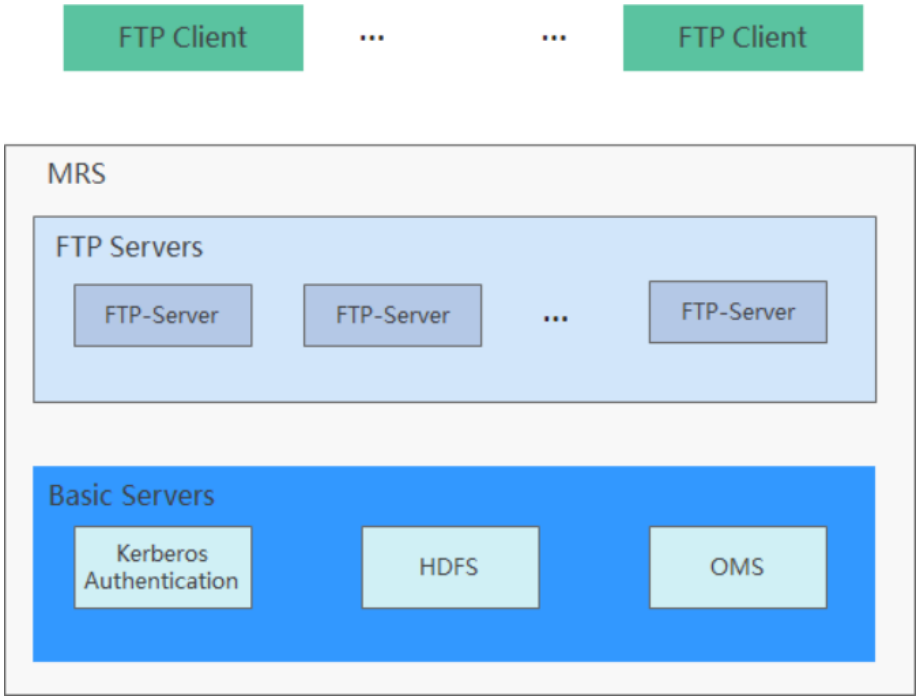


Figure 6: Structure of FTP Server

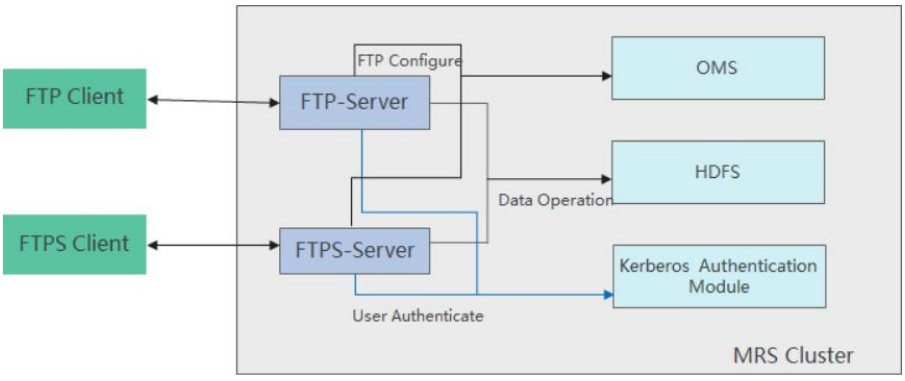
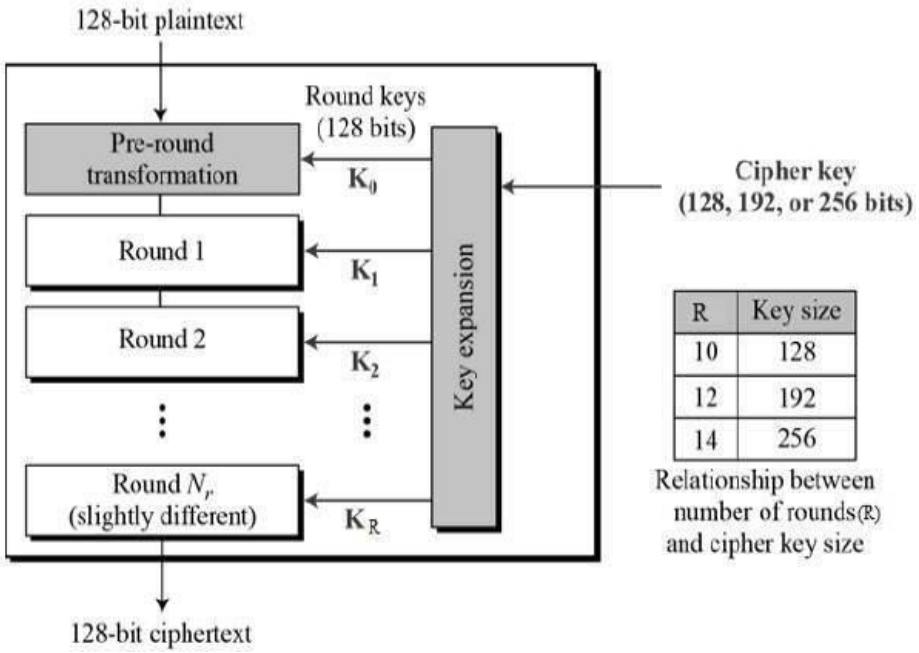


Figure 7: FTP-Server data access process



R	Key size
10	128
12	192
14	256

Relationship between number of rounds(R) and cipher key size

Figure 8: Architecture of AES Algorithm

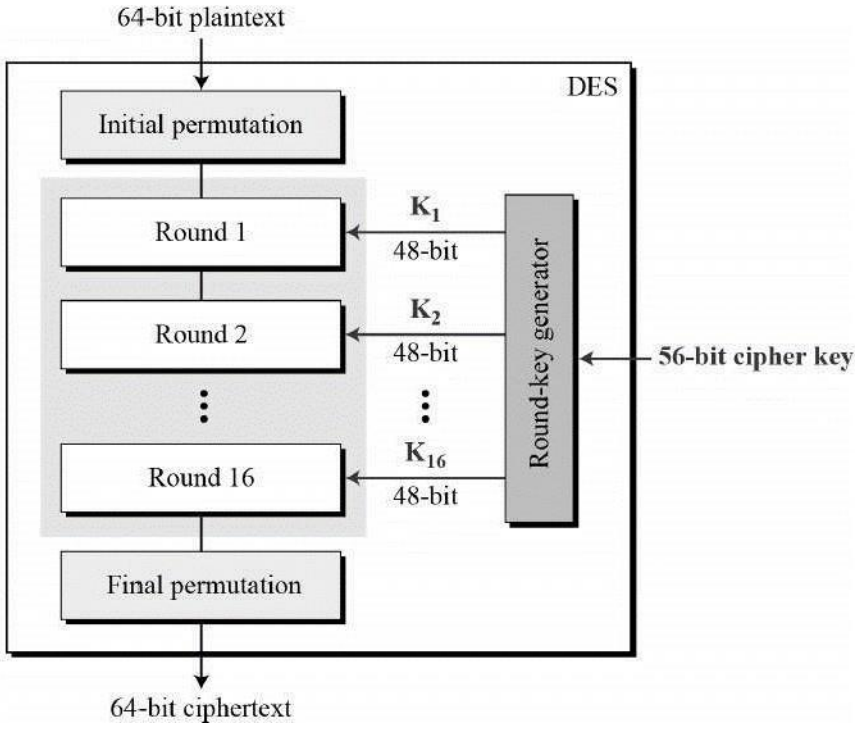


Figure 9: Architecture of DES Algorithm

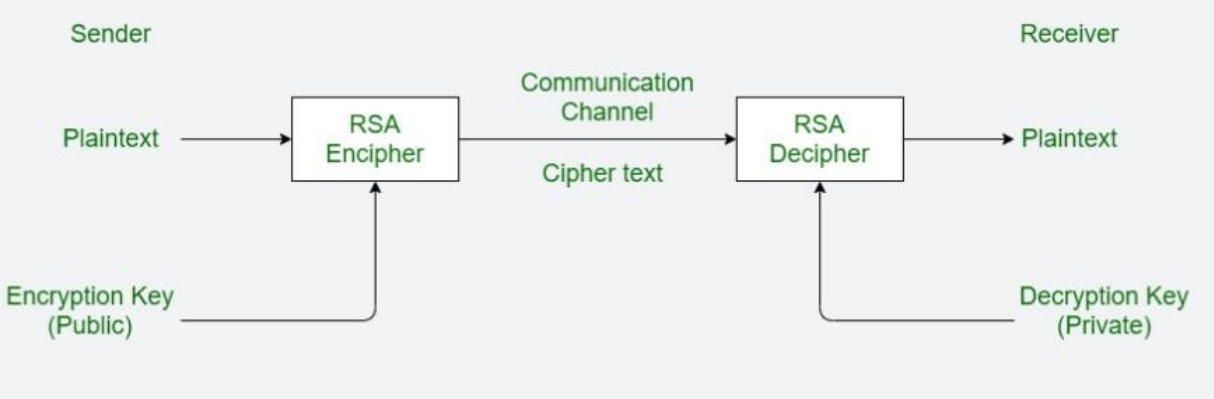


Figure 10: Architectural form of RSA

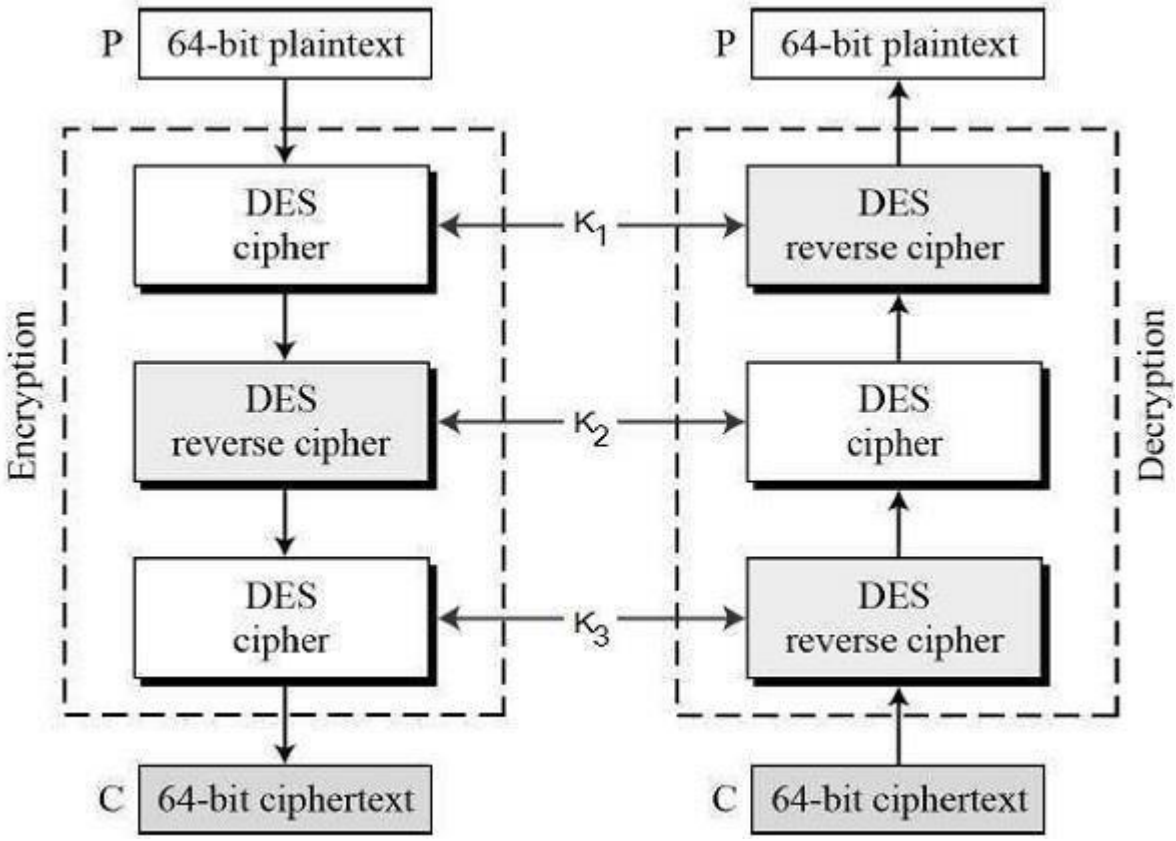


Figure 11: Architecture of Triple-DES

3.3 Data Preparation

Data preparation plays a vital role in ensuring the success and limitations of a project. This part of the report works on structuring raw form of the data into a well-structured format, thus making it suitable for processing and analysis.

We developed the Entity-Relationship diagram to manage the flow of our data and with regards to the ER diagram, a Data Flow Diagram (DFD) helps in further understanding of the FFE.

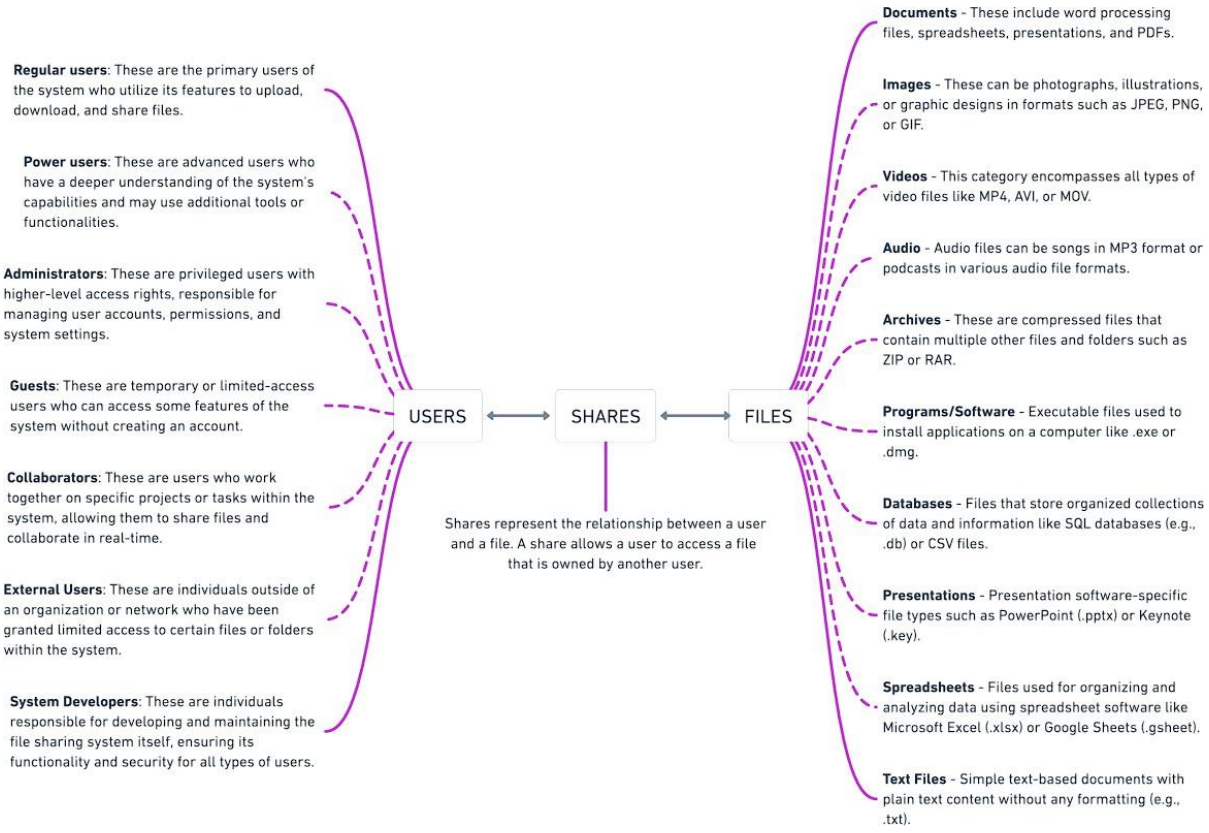


Figure 12: ER Diagram

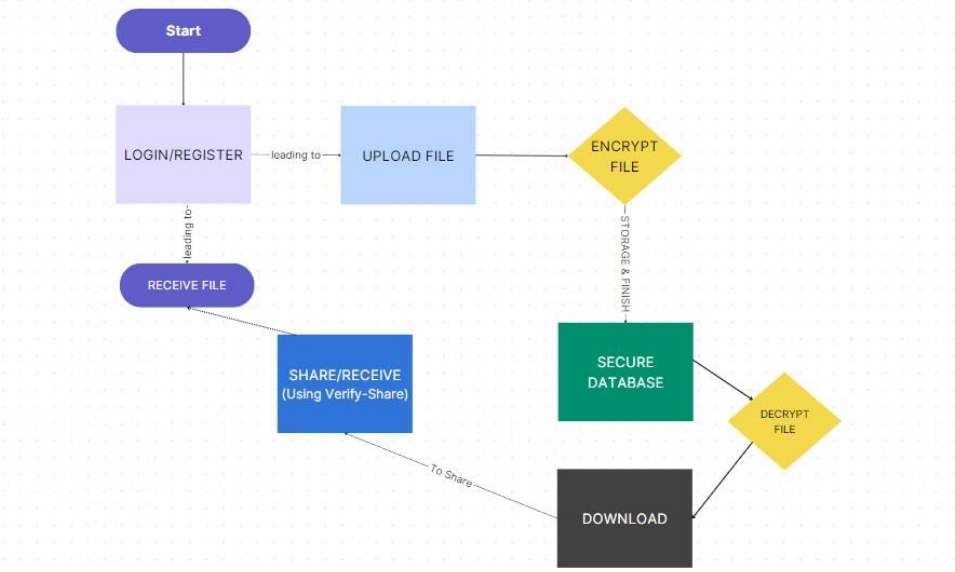


Figure 13: Data- Flow Diagram

3.4 Implementation

Here we have attached code snippets of the variety of algorithms we plan on using.

```
def encrypt(message, key):
  """Encrypts a message using DES."""
  cipher = DES.new(key, DES.MODE_ECB)
  encrypted_message = cipher.encrypt(message)
  return base64.b64encode(encrypted_message)

def decrypt(encrypted_message, key):
  """Decrypts a message using DES."""
  cipher = DES.new(key, DES.MODE_ECB)
  decrypted_message = cipher.decrypt(base64.b64decode(encrypted_message))
  return decrypted_message
# Example usage:
key = "This is a secret key."
message = "This is a secret message."
encrypted_message = encrypt(message, key)
decrypted_message = decrypt(encrypted_message, key)
print(decrypted_message)
```

Figure 14: DES

```python
def encrypt(message, key):
  """Encrypts a message using AES."""
  cipher = AES.new(key, AES.MODE_ECB)
  encrypted_message = cipher.encrypt(message)
  return base64.b64encode(encrypted_message)

def decrypt(encrypted_message, key):
  """Decrypts a message using AES."""
  cipher = AES.new(key, AES.MODE_ECB)
  decrypted_message = cipher.decrypt(base64.b64decode(encrypted_message))
  return decrypted_message

# Example usage:
key = "This is a secret key."
message = "This is a secret message."
encrypted_message = encrypt(message, key)
decrypted_message = decrypt(encrypted_message, key)
print(decrypted_message)
```

Figure 15: AES

```python
def generate_keypair():
  """Generates an RSA keypair."""
  p = random.prime(2048)
  q = random.prime(2048)
  n = p * q
  phi = (p - 1) * (q - 1)
  e = random.randint(2, phi - 1)
  d = pow(e, -1, phi)
  return n, e, d

def encrypt(message, n, e):
  """Encrypts a message using RSA."""
  ciphertext = pow(message, e, n)
  return ciphertext

def decrypt(ciphertext, n, d):
  """Decrypts a message using RSA."""
  plaintext = pow(ciphertext, d, n)
  return plaintext

# Example usage:
n, e, d = generate_keypair()
message = "This is a secret message."
ciphertext = encrypt(message, n, e)
plaintext = decrypt(ciphertext, n, d)
print(plaintext)
```

Figure 16: RSA

```python
from Crypto.Cipher import DES3

# Define the keys
key1 = b"0123456789012345"
key2 = b"6789012345678901"
key3 = b"2345678901234567"

# Create a DES3 cipher object using the three keys
cipher = DES3.new((key1, key2, key3), DES3.MODE_ECB)

# Define the plaintext message
plaintext = b"This is the plaintext message."

# Encrypt the plaintext message
ciphertext = cipher.encrypt(plaintext)

# Decrypt the ciphertext
decryptedtext = cipher.decrypt(ciphertext)

# Print the plaintext, ciphertext, and decryptedtext
print("Plaintext:", plaintext)
print("Ciphertext:", ciphertext)
print("Decryptedtext:", decryptedtext)
```

Figure 17: Triple-DES

Figure 18: Working Hierarchy of SFTP

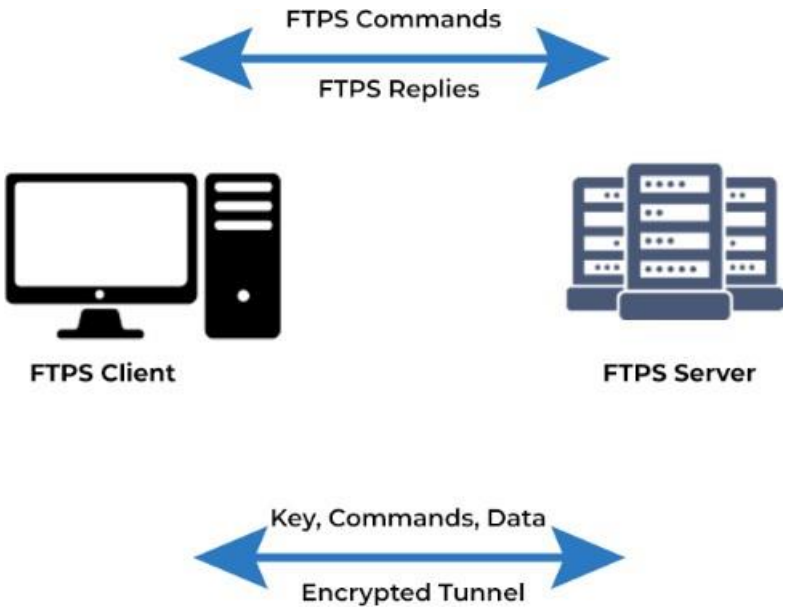


Figure 19: Working Architecture Design of FTPS

After the snapshots of the algorithm and architecture being used in our systems, here are the snapshots of the UI that is created by us for the Fortified File Exchange using python's tkinter.

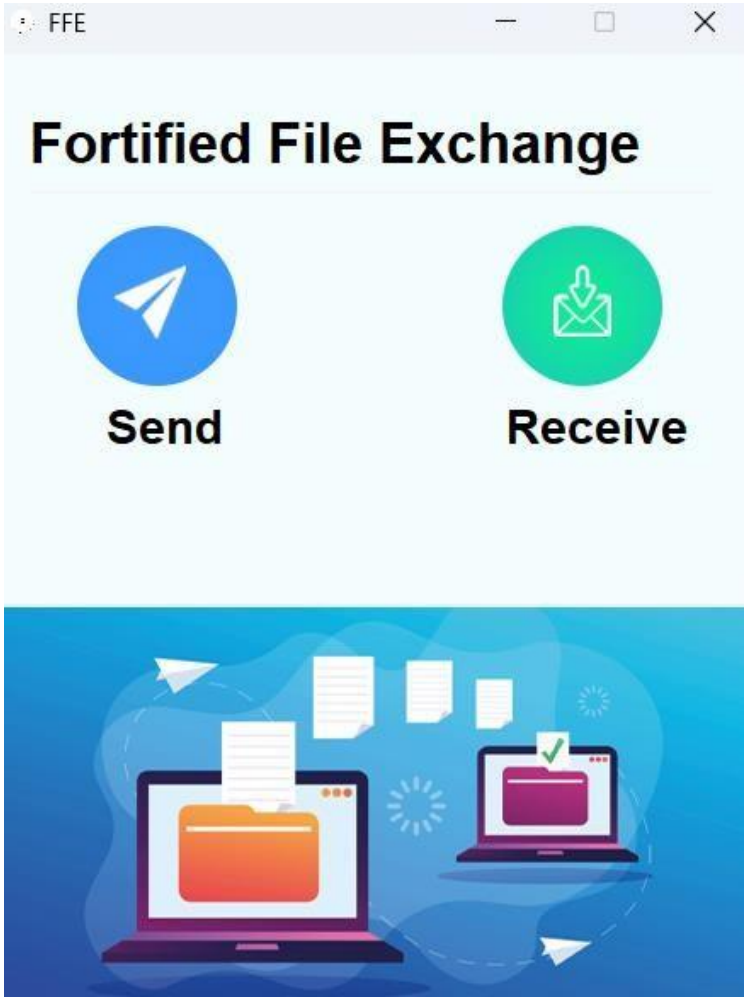


Figure 20: Send and Receive Page (Home Page)

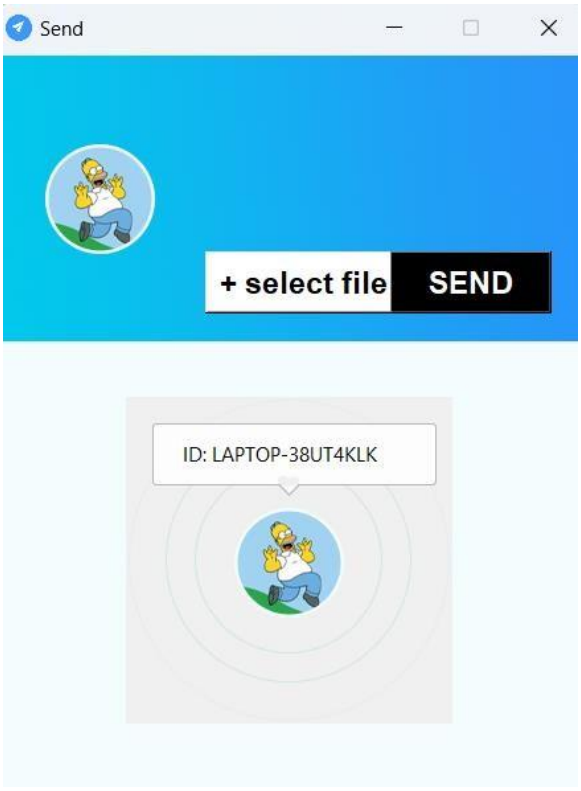


Figure 21: Once the send button is clicked

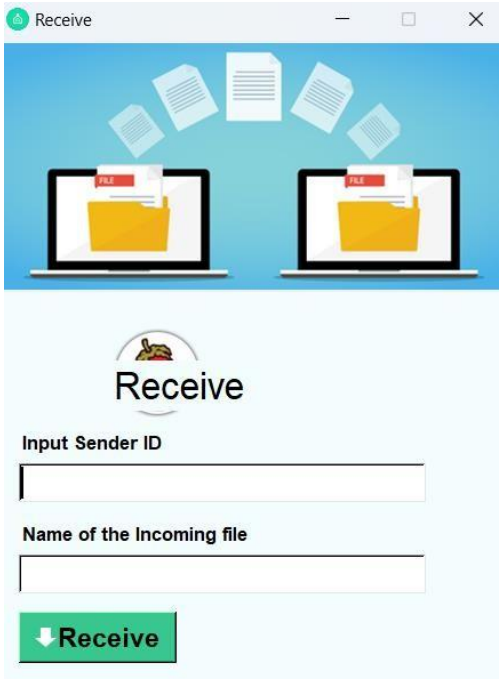


Figure 22: Once the receive button is click

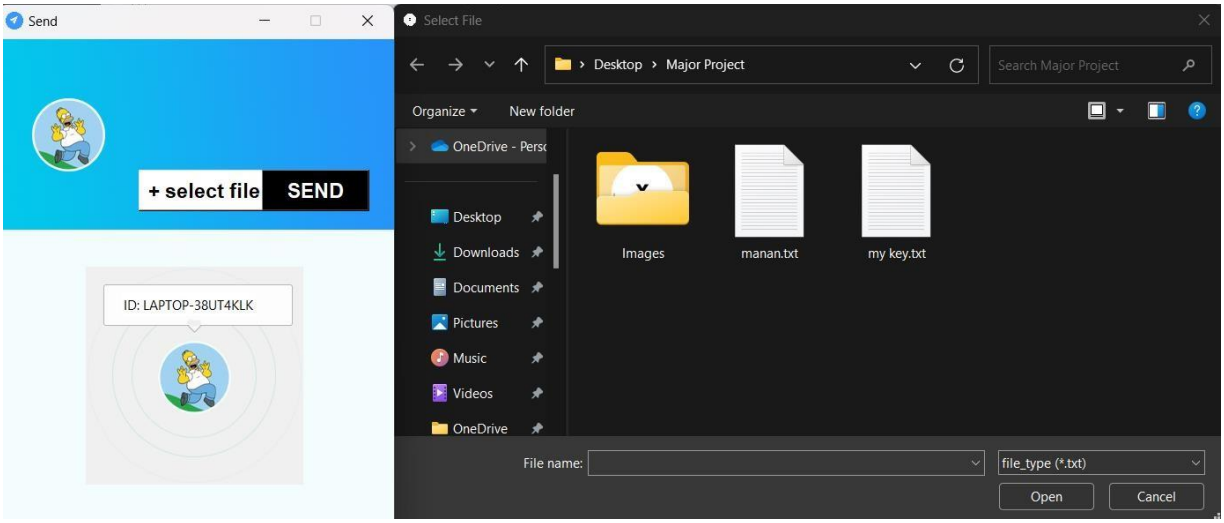


Figure 23: Once the select file option is clicked on the send page, the file explorer opens with default file type to select as .txt

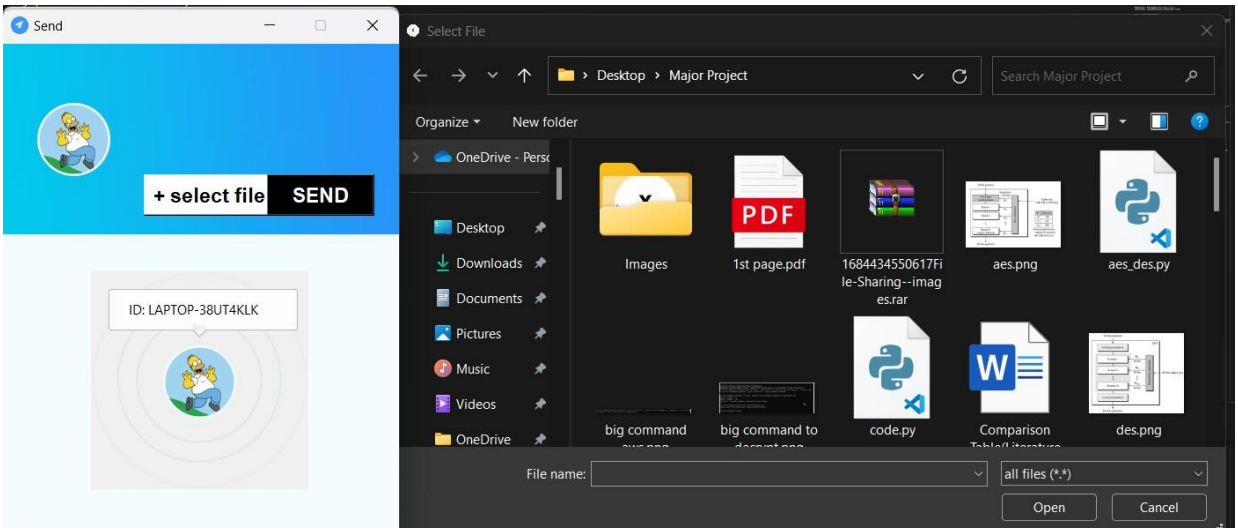


Figure 24: Clicking on the arrow adjacent to file_type helps us in selected all files as the option

3.5 Key Challenges

As college students developing a secure data transfer system like Fortified File Exchange (FFE), we encountered several challenges:

1. Technical Challenges

Understanding Complex Cryptographic Concepts: Implementation of encryption algorithms and understanding their limitations was challenging.

Mastering Networking Protocols: This is still a work in progress for us, as Networking protocols are difficult to understand at times.

Large File Transfers: Transferring large files over a network is still not a thing that we have mastered.

Addressing Security Vulnerabilities: We don't know about every vulnerability present in the cyber world, and thus we are still addressing this issue.

2. Project Management Challenges

Effective Team Collaboration: It was challenging for us to manage tasks and workload initially, but we were able to coordinate with each other soon. Timely submission of drafts to our supervisor was also a point of concern in the beginning but it is the part of our life process now.

Weekly Logs and Meeting with the supervisor: We maintained our weekly log from the first day of Week-I and met with our supervisor from time to time.

3. Time and Resource Constraints

Balance of Academics and Project Work: Managing both Project work, Lab Hours and the classes was a difficult task at first but we inherited ourselves into the situation and did well after the first few weeks.

To overcome these challenges, we followed the following:

- Seeking Expert Guidance: We went to our supervisor time and time again for his guidance and he never refused to help.

- Resolving the bugs in our codes and algorithms: We have been working on a near 100% efficient code that works and follows all the network guidelines for a secure file transfer.

Chapter 4

Testing

We have tested Fortified File Exchange implementation and encryption algorithms used on Visual Studio Code and the strategy and test cases along with the outcomes are shared below:

4.1 Testing Strategy

For us, ensuring security, performance and reliability is top-tier important and for the same we need to develop a good testing strategy.

Our objectives going with our testing strategies are listed as follows:

- We need to introduce 2FA at the initial phase, so that major malicious activates are terminated at a phase prior to file transfer.
- Any system depends on the amount of database it can handle, similarly in our case we need to send our data from one device to another without the data being intercepted by hackers.
- Two more aspects come into the picture, one is the large file transfer and other is the large volume of file transfer.
- Currently our code and algorithms align to Windows OS, but we plan to introduce our UI to other Operating Systems like Linux and MacOS.

We will also be using particular testing tools for our Fortified File Exchange:

- Implementing Encryption Algorithms like AES, DES, Triple-DES and RSA require initial testing on tools like OpenSSL and pycrypto, we haven't used them as of now but will surely be using them in our future testing.

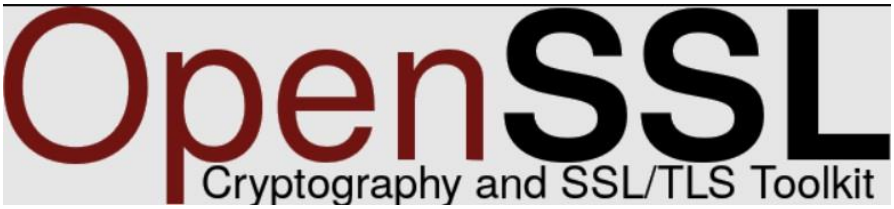


Figure 25: OpenSSL Logo

Figure 26: pycrypto visual

- Using tools like Wireshark to analyze the packets during file transfers and debugging the protocols is boon to cybersecurity enthusiasts.

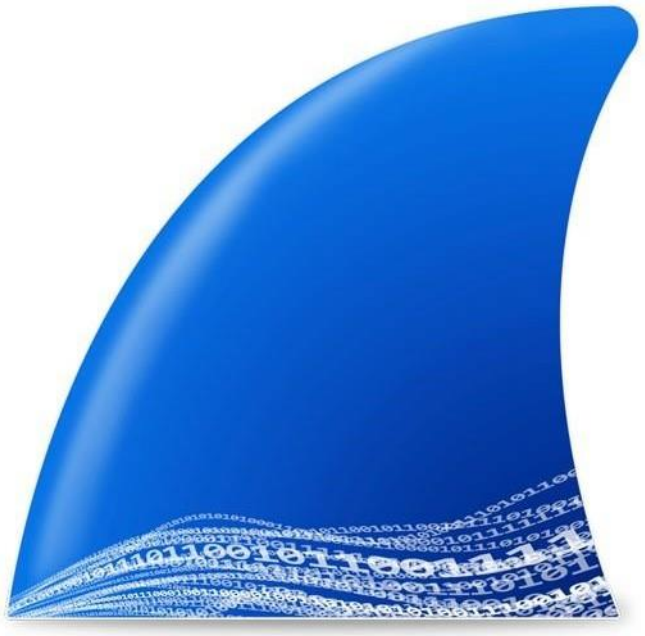


Figure 27: Wireshark Logo

4.2 Test Cases and Outcomes

- **File Transfer UI**

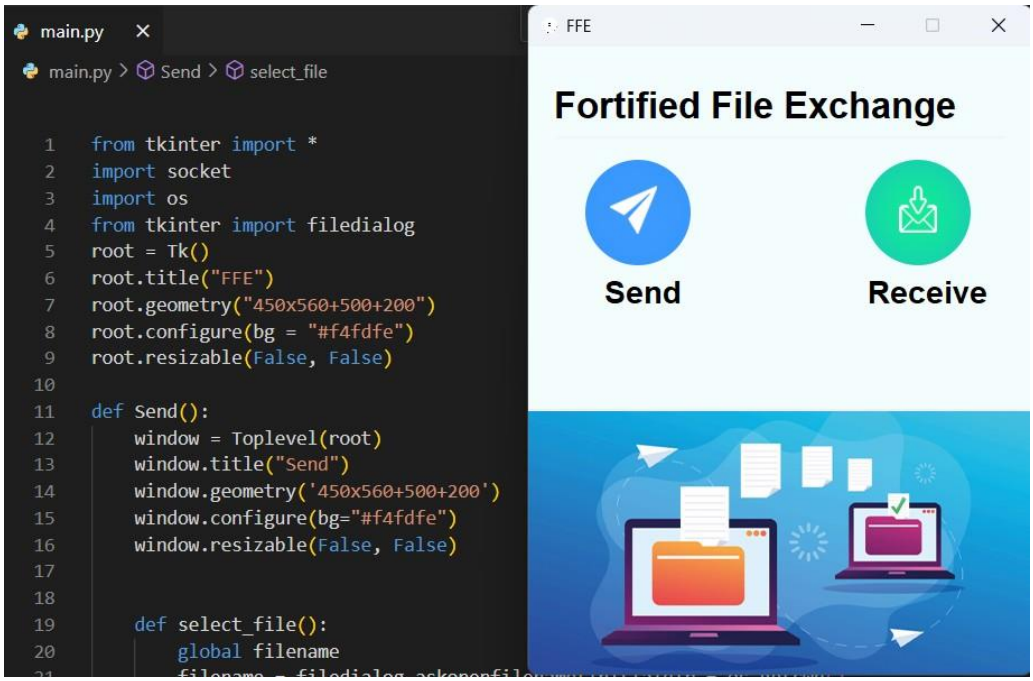


Figure 28: Testing Phase 1

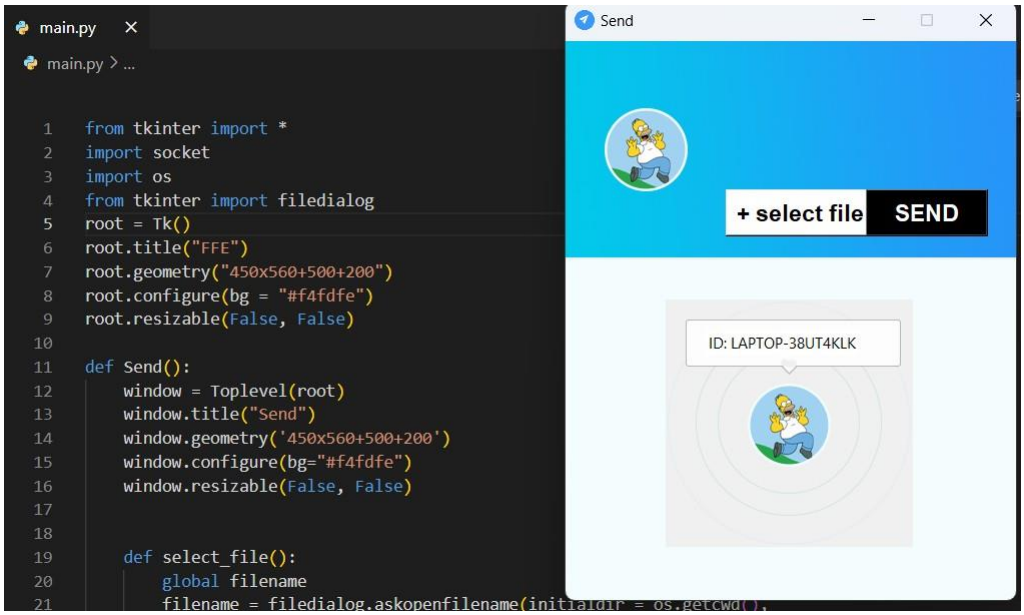


Figure 29: Testing Phase 2

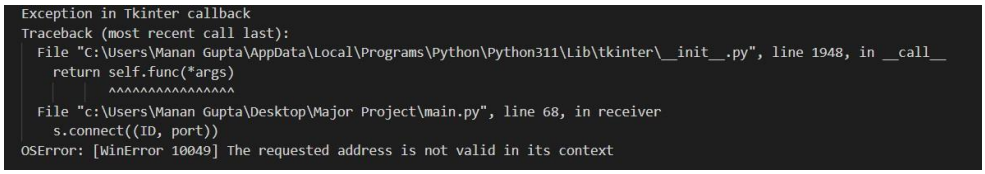


Figure 30: Testing Phase 2 error

In regards to Figure 30, we worked on reimplementing the system on the basis of socket programming to first resolve the error and then embed it into our main system. This is still a work in progress.

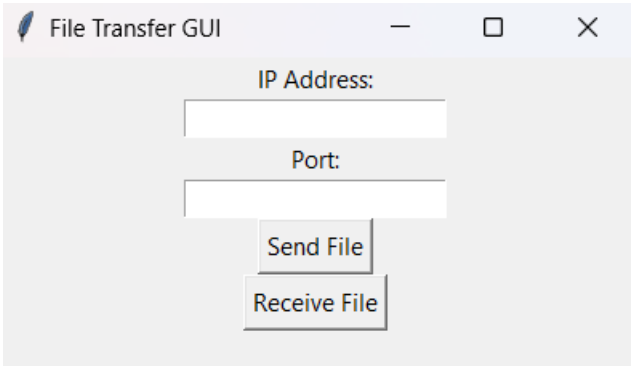


Figure 31: Developed only for fixing Phase 2 error

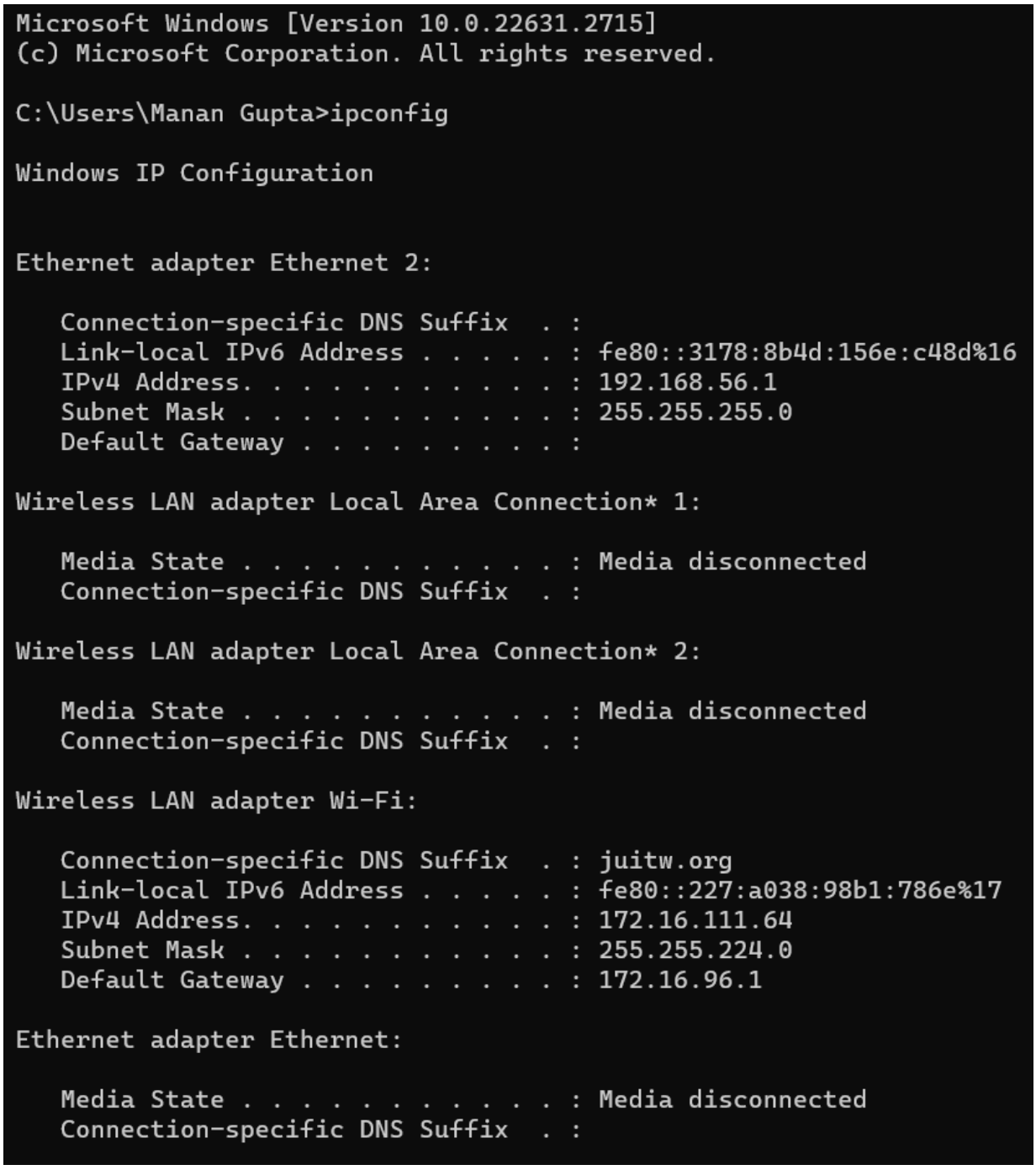


Figure 32: Getting the IP Address of the destination machine using ipconfig command in the command prompt

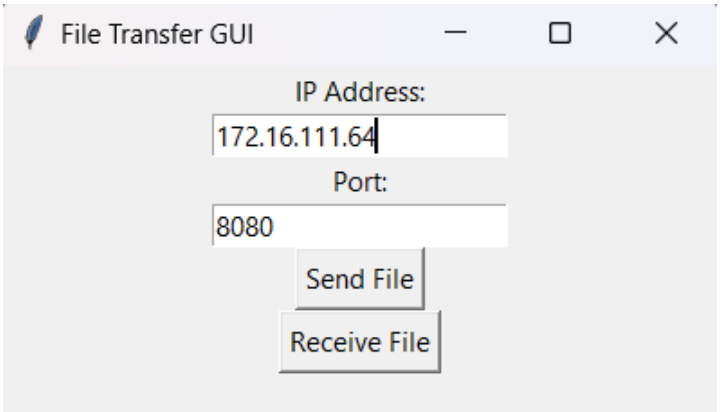


Figure 33: Entering the IP and Port Number in the designated columns.

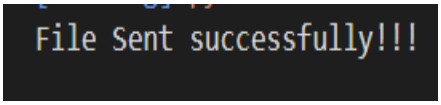


Figure 34: Sent Successfully message

Figure 35: UnboundLocalError received on receiver end

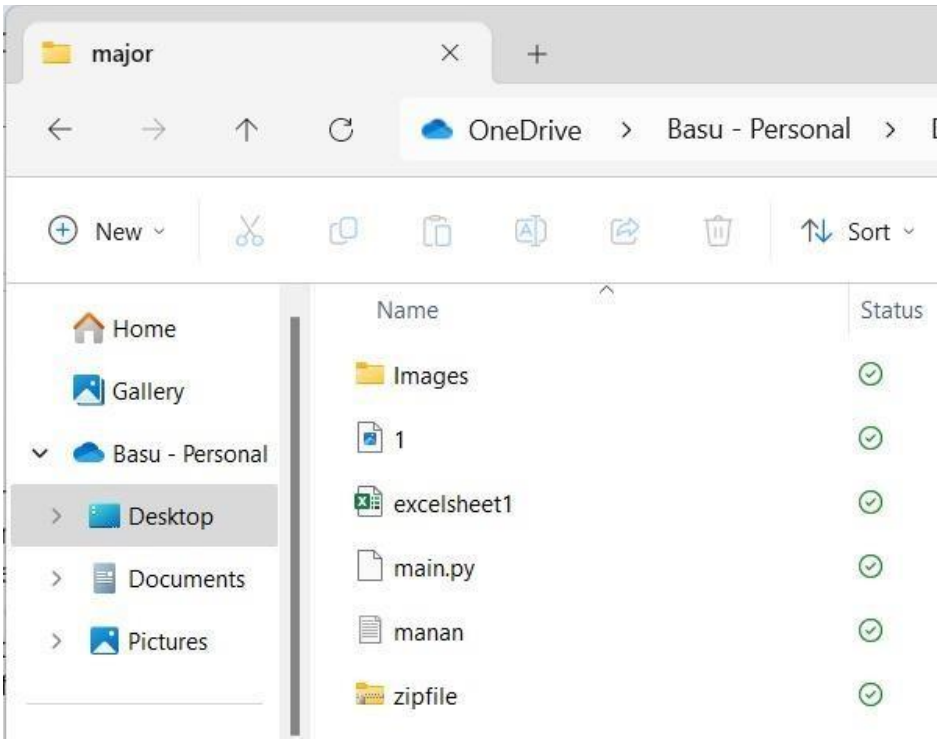


Figure 36: Resolved UnboundLocalError(Receiver Folder)

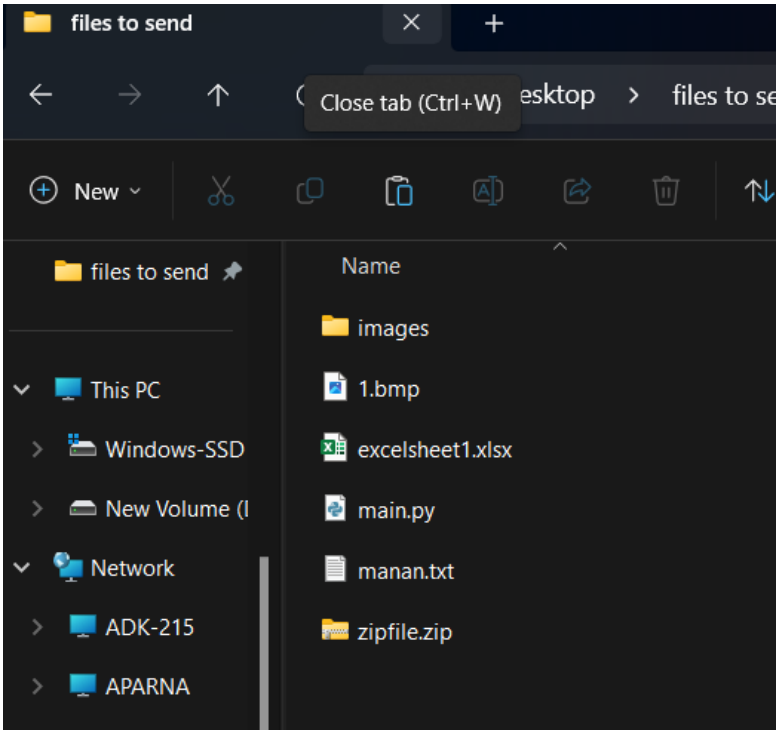


Figure 37: Sender Folder

Chapter 5

Results and Evaluation

5.1 Results

After researching and implementing the code for a successful and secured Fortified file exchange system, in which we tried and include various types of the cryptographic algorithms such as AES, DES, T-DES, ECC and RC6. this system provides an enhanced level of security by implementing a combination of encryption techniques (AES and ECC) and choosing a transferring protocol which is safest and easy to manage to create the system efficient and reliable at the same time.

After using all the cryptography techniques such as AES, DES, T-DES and RC-6, the most secured and efficient technique which has lesser complexity and higher susceptibility i.e., AES and ECC.

In the coming sections, we evaluated our learnings and presented them in form of tables and a graph. These comparisons not only set a benchmark for the future considerations but also help us in distinguishing various encryption standards and file transfer techniques.

In order to develop these tables, we have used existing data from various research papers that we used as our references, we then compared our findings with the existing ones and formulated the comparison tables from them. When the graph is considered, we use matplotlib, which is the plotting library for Python programming language. The purpose behind using matplotlib is its ability to provide an object-oriented API for embedding plots into applications.

5.2 Comparison with Existing Solutions

	DES	T-DES	AES	RC6	ECC
No. of rounds	16	48	10	20	-
XOR	64	192	84	40	80
S-Box	16	48	10	-	20
Rotate	1 bit, 2 bit	1 bit,2 bit	1 byte, 2-byte, 3-byte	5,3	-
Addition mod 2^32	-	-	-	523	256
Subtract from 2^32	-	-	-	<=20	-
Multiply mod 2^32	-	-	-	40	-
Key size	56-bits	3*56 bits	128,192, 256 bits	128,192,256 bits	256 bits
Block size	64 bits	64 bits	128 bits	256 bits	No fixed block size
Ciphertype	Block cipher	Blockcipher	Block cipher	Block cipher	Public key cipher
Shift left	4,12	12,36	1,1,1	40,132	128

Table 3: Comparison Table of Encryption Standards

Afterwards, we will look into the various protocols such as UTP, FTP, SFTP and STP which we used to create a centralized system which generated a system that could carry large volume of data with ease and capable supporting the cryptographic algorithms we used to secure the data or any attacks over the network. The protocol we used to create this scenario.

Comparison	**UTP**	**STP**	**FTP**	**S/FTP**
Full Name	Unshielded twisted pair	Shielded twisted pair	Foil twisted pair	Shield foil twisted pairs
Cable Shield	None	Braiding	Foil	Braiding
Twisted pair	None	None	None	Foil
Crosstalk	High crosswalk	Low crosstalk	Low crosswalk	Low crosstalk
Termination	Easier to terminate and saves time	Difficult and talk a longer time	Difficult and takes a longer time	Difficult and takes longest time
Bonding and Grounding	Not required	Necessary	Necessary	Necessary
Data rate	Low	High	High	High
Application	Home network	Longer distance network	Longer distance network	Longer distance network
Costs	Cheaper and does not require much maintenance	Moderately expensive	Moderately expensive	Very expensive

Table 4: Comparison Table of Transfer Protocols

Parameter	SFTP (SSH File Transfer Protocol)	FTPS (FTP over SSL/TLS)	Managed File Transfer
Encryption Standard	AES-256, RSA	AES-256, RSA	AES-256, RSA (and others)
Data Security	Data Encryption is done in transit and when the data is at rest	Data Encryption is done in transit only	Data Encryption is done in transit and when the data is at rest with t he help of additional security features
Ease of Use	Relatively easy to set-up	Requires additional FTPS sever	Requires dedicated MFT server
Performance	Slower than FTP due to encryption overhead	Similar to FTP	Variable, depending upon MFT software
Scalability	Suitable for small and medium file transfers only	Suitable for small and medium file transfers only	Highly scalable (Suitable for any file transfer)
Cost	Open-source (Free)	Requires paid server software	Requires MFT software.

Table 5: File Transfer Comparison: Encryption and Features

```python
import matplotlib.pyplot as plt
data = [
    {'Algorithm': 'AES', 'Data': 32, 'Time': 1.81, 'Memory': 84261},
    {'Algorithm': 'DES', 'Data': 64, 'Time': 1.83, 'Memory': 66531},
    {'Algorithm': 'Triple-DES', 'Data': 128, 'Time': 2.03, 'Memory': 54395},
    {'Algorithm': 'RSA', 'Data': 256, 'Time': 2.14, 'Memory': 22189},
    {'Algorithm': 'AES', 'Data': 512, 'Time': 2.43, 'Memory': 41113},
    {'Algorithm': 'DES', 'Data': 512, 'Time': 2.63, 'Memory': 33207},
    {'Algorithm': 'Triple-DES', 'Data': 512, 'Time': 2.74, 'Memory': 26865},
    {'Algorithm': 'RSA', 'Data': 512, 'Time': 4.57, 'Memory': 15679},
]
plt.figure(figsize=(10, 6))
plt.bar(
    [item['Algorithm'] for item in data],
    [item['Memory'] for item in data],
    label='Memory (KB)',
    color='orange',
    bottom=[item['Data'] + item['Time'] for item in data],
)

plt.title('Comparison of Data for Encryption Algorithms')
plt.xlabel('Encryption Algorithm')
plt.ylabel('Data (KB)')
plt.legend()

plt.savefig('encryption_comparison.png')
```

Figure 38: Code snippet for plotting Comparison of Data for Encryption Algorithms. (Based on detected data)

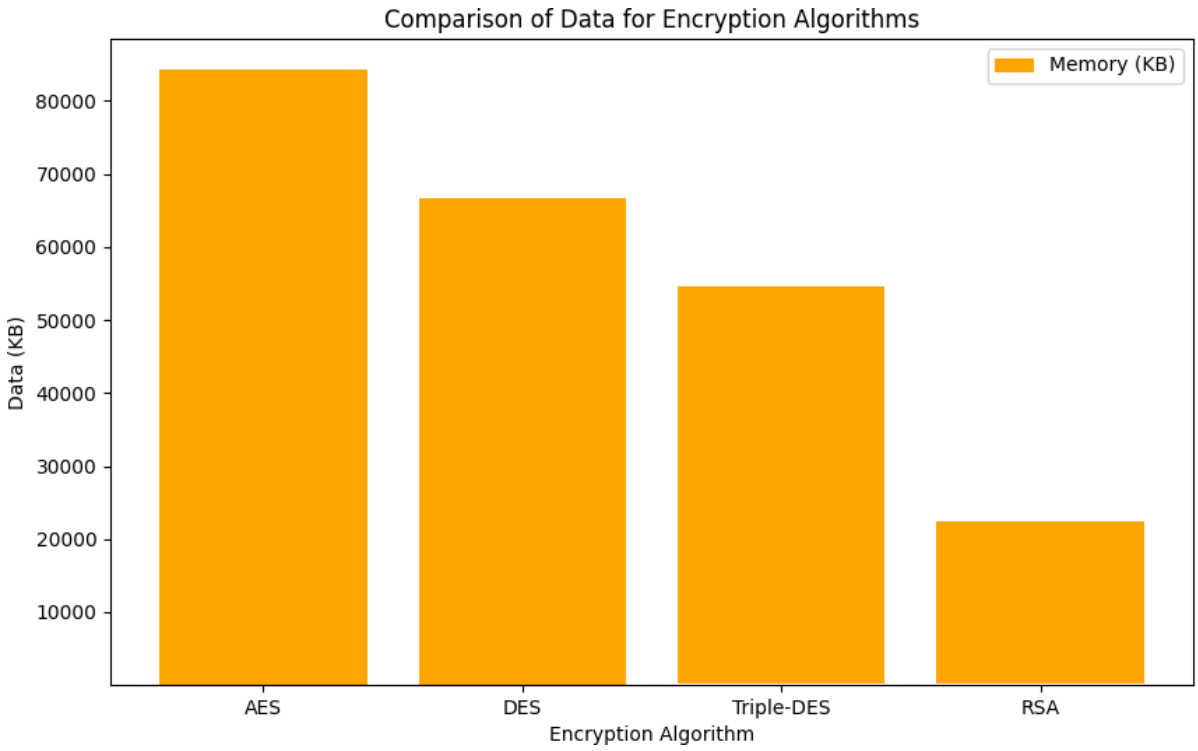


Figure 39: Plot showing the Comparison of Data for Encryption Algorithms

Chapter 6

Conclusions and Future Scope

6.1 Conclusion

When we started this project, we were unaware of the fact that it would have so many challenges and lots of different levels of learning. We started by review Research papers of different authors who have published their literature on file transfer techniques and various encryption standards. Integration of Blockchain Technology is something which was far beyond our understanding and we definitely ought to work in that sector in the near future because of its high demand.

Working on projects like these and that too with a group partner is definitely good for the interpersonal growth of us as a human being and we tend to do such projects if given another chance. By the means of Fortified File Exchange: A Secure Data transfer technique, we learned about processes involved in the data transmission from a sender to a receiver and combining Secure File Transfer Protocols (SFTPs) with End-to-end encryption standards in combination to user authentication makes project like FFE a secure and reliable solution for both the individuals as well as the organizations.

Such techniques are a boon to today's digital landscape and ideas like these should keep coming to young minds like us for a better digital future.

Learning and Moving Forward

Adding on to the valuable learning experience, highlighting the complexities and depth of cyber security. When we began exploring file transfer methods through research papers and looking into the established file transfer techniques and encryption standards, we were constantly coming across the concept of Blockchain technology and it is a frontier beyond our initial grasp. Recognizing its growing importance and vast potential in cyber security, we acknowledge the need to explore this area further in the near future.

Beyond File Transfer Security: Embracing Broader Horizons

Apart from the technical understanding, the collaborative task promoted interpersonal growth. We are confident in our ability and enthusiasm to tackle future projects, both individually and collaboratively. Through "Fortified File Exchange" (FFE), we gained a deeper understanding of data transmission processes, the synergy between Secure File Transfer Protocols (SFTPs) and encryption techniques that promote end-to-end security, and the crucial role of user authentication. This combined approach paves the way for secure and reliable data transfer solutions for individuals and organizations alike.

AI advancements

Artificial Intelligence and tools like Gemini (formerly Bard) and Chat-GPT can be used for multiple level of advancements in both the scope of learning as well as developing codes. Use of AI is not at all bad in our opinion until it helps you understand the trend better. Additionally, use of tools like Whimsical to generate flowcharts using the attached AI model definitely helps us in understanding the technology better. With our experience and exploration around the field of cybersecurity there is a small gap between what a normal market search and using AI to enhance your search. This small gap can be termed as innovation. AI helps you find articles, tools and databases that are still not popularly known to public. Adding AI tools like Whimsical, helped us develop visuals that are not only engaging but also less text heavy to enhance readability. Moreover, we also prefer use of these tools to such an extent that it helps reduce human effort and leads in lot of time saving.

6.2 Future Scope

As stated in the conclusions, AI will definitely be a forefront to develop techniques like this in the future. But apart from AI, our scope considerations also consider the following things for the future of FFE:

- There is a lot of scope of improvement in the encryption and decryption in terms of their performance.
- Latency of File Transfers can be further reduced.
- Adding options to send/receive more file types and of larger size too.

- Keep developing the User-Interface based on user demands
- Integrate Cloud services and cloud storage providers for a more secure and fast Fortified File Exchange Server.
- Taking our GUI to a Web-based GUI or an App-based GUI in the near future.
- Apart from the research papers and various scholarly articles available in the market, we would also prefer looking into patents that various entities like Apple, Samsung may be filing to enhance their existing file sharing techniques.

References

1. R. Mathwale and R. Ramisetty, "Blockchain Based Inter-Organizational Secure File Sharing System," 2023 2nd International Conference for Innovation in Technology (INOCON), Bangalore, India, 2023, pp. 1-5, doi: 10.1109/INOCON57975.2023.10101350.
2. Kumar, M., Jadon, K.S., Gupta, N. (2023). T-AES- and ECC-Based Secure Data Communication in Peer-to-Peer Networks. In: Woungang, I., Dhurandher, S.K. (eds) 5th International Conference on Wireless, Intelligent and Distributed Environment for Communication. WIDECOM 2023. Lecture Notes on Data Engineering and Communications Technologies, vol 174. Springer, Cham. https://doi.org/10.1007/978-3-031-33242-5_4.
3. C. Susmitha, S. Srineeharika, K. S. Laasya, S. K. Kannaiah and S. Bulla, "Hybrid Cryptography for Secure File Storage," 2023 7th International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 2023, pp. 1151-1156, doi: 10.1109/ICCMC56507.2023.10084073.
4. C. Vaidya, K. Takalkar, A. Ghosekar, S. Nimgade and V. Ghode, "Decentralized File Sharing," 2023 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS), Bhopal, India, 2023, pp. 1-6, doi: 10.1109/SCEECS57921.2023.10062977.
5. Mhatre, V. S., & Patel, S. N. S. (2023, August). Secure file storage on cloud using elliptic curve cryptography (ECC) algorithm. In 2023 International Conference on Information Communication and Embedded Systems (ICICES) (pp. 1-6). IEEE.
6. Ghosh, R., Yadav, P., Khan, Z., & Panika, L. (2023). Decentralized file storing and sharing system.
7. Xi, Peng, Xinglong Zhang, Lian Wang, Wenjuan Liu, and Shaoliang Peng. "A review of blockchain-based secure sharing of healthcare data." IEEE Access 11 (2023): 102063-102081
8. Ranadive, S. S., Sawant, H. S., and Pinjarkar, J. E. "Secure file storage on cloud computing using cryptographic algorithm." 2023 IEEE International Conference on

Advances in Computing, Communication and Information Technology (ICACCI). IEEE, 2023.

9. Li, X., & Wang, C. (2021). Lightweight and secure data transfer scheme for cloud storage using certificateless cryptography. In 2021 IEEE International Conference on Cloud Computing (ICCC) (pp. 198-203). IEEE.
10. Nevpurkar, M., Bandgar, C., Deshmukh, R., Thombre, J., Sadafule, R., & Bhat, S. (2020). Decentralized File Storing and Sharing System using Blockchain and IPFS. International Research Journal of Engineering and Technology (IRJET), 7(5), 560-565.
11. Meng, W., & Ding, M. (2020). A secure and efficient data transfer scheme for cloud storage. IEEE Transactions on Cloud Computing, 8(2), 493-505.
12. Schneider, M. (2020) "Secure Automated File Exchange (SAFE) – Enabling More Efficient Transfers of Sensitive Data", International Journal of Population Data Science, 5(5). doi: 10.23889/ijpds.v5i5.1599.
13. Shitami, J., Honjo, G., Hiraki, K., & Inaba, M. (2020). Secure file transfer on 100 Gbps network using single server. IEICE Technical Report, 119(435), 1-6.
14. Alves, M. A., Barbosa, M. P., & Falcão, D. (2019). Secure and efficient data transfer for cloud storage using attribute-based encryption. In 2019 IEEE International Conference on Cloud Computing (ICCC) (pp. 411-418). IEEE.
15. Huang, S., Gao, W., & Zhang, K. (2019). A Method of Bluetooth Large File Transfer Based on BlueSolei+ FTP. Computer Science and Applications, 97(137), 1224.
16. Zheng, Q., Xu, T., & Cui, J. (2018). A secure and efficient data transfer scheme based on identity-based cryptography for cloud storage. IEEE Transactions on Cloud Computing, 6(3), 599-610.
17. Srivastav, A., & Bhartee, A. (2016). Design of secure file transfer over internet. International Journal of Advanced Research in Computer and Communication Engineering, 5(5), 233-240.
18. Zuo, L., & Zhang, W. (2016). A secure and efficient data transfer scheme for mobile cloud storage. IEEE Transactions on Information Forensics and Security, 11(9), 1691-1704.
19. Rhee, W., & Park, J. H. (2016). A novel secure data transfer protocol for cloudcomputing environments. IEEE Transactions on Cloud Computing, 4(1), 57-69.

20. Prajapati, P., Patel, N., Macwan, R., Kachhiya, N., & Shah, P. (2014). Comparative Analysis of DES, AES, RSA Encryption Algorithms.
21. Northern Illinois University. [Online]. Available: https://www.niu.edu/presentations/deliver/index.shtml [Accessed: November 25, 2023].
22. Pycrypto: The Python Cryptography Toolkit [Online]. Available: https://pythonfix.com/pkg/p/pycrypto/ (Accessed on: November 19, 2023).
23. GeeksforGeeks. (2023, November 29). Difference between RSA algorithm and DSA. [Online]. Available: https://www.geeksforgeeks.org/difference-between-rsa-algorithm-and-dsa/ [Accessed: October 29, 2023]
24. Triple DES Encryption Algorithm. Tutorialspoint. [Online]. Available: https://www.tutorialspoint.com/cryptography/triple_des.htm [Accessed 29 September 2023].
25. Wallarm (2023). What is AES? Advanced Encryption Standard. Retrieved from https://www.wallarm.com/what/what-is-aes-advanced-encryption-standard
26. A. Menegai, P. Mittal, and S. Panwar, "A lightweight and provably secure attribute-based encryption scheme for secure data transfer in cloud storage," Journal of Network and Computer Applications, vol. 205, pp. 103384, 2023.
27. X. Chen, J. Li, J. Ni, Y. Zhang, and X. Li, "Privacy-preserving and verifiable secure file transfer for fog computing," IEEE Transactions on Sustainable Computing, doi: 10.1109/TCSC.2023.2290221, 2023.
28. M. Singh, N. Kumar, and J. Singh, "Blockchain-enabled secure and privacy-preserving data transfer for internet of things," Journal of Information Security and Applications, vol. 48, p. 102852, 2023.
29. L. Xu, C. Chen, Z. Liu, and J. Zhou, "Provably secure identity-based signcryption for secure and efficient data transfer in fog computing," Security and Communication Networks, doi: 10.1155/2023/6882402, 2023.
30. D. Liu, Y. Zhang, X. Chen, J. Ni, and X. Li, "Efficient and privacy-preserving verifiable secure file transfer for fog-assisted IoT," Future Generation Computer Systems, vol. 140, pp. 422-433, 2023.
31. S. Kumari, S. Ghosh, and S. Banerjee, "A hybrid blockchain-based approach for secure file sharing in fog computing environments," Journal of Information Security and Applications, vol. 47, p. 102793, 2023.

32. A. Kumari and S. Ghosh, "Secure and efficient data sharing using consortium blockchain in fog computing," Sustainable Computing: Informatics and Systems, vol. 38, p. 101323, 2023.
33. T. Ma and J. Zhang, "Blockchain-based secure data sharing scheme with verifiable auditing for fog computing," Security and Communication Networks, vol. 2023, no. 1, pp. 1-13, 2023.
34. J. Wan, J. Li, Y. Zhang, C. Zhao, and X. Li, "Blockchain-enabled secure and efficient data sharing for mobile edge computing in vehicular networks," IEEE Transactions on Intelligent Transportation Systems, doi: 10.1109/TITS.2023.3242224, 2023.
35. M. Imran, S. Kumari, S. Ghosh, and S. Banerjee, "Attribute-based encryption with blockchain for secure and scalable data sharing in fog computing," Computer Networks, vol. 242, p. 106812, 2023.