EFTS: An Encryption File Transfer System Applying Advanced Encryption Standard (AES) Algorithm

Omar Mahdi Eldood Mahdi[1], Julia Juremi[2]
Forensics and Cyber Security Research Center
Asia Pacific University of Technology & Innovation
TP049185@mail.apu.edu.my, julia.juremi@apu.edu.my

***Abstract*— Since the start of the Covid19 pandemic, most of the work has been done online, including teaching, consulting, and exchanging confidential data between coworkers. Transferring data over the internet, on the other hand, is always risky. Some data can be lost or delivered to the wrong person by mistake. When a worker types in an email address to send a confidential attachment, they might type in a letter or a number by mistake that will send the email to another person. If the data ends up in the wrong hands, the company can suffer a significant loss. Therefore, an encryption is important incase an unauthorized user obtains access to the data, they will be unable to read it due to encryption. It secures sensitive data as well as increases the reliability of client-server communication. EFTS is an encryption file transfer system, developed to ensure safe and secure data transferring over the internet and to reduce the impact cased to the companies in case of a human error.**

***Index Terms*—Information Security, Cryptography, AES, Encryption Algorithm**

1. Introduction

Cryptography can be traced back to 4000 years ago when the ancient Egyptians uses hieroglyphs where the scribes used as a form of communication to convey messages on behalf of the king. After a few decades, scholars began to develop a new cryptographic method which is the Caesar Shift Cipher where the letters of the message are being shifted by a certain number, and during World War 2 between Korea and Vietnam, it is used to prevent enemies or spies from obtaining vital information and Morse Code was used during the war to safely set up supplies and planning strategy and the same concept is being implemented in real-life scenarios such as companies have utilized and set up secured communication in their company network to prevent a rival company from getting vital information such as a business plan or user authentication.

Nowadays, we transfer and exchange data using the internet, we use our mobile phones and computers to send and receive messages. As a result, information security has become a major concern. When many organizations take advantage of technology services for improving their productivity, protecting data under risk must be of paramount concern. An information security policy is developed and implemented by organizations to ensure that all sensitive data is handled and protected uniformly, and the policy should apply to all users on the network. Access to various types of data is determined, as is the authentication process and the measures used to keep data secured. If a firm's information security policy is going to be effective, it must specify exactly what the company and its employees must do to protect user information [8].

2. Cryptography Encryption

Cryptography is the most widely used method of data security [7]. The Caesar Cipher was one of many cryptographic ciphers. Caesar ciphers were much easier to decipher compared to the current cryptographic techniques because algorithms and cryptosystems are far more sophisticated nowadays. Several rounds of ciphers and encryptions are used to provide highly secure transit and preservation [4]. Various types of irreversible encryption are now in use, ensuring the message's security indefinitely. Decrypting today's modern algorithms is possible but deciphering the meaning of a single message would take years, if not decades. As a result, the race to create newer, more powerful cryptography algorithms continues.

Encryption is the process of converting plaintext to ciphertext. A ciphertext should be readable only to authorized users with the decryption key. If the encryption method is properly applied, data will be completely safe from unauthorized access, and messages will be encrypted in such a way that anyone listening in on the conversation will not comprehend what is said [9]. When it comes to encryption mainly 2 algorithms methods are mainly used which are the Symmetric which use the same key for encryption and decryption and Asymmetric algorithms which uses different keys.

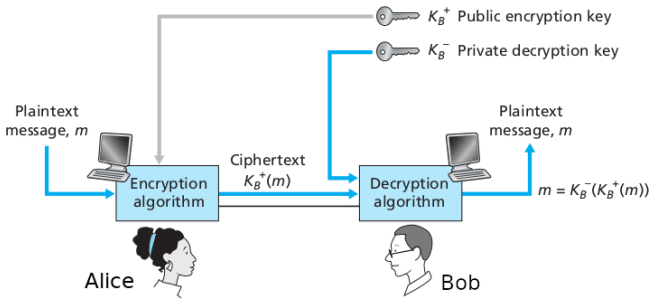

Figure 1: Asymmetric Encryption Method

Starting with the asymmetric encryption method as shown in Figure 1, each party must share a public key. To ensure the security of their communications, the two parties must keep their private keys hidden. After obtaining their recipient's public key, the sender can use it to encrypt the data that they want to keep secure. After the message has been encrypted with the public key, the secret key from the same key pair must be used to decode it. Even if you have the same private key, you will be unable to decode the data because the technique is easy to compute in one direction but nearly impossible in the opposite direction. After receiving the communication, the

receiver decrypts it is using their private key. If the receiver wants an encrypted connection in return, they can encrypt their message with the other party's public key. Once encrypted with the public key, the data can only be decrypted with the corresponding private key. As a result, previously unknown individuals can now exchange data securely.

In the symmetric algorithm, the secret key used by the transmitter and receiver is typically between 80 and 256 bits long. These encryptions use either a stream cipher or a block cipher which are quick because they rely on simple mathematical equations [9]. The Block Cipher encryption method encrypts data in units of k bits per block. If k is equal to 64, the message is divided into 64-bit blocks and encoded separately for each block. Figure 2 illustrates the 64-bit encryption process.

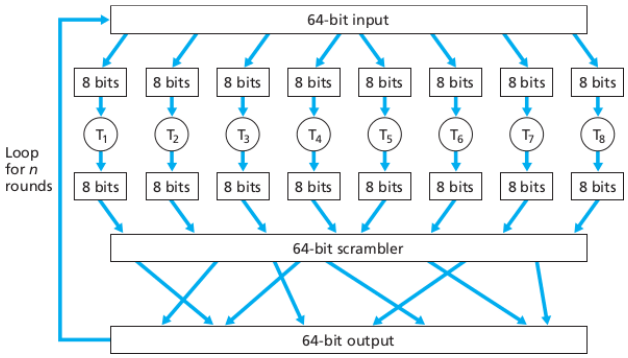

Figure 2: Encryption Methods using 64-bit

Data Encryption Standard (DES), and Advanced Encryption Standard (AES) are both popular block ciphers. DES employs 64-bit blocks and a 56-bit key for its encryption. Because of this, cracking a DES key will roughly 6.4 days, on the other hand, using 128-bit blocks, AES encrypts data up to a key length of 128, 192, or 256 bits. Because of its low resource consumption, this algorithm key can 149 trillion years to be cracked.

When using symmetric cryptography, just one key is needed to encrypt the data being transmitted. In symmetric encryption, the same key is used for encryption and decryption, making it the simplest type of encryption [4]. The cryptographic technique encrypts the data using the private key, and when the data must be retrieved again, only the authorized person with the private key may decrypt it. If the key is not sent to the intended receiver, Private Key Encryption can be utilized on both in transit and at rest data.

Advanced Encryption Standard (AES) is an example of symmetric encryption with a high level of security [3]. To encrypt and decode data, AES encryption utilizes block ciphers with widths of 128, 192, or 256 bits, and carking it is almost impossible, therefore it is commonly employed to protect highly confidential information in a variety of businesses, including government, healthcare, and finance. Both asymmetric and symmetric encryption methods have been used in developing various tools and applications.

3. Encryption Tools

BitLocker

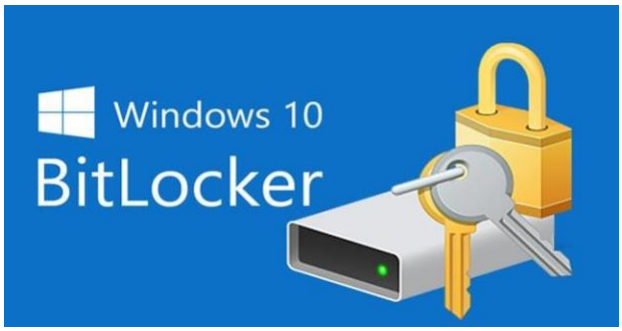

Figure 3: Windows BitLocker [1]

BitLocker is an Encryption feature that secures all data on a hard drive using the AES algorithm. It is free and easy to configure as it comes with Windows as shown in Figure 3. BitLocker is designed to protect sensitive information stored in the PC by creating a recovery key that will be needed to access all the PC settings once it starts [6]. In addition, the TPM module proves an extra security level by encrypting the whole device. The TPM is an embedded system that computer manufacturers install in many modern systems. It works in combination with BitLocker to help secure the user's data and ensure that a device is not tampered with while it is down. Aside from the TPM, BitLocker also can lock the startup process until the user inserts a physical device that contains a startup key or types in a personal identification number (PIN) number. This security measure can prevent the computer from starting or hibernating until the user enters the pin or present the startup key [6].

AxCrypt

Figure 4: AxCrypt [5]

AxCrypt is an open-source encryption software with simple-to-use security measures. It is great for users who prioritize the security of their data in both the public and private realms. AxCrypt includes encryption, encrypted file editing, secure file deletion, password generating, and password store capabilities. Users may simply encrypt and decrypt any data using the simplified interface's drag-and-drop functionality. AxCrypt is exceptional because it provides basic yet effective tools for encryption that are simple, and straightforward. It also allows

users to access their encrypted data from their mobile phones [2].

4. Encryption File Transfer System (EFTS)

The Encryption File Transfer System (EFTS) was developed using Visual Studio Code editor and Python programming language. Several libraries were used such as the cryptography library which is a python cryptography module for encrypting files and data with AES-256 algorithm. Tkinter library is also in used as part of the GUI development. It is a Python standard library for graphical user interfaces. Tkinter includes several controls that can be used to create a graphical user interface. Smtplib is also used as part of the EFTS development where it is a Python package that allows you to send emails by using Simple Mail Transfer Protocol (SMTP). The smtplib module is pre-installed. It abstracts all of SMTP's intricacies.

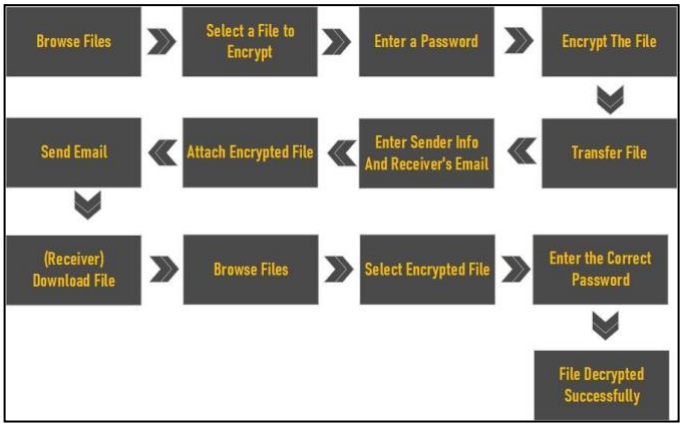

Figure 5: EFTS Architecture

Figure 5 above displays the EFTS architecture.In order to use EFTS tool, the user must first browse their computer and then select any file type, such as MP3, JPG, TXT, and so on. The path to the selected file will be displayed in a dialogue box for the user. After selecting a file, the system requires the users to encrypt it with a password that they must type into the entry bar; there is also an option to "show password" that allows them to see the password as they are typing it. When it comes to decryption, the process is just as simple as it is when it comes to encryption. The user simply needs to select the encrypted file and enter the same password that they used to encrypt it. Furthermore, in case the user forgets their password, EFTS provides them with a copy of the password in their emails as a backup the moment they click "ENCRYPT." The system will also allow users to email the encrypted files from within the system, eliminating the need to visit their email provider's website.

Figure 6 demonstrates the system's primary window; the user must click on the "Browse" button which will launch the File Explorer and allow the user to select any file to encrypt. The user must then input a password to encrypt the selected file.

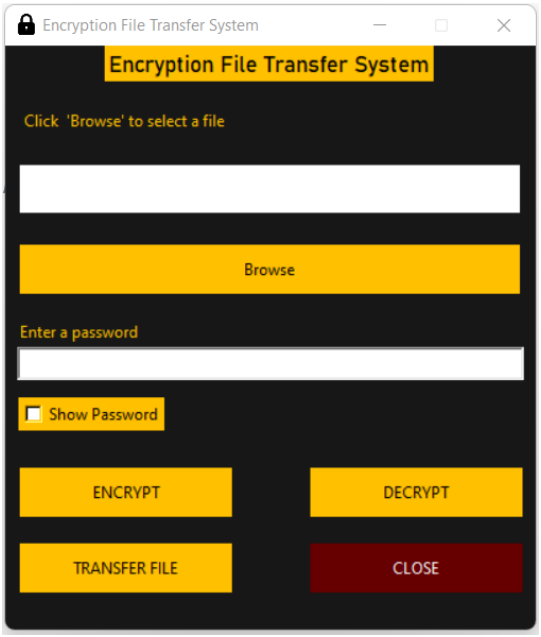

Figure 6: EFTS Main Window

User can review their password by clicking the "Show Password" option. By clicking "ENCRYPT," the selected file will be encrypted with the written password and a copy of the password will be send to an email in case the user forgot the password. Also, the selected file can be decrypted later by clicking "DECRYPT" using the same encryption password, and once it's finished, the password field will be cleared automatically.

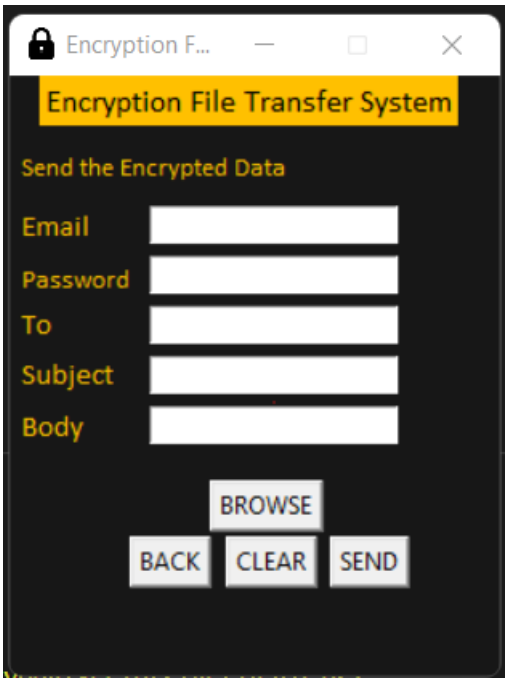

Figure 7: EFTS Transfer File Window

The "TRANSFER FILE" button will lead you to the window depicted in Figure 7. The user must first fill out all the forms

and enter their email address and password, as well as the recipient's email address, the subject, and the body of the message. Then, by selecting the "BROWSE" button, the file explorer will open, allowing the user to select their encrypted file, and it also enables the user to pick multiple files at a time. The "SEND" button will deliver the encrypted message to the receiver's email address. Moreover, the "CLEAR" button will clear all the fields, and the "BACK" button will return the user to the EFTS main window and the "CLOSE" button will allow the user to exit the system.

Figure 8 shows the code snippet of the Encryption code used to encrypt the files. The encryption is present inside a “try” block which is used to display all the possible outcomes when the selected files are encrypted. The type of encryption used in the code is the AES encryption and SHA 256 algorithm. The encryption gets the password input and use it as an encryption key on the files. Inside the “try” block if encryption condition is not met it displays an error message based on the condition that was not met.

```
def encrypt_file(pass_word,):
    try:
        temp_key = pass_word.get()
        temp_key = ''.join(e for e in temp_key if e.isalnum())
        key = temp_key + ("s" * (43 - len(temp_key)) + "=")

        fernet = Fernet(key)

        with open(browse_files.filename, 'rb') as file:  original = file.read()
        encrypted = fernet.encrypt(original)

    except AttributeError:
        messagebox.showinfo ("Encryption File Transfer System","Please Select A File")
    if temp_key == "":
            messagebox.showinfo ("Encryption File Transfer System"," Plese Enter Password")
    else:

        with open(browse_files.filename, 'wb') as encrypted_file: encrypted_file.write(encrypted)
        messagebox.showinfo("Encryption File Transfer System","File Encrypted Successfully")
    password.set("")
```

Figure 8: Encryption code in EFTS

The decryption function takes the encrypted file and it decrypt and restore to its original state. To get the encrypted file back to its original state, the password entered must match the password used during the encryption otherwise the data will still be inaccessible. Figure 9 depicts how the decryption function will be given a string containing the encryption key's path and where the decrypted file would be saved.

```
def decrypt_file(pass_word):
    try:
        temp_key = pass_word.get()
        temp_key = ''.join(e for e in temp_key if e.isalnum())
        key = temp_key + ("s" * (43 - len(temp_key)) + "=")

        fernet = Fernet(key)

        with open(browse_files.filename , 'rb') as enc_file:  encrypted = enc_file.read()
        decrypted = fernet.decrypt(encrypted)
    except AttributeError:
        messagebox.showinfo ("Encryption File Transfer System","Please Select An Encrypted File")
    if temp_key == "":
        messagebox.showinfo ("Encryption File Transfer System"," Plese Enter Password")
    else:

        with open(browse_files.filename, 'wb') as dec_file:  dec_file.write(decrypted)
        messagebox.showinfo ("Encryption File Transfer System","File Decrypted Successfully")
    password.set("")
```

Figure 9: Decryption code in EFTS

Figure 10 depicts the code for sending encrypted files through email. It accepts user input (email, password, receiver email, subject, body) and allows the user to send multiple attachments with the email. It supports the transmission of all file types, including text, jpg, mp3, and mp4. An error message is displayed if something goes wrong. But when all the conditions are met, the email is sent to the recipient via SMTP with Transport Layer Security.

```
def send():
    try:
        Message  = EmailMessage()
        username = Email.get()
        password = Password.get()
        to       = Receiver.get()
        subject  = Subject.get()
        body     = Body.get()
        Message['Subject'] = subject
        Message['From'] = username
        Message['To'] = to
        Message.set_content(body)
        for filename in attachments:
            filetype = filename.split('.')
            filetype = filetype[1]
            if  filetype == "":
                import imghdr
                with open(filename, 'rb') as f:
                    file_data = f.read()
                    image_type = imghdr.what(filename)
                Message.add_attachment(file_data, maintype='image', subtype=image_type, filename=f.name)
            else:
                with open(filename, 'rb') as f:
                    file_data = f.read()
                Message.add_attachment(file_data, maintype='application', subtype='octet-stream', filename=f.name)
        if username=="" or password=="" or to=="" or subject=="" or body=="":
            outcome_msg.config(text="All fields required", fg="red")
            return
        else:
            server   = smtplib.SMTP('smtp.gmail.com',587)
            server.starttls()
            server.login(username, password)
            server.send_message(Message)
            outcome_msg.config(text="Email has been sent successfully", fg="green")
    except:
        outcome_msg.config(text="Error sending email", fg="red")
```

Figure 10: File Transfer code in EFTS

EFTS is also equipped with password recovery option. In Figure 11, the code is used to send a copy of the password to the user email address if the user forgets the password, so it functions as a password recovery feature. This function communicates with the SMTP server. The communication is transmitted over email and is protected by Transport Layer Security (TLS). An email is written in the code to be used for sending out the passwords for the recipients' email so that they will automatically receive the email following encryption.

```
    # Send Password to Manger Email
    server = smtplib.SMTP('smtp.gmail.com', 587)
    server.ehlo()
    server.starttls()
    server.ehlo()

    with open('PasS.txt', 'r') as PasS:
     PasS= PasS.read()
    server.login('efts.recovery@gmail.com', PasS)

    subject = "Encrypted File Password !"
    msg = f"subject: {subject} \n\n\n Hello Mr.Omar Mahdi\n The password for your encrypted File is:{temp_key}
    "\n\n\n\n Thank you for using EFTS"

    server.sendmail(
        'efts.recovery@gmail.com',
        'fypomar1@gmail.com',
        msg
    )
```

Figure 11: Password Recovery Option in EFTS

5. User Acceptance Testing

As part of the software development lifecycle, testing is extremely important. This is due to the nature of testing, which allows the developer to fix the bugs or errors in the system before getting it launched. These as test strategies are required to track and demonstrate if system features are working properly or not. A bug-free system gives users the best possible experience and efficiency, allowing them to use the system's functions without issue. The Unit Testing and User Acceptance

Testing (UAT) has been conducted and results are shown in Figure 12 – 14 below.

Function	Encryption				
Case ID	Description	Test Condition	Expected Result	Actual Result	Status (Pass/Fail)
1.1	Click on the Encrypt button after choosing a file and typing in the password	Browsing a file: File exist Password Field: Password exist	Message "File Encrypted Successfully"	Message "File Encrypted Successfully"	Pass
1.2	Click on the Encrypt button when a file exists while password field is empty.	Browsing a file: File exist Password Field: Empty	Message "Enter a password"	Message "Enter a password"	Pass
1.3	Click on the Encrypt button without choosing a file while a password is written	Browsing a file: No file is selected Password Field: Password exist	Message "Please select a File"	Message "Please select a File"	Pass
1.4	Click on the Encrypt button without selecting a file or typing in a password	Browsing a file: No file is selected Password Field: Empty	Message "Please select a File" and "Please Enter a Password"	Message "Please select a File" and "Please Enter a Password"	Pass

Figure 12: EFTS Unit Testing – Encryption

Function	Decryption				
Case ID	Description	Test Condition	Expected Result	Actual Result	Status (Pass/Fail)
2.1	Click on the Decrypt button after choosing the encrypted file and typing in the correct password (same password used in encryption)	Browsing a file: Encrypted File exist Password Field: Correct password exist	Message "File Decrypted Successfully"	Message "File Decrypted Successfully"	Pass
2.2	Click on the Decrypt button when the encrypted file exists while password field is empty, or the password is wrong.	Browsing a file: File exist Password Field: Empty or wrong password	Message "Wrong Password"	Message "Wrong Password"	Pass
2.3	Click on the Decrypt button without choosing any encrypted file while the password is filled	Browsing a file: No file is selected Password Field: Password exist	Message "Please Select An encrypted file"	Message "Please Select An encrypted file"	Pass
2.4	Click on the Decrypt button without selecting a file or typing in a password	Browsing a file: No file is selected Password Field: Empty	Message "Please select An Encrypted File" and "Please Enter a Password"	Message "Please select An Encrypted File" and "Please Enter a Password"	Pass

Figure 13: EFTS Unit Testing – Decryption

Function	File Transfer				
Case ID	Description	Test Condition	Expected Result	Actual Result	Status (Pass/Fail)
3.1	Enter the email address, password, specified email address, subject, and body of the message	If any of the fields are blank	Message "All fields required"	Message "All fields required"	Pass
3.2	Filling out all the blanks, but either one of the user credentials or both are incorrect.	If all fields are entered but any of the Email or password or both are incorrect	Message "Error sending email"	Message "Error sending email"	Pass
3.3	Click on Browse and selecting a file	Browse a File: File is selected	Message "File attached"	Message "File attached"	Pass
3.4	Enter the email address, password, specified email address, subject, and body of the message	If all fields are entered and the email and password are correct	Message "Email has been sent successfully"	Message "Email has been sent successfully"	Pass

Figure 14: EFTS Unit Testing – Decryption

Results of the UAT are displayed in Figure 15 - 19 below.

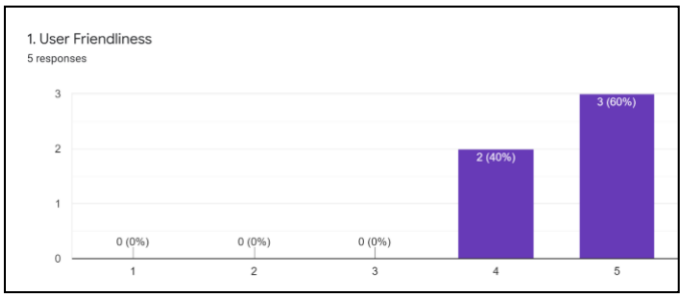

Figure 15: EFTS UAT – User Friendliness

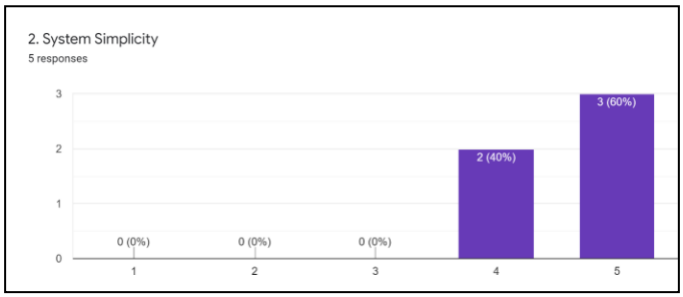

Figure 16: EFTS UAT – System Simplicity

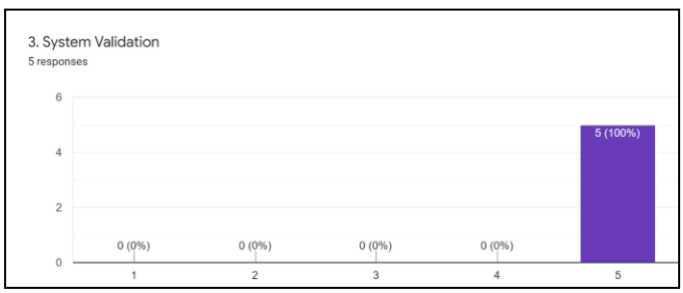

Figure 17: EFTS UAT – System Validation

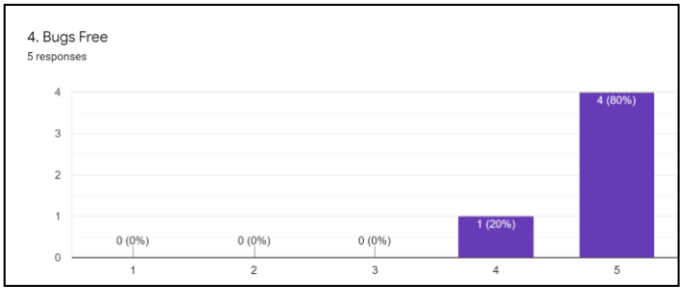

Figure 10: EFTS UAT –Bugs Free

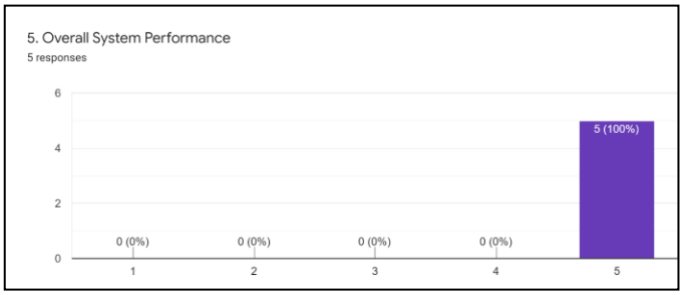

Figure 11: EFTS UAT – Overall Performance

Unit tests and user acceptance tests (UAT) were conducted with thorough preparation to acquire a deeper understanding of the application. To verify that the program performs as expected and is free of any inadvertent malfunctions, all the program's components have been tested sequentially. In addition, the developer conducted a demonstration of the system and gave the users a chance to test it. According to the graphs of user feedback, the majority of testers find the system user-friendly and very simple to use. Furthermore, the system was rated bug-free by 80% of the testers, and the system validation and overall system performance were both rated "Excellent" by all the testers, indicating that they had a smooth experience using the system. However, the developer may gain a better understanding of the consumers' needs and expectations because of the feedback they provide during the test. Based on user feedback, the developer will attempt to improve the overall system functionalities for future enhancement.

Conclusion

EFTS is an encryption file transfer system that was created to give people confidence when transferring files over the internet without worrying about them going to the wrong person by implementing encryption and using Transport Layer Security (TLS) when sending files via email. As of now, the system has resolved the issues raised and made it possible for users with limited computer knowledge to encrypt and transfer their files while maintaining file confidentiality. even though the current version is working properly, the developer is planning to add more features to the system in the future. The developer intends to include a feature that will allow users to hide their files within another using steganography. The developer also plans to add a feature that will verify the user by sending an SMS message to their phones and a face recognition feature to verify the receiver by sending their pics to the system database. Only the person with a similar face will be able to access the encrypted file.

References

[1] Abrams, L. (2018, November 8). Microsoft Releases Info on Protecting BitLocker From DMA Attacks. Retrieved from bleepingcomputer: https://www.bleepingcomputer.com/news/security/microsoft-releases-info-on-protecting-bitlocker-from-dma-attacks/

[2] comparecamp. (2021). What is AxCrypt ? Retrieved from comparecamp: https://comparecamp.com/axcrypt-review-pricing-pros-cons-features/

[3] Daniel, B. (2021, May 4). Symmetric vs. Asymmetric Encryption: What's the Difference? Retrieved from Trenton SYstems: https://www.trentonsystems.com/blog/symmetric-vs-asymmetric-encryption

[4] encryptionconsulting. (2021). History of Cryptography. Retrieved from encryptionconsulting: https://www.encryptionconsulting.com/education-center/what-is-cryptography/

[5] Ivan. (2021, February 20). Proteger archivos con contraseña con AxCrypt. Retrieved from ivanandrei: https://www.ivanandrei.com/proteger-archivos-con-contrasena-con-axcrypt/

[6] Microsoft. (2021). BitLocker. Retrieved from Microsoft.com: https://docs.microsoft.com/en-us/windows/security/information-protection/bitlocker/bitlocker-overview#bitlocker-overview.

[7] Pedamkar, P. (2022). What is Visual Studio Code? Retrieved from educba: https://www.educba.com/what-is-visual-studio-code/

[8] Smart Eye Technology. (2020, October 5). What is an Information Security Policy? Retrieved from Smart Eye Technology: https://getsmarteye.com/confidentiality-integrity-availability-basics-of-information-security/

[9] Zaw, H. K. (2020, October 8). How encryption algorithms provide confidentiality. Retrieved from DEV: https://dev.to/heinkhantzaw/how-encryption-algorithms-provide-confidentiality-4j1b