VOL 1 (2017) NO 4 - 2

e-ISSN : 2549-9904
ISSN : 2549-9610

INTERNATIONAL JOURNAL ON INFORMATICS VISUALIZATION

Secret Channel Using Video Steganography

Lee Kar Yee[#], Chuah Chai Wen[#]

[#] *Information Security Interest Group (ISIG), Faculty Computer Science and Information Technology, University Tun Hussein Onn Malaysia*
E-mail: leekaryee19@gmail.com, cwchuah@uthm.edu.my

***Abstract*— Video steganography is about hiding the secret message into the video as the cover media. Steganography is a technique use to secure the transmission of secret information or hide their existence. Most of the steganography software such as DeEgger Embedder and OurSecret does not have encryption algorithm to preserve the confidentiality for the secret message. OurSecret software only using simple password to keep the secrecy of the message. A simple password is not strong enough to protect the secret message, secret message will be vulnerable once the attacker successfully attack the steganography technique use to hide the secret message in the video. Furthermore, password is not an encryption algorithm. Secret message without encrypted is the vulnerability to the secret message. Confidentiality of the secret message will not be able to preserve. Hence there is a need to encrypt the secret message with stream cipher while not leaving it vulnerable to attacker. Stream cipher provide bit by bit encryption. A video steganography with embedding text with RC4 encryption cipher is design, develop and test in this project.**

***Keywords*— Video steganography, Confidentiality, Encryption, RC4.**

I. Introduction

Cryptography and steganography are two widely used technique in protecting the messages from being observed by unauthorized third parties. These techniques are being applied in applications such as email, web chatting, instant banking, etc. Cryptography algorithms are may preserve the confidentiality, integrity and availability of the message and information. Steganography algorithms are used to hide messages inside a media cover.

Cryptographic algorithms consist of two major processes in preserving the confidentiality of the secret message and information. These processes are known as encryption and decryption. Encryption is the process to convert plaintext together with encryption key into ciphertext. Ciphertext is unreadable by third parties. Decryption is the process to convert the ciphertext with decryption key into plaintext. Encryption key and decryption key are the same and is known as symmetric cipher. Example of symmetric ciphers are block ciphers and stream ciphers. Encryption key and decryption key are different are known as asymmetric cipher. Example of asymmetric ciphers are Diffie – Hellman key exchange, Rivest – Shamir – Adleman (RSA).

Stream cipher is a type of symmetric encryption algorithm. Stream ciphers are designed to be a much faster cipher. Stream ciphers typically operate on smaller units of plaintext, usually bits. With a stream cipher, the transformation of these smaller plaintext units will vary. Its depending on when they are encountered during the encryption process. A stream cipher generates a sequence of bits used as a key name keystream. The keystream is then used with the plaintext with XOR operation. Examples of stream ciphers are RC4. RC4 is a stream cipher designed by Rivest for RSA Data Security. This type of cipher has a variable key – size stream cipher with byte – oriented operations. The key size range is 40 bits to 2048 bits. The algorithm is based in the use of a random permutation.

Steganography is hiding a secret message into another media file. The types of media file can be image, video or audio. The main goal of using steganography is to hide the secret message from third parties. Image steganography is common used and the method used is least significant bit. Audio steganography and video steganography are using the same method as image steganography. However, there is the need of masking for audio steganography and video steganography. This is due to human eye and ear are sensitive to any changes in audio and video respectively. By doing this, it can exploit the properties of human sensory organ to hide the secret message.

In this project, a tool to encrypt and decrypt the secret message, and embed the ciphertext into FLV video is proposed and is developed. The secret message is encrypted using RC4 algorithm and produce the ciphertext. To ensure the messages sent out without third parties' perceptiveness, the encrypted message will be embedded using steganography techniques. Images, audio and video are the three basic mediums that can embed the message. However, images and audios are limited to short message only. The secret messages that are to be embedded are long and thus, videos are the best choice among these three. Least Significant Bit (LSB) is the method use for embedding the messages due to its simplicity. In this method, it usually just focus on the last right-most bit and replaced the bits in order to hide the information. The types of famous video file format are AVI, FLV, MPEG, MP4 and WMV. In this project, FLV files act as the cover media.

II. Literature Review

In this section, background materials that are used to build up the secret channel using video steganography are presented.

A. *Cryptography*

Cryptography is a combination of word that originally from Greek which mean "hidden/secret" and "writing". The purpose of cryptography is to keep the secrecy of the secret message but do not hide the existence of the message by using encryption process that change the plaintext in to ciphertext [1]. There are two basic processes in cryptography which are encryption of plaintext to ciphertext and decryption of ciphertext to plaintext as shown in Fig. 1. There are five important elements in a cryptosystems which is plaintext, encryption algorithm, key, ciphertext, and decryption cryptosystem [1]. There are two types of ciphers which is symmetric cipher and asymmetric cipher.

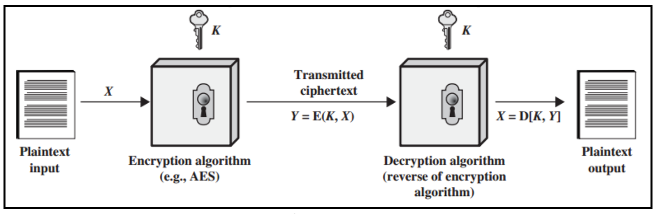

K
K
X
Transmitted ciphertext
Y = E(K, X)
X = D[K, Y]
Plaintext input
Encryption algorithm (e.g., AES)
Decryption algorithm (reverse of encryption algorithm)
Plaintext output

Fig. 1 Overview of Simple Cryptosystem [1].

B. *Symmetric Cipher*

Symmetric cipher uses same key in encryption and decryption process [2]. The key is shared by the sender and the receiver. The same key are uses for encryption and decryption process. Symmetric cipher consists of two categories which are stream ciphers and block ciphers.

C. *Asymmetric Cipher*

Asymmetric ciphers use different keys in encryption and decryption process [3]. For encryption, sender uses receiver public key to encrypt the message and generate the ciphertext. The receiver uses his private key to decrypt the ciphertext and retrieve back the original message.

D. *Stream Cipher*

A typical stream cipher encrypts plaintext one byte at a time, although a stream cipher may be designed to operate on one bit at a time or on units larger than a byte at a time [1]. A key is input to a pseudorandom bit generator that produces a stream of 8-bit numbers that are apparently random. A pseudorandom stream is one that is unpredictable without knowledge of the input key. The output of the pseudorandom key generator is called keystream. The keystream is then combined one byte at a time with the plaintext stream using the bitwise exclusive-OR (XOR) operation. Example of stream cipher are RC4 [1], Trivium Cipher [4] and Grain [5].

RC4 [1]: RC4 is one of the stream cipher that is remarkable for its simplicity. RC4 algorithm is a variable key-size stream with byte-oriented operations. The algorithm is based on the use of random permutation. A variable-length key from 1 to 256 bytes is used to initialize a 256-byte state vector S, with elements S[0], S[1], …, S[255]. S contains a permutation of all 8-bit numbers from 0 through 255 at any times. A byte of key is generated from S by selecting one of the 255 entries in a systematic fashion.

In order to initialize the S, the entries of S are set equal to the values from 0 to 255 in ascending order. In array form, it looks like this, S[0]=0, S[1]=1, …, S[255]=255. A temporary vector, T, is also created. The purpose of T is when the length of the key is 256 bytes, K is transfer to T. Otherwise, for a key of length keylen bytes, the first keylen elements of T are copied from key and then the key is replaced as many times as necessary to fill out the T.

E. *Steganography*

Steganography is a combination of word that originally from Greek which means 'covered" and "writing" [6]. The main objective of steganography is to hide the existence of the secret message by using another type of file as cover media. It is different from cryptography because the result of cryptography is ciphertext that can be seen by third parties while the result of steganography is the secret message is totally hide from human eyes [7]. There are three main elements in the process of steganography which is the hidden data, the cover-media and the stego key. The cover media is the file which will hide the hidden data, which may also be encrypted using the stego key. There are three major type of cover media which is image, audio and video.

LSB method is a method that exploits the fact that the level of precision in many image formats is far greater than that perceivable by average human vision. LSB method basically changing the least bytes of the media used into a new bytes of the secret message that which to hide [8]. Least significant bit changes the colour value by one. Embedding into the second bit-plane can change the colour value by 2. If embedding is performed on the least significant two pixels, the result is that a colour in the cover media can be any of four colours after embedding.

Video Steganography: Video steganography is a combination of image steganography techniques and audio steganography techniques [9]. A successful video steganography should be able to maintain the quality of the video. It should be almost similar to the quality of the video before the embedding process. If the third parties able to differentiate the difference of the video, then it is considering a fail result and the secret message will be review.

F. Least Significant Bit (LSB) Method

LSB method is a common embedding technique. This method is a simple approach to embedding information in a cover file. This method suggests that data can be hidden in the least significant bits of the cover file. This results that human eye would be unable to notice the hidden message in the cover file [8].

G. FLV File Format

The FLV file format is consists of a short header, followed by the metadata, and then interleaving audio and video tags of packets. In the file header of FLV file header, it consists of the first three hex values of 46, 4C and 56. The three hex values are translating to the letter "F", "L", and "V". It then followed by the version, flags, and offset. After these values is a sequence of tags until the end of file (EOF). As for the tags, it consists of audio tags and video tags, 0x08 for AUDIO and 0x09 for VIDEO, and lastly 0x12 for META. In each tags, it contains the type, body length, timestamp, timestamp extended, streamID, and the body (actual data) [10].

H. DeEgger Embedder [11]

DeEgger Embedder is a free video steganography software that works on both 32-bit and 64-bit OS editions of Windows and supports from Windows XP to Windows 8. This free video steganography software is designed to merge two or more files together, supports a wide range of file extensions user can hide and lets user create the bogus file in AVI, JPEG, MP3, MP4, PDF, PNG and other formats. Beside than hiding information, this same software is used to extract and retrieve the secret message that user hide.

I. OurSecret[12]

OurSecret is a free steganography tools. This software operates on Windows only. It can embed multiple secret message into a carrier file. The carrier file type that support are image, audio and video. The same software is used to extract and retrieve the secret message that user hide.

III. Methodology

Object-Oriented Software Development (OOSD) consists of four main phase which are Object-Oriented Analysis Phase, Object-Oriented Design Phase, Object-Oriented Implementation Phase, and Testing and Maintenance Phase [13].

A. Object-Oriented Analysis Phase

This phase is to determine the system requirements and functions by collecting user's needs. The user of this tools is decided to be everyone. A list of specification such as user requirements and system requirements of this tools is being listed down. A further study of cryptography and video steganography is carry out in order to fulfil the user and system requirements. In system requirements, it is divided into two part which is functional requirement and non-functional requirement. The function of the video steganography has been identifying out and included into this video steganography tools. The steganography algorithm is identifying out which is using LSB method. As for the security part of this tools, stream cipher is the one that is been investigate in order to include into this video steganography tools.

B. Object-Oriented Design Phase

In this phase, Unified Modelling Language Diagram is generated base on the requirements gather in analysis phase. The user interface of this video steganography tools is designed base on the requirements in analysis phase. Besides, the system design consists of preserving confidentiality of the secret message by implementing RC4 encryption and steganography techniques.

C. Object-Oriented Implementation Phase

In this phase, it includes all information for tools to execute and to develop the tools. Java programming language is chosen to develop the proposed tools. As for the encryption module, RC4 algorithm is use in order to meet user requirements. The user interface is also developing to connect with functionalities of program.

D. Testing and Maintenance

Testing phase are important for system so that error and bugs can be identify out and fix it. Testing phase is to ensure that the interaction of coding is fully function. Testing phase is carry out during the tools testing. Twenty candidates as user is choose to test the tools to ensure that the quality of tools and it meet the user's needs. The testing of this tools include functional testing for encryption module, functional testing for embedding module, functional testing for extracting module and functional testing for decryption module.

Maintenance phase is another part of phase that come after implementation phase. The results from the testing of this proposed tools are analyzed. Improvement is made in order to fulfil user requirements and system requirements.

IV. Object oriented analysis and design

This section presents the process of analysis and design of proposed tool. UML diagram such as sequence diagram, activity diagram and class diagram are discussed to provide a clearer image of tool flow.

A. Sequence Diagram

Sequence diagram explains the use case diagram. The sequence diagram for user to embed secret message in video is shown in Fig. 2. User first inputs message, video and password before embedding process. Message is encrypted into ciphertext and then the ciphertext is embed into the video.

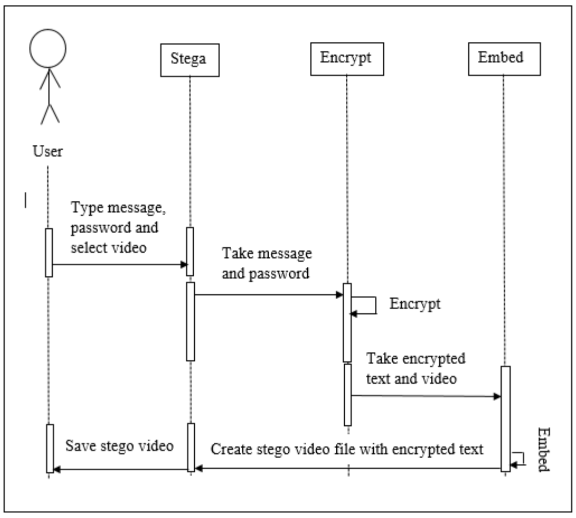
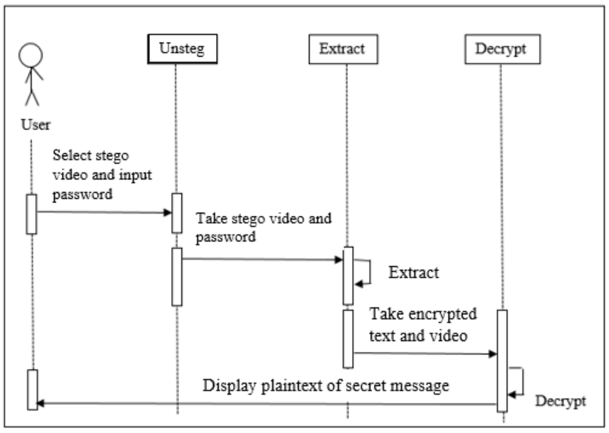

User
Stega
Encrypt
Embed
Type message, password and select video
Take message and password
Encrypt
Take encrypted text and video
Save stego video
Create stego video file with encrypted text
Embed
Fig. 2 Sequence Diagram for user to embed secret message in video

Fig. 3 shows the sequence diagram for user to extract secret message. User inputs the stego video and correct password. Encrypted message is extracted from the video. Ciphertext is decrypted using the password.


User
Unsteg
Extract
Decrypt
Select stego video and input password
Take stego video and password
Extract
Take encrypted text and video
Display plaintext of secret message
Decrypt
Fig. 3 Sequence Diagram for user to extract secret message

B. *Activity Diagram*

Activity diagram is used to present the data flow of this proposed tool. Its notation and origin are based on the widely known flowchart notation. Fig. 4 shows the activity diagram for the proposed tool. Firstly, user can either input the secret message or video file. The secret message area is filled before go to next step which is encryption. The video input is validated to determine the format is in FLV video format or not. Next, key and cover video are needed to generate the stego video. The area that type the key is validated whether the key is entered or not. After user choose the output folder location, stego video is generated.

In order to extract the secret message, user is required to choose and input the stego video. The same key is needed to extract the secret message back. The whole process is terminate if the key input is not match with the previous key use for embed. The original message is displayed to user if key insert is correct.

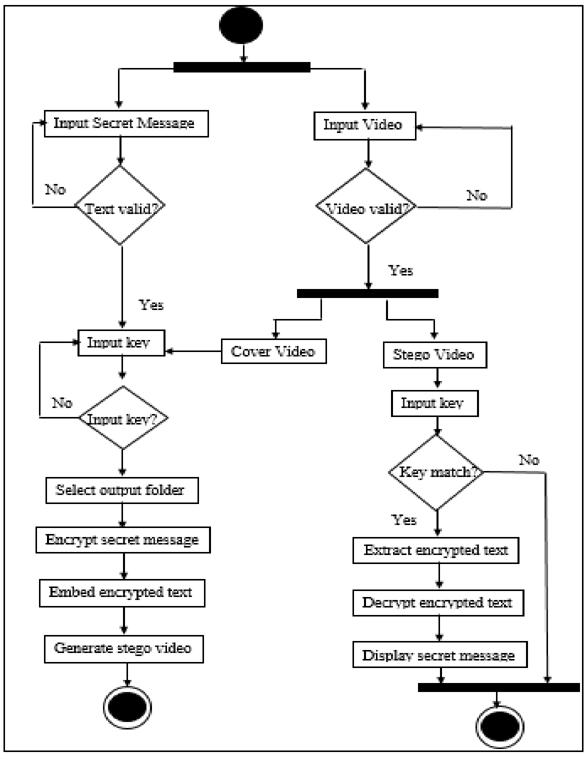
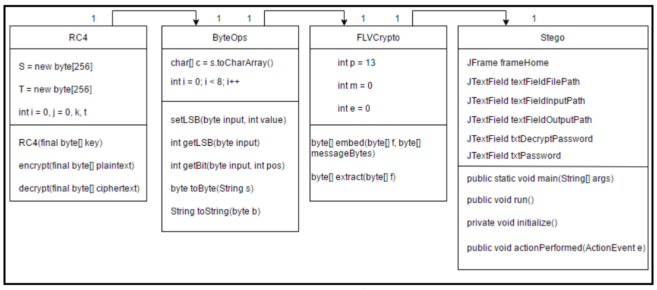


Fig. 4 Activity Diagram for Proposed Tool

C. *Class Diagram*

Class diagram present the relationships between classes. Fig. 5 shows the class diagram for the proposed tool. There are total four main classes in proposed tool. RC4 class is mainly for user to encrypt and decrypt secret message with password. ByteOps class is for carry out byte operations such as changing the byte and the LSB method. FLVCrypto class is for the embedding process and extraction process. Lastly, Stego class is the class that call other class to run it. It also involves the user interface design.


1
1
1
1
1
1
RC4
S = new byte[256]
T = new byte[256]
int i = 0, j = 0, k, t
RC4(final byte[] key)
encrypt(final byte[] plaintext)
decrypt(final byte[] ciphertext)
ByteOps
char[] c = s.toCharArray()
int i = 0; i < 8; i++
setLSB(byte input, int value)
int getLSB(byte input)
int getBit(byte input, int pos)
byte toByte(String s)
String toString(byte b)
FLVCrypto
int p = 13
int m = 0
int e = 0
byte[] embed(byte[] f, byte[] messageBytes)
byte[] extract(byte[] f)
Stego
JFrame frameHome
JTextField textFieldFilePath
JTextField textFieldInputPath
JTextField textFieldOutputPath
JTextField txtDecryptPassword
JTextField txtPassword
public static void main(String[] args)
public void run()
private void initialize()
public void actionPerformed(ActionEvent e)
Fig. 5 Class Diagram for proposed tool

V. System Implementation and Testing

Java programming language is chosen to develop the proposed tool. Fig. 6 shows the RC4 encryption and decryption module implemented in proposed tool.

```
public byte[] encrypt(final byte[] plaintext) {
    final byte[] ciphertext = new byte[plaintext.length];
    int i = 0, j = 0, k, t;
    byte tmp;
    for (int counter = 0; counter < plaintext.length; counter++)
{
        i = (i + 1) & 0xFF;
        j = (j + S[i]) & 0xFF;
        tmp = S[j];
        S[j] = S[i];
        S[i] = tmp;
        t = (S[i] + S[j]) & 0xFF;
        k = S[t];
        ciphertext[counter] = (byte) (plaintext[counter] ^ k);
    }
    return ciphertext;
  }
  public byte[] decrypt(final byte[] ciphertext) {
    return encrypt(ciphertext);
  }
```

Fig. 6 Source Code for Encryption and Decryption Module [1]

Fig. 7 shows the source code for embedding module. The function of this module is to embed the ciphertext into video.

```
public byte[] embed(byte[] f, byte[] messageBytes) throws Exception{
            System.out.println("=== Embedding Process Started ===");
            String message = new String();
            for (int i = 0; i < messageBytes.length; i++) {
                    byte b = messageBytes[i];
                    message += ByteOps.toString(b);}
            if(!((f[0] & 0xFF) == 0x46 && (f[1] & 0xFF) == 0x4c && (f[2] &
0xFF) == 0x56 && (f[3] & 0xFF) == 0x01)){throw new Exception("Not a FLV
file");}
            int typeflags = (f[4] & 0x05);
            boolean audio = ((typeflags >>> 2) & 0x01) != 0;
            boolean video = (typeflags & 0x01) != 0;
            if(audio) System.out.println("Audio tags exist");
            if(video) System.out.println("Video tags exist");
            if(!video) throw new Exception("No video to embed the message in");
            int p = 13, m = 0, e = 0;
            while(p < (f.length - 13)){
                    boolean isVideoTag = false;
                    if((f[p] & 0xFF) == 8){}
                    else if((f[p] & 0xFF) == 9){isVideoTag = true;}
                    else if((f[p] & 0xFF) == 18){}
                    else {}
int i = ( (f[p+1] & 0xFF) << 16) + ((f[p+2] & 0xFF) << 8) + (f[p+3] & 0xFF);
                    int q = (p + i + 10);
                    if(isVideoTag){if (m < message.length()) {
                                int c = message.charAt(m) - '0';
                                f[q] = ByteOps.setLSB(f[q], c);
                                System.out.print(c);
                                m++;}
                                else if (e < EOF_SIZE) {
                                f[q] = ByteOps.setLSB(f[q], 0);
                                System.out.print(0);
                    e++;} else {System.out.println("E before Loop exited : " + e);
                                break;}}
                    q++;
int r = ((f[q] & 0xFF) << 24) + ((f[q+1] & 0xFF) << 16) + ((f[q+2] & 0xFF) << 8) +
(f[q+3] & 0xFF);
                    if(q - p == r) {p = q + 4;}
                    else {
            throw new Exception("Tag size mismatch. Check your logic");}}
            System.out.println("All tags iterated");
            System.out.println("=== Embedding Process Completed ===");
            return f;
        }
```

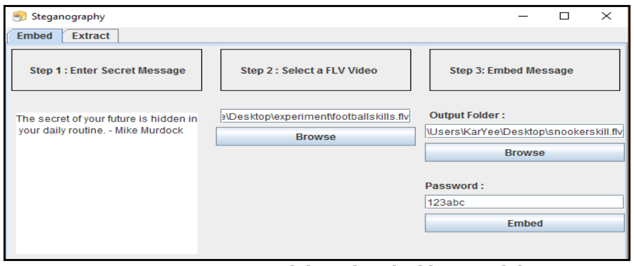
Fig. 7 Source Code for Embedding Module

Fig. 8 shows the source code for extraction module. The function of this module is to extract the ciphertext from the video.

```
public byte[] extract(byte[] f) throws Exception{
            System.out.println("=== Extraction Process Started ===");
if(!((f[0] & 0xFF) == 0x46 && (f[1] & 0xFF) == 0x4c && (f[2] & 0xFF) == 0x56
&& (f[3] & 0xFF) == 0x01)){throw new Exception("Not a FLV file");}
            int typeflags = (f[4] & 0x05);
            boolean audio = ((typeflags >>> 2) & 0x01) != 0;
            boolean video = (typeflags & 0x01) != 0;
            if(audio) System.out.println("Audio tags exist");
            if(video) System.out.println("Video tags exist");
            if(!video) throw new Exception("No video to embed the message in");
            int p = 13;
            String message = new String();
            int e = 0;
            while(p < f.length){boolean isVideoTag = false;
                    if((f[p] & 0xFF) == 8){}
                            else if((f[p] & 0xFF) == 9){isVideoTag = true;}
                            else if((f[p] & 0xFF) == 18){} else {}
int i = ( (f[p+1] & 0xFF) << 16) + ((f[p+2] & 0xFF) << 8) + (f[p+3] & 0xFF);
            int q = (p + i + 10);
                    if(isVideoTag){int b = ByteOps.getLSB(f[q]);
                            message += b;
                            System.out.print(b);
                    if(b == 0){e = e + 1;} else {e = 0;}
            if(e == EOF_SIZE) {System.out.println("EOF Reached. Breaking loop. ");
break;}}
                    q++;
int r = ((f[q] & 0xFF) << 24) + ((f[q+1] & 0xFF) << 16) + ((f[q+2] & 0xFF) << 8) +
(f[q+3] & 0xFF);
if(q - p == r) {p = q + 4;} else {throw new Exception("Tag size mismatch. Check
your logic");}}
                    System.out.println(message.length());
                    int len = (message.length() - e) / 8;
                    boolean k = false;
                    byte[] result = new byte[len + 1];
                    for (int i = 0; i < len + 1; i++) {int d = 0;
                    if(8 * (i+1) > message.length()){d = 8 * (i+1) - message.length();}
            String byteString = message.substring(8 * i, 8 * (i+1) - d);
                            if(d > 0){for (int j = 0; j < d; j++) {byteString += "0";}}
                            byte b = ByteOps.toByte(byteString);
                            if( i != len){result[i] = b;}
                            else if(i == len && b != 0){result[i] = b;
                                    k = true;}}
                    System.out.println("=== Extraction Process Completed ===");
                    if(k){return result;} else {return Arrays.copyOfRange(result, 0,
result.length - 1)}}
```

Fig. 8 Source Code for Extracting Module

Functional testing is technique used in order to test the functionality of this tool. This phase is to ensure the functions work properly and meet the user requirements. Input is use for the functional testing and output is examine. The testing of the main functionalities of secret channel using video steganography are presented. Fig. 9 shows the encryption module and embedding module. User must choose the cover file, enter secret message choose the output folder and enter the password. The "Embed" button will run the encryption module and the embedding module. Fig. 10 shows the extracting module and decryption module. User must choose the correct stego video and enter the correct password in order to retrieve the secret message. The "Extract Message" button will start to extract once the user click. Both major functions are operated as expected.



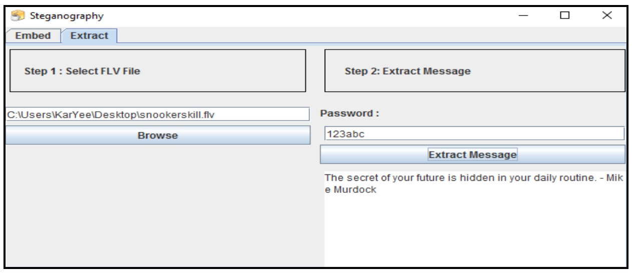
The secret of your future is hidden in your daily routine. - Mike Murdock

e\Desktop\experiment\footballskills.flv

Output Folder : \Users\KarYee\Desktop\snookerskill.flv

Password : 123abc

Fig. 9 Encryption Module and Embedding Module



C:\Users\KarYee\Desktop\snookerskill.flv

Password : 123abc

The secret of your future is hidden in your daily routine. - Mike Murdock

Fig. 10 Extracting Module and Decryption Module

VI. Result and Discussion

In this section, acceptance testing is provided and is carried out with 20 users. The two major feedbacks are required from the users are functionality test shows in Fig. 11. User friendly test in Fig. 12. At the end of this testing, the secret channel using video steganography tool operates 90% successfully as shown in Figure 7.1. The other 10% happened on the extraction module where the users are not able to get back the secret message due to enter the password incorrectly. The correct secret message will only display when the users input the correct video and the correct password.

The next test is about the user friendly test for the proposed tool and is show in Figure 7.2. There are 85% of users agree to the design is user friendly, 5% of users that disagree and lastly there are 10% of users not sure either the design is user friendly or not. The reason that the 5% person user disagree is because they prefer the both embed and extract function appear on the same tab page. So that they don't need to change between tabs when they want to use the tool.

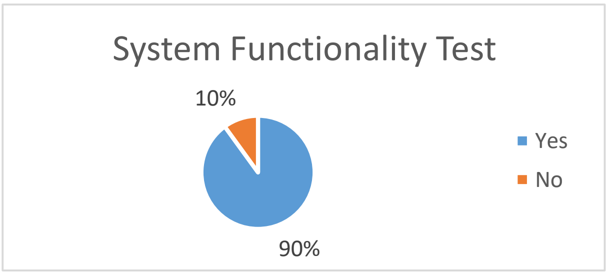


Fig. 11 System Functionality Test

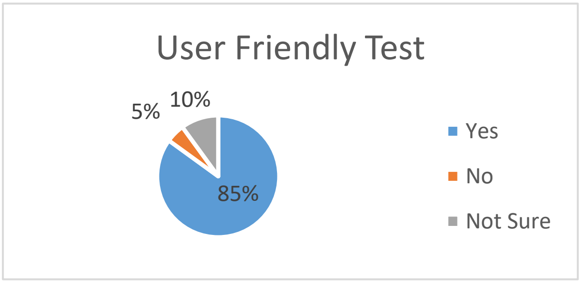


Fig. 12 User Friendly Test

VII. Conclusions

The secret channel using video steganography tool is designed, developed and tested. This tool has been successfully developed and meet this targeted project aim. This tool provide confidentiality to the users' message, as user can embed the secret message into a FLV file. Besides, this tool provides encryption algorithm which is RC4 algorithm to encrypt and decrypt the plaintext. However, the limitation for this tool is that the tool only support one video file which is FLV video. Therefore, more variety of video files format can be added to the tool as cover media.

Acknowledgment

This research was supported by RMC UTHM and Gates IT Solution Sdn. Bhd.

References

[1] W. Stallings, *Cryptography and Network Security: Principles and Practices 7th Edition*, Pearson Education India, 2017.
[2] H. Delfs and H. Knebl, *Symmetric-key Encryption*, Journal of Introduction to Cryptography, Publisher Springer, pp. 11-31, 2007.
[3] N. Daswani, C. Kern and A. Kesavan, *Asymmetric Key Cryptography*, Journal of Foundations of Security: What Every Programmer Needs to Know, Publisher Springer, pp221-226, 2007
[4] C. D. Canniere and B. Preneel, *Trivium*, Journal in New Stream Cipher Designs, Publisher Springer, pp. 224-266, 2008.
[5] M. Hell, T. Johansson and W. Meier, *Grain: A Stream Cipher for Constrained Environments*, Journal in International Journal of Wireless and Mobile Computing, Publisher Inderscience, Vol 2, No. 1, pp. 86-93, 2007.
[6] J. Neil F and J. Sushil, *Exploring Steganography: Seeing the Unseen*, Journal of Computer, Publisher IEEE, Vol. 31, No. 2, 1998.
[7] P. Fabien AP, A. Ross J and K. Markus G, *Information Hiding-A Survey*, Proceeding of the IEEE, Publisher IEEE, Vol. 87, No. 7, pp. 1062-1078, 1999.
[8] BS. Champakamala, K. Padmini and DK. Radhika, *Least Significant Bit Algorithm for Image Steganography*, Journal of International Journal of Advance Computer Technology, Vol. 3, No. 4, pp.34-38, 2013.
[9] AP. Sherly and PP. Amritha, *A Compressed Video Steganography Using TPVD*, International Journal of Database Management System (IJDMS), Vol. 2, No. 3, 2010.
[10] J. P. Cruz, N. J. Libatique and G. Tangonan, *Steganography and Data Hiding in Flash Video (FLV)*, TENCON 2012-2012 IEEE Region 10 Conference, pp. 1-6, 2012.
[11] DeEgger Embedder, (2017). URL: http://deegger-embedder.findmysoft.com/.
[12] OurSecret, (2017). URL: http://oursecret.soft112.com/.
[13] J. B. Dundas, *Object-Oriented Software Development Life Cycle*, Proceedings of the 2nd National Conference, INDIACom, 2008.