Applied Information Technology And Computer Science Vol. 3 No. 2 (2022) 050-067
© Universiti Tun Hussein Onn Malaysia Publisher's Office

AITCS

Homepage: http://publisher.uthm.edu.my/periodicals/index.php/aitcs
e-ISSN :2773-5141

Secure Document Fragmentation Tool with Two Layers of Encryptions: SecDoc

Armisha Hazrin Fitri Abdul Halim[1], **Chuah Chai Wen**[1*],

[1]Faculty of Computer Science and Information Technology,
Universiti Tun Hussein Onn Malaysia, Parit Raja, Batu Pahat, 86400, MALAYSIA

DOI: https://doi.org/10.30880/aitcs.2022.03.02.004
Received 11 August 2022; Accepted 06 October 2022; Available online 30 November 2022

Abstract: Electronic document sharing through an email attachment is the main communication platform in a corporate setting. As it is a convenient way to share information. However, sharing documents through email attachments required security. This is because of the fact that email is never intended to be a secure channel for sending information. There are various types of email attacks. Attackers may intercept the network traffic and steal the data. Confidentiality of sensitive information can be abused. As a result, sensitive data or information should be protected before it is sent in an email as an attachment. Thus, the electronic document must be protected before it is transferred. In this project, fragmentation and hybrid encryption tool are proposed to provide additional security to the electronic document. Fragmentation is used to break off a document into a fragment that has no understandable meaning. Hybrid encryption is to encrypt the fragmented document for making it more secure. In a nutshell, the SecDoc assists users to protect sensitive data of their valuable documentation from illegal third-party users. The methodology chosen for this project is Object-Oriented Software Development (OOSD). The proposed tool is developed using Java programming language. The tool has five main modules. The tool allowed users to select, split, encrypt, decrypt, and join the desired documents before it is attached to an email. The key entered by the users is saved in a file and the key file is encrypted with the user's public key. Overall, the developed tool is able to protect the confidentiality of sensitive documents.

Keywords: Fragmentation, Encryption, Decryption, Document, Sensitive Data

1. Introduction

File sharing refers to the act of spreading or providing access to digitally recorded information [1]. For instance, computer programs, multimedia (audio, pictures, and video), documents, and electronic books. In a corporate setting, email remains the main official communication platform for sharing files [2]. Big profile companies, enterprises, and normal shops owners use email as a convenient way for file sharing in email attachments.

Sharing files through the internet required security. Communication between two users is not confidential through the internet. Companies consider online transmission and storage as feasible file

*Corresponding author: cwchuah@uthm.edu.my sharing because of its low cost. However, the third-party hosting model of online storage increases the risks of information leakage [2]. Thus, the confidentiality of a sensitive data can be abused. This shows how important it is to ensure the security of file routes. For instance, the movement of documents within the organization and the movement of documents in communication with the outside world.

Email is never intended to be secured. This is because when document is shared via email, this document may be visible to anyone with the access to the intermediate email server and to attackers who intercept the network traffic. The communication between two users through the internet is not confidential. Due to the advance technology nowadays, software has been developed to sniff packet and information such as sensitive data. For example, Yahoo said data from more than 1bn user accounts was compromised in August 2013, making it the largest such breach in history [3]. Yahoo said the stolen user account information may have included names, email addresses, telephone numbers, dates of birth, hashed passwords and, in some cases, encrypted or unencrypted security questions and answers [3].

Thus, this shown that it is very important to protect sensitive data. Any document attachment should be protected before it is transferred through the network or third party. As the solution for corporate setting that use electronic document for storing sensitive data, applying cryptography such as symmetric and asymmetric encryption and fragmentation is proposed in this project to protect the valued document.

The proposed tool uses the combination of fragmentation and encryptions to ensure the data protection and privacy of sensitive information for file sharing. The purpose system applies two cryptography mechanism. There are Advanced Encryption Standard (AES) encryption algorithm and Rivest-Shamir-Adleman (RSA) public key cryptosystem. In this system, the file to be protected is first split into different fragments. This is to minimize the information presented and make it unable to be read for human or even for machine. The resulting fragments then is encrypted with AES encryption. This is to protect the fragments in transit and at rest from being accessed by unauthorized users. Then the encryption key is encrypted with RSA public key. This is to ensure the security of the encryption key. Only person with the match private key may access the encryption key to decrypt the fragments file. All this mechanism is important to protect digital information.

2. Related Work

2.1 Asymmetric Cryptography

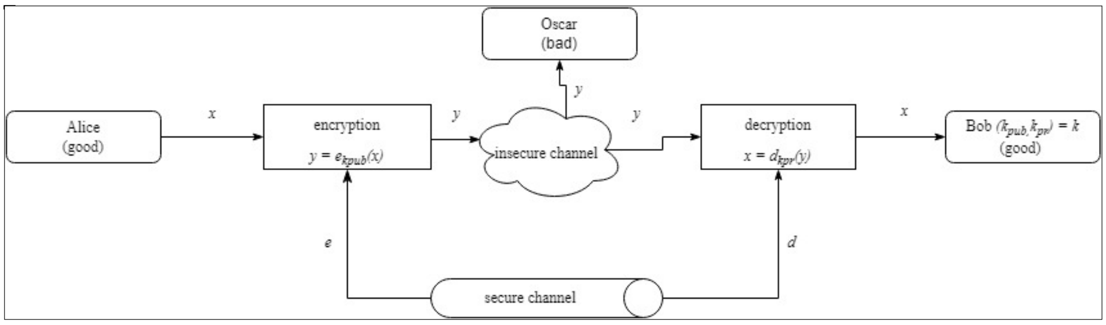

Oscar (bad)
y
Alice (good)
x
encryption $y = e_{k_{pub}}(x)$
y
insecure channel
y
decryption $x = d_{k_{pr}}(y)$
x
Bob $(k_{pub}, k_{pr}) = k$ (good)
e
d
secure channel

Figure 1: Basic Protocol for Public-Key Encryption [4]

Asymmetric cryptography is also known as public-key cryptography. It uses two keys for encryption and decryption process. The two keys are public key, k_{pub} and private key, k_{pr}. Public key is used for encryption process. Private key is used for decryption process. As shown in Figure1, Bob has two keys. Alice encrypts message, x using Bob's public key, $k_{pub.}$ The encrypted message is sent to Bob. Bob decrypts y using his private key, k_{pr}. This simple analogy is the basic protocol of public-key encryption.

RSA is an extensively used public key cryptosystem for secure data transmission. The abbreviation is derived from Ron Rivest's, Adi Shamir's and Leonard Adleman's surnames. An RSA user generates and distributes a public key consisting of two huge prime integers and an additional value. The prime numbers are a closely guarded secret. Anyone can encrypt a message using the public key, but only someone who knows the prime number can decode it.

A distinctive feature of all asymmetric schemes is that there is a set-up phase during which the public and private key are computed [4]. Depending on the public-key scheme, key generation can be quite complex. Below are the steps involve in computing key generation to generate public and private key for RSA cryptosystem [4]:

Output: public key: *kpub* = (*n,e*) and private key: *kpr* = (*d*)

1. Choose two large primes p and q.

2. Compute $n = p \cdot q$.

3. Compute $\Phi(n) = (p-1)(q-1)$.

4. Select the public exponent $e \in \{1, 2, \ldots, \Phi(n)-1\}$ such that $\gcd(e, \Phi(n)) = 1$.

5. Compute the private key d such that $d \cdot e \equiv \bmod \Phi(n)$

The condition $\gcd(e, \Phi(n)) = 1$ ensures the inverse of e exists modulo $\Phi(n)$, so that there is always a private key d [4]. Two large primes p and q are chosen in step one. Public and private key are computed in step four and five. RSA encryption and decryption can be seen as a modular exponentiation [4].

$$y = ekpub\ (x) \equiv xe \bmod n \text{ (encryption)}$$

$$x = dkpr\ (y) \equiv yd \bmod n \text{ (decryption)}$$

The exponents e and d are in general very large numbers. The exponents are in range of 1024 to 3072 bit or even larger. For encryption, after Alice obtain Bob's public key, she can send encrypted message y to Bob. Alice first turns the unpadded plaintext x to padded plaintext, so that $0 \leq x < n$. Alice then compute the ciphertext y using Bob's public key e. This can be done quickly using modular exponentiation. Alice then transmits y to Bob. For decryption, Bob recovers x from y by using his private key exponent d by computing $x = dkpr\ (y) \equiv yd \bmod n$. Bob can recover the original message x by reversing the padding scheme.

2.2 Symmetric Cryptography

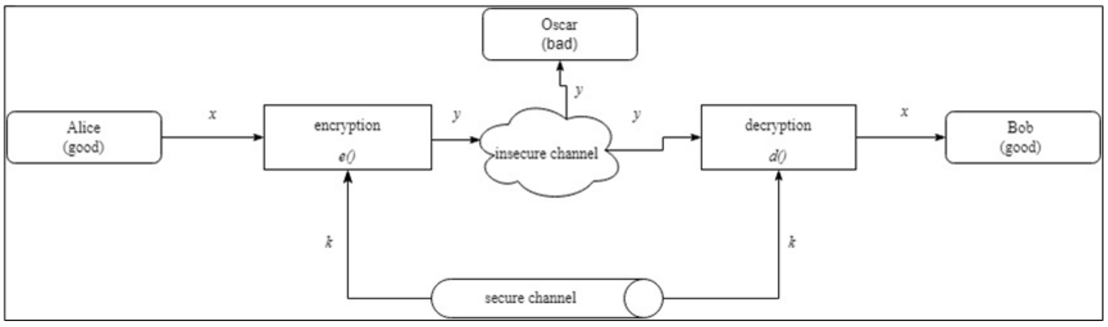


Figure 2: Symmetric key cryptography [4]

As shown in Figure 2, Alice and Bob communicate through an insecure channel. They use symmetric key cryptography to send message. Firstly, Alice encrypts message, x using key, k to produce ciphertext, y. Alice sends ciphertext, y through insecure channel to Bob. Bob decrypt the ciphertext, y using the same key, k to read the message. Oscar is a bad person who is trying to eavesdrop the communication.

Oscar may only see ciphertext, *y* as a random bit if the encryption algorithm is strong. Furthermore, without the key, *k* Oscar may not read the plaintext. The distribution of key between Alice and Bob must through the secure channel.

Symmetric key cryptography uses the same pair of secret keys for encryption and decryption process. Symmetric key cryptography is simpler and faster than asymmetric cryptography. This is because sender and receiver, both use the same secret key to do the encryption and decryption process. One of the symmetric key cryptographies is Advanced Encryption Standard (AES). AES has 128-bit data block. This data block length may avoid possible collision attack. The design structure of AES is iterated cipher, substitution-permutation, and byte-based. This design makes AES become more secure. The longer key length of AES makes exhaustive key search infeasible.

AES has number of 10 rounds for 128-bit master key, 12 rounds for 192-bit master key, and 14 rounds for 256-bit master key. The data block size is always 128-bit, but the key length may be 128, 192, or 256 bits. Therefore, AES has three version such as AES-128, AES-192, and AES-256. AES has four basic operations in each round. There are key addition layer, byte substitution layer, and diffusion layer that consists of shift rows and mix column layers [4]. Each round in AES consists of these four layers except for the first round. While the last round does not have mix column transformation.

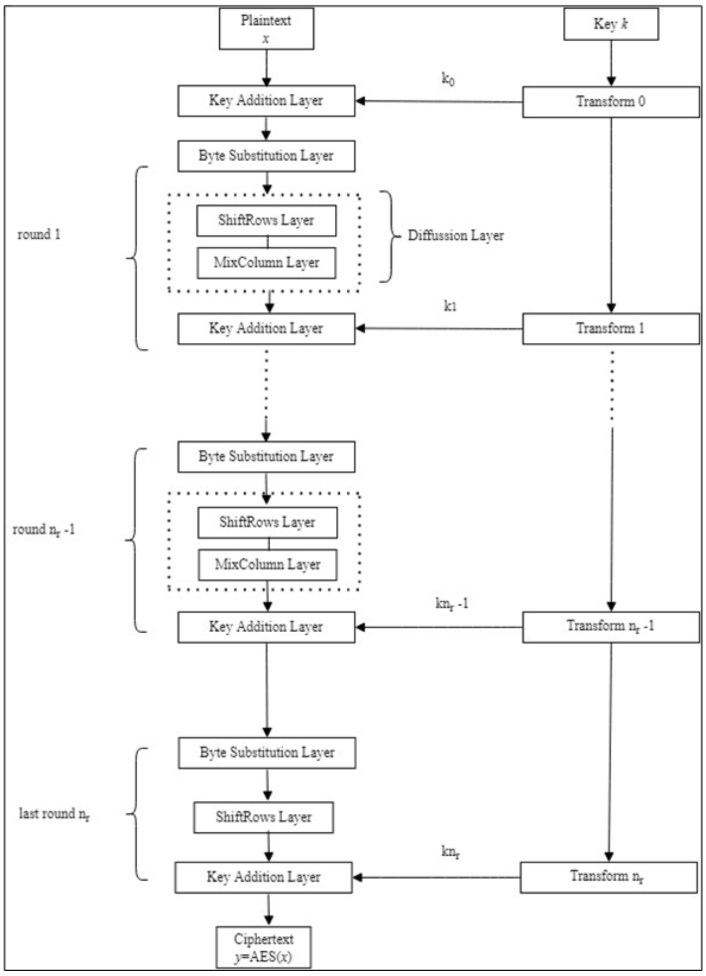
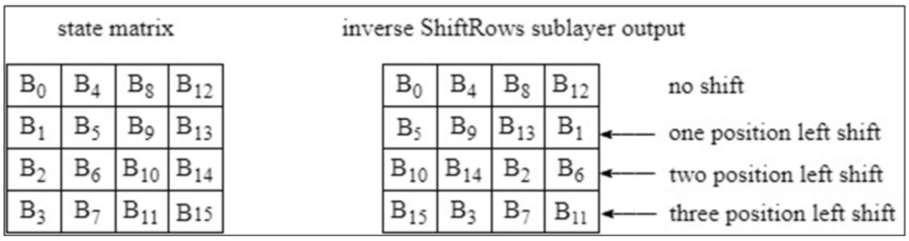


Figure 3: AES encryption block diagram [4]

Figure 3 shows AES encryption block diagram. Key schedule is used to divide a single short key into many subkeys. Every version of AES (AES-128, AES-192, AES-256) has different number of subkeys [5]. The number of subkeys for each version is equal to number of rounds plus one. For example, AES-128 has 10 rounds and 11 subkeys, AES-192 has 12 rounds and 13 subkeys, and AES-

256 has 14 rounds and 15 subkeys. The key schedule generates the required round keys from the original key.

Byte substitution layer is the first layer in each round of AES. The Byte Substitution layer is made up of 16 parallel S-Boxes with 8 input and output bits apiece. AES is byte-matrix based. Each byte in the state matrix array is replaced with its corresponding value in the S-Box [5].



state matrix

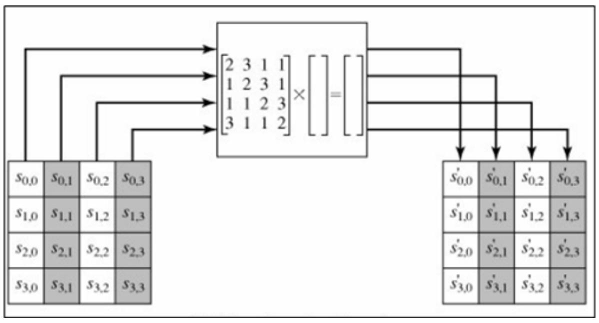
B_0	B_4	B_8	B_{12}
B_1	B_5	B_9	B_{13}
B_2	B_6	B_{10}	B_{14}
B_3	B_7	B_{11}	B_{15}

inverse ShiftRows sublayer output

B_0	B_4	B_8	B_{12}	no shift
B_5	B_9	B_{13}	B_1	← one position left shift
B_{10}	B_{14}	B_2	B_6	← two position left shift
B_{15}	B_3	B_7	B_{11}	← three position left shift

Figure 4: ShiftRows Operation [4]

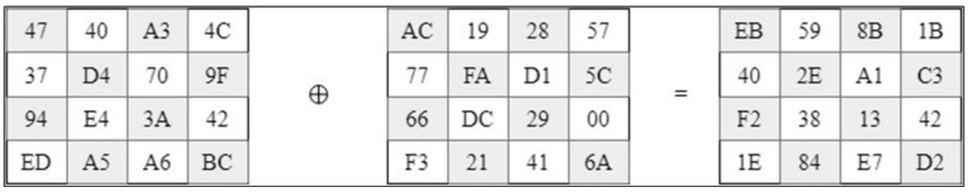
As shown in Figure 4, the diffusion layer in AES consists of two sublayers. The two sublayers are ShiftRows and MixColumn. In diffusion layer, a change of one bit impacts the entire state. Figure 4 shows the shiftRows operation. In shiftrows operation, each row except for the first row in the state matrix, is shifted. Three bytes in second row is shifted to the right. Two bytes in third row is shifted to the right and one byte in forth row is shifted to the right.



$\begin{bmatrix}2&3&1&1\\1&2&3&1\\1&1&2&3\\3&1&1&2\end{bmatrix} \times \begin{bmatrix}\\\\\end{bmatrix} = \begin{bmatrix}\\\\\end{bmatrix}$
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s'_{0,0}$ $s'_{0,1}$ $s'_{0,2}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{1,1}$ $s'_{1,2}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,1}$ $s'_{2,2}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,1}$ $s'_{3,2}$ $s'_{3,3}$

Figure 5: Mix Column Transformation [6]

Figure 5 shows mixColum operation. In mixcolumn operation, each column of the state matrix is transformed to a new column. The transformation is produced by the multiplication of the state matrix and a constant square (4 x 4) matrix. The four bytes of one column in state matrix is multiple by four bytes of a column in the constant square matrix. As a result, it produces a new column replacing the original column.



47	40	A3	4C		AC	19	28	57		EB	59	8B	1B
37	D4	70	9F	⊕	77	FA	D1	5C	=	40	2E	A1	C3
94	E4	3A	42		66	DC	29	00		F2	38	13	42
ED	A5	A6	BC		F3	21	41	6A		1E	84	E7	D2

Figure 6: Add Round Key Transformation [6]

Figure 6 shows key addition layer or add round key transformation. In add round key, the state matrix from previous operation is bitwise XORed with subkey. The first matrix is the state matrix and the second is the subkey matrix. Both matrixes are 16-byte (128-bit). The subkey matrix is produced from key schedule.

2.3 Comparison of Existing System

In this section, three encryption systems are compared to one another. This includes HES, Sendinc, and the proposed system, SecDoc. The comparison is based on different features of the three encryption systems.

The first existing system is An Email Framework Uses Symmetric and Asymmetric Key Algorithm Encryption for Digital's Signature or Hybrid Encryption System (HES). HES is developed by En. Azhar Abdul Halim during his final year project back in 2016 [7]. The Hybrid Encryption System is the name of the email framework prototype use in the system. The HES is a combination of the RSA (Rivest-Shamir-Adleman) and AES (Advance Encryption Standard) algorithms. The system encrypts the electronic document before it is sent through email attachment. The system can be used for encryption and decryption of files.

The second existing system is Sendinc by j2 Global company. This company provides Internet services and digital media. Sendinc is an email encryption service to send encrypted email. Sendinc is a SaaS-based electronic messaging and collaboration platform. It helps users to securely transmit emails. SendInc employs email encryption frameworks to provide a secure email service for businesses of all sizes, professionals, non-profit organisations, and individuals. Sendinc allow user to guarantee that their messages are encrypted to the utmost. Sendinc promotes its encrypted email service as an easy, safe, and free way to send a secure email. It provides military-grade email encryption based on the highest possible standards. Sendinc does not store the encryption key but only the recipients have the ability to decrypt the messages.

Table 1: Comparison of Existing Systems and Proposed System

Features/System	HES	Sendinc	SecDoc
Log In	Yes	Yes	No
Attach Document	Yes	Yes	Yes
Split Document	No	No	Yes
Encrypt Document	Yes	Yes	Yes
Decrypt Document	Yes	Yes	Yes
Join Document	No	No	Yes

Based on Table 1, the three systems have differences in three out of six features. These three features are log in, split document, and join document. The first feature is log in. Only HES and Sendinc have log in feature. The users must input their username and password to log into the systems. SecDoc does not require user to register an account to use the tool. The second feature is attach documents. All the three systems have attach documents feature to enable the user to attach documents for encryption and decryption process.

The third feature is split document. The two existing systems, HES and Sendinc, do not have the feature. Only SecDoc, has the split document feature. This module is to split document into fragments so that the information inside the document is unreadable. The fourth feature is encrypt document. All the three systems have encrypt document feature. The HES and SecDoc encrypt attached documents only. They do not encrypt the message of the email. However, the Sendinc encrypt the message of the email and the attached documents.

The fifth feature is decrypt document. All the three systems have decrypt document feature. For HES and SecDoc, receivers who have the match keys only may decrypt the encrypted documents. While for the Sendinc, the recipients have the ability to decrypt the messages. The sixth feature is join document. Only the purposed system has join document feature. This feature is to join the fragmented document into a full complete document that readable. So that user may read the content of the document.

3. Methodology/Framework

The object-oriented methodology is used for completing the project. Object-oriented Software Development (OOSD) consists of object-oriented requirement analysis, object-oriented analysis, object-oriented design, object-oriented implementation, and testing.

3.1 Object Oriented Requirement Analysis

In object-oriented requirement analysis phase, the requirement of the proposed system is analyzed. An interview with En. Azhar Abdul Halim is conducted with the help of the supervisor. En. Azhar is the owner of Email Framework Uses Symmetric and Asymmetric Key Algorithm Encryption for Digital's Signature Confidentiality[7]. The system is to provide confidentiality to the electronic document as well as to authenticate the owner of the document. Based on the interview, the requirement for the proposed system is analyzed.

The first requirement is that the proposed system should provide confidentiality of the document before it is transferred. Cryptography should be applied to protect the sensitive information. The symmetric and asymmetric cryptography should be implement together become hybrid encryption. There are Advanced Encryption Algorithm and River Shamir Adelman algorithm. This is to add more security to the document. RSA public key can be used to encrypt the AES encryption key.

Next, the document should be fragmented before the encryption process. This is to make the sensitive data become unreadable by the attackers. This is also to avoid data from being altered to protect the integrity of the document. As the encrypted document is split into several fragments. The combination of two layers of encryption and fragmentation provide multiple protection towards the sensitive document before it is transferred as example, via email attachment.

Furthermore, two existing systems are analyzed and taken as reference for the development of the proposed system. The two existing systems are the HES and Sendinc. Based on the two systems, the differences of the systems' features, and the requirements needed for the proposed system are examined. Both HES and Sendinc have encryption and decryption features to encrypt and decrypt electronic documents. However, both do not have split and join document features to fragment and defragment the documents.

3.2 Object Oriented Analysis

In object-oriented analysis phase, the functional and non-functional requirements of the proposed system is analyzed. The functional requirements of the proposed system are the system module. There are five important modules needed. The modules are document selection module, split document module, encrypt document module, decrypt document module, and join document module.

The non-functional requirements are the operational, performance, and security. The operational is to ensure the system should run on windows operating system and should be able with or without the internet connection. The performance is to ensure the five modules proposed are function as it should be. The security requirements are the system should enforce 16 digits key and only the match key can be used for encryption and decryption process.

Table 2: Functional Requirement Analysis

Module	Description
Document Selection	Users select the desired document. The format of the document can be a pdf, Docx, txt, png, and mp4
Split Document	The document is fragmented into *n* parts. The user enters the number of n. Noted, n must be greater than 1. Otherwise, an error message is prompted out to request the user re-enter the value.
Encrypt Document	The fragmented documents are encrypted using AES encryption, and the key is then encrypted by RSA encryption The length of the key must be 16 digits
Decrypt Document	The fragmented documents are decrypted using the same AES key and RSA private key to decrypt the encrypted key
Join Document	After the decryption process, all the fragmented documents are joined together into one complete document

Table 3: Non-Functional Requirement Analysis

Requirements	Description
Operational	The tool should run on the Windows operating system. The tool should be available with and without an internet connection.
Performance	The tool should split documents based on user input. The tool should encrypt documents after the split process. Only users with the key file can decrypt the documents. The tool should join the fragmented documents after the decryption process.
Security	The key entered by the user is encrypted with RSA encryption to secure the key file. The user's key must be 16 digits.

3.3 Object Oriented Design

In Object oriented design, the general system architecture, Unified Modelling Language, test plan and user interface are designed. General system architecture presents the structure and process of the proposed system. Unified Model Language is used to give a standard view of describing the design of the proposed system. This includes use case diagram, sequence diagram, activity diagram, and class diagram of the proposed system.

a. General System Architecture

Figure 7 shows the general system architecture of the proposed system. The users, sender and receiver may perform five functions. There are document selection, split document, encrypt document, decrypt document, and join document modules. Sender sends the encrypted documents. Receiver receives the encrypted documents. The documents are sent through an email attachment. Sender selects a document, split the selected document, and encrypt the fragmented documents. Then, the encrypted document is attached with email sends to receiver. Receiver receives the encrypted documents. Then, downloads the files and save to hard disk. Receiver selects the encrypted documents, decrypt the split documents, and join them into one complete document.

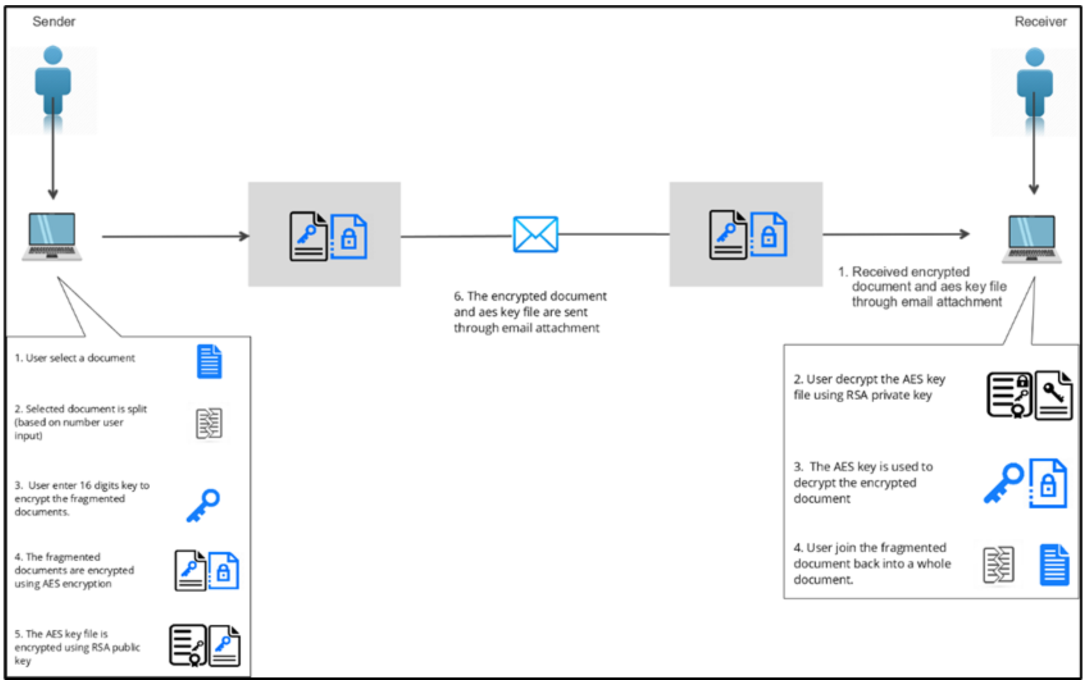

Sender
Receiver
6. The encrypted document and aes key file are sent through email attachment
1. Received encrypted document and aes key file through email attachment
1. User select a document
2. Selected document is split (based on number user input)
3. User enter 16 digits key to encrypt the fragmented documents.
4. The fragmented documents are encrypted using AES encryption
5. The AES key file is encrypted using RSA public key
2. User decrypt the AES key file using RSA private key
3. The AES key is used to decrypt the encrypted document
4. User join the fragmented document back into a whole document.

Figure 7: General System Architecture of The Proposed System

b. Use Case Diagram

Figure 8 shows the use-case diagram for the user. The user may perform five actions. The actions are document selection, split document, encrypt document, decrypt document, and join document.

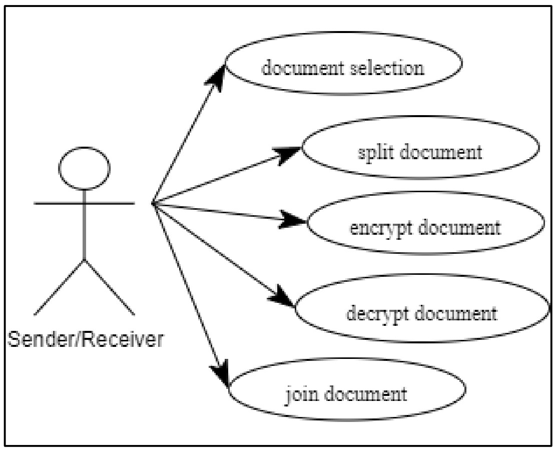


Figure 8: Use-case for user

c. Sequence Diagram

Figure 9 shows the sequence diagram for the user. The user must select a document to split. Next, the user must enter a number of splits to split the chosen document. After the document is split, the user must input16 digits number of key, AES key file, and public key to encrypt the fragmented documents. The user must select the encrypted documents, encrypted AES key file, and private key to decrypt the documents. After decryption, the fragmented documents are joined.

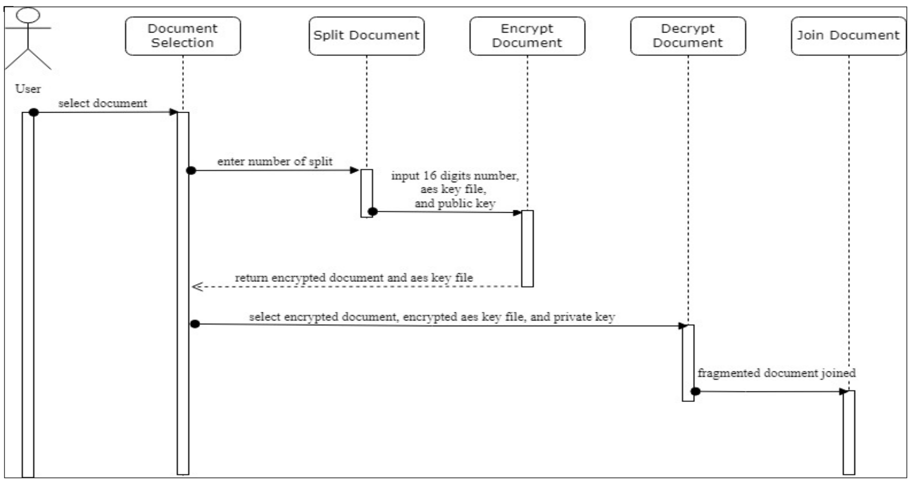

User
Document Selection
Split Document
Encrypt Document
Decrypt Document
Join Document
select document
enter number of split
input 16 digits number, aes key file, and public key
return encrypted document and aes key file
select encrypted document, encrypted aes key file, and private key
fragmented document joined

Figure 9: Sequence Diagram for User

d. Activity Diagram

Based on Figure 10, the user must choose to encrypt a document or decrypt a document. If the user chooses to encrypt a document, the user must select a document. The document is then split based on the number user enters. If the number is one, an error message pop out. After that, the user enters 16 digits key. If the key entered is more or less than 16 digits, an error message pop out. The user selects AES key file and public key file to complete the encryption process. If the user chooses to decrypt a document, the user must select the encrypted document. Next, the user selects the encrypted AES key file and a private key file to complete the decryption process. The file is joined after the fragmented document is decrypted.

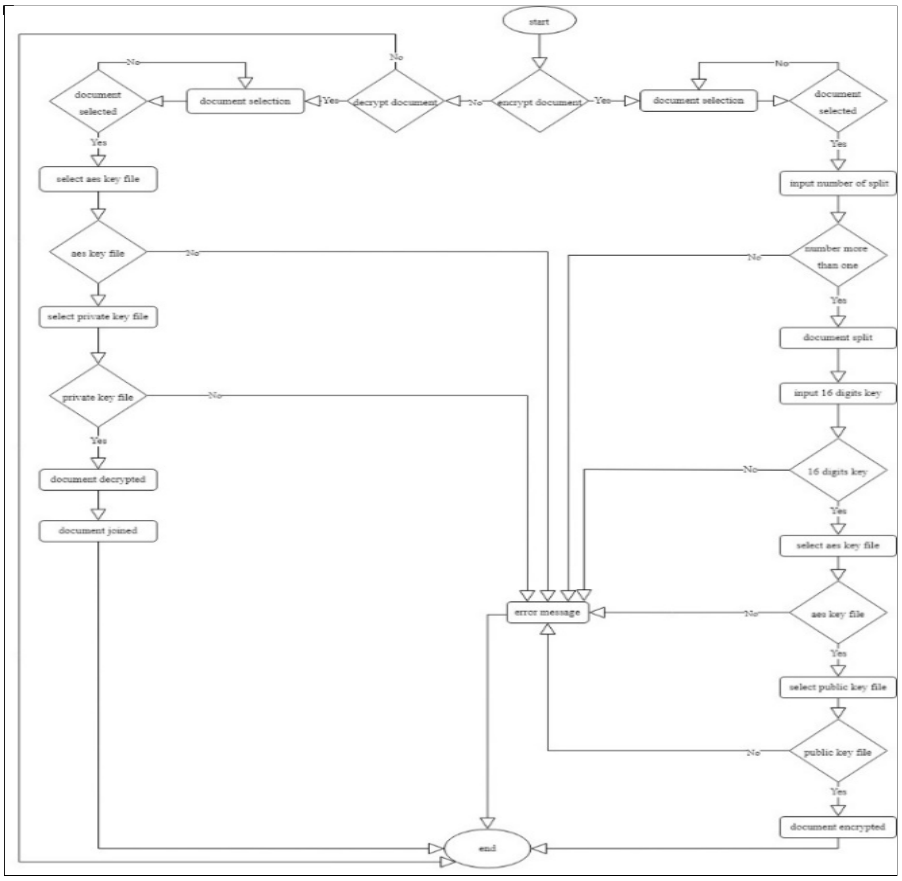


Figure 10: Activity Diagram of the Proposed System

e. Class Diagram

Figure 11 illustrates the class diagram of the proposed system. The class diagram contains JFrame and five classes. The classes are encryption, decryption, split, and join. There are a total of five classes exist in the proposed system. This include JFrame, Split, Encrypt, Decrypt, and Join. Each class has its attributes and methods. Encryption and decryption classes are aggregations of JFrame. The split class is a composition of the encryption class. Join class is a composition of decryption class.

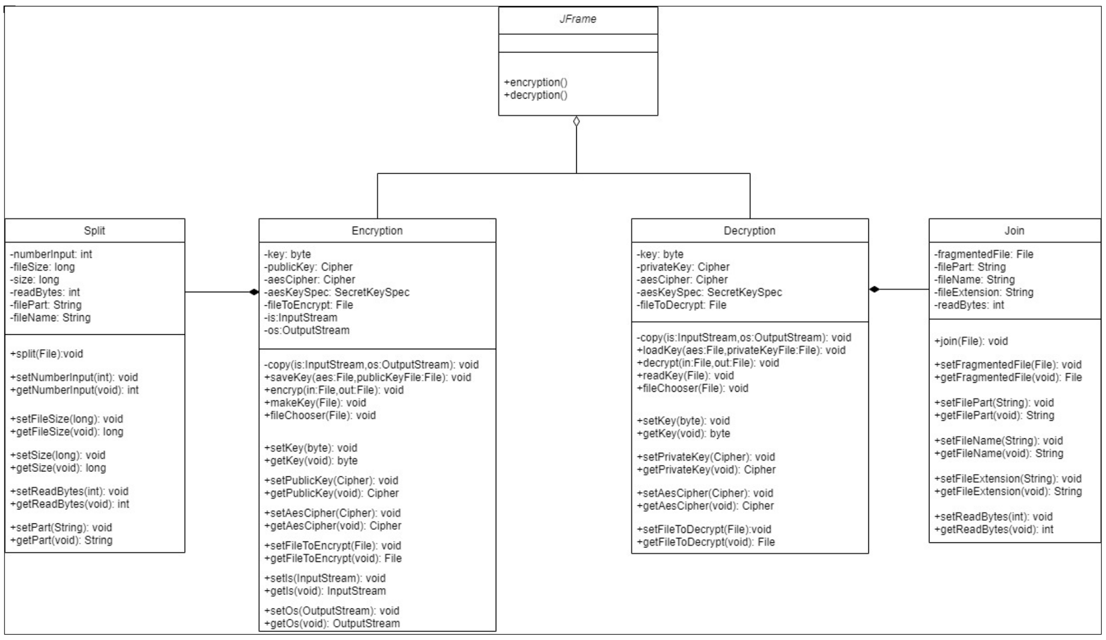

JFrame
+encryption()
+decryption()
Split
-numberInput: int
-fileSize: long
-size: long
-readBytes: int
-filePart: String
-fileName: String
+split(File):void
+setNumberInput(int): void
+getNumberInput(void): int
+setFileSize(long): void
+getFileSize(void): long
+setSize(long): void
+getSize(void): long
+setReadBytes(int): void
+getReadBytes(void): int
+setPart(String): void
+getPart(void): String
Encryption
-key: byte
-publicKey: Cipher
-aesCipher: Cipher
-aesKeySpec: SecretKeySpec
-fileToEncrypt: File
-is:InputStream
-os:OutputStream
-copy(is:InputStream,os:OutputStream): void
+saveKey(aes:File,publicKeyFile:File): void
+encryp(in:File,out:File): void
+makeKey(File): void
+fileChooser(File): void
+setKey(byte): void
+getKey(void): byte
+setPublicKey(Cipher): void
+getPublicKey(void): Cipher
+setAesCipher(Cipher): void
+getAesCipher(void): Cipher
+setFileToEncrypt(File): void
+getFileToEncrypt(void): File
+setIs(InputStream): void
+getIs(void): InputStream
+setOs(OutputStream): void
+getOs(void): OutputStream
Decryption
-key: byte
-privateKey: Cipher
-aesCipher: Cipher
-aesKeySpec: SecretKeySpec
-fileToDecrypt: File
-copy(is:InputStream,os:OutputStream): void
+loadKey(aes:File,privateKeyFile:File): void
+decrypt(in:File,out:File): void
+readKey(File): void
+fileChooser(File): void
+setKey(byte): void
+getKey(void): byte
+setPrivateKey(Cipher): void
+getPrivateKey(void): Cipher
+setAesCipher(Cipher): void
+getAesCipher(void): Cipher
+setFileToDecrypt(File):void
+getFileToDecrypt(void): File
Join
-fragmentedFile: File
-filePart: String
-fileName: String
-fileExtension: String
-readBytes: int
+join(File): void
+setFragmentedFile(File): void
+getFragmentedFile(void): File
+setFilePart(String): void
+getFilePart(void): String
+setFileName(String): void
+getFileName(void): String
+setFileExtension(String): void
+getFileExtension(void): String
+setReadBytes(int): void
+getReadBytes(void): int

Figure 11: Class Diagram of the Proposed System

f. Test Plan

The following tables show the test plan that are designed for the proposed tool. The test plan includes the functionality test plan and the security test plan.

Table 4: Document selection function test plan

No	Test Cases	Expected Result
1	Attach single file	Able to attach a single file
2	Attach multiple files	Able to attach multiple files

Table 5: Split Function Test Plan

No	Test Cases	Expected Result	Result
1	User input number of split more than one. Example: 20	Able to split the file/successful message pop out	Pass/Fail
2	User input number of split less than two. Example: 1	Unable to split the file/error message pop out	Pass/Fail
3	User input number of split less than one. Example: 0	Unable to split the file/error message pop out	Pass/Fail
4	User do not input any number of splits.	Unable to split the file/error message pop out	Pass/Fail
5	Split selected file into number of splits user input.	Able to split the file/successful message pop out	Pass/Fail

Table 6: Encrypt function test plan

No	Test Case	Expected Result	Result
1	User enters 16 characters key. Example: 1234hgnjko*&%)(!	Able to encrypt file/successful message pop out	Pass/Fail
2	User enters less than 16 characters key. Example: document123	Unable to encrypt file/error message pops out	Pass/Fail
3	User do not input any key.	Unable to encrypt file/error message pops out	Pass/Fail
4	User selects the right key file. Example: aes.key	Able to encrypt file/successful message pop out	Pass/Fail
5	User selects the wrong key file. Example: 123.pdf	Unable to encrypt file/error message pops out	Pass/Fail
6	User do not select any key file.	Unable to encrypt file/error message pops out	Pass/Fail
7	User selects the right public key file. Example: public_key.der	Able to encrypt file/successful message pop out	Pass/Fail
8	User selects the wrong public key file. Example: 123.pdf	Unable to encrypt file/error message pops out	Pass/Fail
9	User do not select any public key file.	Unable to encrypt file/error message pops out	Pass/Fail
10	Encrypt selected split files. Example: test.pdf.001	Able to encrypt the selected split files/successful message pop out	Pass/Fail

Table 7: Decrypt function test plan

No	Test Case	Expected Result	Result
1	User selects the right key file. Example: aes.key	Able to decrypt the encrypted file/successful message pop out	Pass/Fail
2	User selects the wrong key file. Example: aes.pdf	Unable to decrypt the encrypted file/error message pops out	Pass/Fail
3	User do not select any key file.	Unable to decrypt file/error message pops out	Pass/Fail
4	User selects the right private key file. Example: private_key.der	Able to decrypt file/successful message pop out	Pass/Fail
5	User selects the wrong private key file. Example: 123.pdf	Unable to decrypt file/error message pops out	Pass/Fail
6	User do not select any private key file.	Unable to decrypt file/error message pops out	Pass/Fail
7	Decrypt selected encrypted files. Example: test.pdf.encrypted	Able to decrypt the selected encrypted files/successful message pop out.	Pass/Fail

Table 8: Join function test plan

No	Test Case	Expected Result	Result
1	User can only select one fragmented file; the system will find the other parts of the file automatically. Example: test.pdf.001	Able to join the fragmented files/successful message pop out.	Pass/Fail

Table 9: Security test plan

No	Test Case	Result
1	Enforce 16-characters key length. The user input is limit to 16 characters length to encrypt files. Example: Wrong message pop out if keys enter less than 16 characters length.	Pass/Fail
2	The key entered by user is encrypted using public key file. Example: The key inside the aes.key file is encrypted, so it is unreadable.	Pass/Fail
3	The system only decrypts files with the match private key. Example: Only private_key.der of user's private key are accepted for decryption process.	Pass/Fail
4	The system must split files before encryption. Example: The system only will do encryption process after split process is done.	Pass/Fail

g. User Interface Design

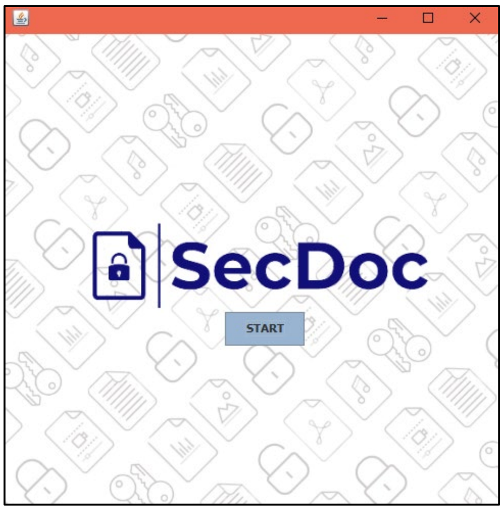


Figure 12: Start interface design

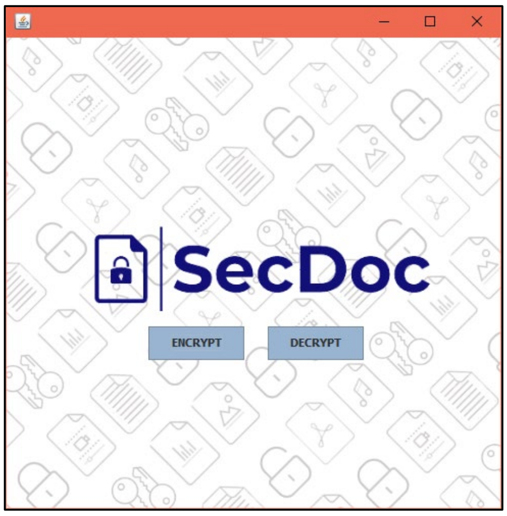


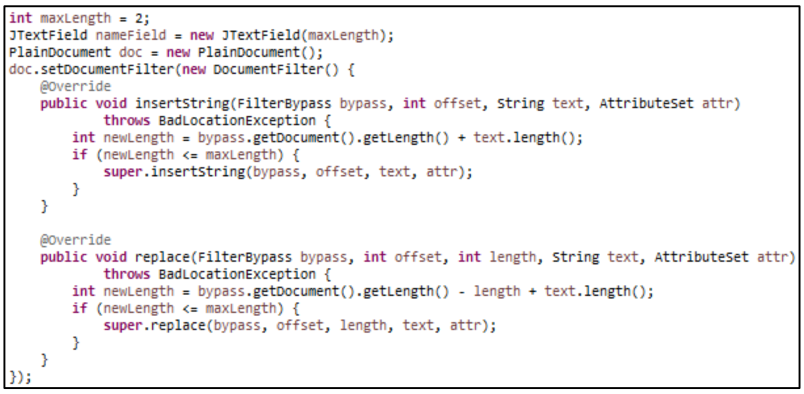
Figure 13: Main menu interface design

Figure 12 shows the first interface that user will see when they open the SecDoc tool. The user must click the start button to start using the tool. Figure 13 shows the main menu interface after the user click the start button. In the main menu, there are two buttons for split, encryption, decryption, and join files process. The buttons are encrypt and decrypt button.

4. Results and Discussion

In results and discussion, the implementation and testing result of proposed system are discussed. The implementation of the code is explained as the following.

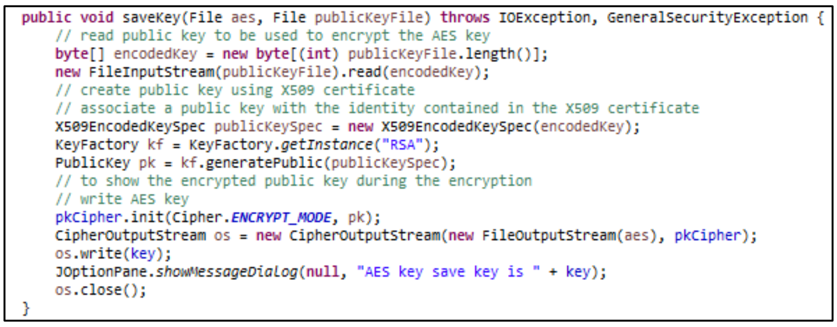
As shown in the Figure 14, the maxLength is set to 2 which means the input that user can put to split the file is two digits. This limit is to ensure that user do not enter too large number of splits. The DocumentFilter is a filter for document modification methods. When a document containing a DocumentFilter is changed, the DocumentFilter gets the relevant method invocation. The insertString method is called before inserting text into the specified document. The replace method is called before replacing a text block in the specified document.



```java
int maxLength = 2;
JTextField nameField = new JTextField(maxLength);
PlainDocument doc = new PlainDocument();
doc.setDocumentFilter(new DocumentFilter() {
    @Override
    public void insertString(FilterBypass bypass, int offset, String text, AttributeSet attr)
            throws BadLocationException {
        int newLength = bypass.getDocument().getLength() + text.length();
        if (newLength <= maxLength) {
            super.insertString(bypass, offset, text, attr);
        }
    }

    @Override
    public void replace(FilterBypass bypass, int offset, int length, String text, AttributeSet attr)
            throws BadLocationException {
        int newLength = bypass.getDocument().getLength() - length + text.length();
        if (newLength <= maxLength) {
            super.replace(bypass, offset, length, text, attr);
        }
    }
});
```

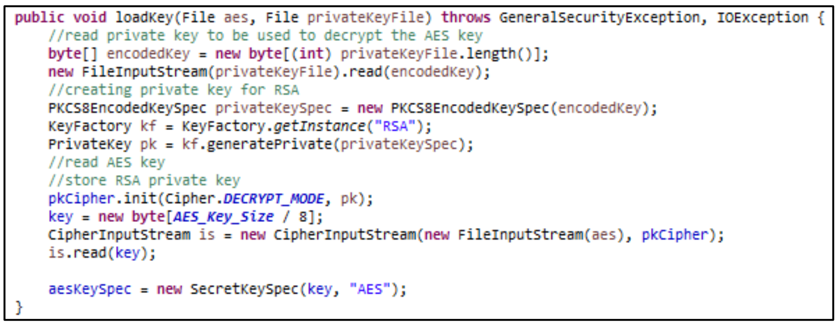
Figure 14: Java source code to limit the number of splits



```java
public void saveKey(File aes, File publicKeyFile) throws IOException, GeneralSecurityException {
    // read public key to be used to encrypt the AES key
    byte[] encodedKey = new byte[(int) publicKeyFile.length()];
    new FileInputStream(publicKeyFile).read(encodedKey);
    // create public key using X509 certificate
    // associate a public key with the identity contained in the X509 certificate
    X509EncodedKeySpec publicKeySpec = new X509EncodedKeySpec(encodedKey);
    KeyFactory kf = KeyFactory.getInstance("RSA");
    PublicKey pk = kf.generatePublic(publicKeySpec);
    // to show the encrypted public key during the encryption
    // write AES key
    pkCipher.init(Cipher.ENCRYPT_MODE, pk);
    CipherOutputStream os = new CipherOutputStream(new FileOutputStream(aes), pkCipher);
    os.write(key);
    JOptionPane.showMessageDialog(null, "AES key save key is " + key);
    os.close();
}
```

Figure 15: Java source code for encrypting key

As shown in Figure 15, public key is read to encrypt the AES key. The public key is created using the identity contained in the X509 certificate. The byte[]encodedKey returns the key bytes and encoded according to the X509 standard. The constructor X509EncodedKeySpec creates a new X509EncodedKeySpec with the given encoded key. The CipherOutputStream is composed of an OutPutStream and a Cipher. The write() methods is first process the data which is the AES key before writing it out to the underlying OutputStream. As shown, the CipherOutputStream will attempt to encrypt the AES key before writing out the encrypted AES key. Then, the message dialog will show the save key.

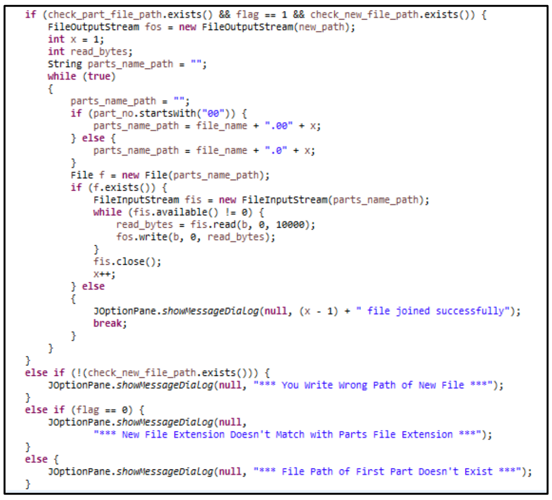


```java
public void loadKey(File aes, File privateKeyFile) throws GeneralSecurityException, IOException {
    //read private key to be used to decrypt the AES key
    byte[] encodedKey = new byte[(int) privateKeyFile.length()];
    new FileInputStream(privateKeyFile).read(encodedKey);
    //creating private key for RSA
    PKCS8EncodedKeySpec privateKeySpec = new PKCS8EncodedKeySpec(encodedKey);
    KeyFactory kf = KeyFactory.getInstance("RSA");
    PrivateKey pk = kf.generatePrivate(privateKeySpec);
    //read AES key
    //store RSA private key
    pkCipher.init(Cipher.DECRYPT_MODE, pk);
    key = new byte[AES_Key_Size / 8];
    CipherInputStream is = new CipherInputStream(new FileInputStream(aes), pkCipher);
    is.read(key);

    aesKeySpec = new SecretKeySpec(key, "AES");
}
```

Figure 16: Java source code to decrypt the AES key using RSA private key

As shown in the Figure 16, PKCS8 is the standard syntax for storing private key information. The PKCS8EncodedKeySpec class is to handlePKCS8 key material to get a private key. The private key is then read to decrypt the AES key. The byte[]encodedKey returns the key bytes and encoded according to the c. The constructor PKCS8EncodedKeySpec creates a new PKCS8EncodedKeySpec with the given encoded key. The CipherInputStream is composed of an InputStream and a Cipher. The read() methods is returning the data that are read in from the underlying InputStream but have been additionally processed by the Cipher. As shown, the Cipher is initialized for decryption, so the CipherInputStream will attempt to read and decrypt the AES key before returning the decrypted AES key.



```java
if (check_part_file_path.exists() && flag == 1 && check_new_file_path.exists()) {
    FileOutputStream fos = new FileOutputStream(new_path);
    int x = 1;
    int read_bytes;
    String parts_name_path = "";
    while (true)
    {
        parts_name_path = "";
        if (part_no.startsWith("00")) {
            parts_name_path = file_name + ".00" + x;
        } else {
            parts_name_path = file_name + ".0" + x;
        }
        File f = new File(parts_name_path);
        if (f.exists()) {
            FileInputStream fis = new FileInputStream(parts_name_path);
            while (fis.available() != 0) {
                read_bytes = fis.read(b, 0, 10000);
                fos.write(b, 0, read_bytes);
            }
            fis.close();
            x++;
        } else
        {
            JOptionPane.showMessageDialog(null, (x - 1) + " file joined successfully");
            break;
        }
    }
}
else if (!(check_new_file_path.exists())) {
    JOptionPane.showMessageDialog(null, "*** You Write Wrong Path of New File ***");
}
else if (flag == 0) {
    JOptionPane.showMessageDialog(null,
            "*** New File Extension Doesn't Match with Parts File Extension ***");
}
else {
    JOptionPane.showMessageDialog(null, "*** File Path of First Part Doesn't Exist ***");
}
```

Figure 17: Java source code for looping to join files

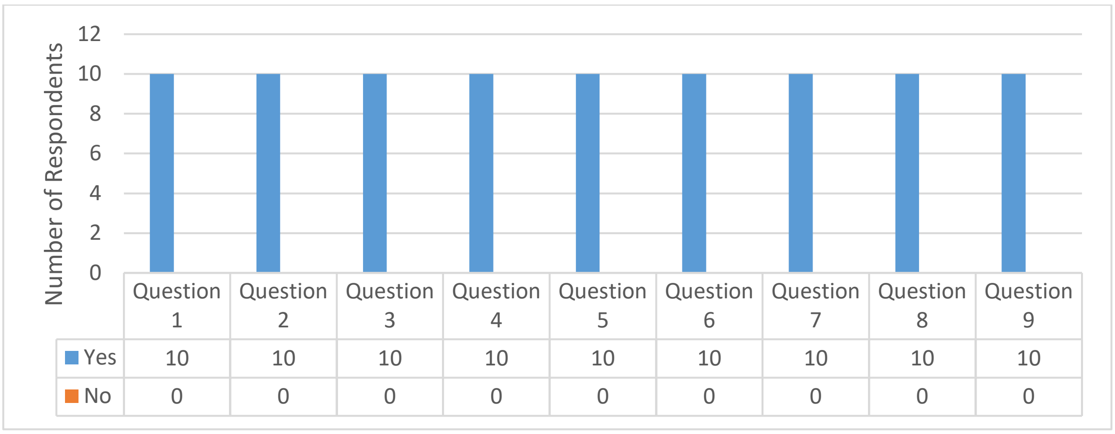
As shown in the Figure 17, the if statement is used to check if part path and the original parent path exist and true. If they exist and true, the parts will be joined or otherwise error message will pop out. Furthermore, the while loop is executed until all parts are joined. The FileInputStream class is used to read data from file. The loop will execute till the fis.available() method is not equal to 0. The fis.close() method is executed every time after every parts completion. The x value is incremented by 1 and the while loop is repeated till all parts are join and successful message will be printed. If all part of the fragmented files is joined, the dialog message will show a successful message saying x file joined successfully. The x is number of files that is successfully joined. Otherwise, if user entered wrong original or new file path, the dialog message will print the user write a wrong path of new file. Next, if the original file extension and parts file extension does not match, the system will tell that the new file extension does not match with parts file extension. If user entered wrong part file name, the system would tell that the file path of first part does not exist.

There are two type of testing result of the proposed system. These include test plan result and user acceptance result. The results of the functionality test plan and security test plan are all passes. The Table 10 shows the user acceptance form that is used in the Section B of the Google Form.

Table 10: User acceptance form

No	Test Case	Result	
		Yes	No
1	User must click start button to start using the tool.		
2	User can choose whether to encrypt or decrypt files.		
3	User must split the files first before encryption function can be done.		
4	User can only join the files after the decryption process is done.		
5	User is prompt to the correct file path for each process.		
6	Files for each process is saved to the specific file path.		
7	Successful or error message will pop out for successful or unsuccessful process.		
8	Cancel button end the process and will direct user to the main menu.		

The respondents for the user acceptance test are the students of Fakulti Sains Komputer Teknologi Maklumat in Universiti Tun Hussein Onn Malaysia. Figure 18 shows the result of Section B questions in Google Form.



	Question 1	Question 2	Question 3	Question 4	Question 5	Question 6	Question 7	Question 8	Question 9
🟦 Yes	10	10	10	10	10	10	10	10	10
🟧 No	0	0	0	0	0	0	0	0	0

Figure 18: User Acceptance Result for Section B in Google Form

Based on Figure 18, all the ten respondents answered "Yes" for question 1 until question 9. This indicates that they must click start button to start using the tool. They also can choose whether to encrypt or decrypt files. The encryption function can be done only after users split the files and users can only join the files after the decryption process is done. Next, all users are prompt to the correct file path for each process. User's files for each process are saved to the specific file path. The users will see a successful or error message pop out for successful or unsuccessful process. The cancel button ends the process and will direct users to the main menu. Lastly, users have to generate their own public and private key using command prompt.

5. Conclusion

The proposed tool, Secure Document Fragmentation Tool with Two Layers of Encryption has several advantages and disadvantages. The advantages are the tool splits files making the files fragmented and unreadable, encryption of the fragmented files making it more secure, enforces 16 characters key length for encryption process, requires users to use the right AES key files for decryption process, requires users to use their own public and private key for encryption and decryption process, encrypts the AES key file using public key, only encrypts file after fragmentation process is done and only joins the files after the decryption process is done.

The disadvantages are firstly, the tool writes the 16 keys input by user in one aes.key file before the key is encrypted. It will then delete the keys in the file and writes the new 16 keys in the same aes.key file if the next encryption process is happened. So, users must save the aes.key into another folder first before they do another encryption process. This may inconvenience when they want to perform many encryptions of files. Secondly, the tool allows user to rename and change extension of their desired files when they do the split, encryption, decryption, and join process. If the extension of the selected files is changed, the files will be corrupted or unreadable even if the decryption and join process is done.

During the encryption process, the tool writes the 16 keys input by user in one aes.key file before the key is encrypted. It will then delete the keys in the file and writes the new 16 keys in the same aes.key file if the next encryption process is happened. So, users must save the aes.key into another folder first before they do another encryption process. This may inconvenience when they want to perform many encryptions of files. For future implementation, the tool should create a new aes.key file for a new encryption process. Next, the tool allows user to rename and change extension of their desired files when they do the split, encryption, decryption, and join process. If the extension of the selected files is changed, the files will be corrupted or unreadable even if the decryption and join process is done. For future implementation, the tool should not allow users to change the extension of the files. It should only allow users to change the name of the files.

At the end of the tool development, the objectives of the proposed tool are achieved. The achieved objectives are to design a secure document fragmentation tool, to develop the secure document fragmentation tool with two layers of encryption, and to test the implementation secure document fragmentation tool. Throughout the system development process, supervisory supervision, project planning, and schedule management are all important considerations in completing the system on time. In conclusion, the main findings of the study are sensitive document must be protected before it is transferred via email attachment. Therefore, encryption must be done to encrypt the document. However, fragmentation helps to add more security towards the document. Java programming language is used to build the proposed system.

Acknowledgement

The authors would like to thank the Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia for its support.

References

[1] S. Malgaonkar, S. Surve, and T. Hirave, "Distributed files sharing management: A file sharing application using distributed computing concepts," in *2012 IEEE International Conference on Computational Intelligence and Computing Research*, 2012, pp. 1–4, doi: 10.1109/ICCIC.2012.6510207.

[2] K. K. Mar and Y. L. Chee, "Secure Email Attachment as a Service (SeaS)," in *2015 International Conference on Cloud Computing Research and Innovation (ICCCRI)*, 2015, pp. 106–111, doi: 10.1109/ICCCRI.2015.33.

[3] S. Thielman, "Yahoo hack: 1bn accounts compromised by biggest data breach in history," *Guard.*, vol. 15, p. 2016, 2016.

[4] C. Paar and J. Pelzl, *Understanding Cryptography: A Textbook for Students and Practitioners*. 2010.

[5] B. A. Forouzan , *Cryptography and network security*. Boston: McGraw-Hill Higher Education, 2008.

[6] W. Stallings, *Cryptography and network security : principles and practices*. Upper Saddle River, N.J.: Pearson/Prentice Hall, 2006.

[7] M. Azhar, B. I. N. Abdul, F. Access, M. Azhar, and B. I. N. Abdul, *An Email Framework Uses Symmetric And Asymmetric Key Algorithm Encryption For Confidentiality* 2016.