World Applied Sciences Journal 35 (New Advancement of Research & Development in Computer Science): 122-132, 2017
ISSN 1818-4952

DOI: 10.5829/idosi/wasj.2017.122.132

Securing File Transferring System by Implementing AES Algorithm

Nor Surayati Mohamad Usop, Ahmad Faisal Abidin, Fauziah Ab. Wahab and Norlina Udin@Kamaruddin

Faculty of Informatics and Computing, Universiti Sultan Zainal Abidin, Malaysia

Abstract: In a small area location such as a house, office or in a classroom, there is a small network called a Local Area Network (LAN). This project aims to transfer a file peer-to-peer from one computer to another computer using JAVA application in the same LAN. It provides the necessary authentication for file transferring in the network transmission. By implementing the Server-Client technology, we can use a File Transfer Protocol mechanism and through socket programming, the end user is able to send and receive the encrypted and decrypted file in the LAN. An additional aim of this project is to transfer a file between computers securely in LANs. Elements of security are needed in this project because securing the files is an important task, which ensures files are not captured or altered by anyone on the same network. Whenever you transmit files over a network, there is a good chance your data will be encrypted by encryption technique. In this project, an AES algorithm is used to encrypt the file that needs to transfer to another computer. The encrypted file is then sent to a receiver computer and will need to be decrypted before the user can open the file. This research makes use of an AES algorithm because it is fast and secure for transferring a file. The file is expected to be transferred securely and without being modified.

Key words: AES · Netbeans · Java Socket Programming and Wireshark

INTRODUCTION

Today, a computer has multipurpose uses such as for gaming, writing, surfing the internet and transferring data or files. The use shows that our work can be finished quickly. There are various ways to transfer a file from one computer to another such as using medium storage and internet applications. Files are not secure to be transferred because there are so many security threats that need to be considered while sending files over a network. Hackers can use many ways to get the content of the file, for example, by using packet sniffer. Hence, as a sender, we should know the files are securely sent to the intended destination. Encryption is one of the best ways to protect the contents of a file while we send it and the file will decrypt when it has arrived at its destination.

One of the best encryption techniques is using AES algorithm. AES algorithm was designed to have resistance against all known attacks, speed and code compactness on a wide range of platforms and design simplicity. AES with 128-bit keys has stronger resistance to an exhaustive key search. It is a substitution-permutation network. Each round in AES encryption includes four different round transformations: Substitute Bytes, Shift Rows, Mix Columns and Add Round Key. It is the process of converting plaintext into encrypted text and decrypting cipher text to plain text at the other end. The initial unencrypted data is referred to as normal text. It is encrypted into cipher text with a cryptographic algorithm, which will in turn be decrypted into usable plaintext. Symmetric key ciphers like AES, on the other hand, are more suitable for encrypting the actual data (and commands) because they require less resources and are also much faster than asymmetric ciphers.

Typically, computer users can share their files using a built-in application in Windows or a Mac operating system by setup the LAN connection between computers. The current research aim is to develop a secure file transfer system by implementing AES algorithm. AES algorithm works as a security mechanism to encrypt sender data and decrypt receiver data. Without any security mechanism, the hacker could intercept the data transmission by using packet sniffer.

Corresponding Author: Ahmad Faisal Abidin, Department of Computer Science, Faculty of Informatics and Computing, Universiti Sultan Zainal Abidin, Besut Campus, 22200, Terengganu, Malaysia.

Literature Review: In a journal "A Study of Encryption Algorithm AES, DES and RSA for Security" by Dr Prerna Mahajan & Abhishek Sachdeva [1], three encrypt techniques — AES, DES and RSA algorithm are used and their performance of encryption and decryption compared. By using the same text file in four experiments, the time taken for AES to encrypt and decrypt file is the shortest followed by DES and RSA. From the simulation result, it shows that AES algorithm is better than DES and RSA in term of speed of the encryption and decryption process.

From "Three Tier Encryption Algorithm for Secured File Transfer" by Bhargav Balakrishnan [2], the file transferring is secured by using the RSA algorithm (2048 Bits), number conversions, digital encoding and mathematical series. All these processes provided a secured encrypted output of the data, which will prevent the hacker from viewing the data when transmitted over the network. This type of algorithm is needed by organisations that need to have secured data transmission within their network. For an organisation to encrypt and decrypt data, it is a simple process involving each data encryption to be stored in their database. However, using RSA (2048) algorithm has some disadvantages; for instance, it may use larger storage space because it uses 2048-bit keys to encrypt one file compared to AES algorithm that only use 128-bit keys for the encryption process. Using larger storage space can slow down computer processes that only have small storage spaces. Besides, the decryption process would take more time because of longer key bits, namely, 2048 key bits. Therefore, the recipient may get the file from the sender faster when using lower key bits, which is implemented in an AES 128 bits algorithm.

Tayde's "File Encryption, Decryption Using AES Algorithm in Android Phone" [3] indicates AES algorithms are preferable to secure a file compared to many other cryptographic techniques such as Data Encryption Standard, 3DES, RSA and Blowfish. There are two types of encryption algorithm Symmetric keys encryption and Asymmetric keys encryption. In Symmetric keys encryption or secret key encryption, only one key is used to encrypt and decrypt data. The key should be distributed before transmission between entities. In Asymmetric key encryption or public key encryption, two keys are used — private and public keys. A public key is used for encryption and a private key is used for decryption. In this study, the researcher compared the four most commonly used Symmetric key algorithms: DES, 3DES, AES and Blowfish. A comparison was made using the following parameters: round block size, key size, encryption/decryption time and CPU process time in the form of throughput and power consumption. It was concluded that blowfish is better than other algorithms. However, Blowfish has disadvantages in terms of time consumption and serially in throughput. AES also has advantages over 3DES and DES in terms of throughput and decryption time. 3DES has the worst performance among all mentioned algorithms. AES was developed by two scientists, Joan and Vincent Rijmen in 2001[4]. It is fast, compact and has a very simple mathematical structure that would make AES encryption and decryption processes faster than others algorithm.

In a journal "An Efficient Certificateless Encryption for Secure Data Sharing Over the Network Using AES-128 and AES-256" by Sudhir Dhongade [5], Shrikant Bhandare, Amol Davare and Rudraksh Chandel, the encryption process involved in AES algorithm was studied. In this paper, the researchers proposed the AES-128-256 scheme without pairing operations and provided its formal security. AES-128-256 solves the key escrow problem and revocation problem. Using the AES-128-256 scheme as a key building block, they proposed an improved approach to securely share sensitive data in their database. The approach supports immediate revocation and assures the confidentiality of the data stored in an untrusted database while enforcing the access control policies of the data owner. Their experimental results show the efficiency of basic AES -128-256 scheme and improved approach for the database.

MATERIALS AND METHODS

In this project, tools have been used is the latest NetBeans application [6] to develop a file transfer system using JAVA socket programming language. Socket provides the communication mechanism between two computers using Transmission Control Protocol (TCP). TCP is a connection-oriented protocol. To communicate over the TCP protocol, a connection must first be established between the pair of sockets. While one of the sockets listens for a connection request (server), the other asks for a connection (client). A client program creates a socket on its end of the communication and attempts to connect that socket to a server. Once two sockets have been connected, they can be used to transmit data in either one or both directions.

AES is symmetric encryption use the same key for encrypting and decrypting, so both the sender and the receiver must know and use the same secret key. Symmetric encryption requires that the secret key be known by the party encrypting the data and the party decrypting the data. AES-128 refers to bit key lengths that are deemed sufficient to protect classified information up to the "Secret" level with "Top Secret" information. AES works by repeating the same defined steps multiple times. AES algorithm is modified to make it suitable for file transfer environments. AES integrates with systems that have been developed at the SEND button to start encryption processes. The decryption process takes place when ciphertext converts to plaintext at the end of transmission when the file is received by the recipient.

How AES Works: AES [7] is an iterated symmetric block cipher, which means that AES works by repeating the same defined steps multiple times. It is based on a 'substitution–permutation network'. It comprises of a series of linked operations, some of which involve replacing inputs by specific outputs (substitutions) and others involve shuffling bits around (permutations). Interestingly, AES performs all its computations on bytes rather than bits. Hence, AES treats the 128 bits of a plaintext block as 16 bytes. These 16 bytes are arranged in four columns and four rows for processing as a matrix.

Table 1: AES key

Key Size (Bits)	Block Size (Bytes)	Rounds
128	16	10
192	16	12
256	16	14

*The number of rounds of the algorithm depend on the key size.

Table 2: AES Encryption Process in Round

Round	Function
-	Add Round Key (State)
0	Add Round Key (Mix Column (Shift Row (Byte Sub (state))))
1	Add Round Key (Mix Column (Shift Row (Byte Sub (state))))
2	Add Round Key (Mix Column (Shift Row (Byte Sub (state))))
3	Add Round Key (Mix Column (Shift Row (Byte Sub (state))))
4	Add Round Key (Mix Column (Shift Row (Byte Sub (state))))
5	Add Round Key (Mix Column (Shift Row (Byte Sub (state))))
6	Add Round Key (Mix Column (Shift Row (Byte Sub (state))))
7	Add Round Key (Mix Column (Shift Row (Byte Sub (state))))
8	Add Round Key (Mix Column (Shift Row (Byte Sub (state))))
9	Add Round Key (Shift Row (Byte Sub (state)))

This is an example of AES algorithm cipher using a 16-byte key [8]. The only exception being that in the last round, the Mix Column step is not performed to make the algorithm reversible during the decryption process. Before applying the algorithm of data, the block and key sizes must be determined. AES allows for block sizes and key sizes of 128, 192 and 256 bits. Using AES-128 means that each block consists of 128 bits. The original plaintext is stored in bytes inside the block. Each character is stored in a cell of the block. The following example will show how data is broken up into a block. For instance, the text "THIS IS A SECRET" will be stored in a block as shown below:

T		A	C
H	I		R
I	S	S	E
S		E	T

There are four steps in AES algorithm:

Add Round Key

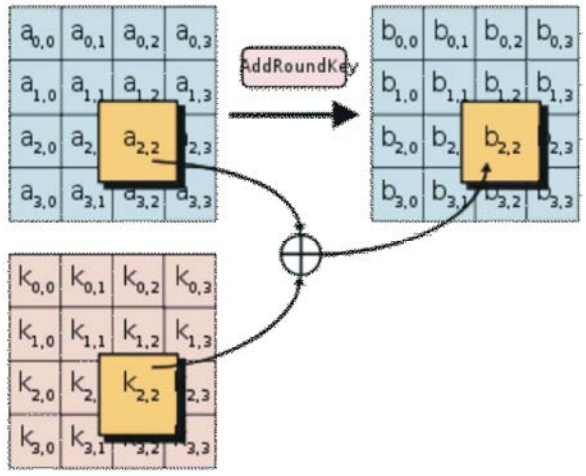

Fig. 1: AES add round key

In the AddRoundKey step, each byte of the state is combined with a byte of the round subkey using XOR operation. A subkey is derived from the main key Rijndael's key schedule.

Subbytes: SubBytes are used at the encryption site to substitute a byte, interpret the byte as two hexadecimal digits by using an ASCII lookup table. As an example, we had the plaintext of character "T" it would be translate to hexadecimal value of 54 by using an ASCII lookup table.

ASCII Table

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	(space)	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	|
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

Fig. 2: ASCII Lookup table

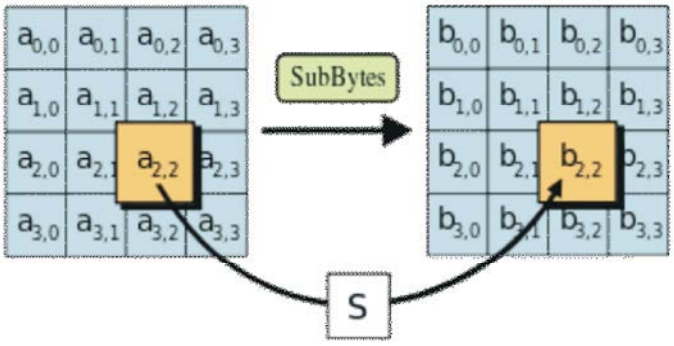

Fig. 3: Idea of SubBytes transformation

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Fig. 4: S-Box Table

The left digit defines the row and the right digit defines the column of the substitution table.

In the SubBytes transformation, the state is treated as a 4x4 matrix of bytes. Transformation is done one byte at a time. The content of each byte is changed but the arrangement of the bytes in the matrix remains the same. In the process, each byte is transformed independently. There are 16 distinct byte-to-byte transformations.

For example, the hexadecimal value that we get from the ASCII Lookup table is 54. By using S-Box, we find row number 5 and column number 4 to find the value in S-box table, which is 20. This is how the S-Box works. Using this method to the plaintext will generate the new hexadecimal values. The transformation provides a confusion effect.

0x20	0xB7	0xEF	0xFB
0x45	0xF9	0xB7	0x40
0xF9	0x8F	0x8F	0x43
0x8F	0xB7	0x43	0x92

Fig. 5: Block/State after SubBytes

Shift Row: Each of the four rows of the matrix are shifted to the left. Any entries that 'fall off' are re-inserted on the right side of the row. The shift is carried out as follows -

- First row is not shifted.
- Second row is shifted one (byte) position to the left.
- Third row is shifted two positions to the left.
- Fourth row is shifted three positions to the left.

The result is a new matrix consisting of the same 16 bytes but shifted with respect to each other.

0x20	0xB7	0xEF	0xFB
0xF9	0xB7	0x40	0x45
0x8F	0x43	0xF9	0x8F
0x92	0x8F	0xB7	0x43

Block/State after Shift Row

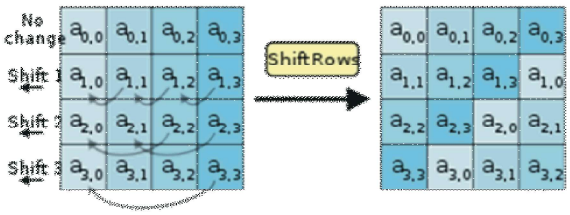

Fig. 6: Operation of Shift Row

Mix Columns: Each column of four bytes is now transformed using a special mathematical function. This function takes as input the four bytes of one column and outputs four completely new bytes, which replace the original column. The result is another new matrix consisting of 16 new bytes. The bytes that are in the state column and constant matrix are interpreted as 8 bits or polynomial with coefficient GF(2). The procedure is repeated with the next column of the state until there are no more state column. It should be noted that this step is not performed in the last round.

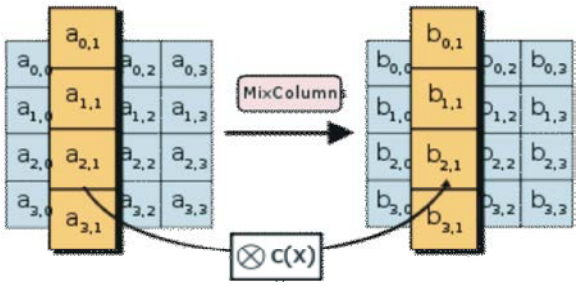

Fig. 7: Operation of Mix Columns process

Implementation

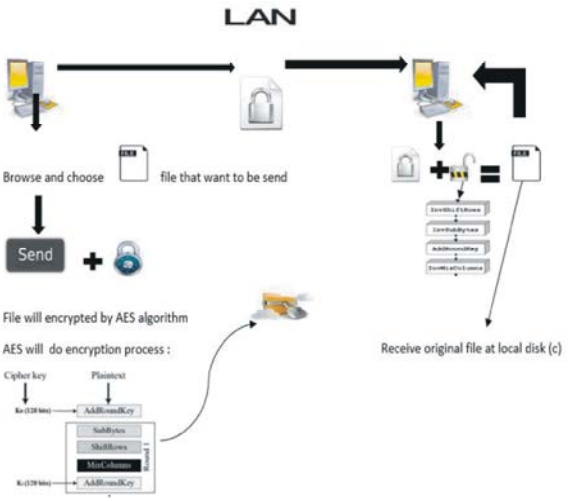

Fig. 8: Framework to secure file transferring system by implementing AES algorithm

A client/sender has a highly confidential file needing to be transferred to the server/receiver over the same network. The file is located within the client's computer. The file should be transferred without being altered or captured by packet sniffers. Firstly, a server needs to run the system to enable the client to send the file. Then, the client runs the system to browse for the file in the computer and choose the file they want to send to the

server/receiver. Next, after finishing browsing and choosing the file, the encryption process begins when the send button is clicked. Now, the file is encrypted and transferred over the LAN securely. When the file is received by the server, the decryption process begins to convert the encrypted file to its original file, which can then be read by the server. The file is stored in the local disk in the server's computer.

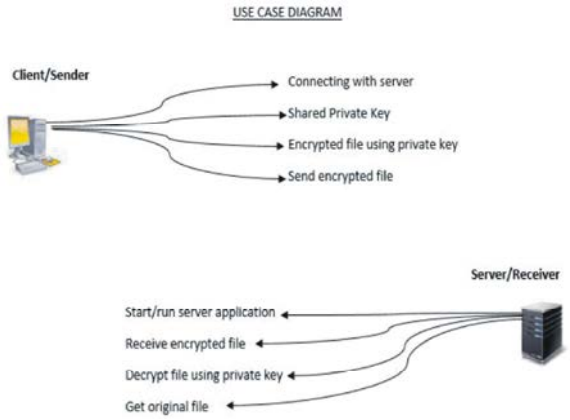

Fig. 9: Use Case Diagram

Research Development

- Create two program client and server side by using Netbeans IDE 7.2 application.
- Use Java Socket Programming to make TCP communication protocol.
- Create a socket object in the server and client side to enable communication.
- Use a method of java.net.Socket class represents a socket in Client side.
- Use a method of java.net.ServerSocket class in the Server side for the server program to listen for clients and establish connections with them.
- The server instantiates a ServerSocket object, denoting which port number communication is to occur on.
- The server invokes the accept() method of the ServerSocket class. This method waits until a client connects to the server on the given port.
- After the server is waiting, a client instantiates a Socket object, specifying the server name and port number to connect to.
- The constructor of the Socket class attempts to connect the client to the specified server and port number. If communication is established, the client now has a Socket object capable of communicating with the server.
- Each socket has both an OutputStream and an InputStream. The client's OutputStream is connected to the server's InputStream and the client's InputStream is connected to the server's OutputStream.
- But for this application, the Server side can only transfer or send data to the client and the Client side can only receive data from the client.
- The InputStream is used to read data from a source and the OutputStream is used for writing data to a destination.
- Then, create the file object that will be transferred by using a method of FileInputStream and FileOutputStream.
- The file object will be converted into a byte stream by using a mechanism of serialization.
- Serialization in java is a mechanism of writing the state of an object into a byte stream.
- AES Encryption algorithm was implemented and starts working after file objects are converted into a byte stream. AES algorithm will encrypt the byte stream and convert it into the ciphertext before it being transferred to the receiver's side by using a Cipher class that provides the functionality of a cryptographic cipher used for encryption and decryption.
- Once the receiver's computer receives the encrypted byte stream or ciphertext, the AES decryption algorithm will start the process of converting the ciphertext into the file object by using Cipher class method.

Graphic User Interface (GUI): This interface had been developed by java programming in Netbeans application The sender's side must be run before the receiver's side. The user needs to know the IP address of the sender's computer and port number that has been used in the application. Then, the IP address field needs to be filled with the sender's IP address and port number that need to be connected.

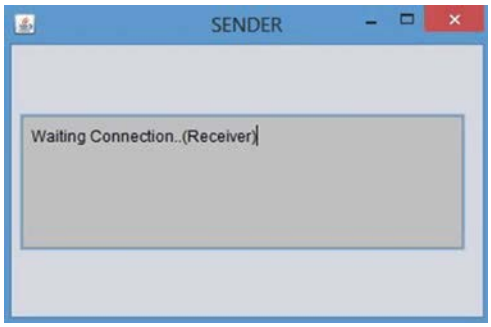

Fig. 10: Interface of Sender

First, when the user runs the sender's side, it will pop-up with the interface as shown in Figure 14. It will show the text "Waiting Connection…(Receiver)", meaning it is waiting for the receiver to connect with the same IP address and port number.

Fig. 11: Interface of Receiver

Next, when receiver's side has been run, the interface will pop-up, as shown in Figure 11. The user needs to input the IP address of the sender's computer and port number that have been used. Then, by clicking the "CONNECT" button, the text "Connected to Sender !" will appear to give an acknowledgement to the user that the connection has been established.

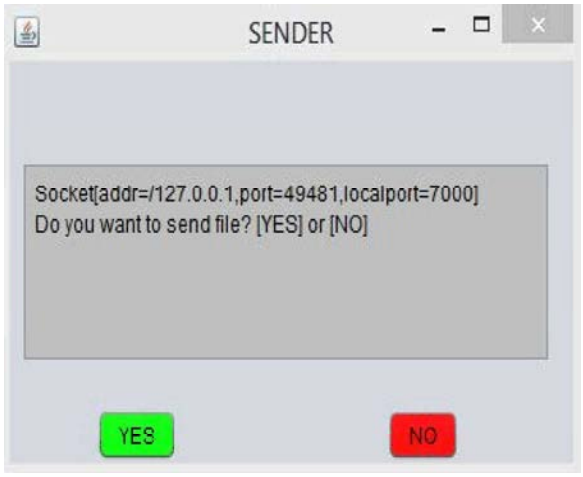

Fig. 12: Connection has been established

Whenever a sender and receiver side have been connected and connection has been established, the interface will change to another interface as shown in Figure 12 It will show the user IP address of the receiver and ask the user whether they want to send the file or not. If the user wants to send a file, they should click the green "YES" button and further to another interface. If not, they must click the red "NO" button and the connection will be terminated.

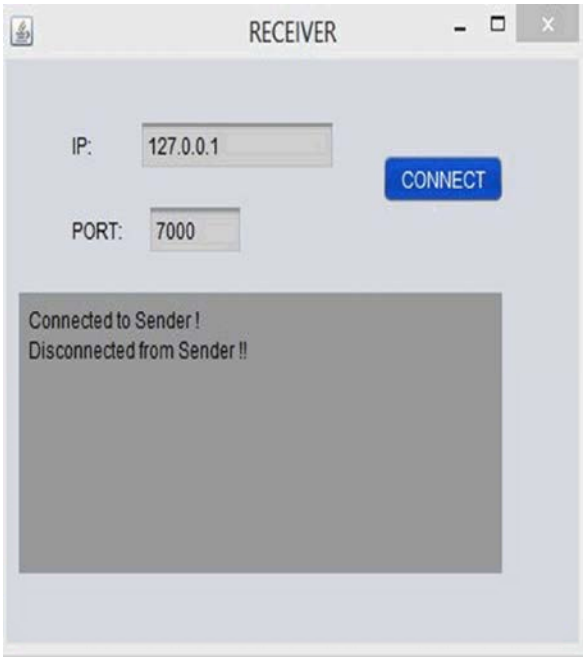

Fig. 13: Connection disconnected

The receiver will get the notification "Disconnected from Sender !" appearing in the text field, as shown in Figure 13 if the sender clicked the red "NO" button.

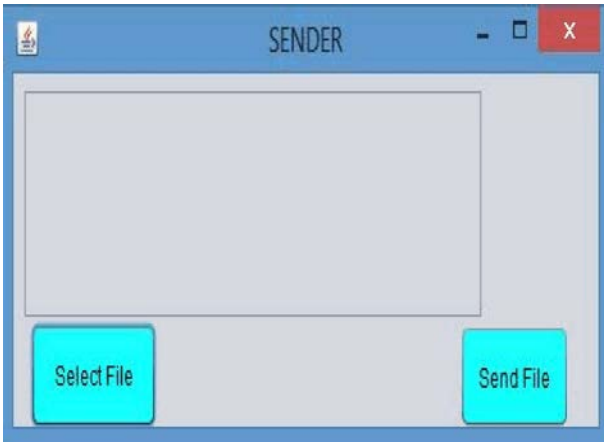

Fig. 14: File transferring interface

If the user clicked the green "YES" button, this interface will appear, requiring the user to browse files by clicking the "Select File" button. The interface in Figure14 will pop up and the user will have the ability to choose the file needing to be transferred to another computer. After the user has chosen a file they want to be transferred, the name of the file will appear in the select file name field.

Fig. 14: Process of transferring file

The file chosen by the user appears in the file name field. The user needs to click the "Open" button then, a cyan "Send File" button will appear, requiring the user to click that button to transfer the chosen file to the receiver's computer.

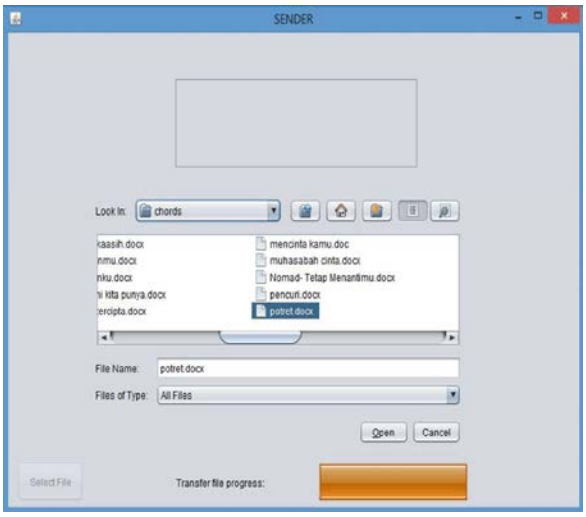

Fig. 15: Transferring file in progress

A progress bar will appear once the user clicks the "Send File" button. The user must wait until the progress bar is full to complete the file transfer. Whenever the file is received from sender's side, the name and format of the file will appear in the text field automatically. This allows the receiver to know the name and type of file that have been sent from the sender's computer. Once the transferred file has been received by the receiver side, the interface, as shown in Figure 16 will pops up. This interface is built to allow the receiver to rename the file and choose the path they want the file to be saved in. Then, after the user inputs the name and type of format of the file, the user must click the "Save" button to save the file.

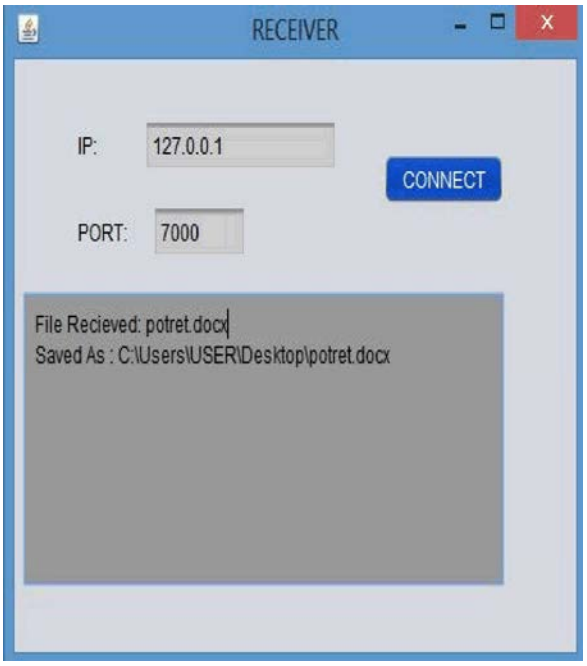

Fig. 16: File transferring completed

Finally, the path and name of the file saved by the receiver will appear in the text field to let the receiver know where to locate the file on the receiver's computer.

Testing and Result: Firstly, the sender and receiver are connected in the same network to establish a connection. Then, as an example for the testing, the "potret.docx" file was chosen by the sender from the "chords" folder in the sender "Documents". The file has been successfully transferred securely and the user can get the encrypted file from the "NetBeans" folder in the "My Document" folder. The encrypted file is named as "SenderEncryptedFile" and assigned a ".txt" format, enabling the user to view the content of the file.

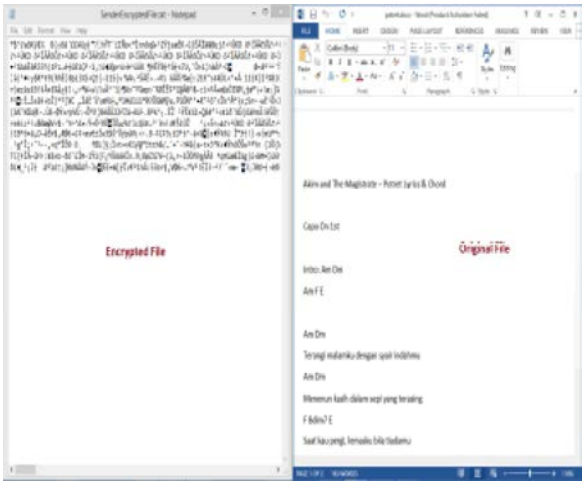

Fig. 17: Encrypted file

On the left is the encrypted file "potret.docx", which has been assigned as "SenderEncryptedFile.txt". On the right is the original file, "potret.docx".

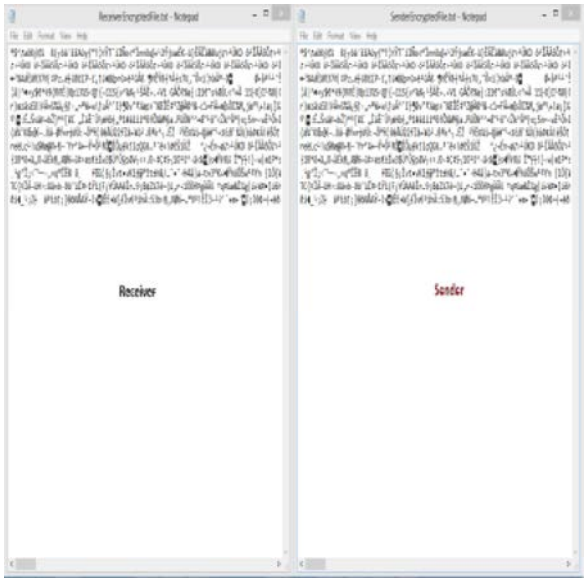

Fig. 18: Receiver received encrypted file

The SenderEncryptedFile is the actual file that has been sent to the receiver for the decryption process. The receiver also can get the encrypted file on the computer's desktop, written as ReceiverEncryptionFile. Figure 19 shows the same encrypted file that has been sent, showing that no alteration has occurred in the file.

On the left is the original file that has been sent and received by the receiver. On the right is the original file from the sender. There is no alteration, as indicated in the figure above. This shows the integrity of the file is high and guaranteed.

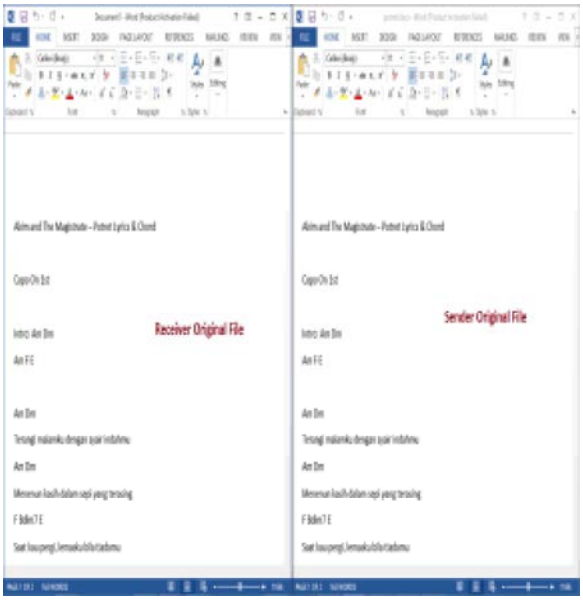

Fig. 19: Receiver received the file without alteration

Wireshark Packet Capture: In this project, AES algorithms are used to establish a secure file to transfer in a secure data connection using Transmission Control Protocol (TCP) between the receiver and sender over the same network. To test the security of the encrypted data, Wireshark [9] [10] is used to capture the packet transfer between the connection, specifically, to capture the packet flow between the client's IP and the server's IP. The Wireshark application can only capture the encrypted file that is transferred from the sender to the receiver. This indicates the transferring process is secure, as the file can only be viewed at the receiver site.

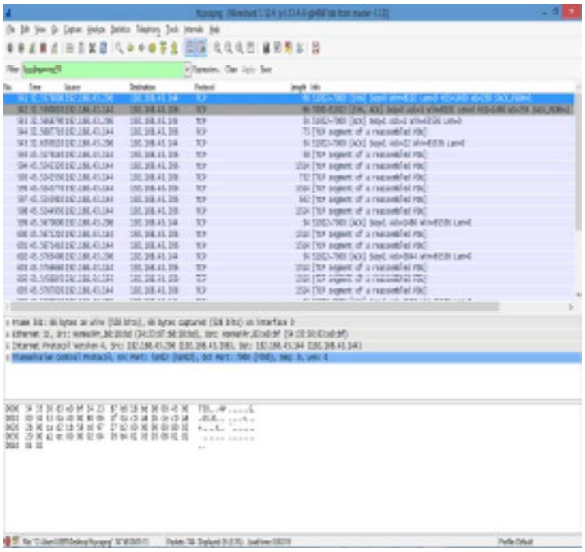

Fig. 20: Wireshark captures the network packets

Figure 20 shows the packet capture of the entire conversation between the IP address of the sender and the IP address of the receiver. In this packet capture, it will know the IP address of sender and receiver. It also can detect the port and protocol being used.

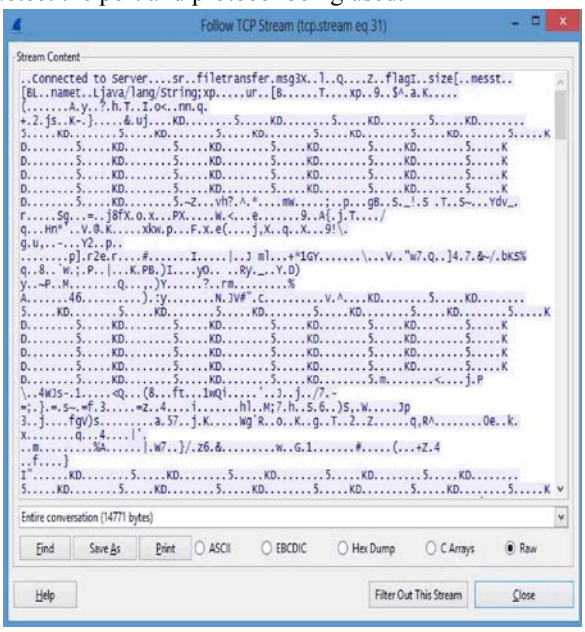

Fig. 21: Packet captured from TCP stream

The figure above shows the TCP stream of the packet capture from the IP address of the sender to the IP address of the receiver. It shows the entire conversation from the beginning until the end. Select the raw data option and save file as (.txt).

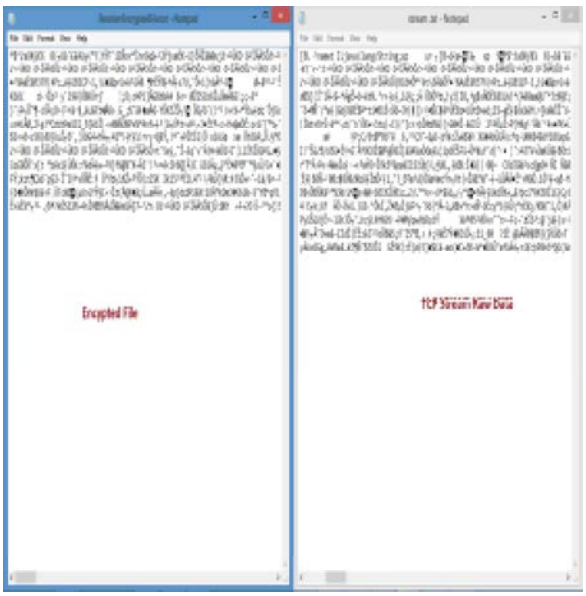

Fig. 22: Raw data captured by wireshark

On the left is the encrypted file from the sender and on the right is the TCP Stream raw data from the Wireshark packet capture. The contents of the two files are the same, as shown in figure 22. This proves that the file transferred is secure from hackers or the blackhat community.

CONCLUSION

During file transfers from sender to receiver, files are exposed to interception by hackers. Without any encapsulation, files can easily be sniffed. Therefore, this research has leveraged AES algorithm to secure file transferring systems. AES is part of a well-known cryptography. It is believed that applying AES in file transferring systems can hinder hackers from trying to steal any information during file transfers.

REFERENCES

1. Mahajan, Prerna and Abhishek Sachdeva, 2013. A Study of Encryption Algorithms AES, DES and RSA for security. Global Journal of Computer Science and Technology.
2. Balakrishnan, Bhargav, 2010. Three Tier Encryption Algorithm for Secure File Transfer. Computer Engineering and Applications (ICCEA), 2010 Second International Conference on. Vol. 2. IEEE, 2010.
3. Tayde, Suchita and Seema Siledar, 2015. File Encryption, Decryption Using AES Algorithm in Android Phone. International Journel of Advanced Research in Computer Science and Software Engineering, 5.5.
4. Rijmen, Vincent and Joan Daemen, 2001. Advanced encryption standard. Proceedings of Federal Information Processing Standards Publications, National Institute of Standards and Technology, pp: 19-22.
5. Dhongade, Sudhir, *et al*., 2015. An Efficient Certificateless Encryption for Secure Data Sharing Over the Network Using AES-128 and AES-256..
6. Benson, Calum, Matthias Muller-Prove and Jiri Mzourek, 2004. Professional usability in open source projects: GNOME, OpenOffice. org, NetBeans. CHI'04 extended abstracts on Human Factors in Computing Systems. ACM, 2004.
7. Granado-Criado, José M., *et al*., 2010. A new methodology to implement the AES algorithm using partial and dynamic reconfiguration. INTEGRATION, the VLSI Journal, 43(1): 72-80.

8. Biryukov, Alex and Ivica Nikoliæ, 2010. Automatic search for related-key differential characteristics in byte-oriented block ciphers: application to AES, Camellia, Khazad and others. Advances in Cryptology–EUROCRYPT 2010, pp: 322-344.
9. Sanders, Chris, Practical packet analysis: Using Wireshark to solve real-world network problems. No Starch Press, 2017.
10. Banerjee, Usha, Ashutosh Vashishtha and Mukul Saxena, 2010. Evaluation of the Capabilities of WireShark as a tool for Intrusion Detection. International Journal of Computer Applications, 6.7.