This Report is Submitted to

The College of Health, Engineering, and Science

Slippery Rock University

In Partial Fulfillment

of the Requirements for the Degree of Bachelor’s in Computing

By:

Richard Schwanbeck

Justin Conklin

Ben Scheirer

A report submitted to the College of Health, Engineering, and Science. Slippery Rock University, in partial fulfillment of the requirements for the Degree of Bachelor of Computing by:

The report is hereby approved by:

Nitin Sukhija, Ph.D.

Assistant Professor, Department of Computer Science

Slippery Rock University of Pennsylvania

**Table of Contents**

Abstract. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4

1. Introduction. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4

2. Requirements. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4

3. Use Cases. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4

3.1 Encrypt File(s). . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4

3.2 Sending Encrypted File(s) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

3.3 Receive Encrypted File(s). . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6

3.4 Decrypt File(s). . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7

3.5 General Use Case Diagram. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8

4. Sequence Diagrams. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8

4.1 Encrypting File(s). . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9

4.2 Decrypting File(s). . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10

4.3 Sending File(s). . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11

4.4 Receiving File(s). . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12

4.5 Selecting Encryption Type. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13

5. Data Flow Diagram. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14

5.1 Context Level. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 16

5.2 Level – 0. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 17

5.3 Level - 1. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18

6. User Manual. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 19

7. Technical Manual. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 21

7.1 Encryption. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 21

7.2 Decryption . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 22

7.3 Sending/Receiving . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 23

**Abstract**

The main ambition of this software was to have a packaged set of encryption methods available to the user to perform secure encrypted file sending from one user to another. This was to answer the problem of sending confidential or identifying information securely from one party to another. The encryptions accomplished are AES and RSA. Multiple file encryption, decryption, sending of and receiving of was achieved for AES. Single file encryption, decryption, sending of and receiving of was achieved for RSA. The key result was a secure system in which users were able to send and receive messages containing files though a secure one-way connection using .NET provided algorithms and methods.

**1. Introduction**

This report will cover; the requirements that were needed to develop the software, the diagrams and methods used, and the technical and evaluation manuals to provide clarity of the use and function of the software.

**2. Requirements**

To develop the software, we used Microsoft’s Visual Studio IDE. Within Visual Studio, our software is coded in the C#/ .NET language. We used Github for version control of our project and to work in parallel. Since this was developed using primarily Microsoft’s workspace, a computer with a recent version of Microsoft Windows is needed to develop or run the software. Also, a network connection is required to send and receive encrypted files.

**3. Use Cases**

**3.1 Encrypt File(s):**

Actors: User, User’s Machine, Encryption System

Inputs: Non-encrypted File(s), Save Location

Outputs: Encrypted File(s)

Normal operation:

The user opens the software. They select the Encryption Tab. They then click “Browse…” and select files from the file system on their machine. They then select the method they would like to use; in this case it will be “Save Locally”. They click the corresponding radio button and are given another file system browser to select a save location for their files. Once the user selects a save location they may now hit Accept. The files, if they are of the correct format, will be sent through the encryption system. A dialog box will pop up if the files are encrypted successfully. The files should now be saved to the location chosen by the user.

Exception:

* Wrong file format. User is prompted to choose files of the correct format.
* Encryption failed. User is shown an error message and prompted to attempt again.
* Do not have access to folder. If the user selects a folder which cannot be saved to, the software will detect this and prompt the user to save to another folder.

**3.2 Sending Encrypted File(s):**

Actors: User, Users Machine, Encryption System

Inputs: Destination IP address, TCP Port Number, Encrypted File(s)

Outputs: Transaction Details Dialog

Normal Operation: The user opens the software. They select the Encryption Tab. They then click “Browse…” and select files from the file system on their machine. They then select the method they would like to use; in this case it will be “Send Files”. Once the radio button is clicked the textbox where they will enter the IP address and port number will be made available. The user enters where they want to send the file. If the port number and IP address are valid, the file is sent and a message with the transaction details is shown.

Exception:

* Wrong file format. User is prompted to choose files of the correct format.
* Connection with target machine failed. Users may attempt to resolve or contact system administrator to solve.

**3.3 Receive Encrypted File(s):**

Actors: User, Users Machine, Encryption System

Inputs: Save Location

Outputs: Encrypted File(s) From Sender Machine, Transaction Details Dialog

Normal operation:

The user opens the software. They click “Search for Connections” and select from the potential machines attempting to connect to transmit files. Once the user ensures that they have found the correct machine they wish to receive files from, they will select it. Once selected and confirmed through a checkbox, the machines will open a connection. The software will then do all tasks required for secure encryption communication and then will receive the encrypted files and place them on the user’s machine. By default, the files will be placed in the downloads folder once received. The user can specify this to be any folder with a setting in the software.

Exception:

* Wrong file format. User is prompted to choose files of the correct format.
* Connection with target machine failed. Users may attempt to resolve or contact system administrator to solve.
* Decryption failed. User is prompted with an error message and asked to repeat the process again.
* Do not have access to folder. If the user selects a folder which cannot be saved to, the software will detect this and prompt the user to save to another folder.
* Timeout searching for connections. If there are no connections within an allotted amount of time, the software will return to main menu after a popup.

**3.4 Decrypt File(s):**

Actors: User, User’s Machine, Encryption System

Inputs: Encrypted File(s), Save Location

Outputs: Decrypted File(s)

Normal operation:

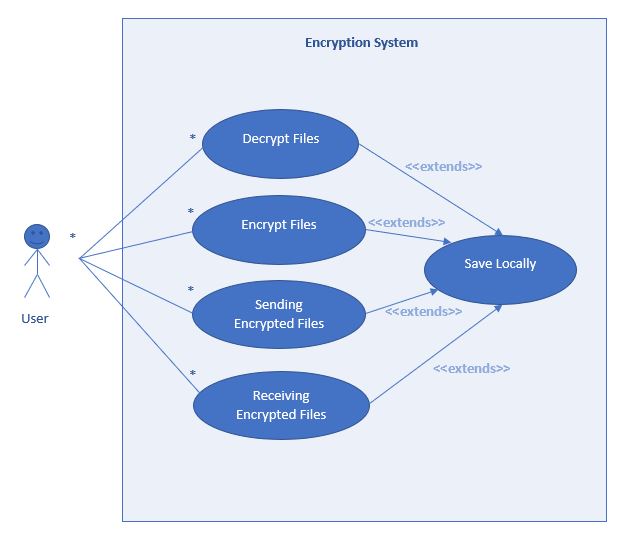
The user opens the software. They click the Decryption Tab. They then click “Browse…” and select the encrypted files from their save location. If the files are of the correct format, the user will be able to initiate decryption through clicking “Decrypt”. If the decryption is successful a dialog box will pop up showing the decrypted files. The decrypted files will be saved in the original save location.

Exception:

* Wrong file format. User is prompted to choose files of the correct format.
* Decryption failed. User is shown an error message and prompted to attempt again.
* Do not have access to folder. If the user selects a folder which cannot be saved to, the software will detect this and prompt the user to save to another folder.

**3.5 General Use Case Diagram**

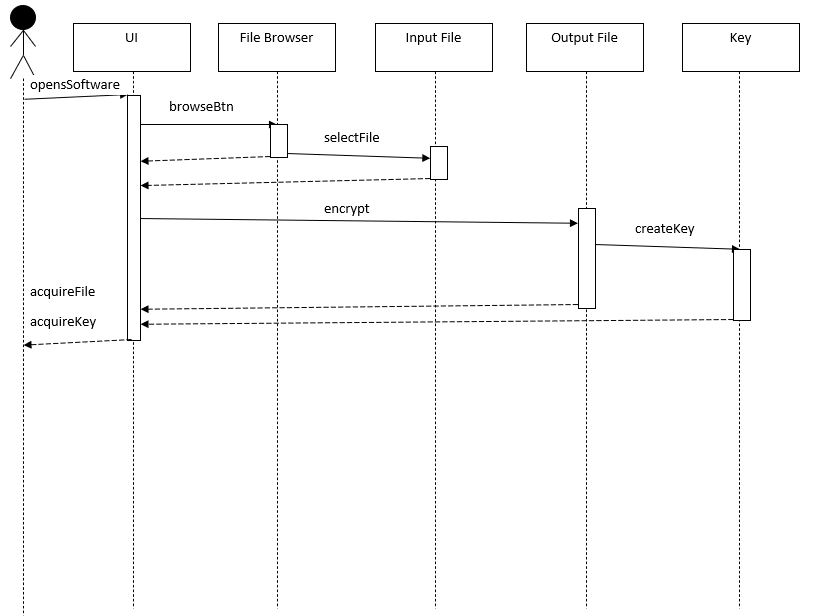
This diagram demonstrates the uses provided by the software to the user.



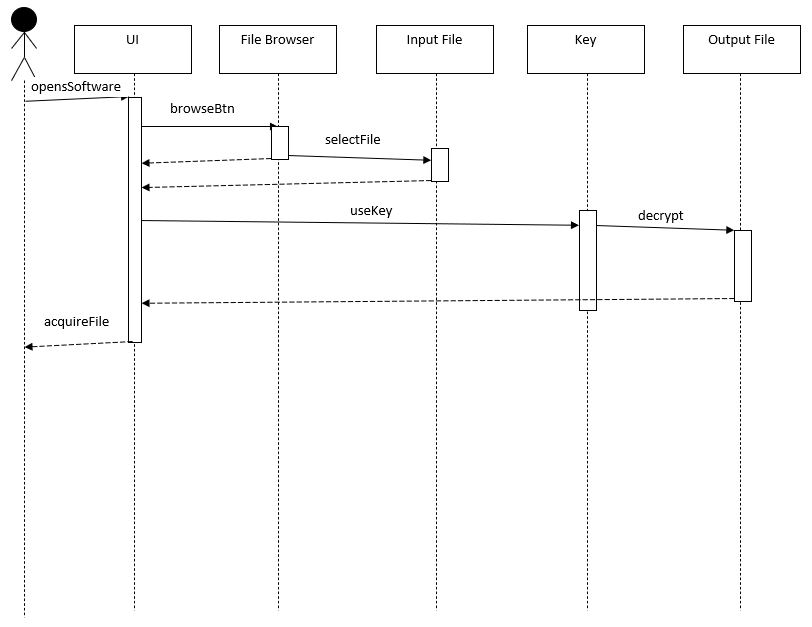
**4. Sequence Diagrams**

These sequence diagrams portray the order of actions taking by the typical user to perform the main functions of the software including encrypting a file, decrypting a file, the sending and receiving of files, and the selection of encryption type.

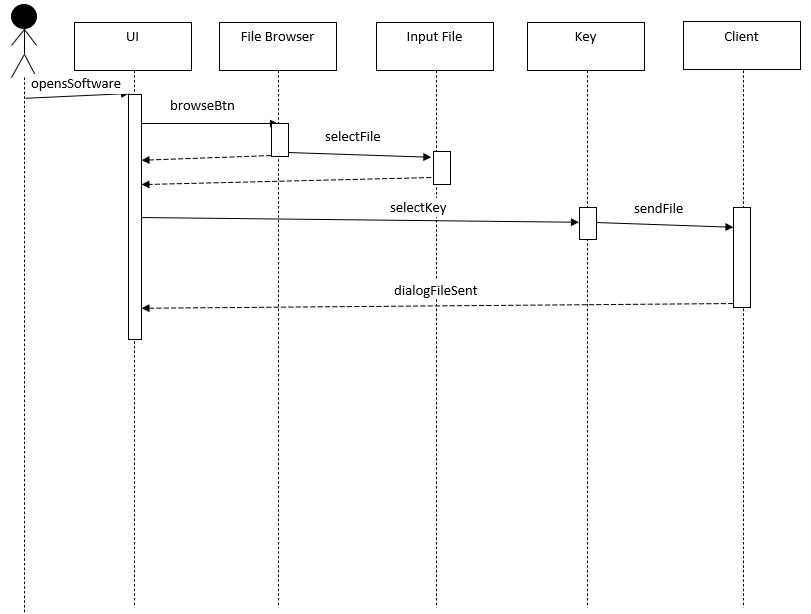
**4.1 Encrypting File**



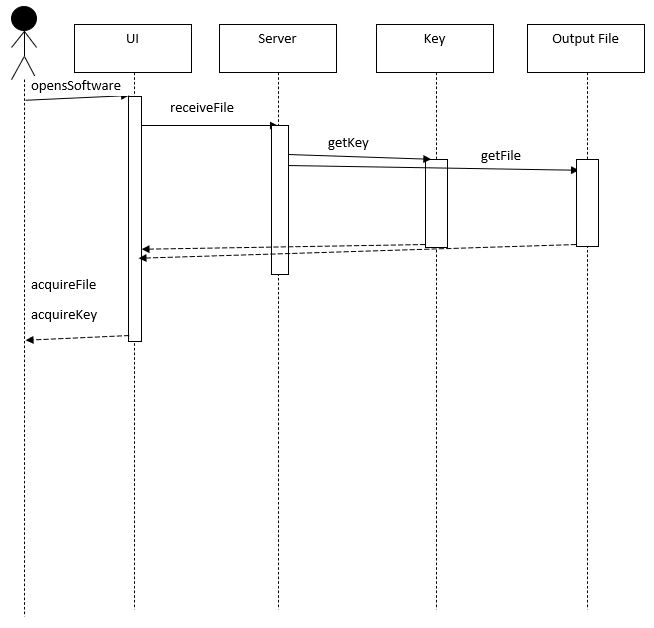
**4.2 Decrypting File**



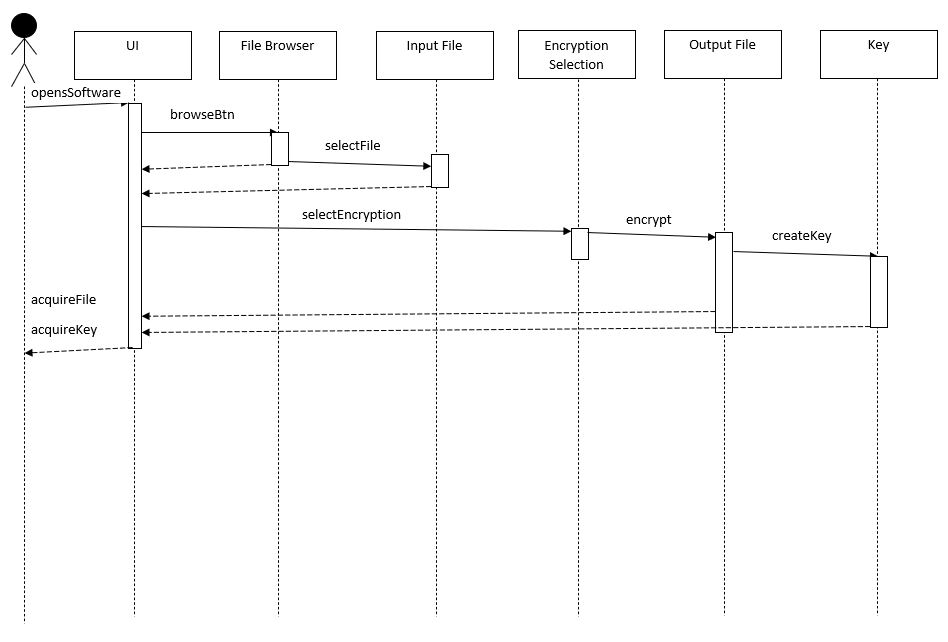
**4.3 Sending File**



**4.4 Receiving File**



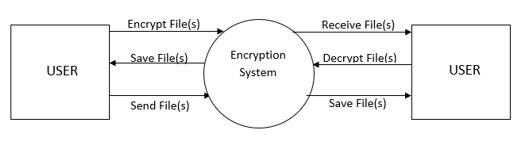
**4.5 Selecting Encryption Type**



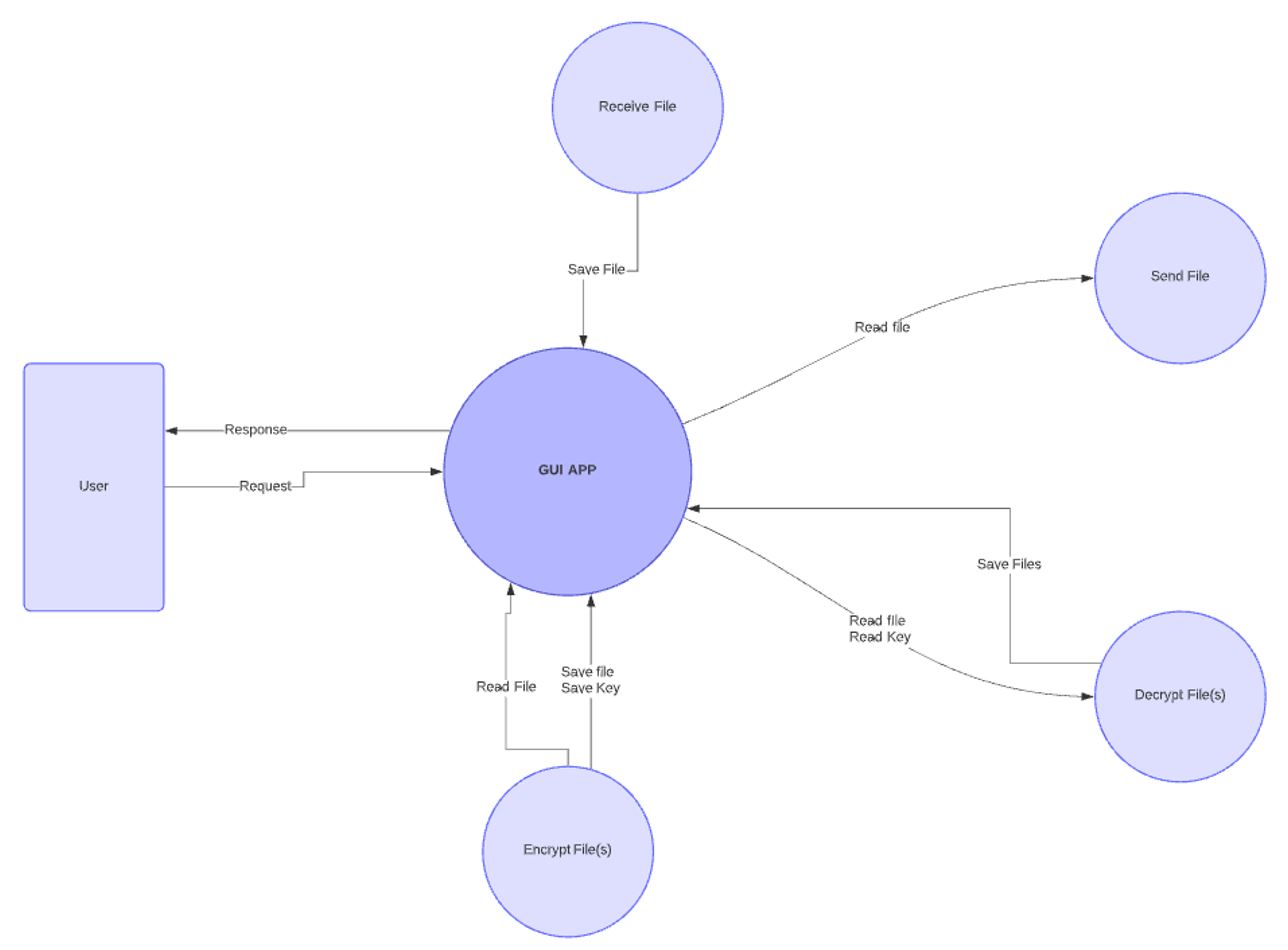
**5. Data Flow Diagrams**

Although the software does send and receive data, there are no instances of data being saved and controlled by the software. The flow of data is represented by these diagrams.

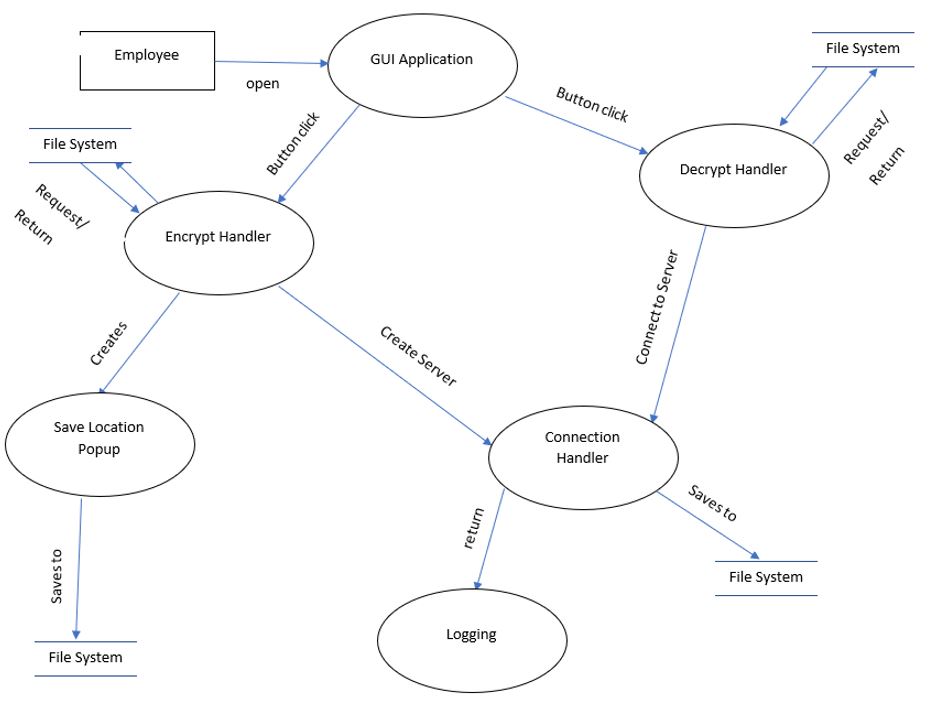
**5.1 Context Level**



**5.2 Level – 0**

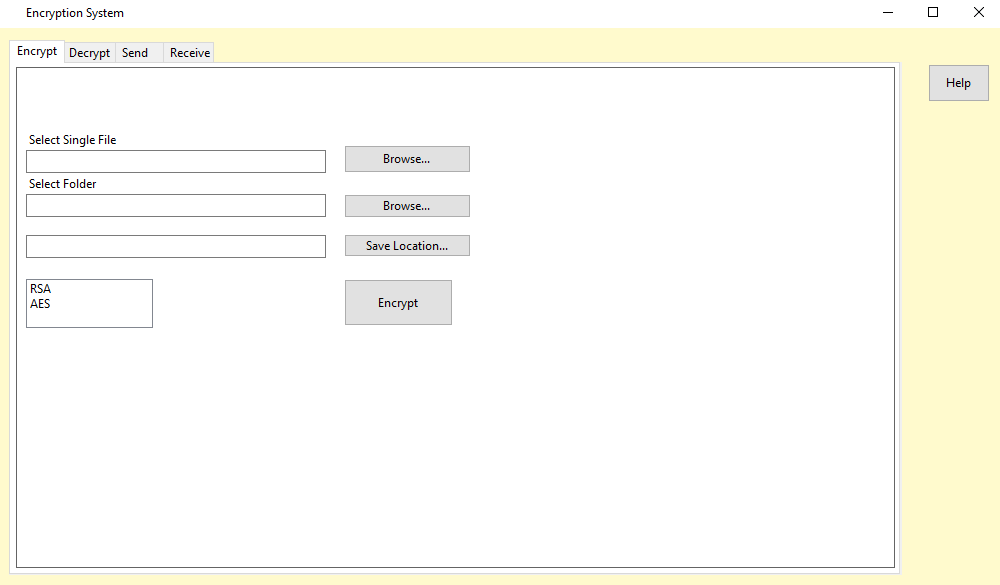


**5.3 Level – 1**

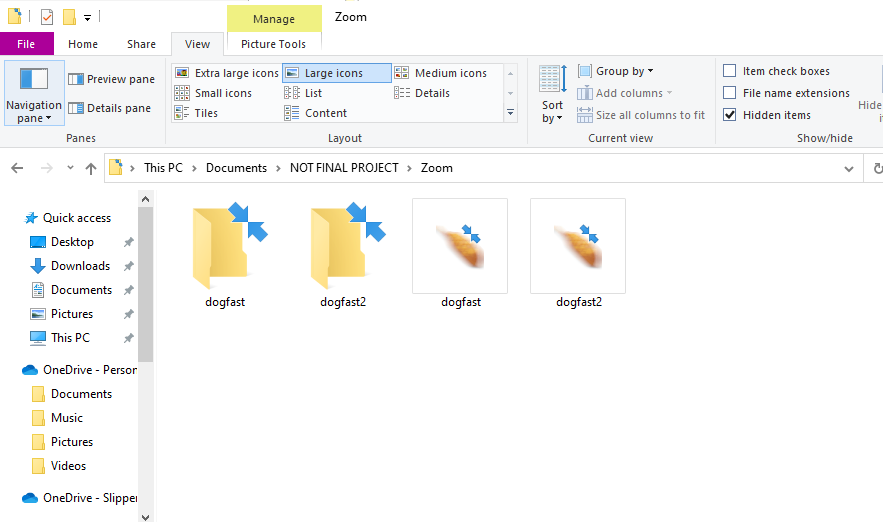


**6.0 User Manual**

Our software interacts with the user primarily through the GUI we have developed. Below is a picture of the GUI.



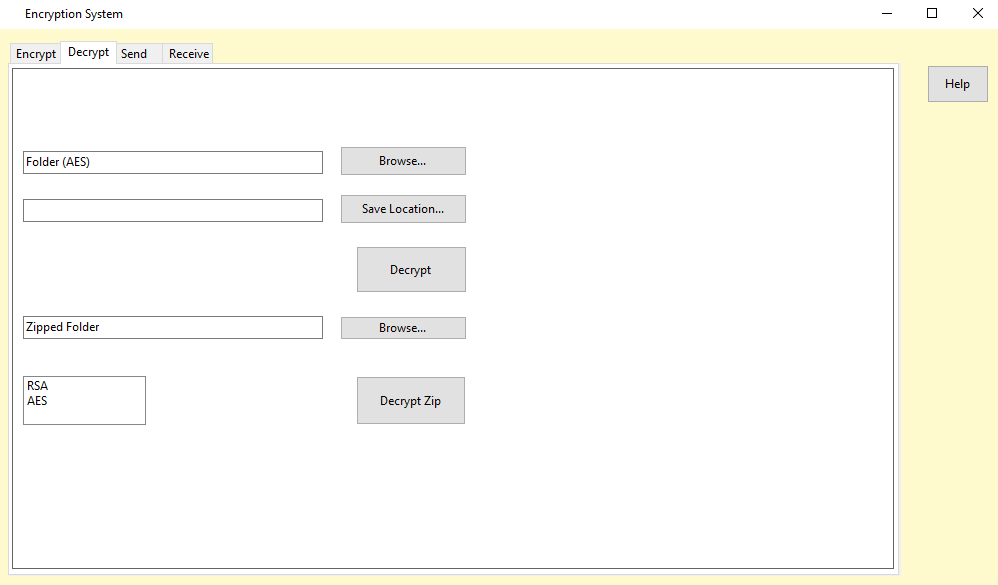
As shown, there are a multitude of buttons, text boxes and tabs to select from. We will begin with detailing what each tab exists to do. There are four tabs which are the following: Encrypt, Decrypt, Send and Receive. The Encrypt tab is where all of the encrypting functionality of the program is held. Decrypting is done within the decrypt tab, and encrypting and decrypting can be done either for singular files, or for entire folders. Whenever the file is either encrypted or decrypted, it is stored on the local machine for a later date or to be sent to another machine. The Send and Receive tabs are where the software handles these file transfers. Now that the tabs have been explained, we will discuss how to encrypt a file. To begin encrypting a file, navigate to the Encrypt tab and click on the browse button that is next to the Select Single File text box. This will open an instance of the machine’s File Explorer application.



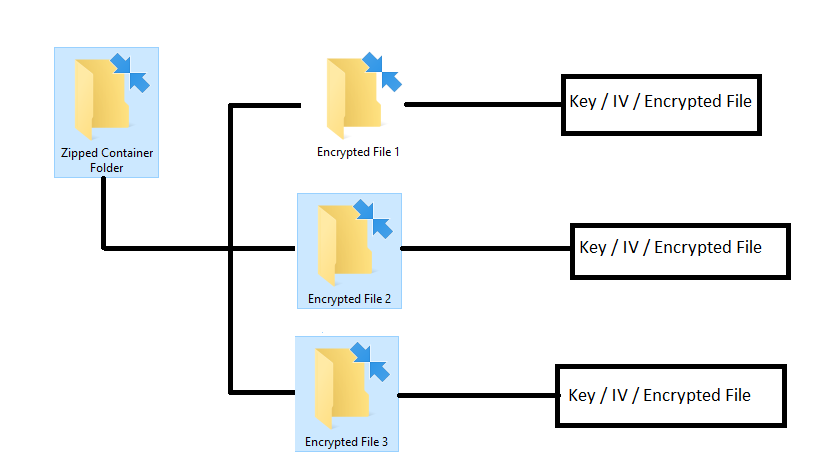
For example, if we wanted to encrypt the dogfast.jpg file, we can select it from the File Explorer window and this will bring the software back to the home window. Then, we can select a save location for the file after it is encrypted. This follows the same procedure as selecting a file, but instead of choosing a file, you are choosing a folder. After a save location and a file are selected, then an encryption method must be chosen. An encryption method is chosen by selection from the box shown here:



In the current version of the software, there are two options available for encryption. AES, and RSA. For more detailed descriptions on how these algorithms work and are used within the program, see the technical manual. After an encryption method is selected, then the Encrypt button can be clicked to encrypt the file. Then the file is encrypted and stored at the selected save location. To Encrypt a folder, the process is largely the same as encrypting a single file with the exception of selecting a folder instead of a file. Once a file or folder is encrypted, then to decrypt it, navigate to the Decrypt tab.



There are two options to decrypt a file/ folder. The first is to select a not encrypted folder containing a file and the key(s) required to decrypt the file. To begin, select an Encryption method from the choice box at the bottom of the panel. Clicking on the browse button uses the machine’s File Explorer to select the folder containing the files needed for decryption and to select a folder for a Save Location. The decrypted file is then stored at the Save Location selected. The process is exactly the same for selecting a zipped folder, but it needs to be noted that the encrypted file still needs to have folders containing individual file and key pairs to work properly. This is to ensure that the correct file and key are used together. Below is a demonstration of the correct format.

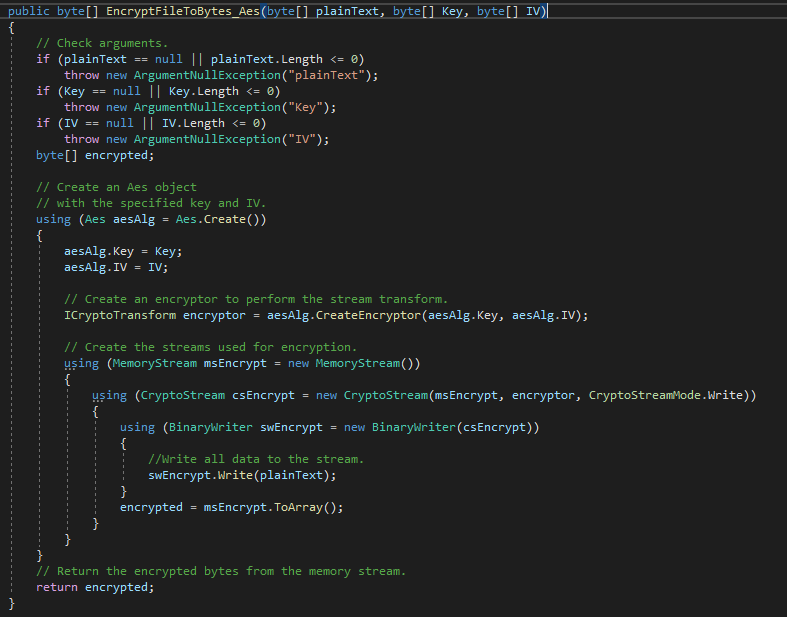


To send files to other machines, navigate to the Send tab. In the current implantation of the software, the sending and receiving requires communication between the operators on sending and receiving. This is because the program is not set up to create a server to wait on requests. This is circumvented by simply having a timeout timer attached to sending. Following from this, the software requires communication between two people on both ends of sending and receiving to hit the button within the timeout timer. First, the sender needs to select a zipped folder or file to send to the recipient. Then, the receiver selects a save location for the file or folder once it is received. The sender needs to input the recipient’s IP address into the text box in the Send tab before they hit the send button, and the receiver needs to put their local IP address in the text box in the Receive tab. Then after the sender hits the send button, the receiver hits the receive button within the timeout timer, and then the recipient receives the file at the save location specified.

**7. Technical Manual**

* 1. **Encrypting file(s)**

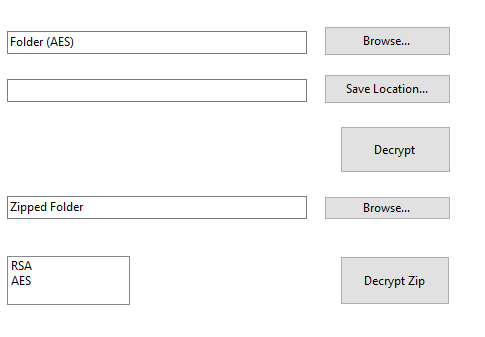
Encrypting files in the program can be done in two different ways AES or RSA. A file must be selected to encrypt and a type also must be selected. Files are selected through the file select button that brings up the windows explorer. The type of encryption is done through a choice box where you simply click which form of encryption you want to use. The encrypt button checks the choice box’s value when clicked and proceeds with the correct encryption form. The C# built in cryptography services from the System.Security.Cryptography is used for both forms of encryption.Encrypting files using AES is done with the EncryptFileToBytes\_Aes function in the Encryption class. To encrypt a file it must be passed into the function in the form of a byte array. This is done by using the FileToBytes method in the EncryptionSystemFrom.cs. As is shown in the function below, a key and an initialization vector (IV) are also needed to encrypt using AES. Those will be generated when the Aes object is created as seen below. An encryptor is then created to perform the stream transform. The file then gets encrypted. After the file is encrypted, it, the IV, and the key are written out into text files so the file can be decrypted in the future in the EncrpytAESfile function.



Encrypting files using RSA is done through the RSAEncryption function in the Encryption class. Like the AES function the file that is passed is in the form of a byte array. The function starts out by creating a new instance of the RSACryptoServiceProvider or csp with 2048 bit RSA keys, one private key and one public key. The public key then is retrieved and used to encrypt the file. A new instance of the RSACryptoServiceProvider is then created, the file then gets encrypted, and in the function that calls RSA encryption, encryptBtn\_Click, the file and private key are written out to files so they can be used for decryption later.

* 1. **Decrypting files**

Like encryption, decryption in this program is done in two different ways, AES and RSA. Decryption is done in the decryption class through a few different methods that are all called from the EncryptionSystemFrom class through button click functions. To decrypt you must first zip the necessary files into a folder within another folder. Meaning for AES zip the file, key, and IV together. And for RSA the file and key are zipped together. AES decryption is handled in the DecryptFileFromBytes\_Aes function. It works in the same way that the AES encryption function works. The zip file is decompressed to get the file, key and, and IV. They are then passed through as parameters, a decryptor is created to perform the stream transform, and then the file is decrypted using the key and IV and written out to a new file. RSA decryption is done through the RSADecryption function in the decryption class, but is called in the decryptBtn\_Click function in the EncryptionSystemFrom class. The RSADecryption function gets called with the file as a byte array and the key as parameters after the zip file is decompressed. Because the private key is just a string, it needs converting back to the type of RSAParameters. First, the function gets the string containing the key. It then  get stream from the string and deserializes it. Finally, the object is retrieved from the stream. The key is then imported to the csp and is used to decrypt the file.



Decryption can be done with folders as well as zip files. To decrypt a folder you choose the folder from the first browse button, choose a save location, and choose your method of encryption from the choice box and then hit the decrypt button. Either the DecryptFolderAES function or the DecryptFolderRSA function is then called to iterate through the folder decrypting each file with its respective key/IV the same way described above until all the files are decrypted.

* 1. **Sending/Receiving files**

Sending and receiving files are done with the Client and Listener classes. The Client class is used for sending files and has one function used to do so, Connect. To send a file you need the external IP address of the person receiving the file, which will be entered into a text box labeled “Receiver’s IP” and the file. The file and IP will be passed as parameters to the Connect function. The Connect function has a constant port of 13000 that is used in creating a Tcpclient used to send the file to another user. That client is created with the IP address of the user as well. It then gets a client stream for writing and reading. That stream is then used to send the file to the connected Tcpserver and the client and stream are closed.

The Receive function within the Listener class is used for the reception of files from another user. The local IP address and the location the file will be saved are passed into it as parameters. The same constant port 13000 is used so the transfer can happen. The local IP address and port are used to set up a receiving connection using the Tcplistener class from the C# provided namespace System.Net.Sockets. That connection then starts listening for client requests. After that the listening loop starts with the pending connection request being accepted. Then, a stream object is created for reading and writing. Finally, a loop is run to receive all data that was sent by the client.