INTERNATIONAL JOURNAL OF RESEARCH IN COMPUTER APPLICATIONS AND ROBOTICS
ISSN 2320-7345

A NOVEL FLV STEGANOGRAPHY APPROACH USING SECRET MESSAGE SEGMENTATION AND PACKETS REORDERING

Atef A. Obeidat, Mohammed J. Bawaneh

Department of Information Technology, AL-Huson Polytechnic, Al-Balqa Applied University, Jordan.
Email: atefob@gmail.com, mohmmed_jazi@yahoo.com

Abstract: - Steganography is not only the art of hiding the secret message in a cover media but also the existence of communication and secure data transferring. Today, there exist a lot of steganography techniques for building the secure data transferring communication between sender and receiver, but all of them are challenged and encountered by the steganalysis. This paper proposes a new secure technique flash video file (FLV) steganography that keeps the frame video quality and statistical undetectability. It looks to hide a secret message of any type inside a given FLV file. The secret message is divided into packets of the same length, reordered the packet bytes and then encrypted the bytes before hiding them at the end of a random selected video tags. The proposed method analyses the cover FLV file in order to find out the number and location of each video tag and then the data of secret message will be distributed randomly inside the random video tags through the Linear Congruent Generator (LCG) random generator. The existence of secret message is hard to be detected by the intruders or steganalysis due to the correct pre-knowledge that must be available for the receiver about the manipulation process, those knowledge are: the packets number, the packet length, the key of reordering secret message bytes, the key of video tag selection, the key agreement of decrypting method, the secret message length and the message extension. Experiment results explained that, the proposed technique satisfied the main requirements of steganography; visual appearance, modification rate, capacity, undetectability and robustness against extraction. It reached the maximum capacity for hiding the secret message with a modification rate equals 0.018.

Key words: FLV Steganography, LCG, Video tags, Message Segmentation, Packet Reordering

1. Introduction

The network environment turns into a substantial portion of humans work life or even of their normal life. Network provides us with a rapid carrier to deliver the digital data from one point to another in a convenience way. Users look to enjoy the merits that networks have provided in sending and receiving process. Sometimes they may want to keep the secret communications or the copyrights, so a various technologies have been recognized as a helpful ways in dealing with transferring data securely. The most common ones are steganography, cryptography and watermarking. Steganography has always been confused with encryption and

watermarking. Although they have the same goal (transferring data securely), but they are different in manipulating the data.

Steganography was derived from the Greek word steganos, meaning covered or secret, and graphy (writing or drawing) (Sneha et al., 2014). On the simplest level, steganography is the process of hidden writing, whether it consists of invisible ink on a paper or copyright information on a digital media in a vision to hide the existence of communication.

Steganography techniques can be clustered according to the type of cover file (Image, Audio, Text or Video steganography) or the manipulation procedure in the embedding process (Injection, Substitution, Distortion or Generation steganography) (Abdelwahab et al., 2008).

FLV file is a good hosting cover in the steganography process due its simplicity structure, ability to keep pictures quality as well as the sound quality and also the popularity within the internet websites.

This paper will present a novel approach for FLV steganography consolidated with the secret message packet segmentation to encounter the security threats against the confidential information. Here, the cover-object, where the data to be embedded, is always a FLV file, the embedded data could be a text, an image or any type of files. The input FLV file will be utilized to find out the positions of video tags. Those tags will be used to compute the number and length of possible packets that will be employed in the hidden process. The selected number of packets and length represent the key for data distribution inside the video tags. The user must insert a master key to be used in the manipulation process for hiding and extracting, also the selection of random video tags for data distribution and cryptography are based on the inserted master key. Secret message length, extension, packet length and packets number must be kept and known by the user in order to use them in extracting the secret message.

The rest of the paper is organized as follows. Section 2 presents related work. Section 3 gives an overview for FLV file structure. Section 4 describes the materials and methods. The experimental result is shown in Section5. Finally, section 6 concludes this paper with directions for future work.

2. Related Work

The ancient Greeks utilized the steganography technology to pass a secret message from one area to another. They employed a several cover hosts such as the head of slaves or the wood part of a wax to write the secret message. The message was inscribed on the head or the wood then sent to the receiver area (Sneha et al., 2014). Today, due to the availability of different types of files such as images, audios, videos and documents the door for building a new steganography techniques that embed the secret message is unfolded. This work focuses on the FLV file steganography, so the ulterior investigation presents some of the published studies in this field.

Shinde et al (2015) proposed a novel approach for video steganography that worked with multiple types of cover format (.mov,.mts,.flv,.mpeg). The secret message might be any type of data such as text, audio, video and image. It combined encryption, compression and embedding technique in one protection technique. Results show that, the existing video steganography techniques were worked only on .AVI files and secret message of type text or image.

Dan et al (2015) proposed a new technique to add a compressed data using the Huffman coding at the end of video tag. The data were compressed and distributed evenly within all video tags. Some data cannot be compressed so, the results show that, the level of success compression method reached 80% and capable of compressing up to 57%.

Bawaneh(2014) proposed a random LSB to embed the secret message inside the RGB color image. It used the linear congruent generator to finds out the location of random pixel in the cover image. The Secret key was a combination of four parameters (Seed, Multiplier, Non-common factor, and Cycle length). The method utilized red, green or blue channel to hide the message bit. The selection of channel to be used for hiding based on the modification rate for each channel. The minimum modified rate was employed to embed secret message. Results show that, the random LSB was better than the sequential LSB in term of visual appearance and satisfied sufficient security to secret message.

Kaur et al (2014) proposed a technique for video steganography that was called Hash-LSB combined with the RSA algorithm. The proposed method generated a mask pattern for the data bits by using a hashing function. It encrypted the secret message bytes before hiding them inside a cover video frame. Results show that, the method was secure against intruders due to the different levels of security.

Alwan(2013) proposed a modified or dynamic based least significant bit (LSB) technique to hide movie inside movie. The method used the least significant pixels from one image (frame) from the cover movie to hide the most significant pixels of the second frame in the hidden movie. The data of stego movie and cover movie were compared in term of noise ratio and mean square error.

Atiea et al (2012) proposed a novel FLV file information-embedding scheme. They used a weak point in the header information of the host FLV file to find out the robustness of compression process. The secret message as they mentioned can be reconstructed without knowing the original host FLV file. Experimental results explained that, the method was robust against lossless and lossy compression.

Cruz et al (2012) presented a study about the design, implementation and automatic tools for analyzing the FLV file. They proposed several methods for hiding data inside the different parts of FLV file; also they discussed the merits and demerits of each method. The methods were tested using the auditory visual test video tags histogram and RGB average analysis.

Dasgupta et al (2012) proposed a video steganography technique that bases on a hashing function for least significant bit (LSB). The data were embedded in the cover frame using LSB. Eight bits of each byte from the secret message was divided into 3,3,2 and embedded into the RGB pixel values of the cover frames. The hashing function was used to select the position of insertion in LSB bits. The data are analyzed in terms noise ratio, mean square error and image fidelity. They found an encouraging result in term of capacity in the tested video file.

3. FLV File Structure

The FLV file can be divided into two basic parts: FLV header and FLV stream. FLV header is a record with size 9 bytes that stores the type of FLV file, version, flow information (has audio and has video) and length of the header itself (ADOBE FLASH, 2010). Each FLV file consists of a stream of different tags. The tag holds information about the length of the previous tag, type of current tag, time stamp, stream ID and data size. The back pointer in each data tag is constructed from four bytes to determine the size of the previous tag. In general, each tag in the FLV file consists of two parts: Header and Data. Tag Header determines the type of tag, length and other information about tag. Data area can be divided into three types according to the type of tag; those areas are audio data, video data, and metadata. Audio data contains the information about the used audio in the tag such as format, sampling rate, length and audio. Video tag contains a data to determine the frame type of video, staring position of tag, size of tag and coded ID. The metadata tag stores a general data about the FLV file such as the information about storage, duration, audio data rate, creator or owner, width, player and date of creation. **Figure 1** shows the structure and components of the FLV file.

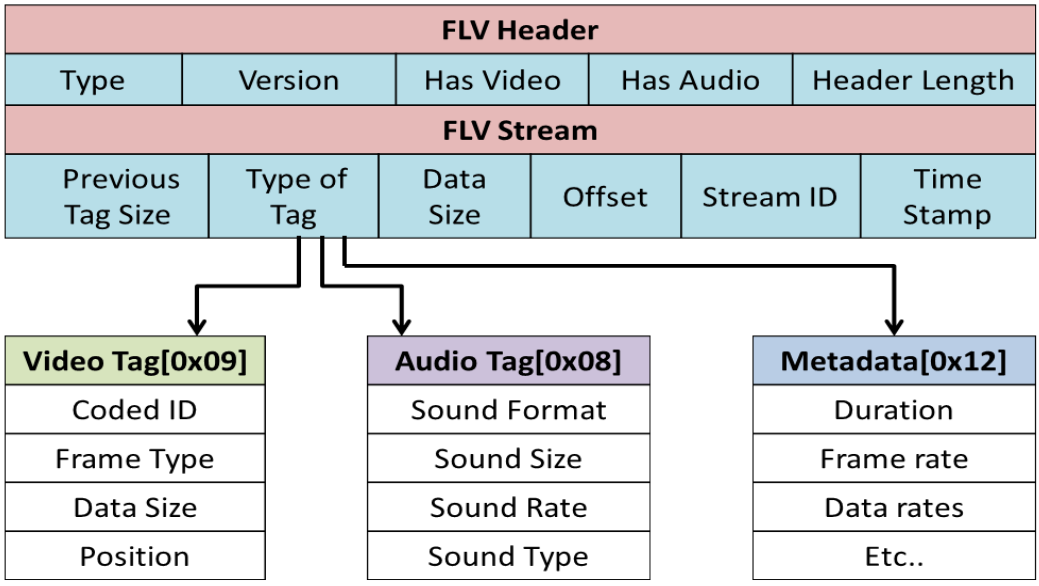

Figure 1: FLV file structure

4. Materials and Methods

The main focus of this paper is the FLV file steganography using the secret message packet segmentation. The proposed method looks to trace and analyze the input FLV file in order to find out the locations of video tags that will be employed in manipulating the secret message into a set of packets with an equal length as mentioned previously. Only one packet segmentation will be utilized in the hidden and the extraction process. The used method utilizes one master key to generate multi keys for video tag selection, message manipulation

and cryptograph. The system deals with secret messages as binary stream of data, so it can manipulate any type of data. Ulterior subsections show how the hidden and the extraction process will be carried out to reach the main goal of this work.

4.1 Hidden Process

The used technique to hide the secret data in the cover file is a substitution method. It replaces a byte from the cover file with another one from the secret message. The number of modified video tags is determined by number of selected secret message packets, also the number of replaced bytes in each selected video tag is determined by the length of selected secret message packet. The proposed method is very fast and simple, but the distortion in the cover file is noticeable when the number of replacement bytes exceeds a specific thresh hold.

The target video tags that will store the secret message packets are selected randomly from the different computed locations. Randomness process has two merits; firstly it increases the security of the data since the message packets are going to be scattered all over the FLV video tags in a random way. Secondly it makes the effect of changing relatively less apparent; due to the replacement packets are not adjacent.

Hidden process needs for several requirements to accomplish their work. Those requirements can be summarized by: selection of FLV file, video tags tracing, possible number and length of packets computation, secret message conversion and keys initialization. **Figure 2** shows the frame work of the proposed hidden process; those steps are summarized as ulterior and illustrated in next subsections.

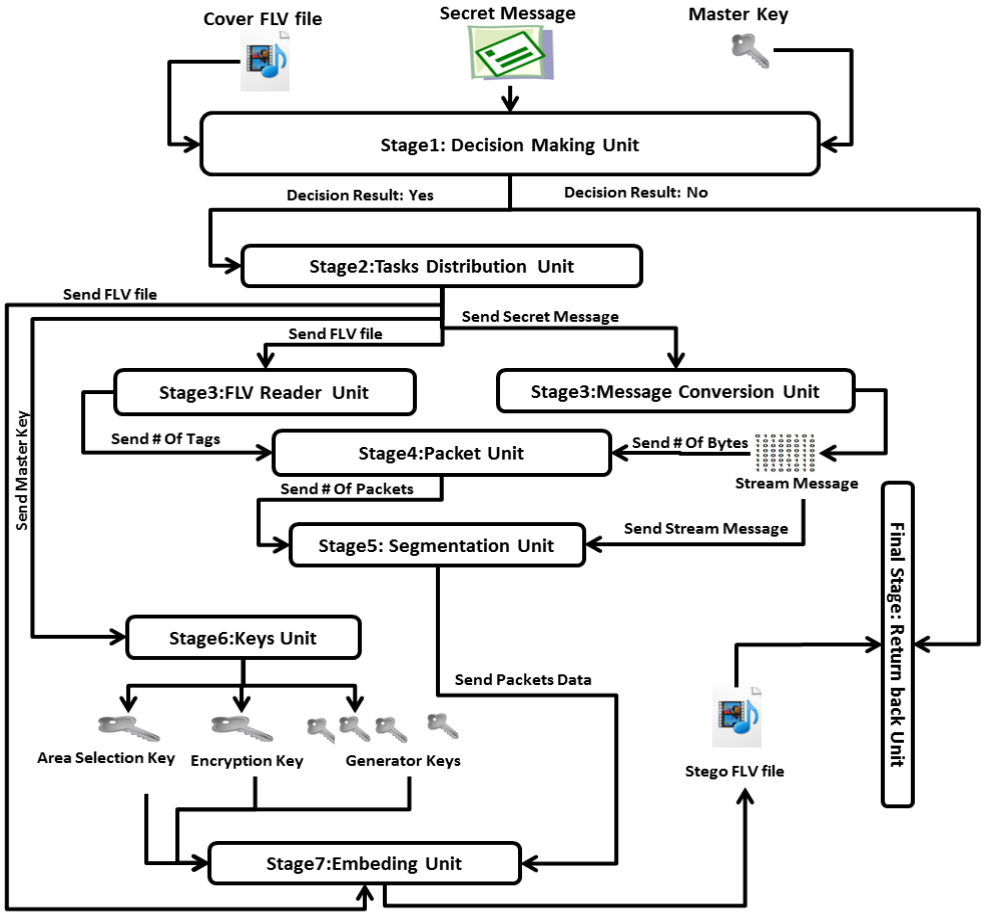

Figure 2: Hidden process framework

Hidden process steps

Step1: Input the cover FLV file, secret message, and master key.

Step2: Find out the location and number of video tags in the input FLV file by using the FLV reader unit

Step3: Convert the secret message to stream of bytes by using message conversion unit.

Step4: Compute the possible number and length of the secret message packets by using the Packet unit
Step5: Divide the secret message to packets by using the segmentation unit.
Step6: Generate the encryption, distribution and reordering keys by using key unit.
Step7: Embed the secret message packets inside the cover FLV file to get the stego FLV file

4.1.1 Video Tags Computation

First of all, the cover object that will hide the secret message must be a FLV file. The system checks the header of input file that consists of type, version, audio and video in order to determine their compatibility with the FLV reader unit. However, the result of compatibility checking determines the continuity for finding the locations of video tags or not. In the continuity state, if the FLV file has a video tags then the system will return a linked list that holds index of tag, start position, tag size and end position as shown in **Figure 3**.

4.1.2 Preparing the Secret Message

The secret message could be any type of files such as text, image, PDF, DOC…etc. The message manipulation subsystem converts message file to a stream of bytes, computes the size of input secret message, finds out their extension and displays the results to user in aim to keep them for the future employment in the extraction process.

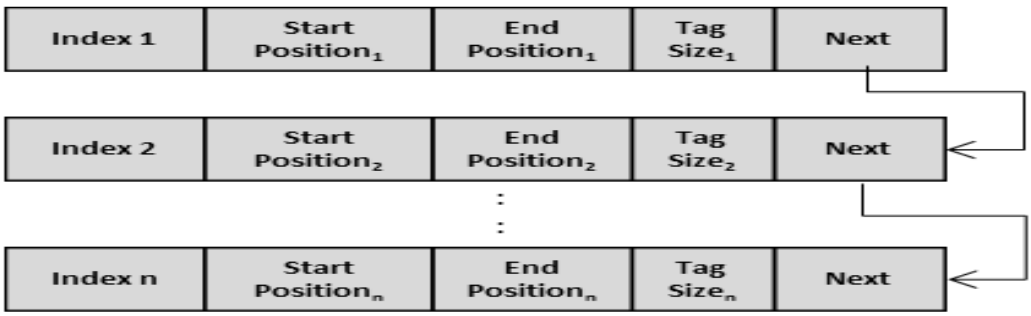

Figure 3: Return List of Video tags

4.1.3 Packets Number and Length Computation

After completing the video tags computation process, the system will take the returned list of video tags with the stream secret message and start the processing. Each possible computed packet is given an index through the process procedure. The determination of each packet is done according to the number of video tags, maximum size of packet and length of secret message that were sent from the caller. Number of Packets, length and index for each one are stored and returned back to user as a linked list as shown in **Figure 4**. However, the user must select one of them to use it through the embedding process.

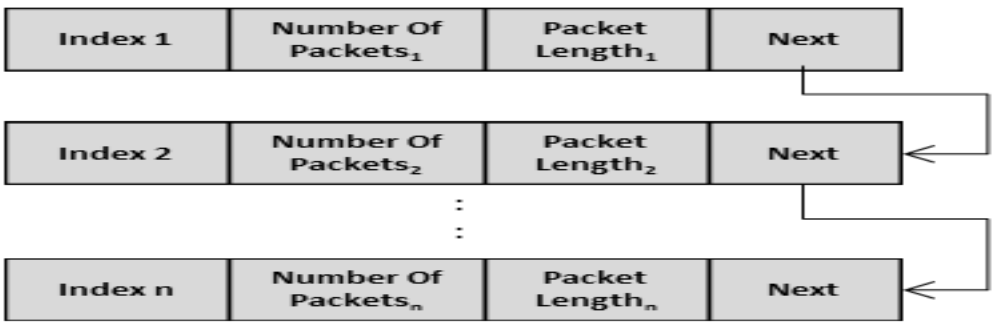

Figure 4: Returned List of Packets Number and Length

4.1.4 Dividing the Secret Message

Here, the user must select a length and number of packets form the returned packet list to be utilized in the embedding process. According to user selection, the system will divide the secret message into equal length packets. The bytes of each packet are reordered using the LCG random function with specific key. Reordering key for each packet is generated from the inserted master key. At the end of this process, the process will return a linked list that stores index and data for each packet as shown in **Figure 5**. The ulterior steps summarized how the dividing process will work:

Step1: Set Input =ReadFile(Secret Message File)
Step2: While Not EOF(Input) Do
*Ordering Key = Ordering Key * 3*
Data= Input.Read(1, Packet Length)
Data= Reordering(Data,Ordering Key)
End While

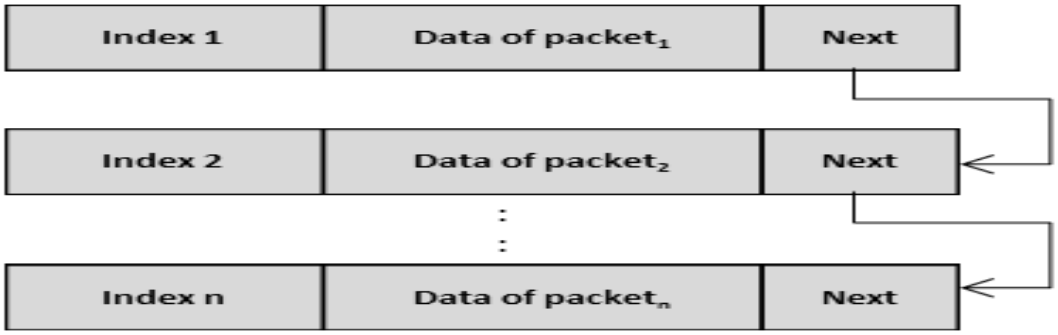

Figure 5: Returned List of packets Data

4.1.5 Keys Value Preparing

The inserted master key value is utilized in multiple operations. It is used for data encryption, packet bytes reordering and video tag selection as ulterior:

*Step1: Set EncryptionKey = Master Key * 23*
*Step2: Set Tag SelectionKey = Master Key * 31*
*Step3: Set Ordering Key=Master Key*41*

4.1.6 Reordering Process

The reordering process of packets is done by using the LCG method that generates a sequence of random numbers over the interval [0, M-1] without any redundancy until completing the cycle M. **Figure 6** shows the general formula of LCG with their parameters. More ever, to specify the values of multiplier **A** and non-common factor **C**, a set of preconditions must be satisfied. The existence of conditions allows the system to generate the sequence correctly without any redundancy (Byron, 1984). Those conditions are: **C** and **M** have no common factors other than the value **1**, (**A-1**) is multiple of every prime number that divides **M** and (**A-1**) is multiple of **4** if **M** Multiple of **4**. In this generator, the ordering key is utilized to initialize the $\mathbf{X_0}$ or seed value for each data packet. The cycle length in the generator is determined by the packet length.

$$X_{i+1} = (A X_i + C) \bmod M$$

- X_{i+1} refers to next random number
- X_i refers to current random number
- C refers to non-common factor
- A refers to multiplier
- M refers to cycle of generator

Figure 6: LCG Generator

Figure 7 shows how the bytes of a packet with length **4** will be reordered according to the LCG random numbers sequence. Each number in the sequence that starts from zero value represents a location in the manipulated packet.

4.1.7 Cryptography Process

Caesar cipher is one of the most common and simplex cryptography techniques (Stallings, 2005). It bases on the idea of substitution for the plain text letters. It must have a predefined list of letters that represents the first part of key, while the second part is the shifting value within the list. In this work the selection of substitution byte (shifting value) is done through the LCG random generator from a predefined list that is common between the sender and receiver.

4.1.8 Embedding Process

After completing the main requirements for embedding process (video tags determination, secret message selection, possible number and length of packets, Key setting and secret message segmentation) the system will start the data manipulation. First of all, it selects the first packet of secret message randomly, encrypts it by using the Caesar algorithm then the result will be replaced at the end of the random selected video tag. For each byte in the selected packet, the system will replace a byte from the video tag. The process stills working until embedding all packets inside the FLV file. After that, a new FLV file that hides the secret message will be set as summarized in the next steps.

```
Step1: Set FileInfo F1 = new FileInfo(CoverFile);
Step2: SetFileInfo F2 = new FileInfo(SecretFile);
Step3: IF ( NotF1.Exists OrNot F2.Exists)
           Return Error Message;
Step4: CopyTo(F1.FullName, StegoFileName );
Step5: IF Packet List Is Empty Then
           Return Error Message;
Step6: Tags Counter = 1;
Step7: While (TagsCounter<=NumberOfPackets) Do
            Set  Temp= Select_Random_Tag (Tag List, Selection key)
            SetP = Temp.EndPosition - LengthOfPacket;
            Set StegoFileName.Position = P;
            Set Packet= Select_Random_Packet (packet List)
           For I =1ToLengthOfPacket Do
              Set byte x= Packet.ReadByte(I);
              Set x=Encrypt(x,key)
              StegoFileName.WriteByte(x);
             Set StegoFileName.Position = P +1 ;
          Next For
         TagsCounter=TagsCounter+1;
      End While
Step8: Return StegoFileName
```

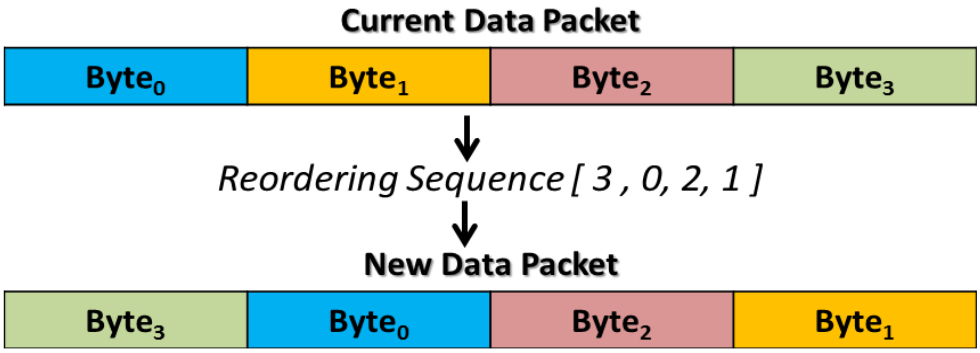

Figure 7: Packets Reordering

4.2 Extraction Process

In order to extract the secret message from the Stego-FLV file, the extractor should be aware of four important things; first you should know the length of secret message, this means you should know the number of bytes that were hidden. Secondly, you should know the master key that was used to hide the data. Thirdly, you should know the packet size that was utilized in hiding the data. And finally, you should know the extension of secret message (type of message). Once you have these information you can extract the message from the stego-image using the ulterior steps as shown in **Figure 8**.

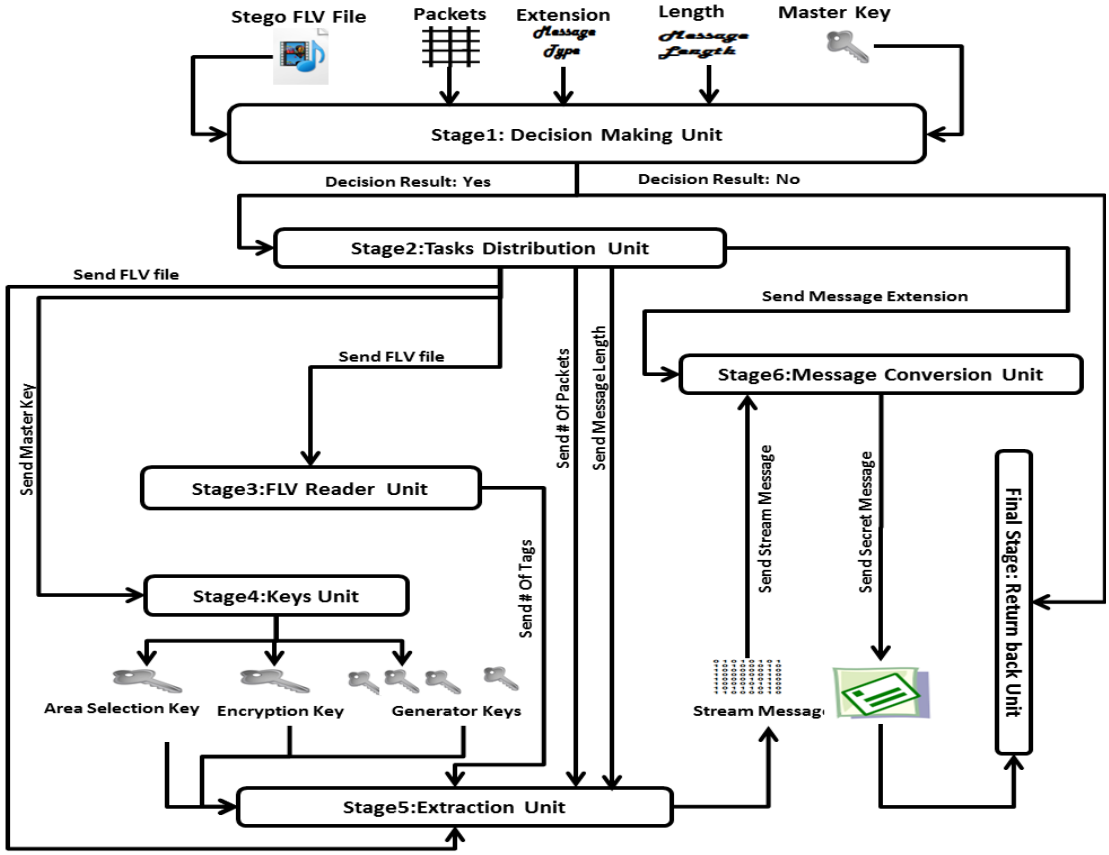

Figure 8: Extraction Process Framework

Step1: Set FileInfo F1 = new FileInfo(StegoFile);
Step2: IF (NotF1.Exists)
Return Error Message;
Step3: Get the master key, packet size, message length and message extension
Step4: Initialize the keys of decryption, selection and reordering
Step5: Tag List=BuildVideoTags(StegoFile)
Step6: IF Tags List Is Empty Then
Return Error Message;
Step7: Tags Counter = 1;
Step8: Set NumberOfPackets=Message Length/Packet Size
Step9: While (Tags Counter<=NumberOfPackets) Do
Set Temp= Select_Random_Tag (Tag List, Selection Key)
Set P = Temp.EndPosition–packet size;
Set StegoFileName.Position = P;
For I =1ToPacket Size Do
Set byte x= StegoFileName.ReadByte(P);
Set x= Decrypt(x, Decryption key)
Set packet[i]=x
Set P = P +1 ;

Next For
Reordering_Packet(Packet, Ordering key);
SecretFileName.WriteByte(Packet);
Tags Counter=Tags Counter+1;
End While
Step10: Return SecretFileName

The extraction procedure carries out the steps of building video tags, selecting the random tag and reordering packets as in the hidden procedure, so no need to re-explain them. More ever, the decryption process uses the Caesar algorithm that was used to encrypt the secret message bytes.

5. Results and Analysis

The proposed system was tested using a lot of FLV files and secret messages with different size. **Table1** displays the data set that was utilized in the evaluation process. Max size of secret message is determined through the number of cover file tags and maximum allowed packet size. In this work the maximum allowed packet was **30** bytes. The threshold value of **30** was selected to hide the visual appearance flicker that may result from the huge size of packets.

Table 1: Evaluation Data Set

Secret Message	Size
Secret 1	48 Bytes
Secret 2	598 Bytes
Secret 3	798 Bytes

Cover FLV	Size	Number Of Video Tags	Max Size of Secret Message
Cover1	669036 Bytes	424	12720 Bytes
Cover2	129444 Bytes	170	5100 Bytes

Secret message is divided into packets with the same length of bytes. The division process bases on the size of maximum allowed packet and FLV cover file.**Table2** shows the number of packets and length for each one when the secret message with 48 bytes taken with the cover file of size 669036 bytes.

The main issues that were taken in the consideration for evaluating the system are: visual appearance of stego-FLV file compared to the cover one, modification rate, capacity of cover FLV file, robustness against modification, detecting ability and security.

Visual appearance is evaluated through the human eyes or the magnifying classes. The stego-FLV file is compared with cover FLV one in order to detect a noise or irregularity in the final form of the file. To do that, secret1, secret2 and secret3 were embedded in the cover1 and cover2 using different packets size for each one. The result FLV file in the different cases gave a result similar to the cover one. The Random distribution of packets within the random selected video tags and the huge size of video tag compared to the size of packets are the main reasons for hiding the noise and flicker in the result stego-FLV file.

Table 2: Packets and Length Computation

Number Of packets	Packet Length
48	1 Byte
24	2 Bytes
16	3 Bytes
12	4 Bytes
8	6Bytes
4	12 Bytes
3	16 Bytes
2	24 Bytes

The best number of packets and length can be computed by utilizing the modification rate (MR). It is computed by fining out the number of modified bytes divided by the total number of bytes from the cover FLV file. The MR value bases on the length of user selected secret message and length of the cover FLV file as shown in

Figure 9. MR varies between one packet and another for the same secret message as shown in **Figure 10**, so the sender should select the minimum MR packet which helps in hiding the effect of visual appearance in the stego FLV file. Variation in MR results from random and different distribution of bytes within the video tags. Huge number of packets may give a low MR due to regular and equal distribution of bytes within the huge number of video tags.

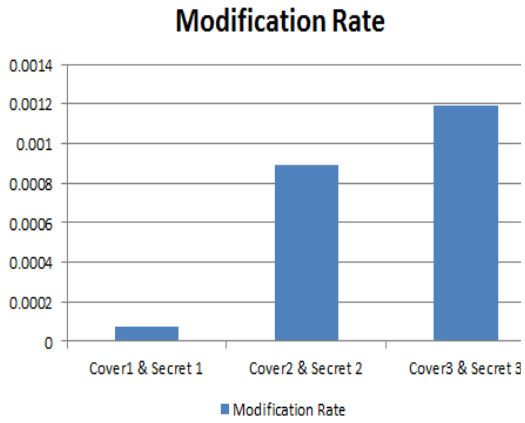

Figure 9: MR for FLV covers and secret

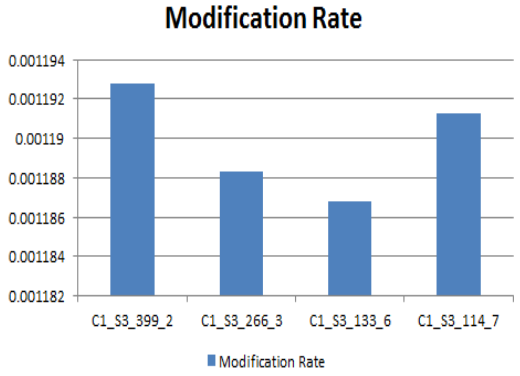

Figure 10: MR for cover1 with secret message3

The capacity of cover FLV file is determined by the number of video tags and the maximum allowed packet size. Each secret message byte requires one byte from cover file, so the maximum capacity is reached when the size of secret message equals the maximum packet size multiplied by the number of video tags. To evaluate the proposed method at the maximum capacity state, a secret message of size 12220 was embedded in cover1 of size 669036 bytes; the result stego-FLV file was very similar in visual appearance to the cover one with a modification rate only 0.018.

Information is considered a robust when it is embedded inside a cover file and encountered any modifications. However, the proposed method is considered a robust one because the secret data are embedded as a part of encoding information for FLV file, so the secret data corrupts only when a damaging occurs to the host FLV file itself.

Undetectability as a feature was applied in the proposed method because the secret data bytes were distributed randomly within a random video tags, according to this the hidden bytes under the host FLV file will not be doubtable or suspicious for attackers.

The proposed method is considered a secure one because no attacks view the hidden message unless they have a full knowledge about the keys. The extraction process requires multi keys for extracting the secret message. Those keys are: master key, length of message, used number of packets and type of secret message. The Master key does not construct only the decrypting key but also the key for selecting the host video tags and the key for reordering the secret message bytes. According to that, the extraction process is very hard, complex and secures one.

6. Conclusion

This paper presented a secure video FLV file stenographic method for information security. It based on the idea of secret message packet segmentation and reordering of packet bytes at the end of a random video tag. The main goal of this paper was to construct a solution that is robust against the attack, effective in generating the stego FLV file and very hard for visual appearance to predict and detect the existence of secret message. Experimental results found out that, the proposed method satisfied most of the security requirements (visual appearance, security, undetectability) and performed adaptability of FLV file as a host to hide the secret messages. The future work of this paper proceeds to extract the secret message from stego FLV file without a pre-knowledge about the master key or secret message. More ever, it looks to check the adaptability of other FLV file fields or parts for hiding the secret message.

REFERENCES

[1] Abdelwahab A. A. and Hassaan L. A.,2008, A DISCRETE WAVELET TRANSFORM BASED TECHNIQUE FOR IMAGE DATA HIDING, Radio Science Conference, Egypt, March 2008, PP: 1-9.
[2] ADOBE FLASH VIDEO FILE FORMAT SPECIFICATION, VERSION 10.1, 2010, Published August 2010, (http://download.macromedia.com/f4v/video_file_format_spec_v10_1.pdf).
[3] Alwan, W, 2013, DYNAMIC LEAST SIGNIFICANT BIT TECHNIQUE FOR VIDEO STEGANOGRAPHY, Journal of Kerbala University, Vol. 11 No.4 Scientific, 2013,PP: 7-16.
[4] Atiea M.A. , Yousef B. M, and Hedar A ,2012,HIDING DATA IN FLV VIDEO FILE, Proceedings of the Second International Conference on Computer Science, Engineering & Applications (ICCSEA 2012), May 25-27, 2012, New Delhi, India. Volume 2, Publisher Springer Berlin Heidelberg, PP: 919-925, Online ISBN978-3-642-30111-7DOI10.1007/978-3-642-30111-7_89,
[5] Bawaneh, M.J. (2014) A NOVEL APPROACH FOR IMAGE STEGANOGRAPHY USING LCG. International Journal of Computer Applications, 102, PP:34-38.
[6] Byron, M.J.T., 1984, ELEMENTS OF SIMULATION. Chapman and Hall, USA, 57-64.
[7] Cruz J. P. , Libatique N. J. and Tangonan,G. ,2012, STEGANOGRAPHY AND DATA HIDING IN FLASH VIDEO (FLV) Published in:TENCON 2012 - 2012 IEEE Region 10 Conference,19-22 Nov. 2012,PP: 1 – 6, ISSN :2159-3442, DOI:10.1109/TENCON.2012.6412279, Publisher: IEEE
[8] Dan, D, A and Ferdinandus F. X. ,2015, OPTIMALISASI STEGANOGRAFI PADA FILE FLV MEMANFAATKAN METODE INJECTED AT END OF ALL VIDEO TAG DENGAN PENAMBAHAN KOMPRESI, Seminar asidalam De- IDeaTech 2015,PP:131-136 ISSN: 2089-1121
[9] Dasgupta K , Mandal J.K. and Dutta P ,2012, HASH BASED LEAST SIGNIFICANT BIT TECHNIQUE FOR VIDEO STEGANOGRAPHY(HLSB) ,International Journal of Security, Privacy and Trust Management (IJSPTM), Vol. 1, No 2, April 2012,PP:1-11, DOI : 10.5121/ijsptm.2012.2201
[10] Kaur , M, and Kaur, A ,2014, IMPROVED SECURITY MECHANISM OF TEXT IN VIDEO USING STEGANOGRAPHIC TECHNIQUE Volume 2, Issue 10, October 2014 International Journal of Advance Research in Computer Science and Management Studies Research Article IJARCSMS www.ijarcsms.com, PP: 44-51 , ISSN: 2321-7782
[11] Shinde , P, and Bano, T ,2015, A NOVEL VIDEO STEGANOGRAPHY TECHNIQUE , International Journal of Advanced Research in Computer Science and Software Engineering, Volume 5, Issue 12, December 2015, PP: 676-684, www.ijarcsse.com , ISSN: 2277 128X
[12] Sneha, B. and Gunjan, B, 2014, DATA ENCRYPTION BY IMAGE STEGANOGRAPHY. International Journal of Information and Computation Technology, volume 4, PP: 453-458. http://www.irphouse.com/ijict.htm
[13] Stallings, W., 2005, CRYPTOGRAPHY AND NETWORK SECURITY PRINCIPLES AND PRACTICES. 4th Edition, Prentice Hall, Upper Saddle River, PP: 36-38.

A Brief Author Biography

Atef Obeidat received the B.S. Degree in Computer science from Yarmuk University in 1991 and M.S. degrees in Computer science from Jordanian University in 2001. But the PhD degree in communication and network systems, he received from Novosibirsk State Technical University in 2009. Areas of his interest are p2p, operating systems, algorithms and security

Mohammed Bawaneh received the B.S. Degree in Computer science from Yarmuk University in 2001 and M.S. degrees in Computer science and applications from Yarmuk University in 2004. But the PhD degree in computer information systems, he received from Arab Academy of banking & financial Science Faculty of Systems & Information Technology, Department of information system, 2010. Areas of his interest are computer graphics, data structure, programming languages, algorithms and security.