An Adaptive FLV Steganography Approach Using Simulated Annealing

Mohammed J. Bawaneh[1], Atef Ahmed Obeidat[1], Majd M.Al-kofahi[1]

[1]Department of Information Technology, Al-Huson University College, Al-Balqa Applied University, Salt, Jordan

Abstract**:** Steganography is not only the art of hiding secret messages in cover media but also a process of communication and secure data transfer. Secret messages can be sent over the Internet with security by using several steganography techniques, but all of them present challenges in steganalysis. This study proposes a new secure technique called flash video (FLV) file steganography that keeps the frame video quality and is difficult to detect. The technique can hide any type of secret message inside a given FLV file. The secret message is divided into packets of the same length, reordered packet, and encrypted bytes before being hidden at the end of a selected video tag. A simulated annealing (SA) approach to select tags for steganography is presented to reduce or avoid the challenge of steganalysis. The proposed method uses SA as supporting framework to deal with the FLV file as a host for different types of secret messages. The system determines the minimum path within the host FLV file by using SA and hides the message bits inside each location in the minimum computed path. Analysis of the host FLV file cannot be performed without proper knowledge on the transformation process. Thus, the existence of the secret message is difficult to detect by steganalysis. Knowledge is represented by the key of finding the minimum path in the host FLV file, key of secret message length, key of additional bytes, key of message packets reordering and key of message extension. Experimental results show that the proposed technique satisfies the main requirements of steganography with regard to visual appearance, capacity, undetectability, and robustness against extraction**.**

Keywords**:** Steganography, FLV, LCG, Video tags, Simulated Annealing, Optimization**.**

1. Introduction

With the development and popularization of the Internet, the need for the exchange of secret data among users, particularly in commercial operations, has increased. This exchange requires that the data be kept confidential during the transfer process. Many methods, such as steganography, cryptography, and watermarking, have been developed to achieve this goal. All of these methods have the same goal of transferring data safely but in different ways and techniques.

Steganography is an old science; the term was derived from the Greek word "steganos," which means covered or secret, and "graphy," which means writing or drawing[1]. In simple words, steganography refers to the process of hiding writing inside a communication channel. The hidden message may consist of invisible ink on a paper or copyright information on digital media.

Different types of cover files, such as text, image, audio, or video steganography, or manipulation procedures in the embedding process, such as injection, substitution, distortion, or generation steganography, can be employed to classify steganography techniques[2].

This study proposed a robust steganography system that merges the simulated annealing (SA) algorithm with FLV file structure. SA is a common random search method that can be applied to several types of linear and nonlinear problems [3]. SA bases on the idea of metal heating and cooling. Thus, the algorithm starts with a high temperature (worst case) until the minimum temperature (best case) is reached. Furthermore, SA utilizes a random searching process for maximization or minimization of the solution. Thus, SA does not accept changes that will modify the solution to become poor. The merit of SA is its capability to manipulate nonlinear models, noisy data, and different constraints, whereas the demerit of SA is the large number of options in the random manipulation process that needs to be processed.

Here, the cover object, where the message is to be hidden, is a flash video (FLV) file. FLV is a good hosting cover in the steganography process because of its simple structure, capability to maintain picture and sound quality, and popularity on Internet websites.

The method consists of encoding and extracting secret byte stream algorithms, which will be discussed in the subsequent sections. In contrast to recent video steganography methods, the proposed method characterizes by the following features. First, the method selects substation locations through SA. Thus, the proposed technique is novel. Second, the method reaches the maximum capacity for hiding a secret message with a low modification rate. Although the capacity of the video used to hide data is high, the proposed method exhibits satisfactory robustness against modification, and the visual appearance of the stego FLV file is identical to that of the cover FLV file without attracting attention.

Today, some of the digital media hosts give an opportunity to change the inner content without leaving any evidence in outer information about modification or injection of data. FLV file is one of those hosts that grant a chance for transferring data securely. The main motivation of proposed approach is the prominence, simplicity of structure and capability of FLV file to carry a huge amount of data. It makes the eyes turn to utilize the inner structure as transporters for secure data. Furthermore, the robustness of SA parameters or keys against extraction played a role in motivation area of system. The simplicity of FLV file and robustness of SA provide us with a well and excellent framework that has a low modification rate.

The main challenges that encounter the proposed system are summarized by preventing sniffing, message extraction and errors or modifications in data transferring channel. To solve the dilemma of detection, a number of keys were used in extraction and hiding process make the mission close to impossibility. The various numbers of keys inspire a problem for authenticated user about remembering and insertion. One solution for such problem was to use a master key that will be utilized to find out the other keys of SA procedure and

message segmentation. The problem of channel errors or modifications is out of scope in this work because it is a network challenge and requires additional techniques for manipulating it.

The rest of the paper is organized as follows. Section 2 presents related work. Section 3 describes the structure of FLV file. Section 4 describes the materials and methods. The experimental result is shown in Section 5. Section 6 concludes this paper and presents the directions for future work. Finally, Acknowledgment is given in section 7.

2. Literature review

The ancient Greeks utilized steganography to pass secret messages from one area to another. They employed several cover hosts, such as the heads of slaves or the wooden part of wax, to write secret messages. Messages were inscribed on the heads or the wood and sent to the receiver area [1]. Today, the opportunity for generating new steganography techniques that host secret messages has unfolded because of the availability of different types of files, such as images, audio, videos, and documents. This work focuses on FLV file steganography. Thus, the investigation presents several published studies in this field.

Atiea et al. [4] proposed a novel flash video (FLV) file information-embedding scheme. They used a weak point in the header information of the host FLV file to evaluate the robustness of compression process. The secret message, as they mentioned, could be reconstructed without knowing the original host FLV file. The method was robust against lossless and lossy compression.

Cruz et al. [5] presented a study about the design, implementation, and automatic tools for analyzing FLV files. They proposed several methods for hiding data inside the different parts of FLV files and discussed the merits and demerits of each method. The methods were tested using auditory–visual test video tag histogram and red–green–blue (RGB) average analysis.

Bawaneh [6] proposed a random least significant bit (LSB) to embed secret messages inside RGB color images. The LSB used a linear congruent generator to determine 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 was based on the modification rate (MR) for each channel. The minimum modified rate was employed to embed secret messages. The random LSB was better than the sequential LSB in terms of visual appearance and satisfied sufficient security to secret messages.

Arraziqi and Ferdinandus [7] proposed a new technique to add compressed data using Huffman coding at the end of video tags. The data were compressed and distributed evenly within all video tags. Some data, however, could not be compressed. The success level of the compression method reached 80%, and the method could compress up to 57% of data.

Dasgupta et al. [8] proposed a video steganography technique based on a hashing function for LSB. Data were embedded in cover frames using LSB. Eight bits of each byte from a secret message were divided into 3,3,2 and embedded into the RGB pixel values of the cover frames. The hashing function was used to select the insertion position in LSB bits. The data were analyzed in terms of noise ratio, mean squared error, and image fidelity. They found an encouraging result in terms of capacity in test video files.

Alwan [9] proposed a modified or dynamic-based LSB technique to hide movies in movies. 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 and cover movies were compared in terms of noise ratio and mean squared error.

Kaur et al. [10] proposed a technique for video steganography that was called hash–LSB combined with the Rivest–Shamir–Adleman algorithm. The proposed method generated a mask pattern for data bits by using a hashing function. Secret message bytes were encrypted before hiding them in cover video frames. The method was secure against intruders due to different security levels.

Shinde et al. [11] proposed a novel approach to video steganography that dealt with multiple types of cover format (MOV, MTS, FLV, and MPEG). Secret messages might be any type of data, such as text, audio, video, and image. Encryption, compression, and embedding techniques were combined in a protection technique. Existing video steganography techniques work only on Audio Video Interleaved files and secret messages of text or image type.

Jókay [12] proposed a technique that uses the internal structure of the MPEG-4 standard GOP encoder to hide a secret message. Although the method does not assume decoding of the video stream included in the MP4 file, its suitability depends on the video encoder used and on the type of video scenes. Varying numbers of adjacent P and B frames in the individual GOPs (variable number of video frames in the MP4 chunk structures) are used to decode the hidden data. However, the proposed method increases the size of the cover file and considers only the file with dynamic GOP length.

In [13], multiple bits of pixels were used to embed secret information. Thus, multiple-bit steganalysis could not be easily implemented on the stego video file for the extraction of secret information. This study included text and video data hidden in a single video file component (audio and video).

In [14], a method for hiding and extracting secret data using high-resolution MP4 videos was proposed. Discrete cosine transform (DCT; 8 × 8 block) was performed on any channel (e.g., R channel) of the frames, and the secret information bits were embedded in selected high-order coefficients. Each frame was processed by 8 × 8 inverse DCT block processing and combined to obtain an MP4 video with the hidden message.

In [15], an algorithm for data hiding that utilizes the Round-LSB technique to hide secret data in a cover video was proposed. In the algorithm, secret data are hidden in selected frames of the cover video, which are known as key frames, to improve system security. Key frames are selected using two methods. First, statistical features, such as kurtosis, skewness, standard deviation, and mean, are utilized for key frame extraction. Second, a fixed threshold is used. Hiding secret data in the key frames achieved through these methods

enhances the security of the algorithm by creating increased confusion for hackers.

In [16], secret data were compressed, encrypted, and embedded into cover frames in such a manner that 8 bits of the secret data were divided into 2, 2, 2, and 2 and embedded into the red, green, and blue pixel values of the cover frames; the remaining 2 bits were inserted into the subsequent pixel of the cover frame.

In [17], 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.

In [18], a video was used as a cover medium to hide secret data by using the random encoding/decoding process. However, the message size was difficult to estimate, and the message was detectable by using varying sizes of windows and localized histogram analysis.

In [19], an approach is proposed by combining both video, audio and text signals into a single architecture and securing it before the process of transmission. The approach is worked by embedding the color components of each pixel of the video signal in a quaternion number. The fourth component of the quaternion number is taken with either an audio sample or a textual data. The array of quaternion numbers corresponding to a video frame is converted to the frequency domain, using quaternion Fourier transform, and then multiplied by the quaternion Fourier transform of a digital image. The advantages of the approach, the first is the selected digital image which is used as a complicated secret. The yielded signal is transmitted and when received, both of video, audio and text signals are extracted using simple quaternion mathematics applied to the received signal and a copy of the digital image. The second is increased its complexity by applying one of the well-known cryptographic techniques to the samples of the transmitted signal.

In [20], a novel method is introduced for encrypting/decrypting audio signal using a selected digital image as a complicated key and cover for audio signal. Each sample of the audio signal is combined with the values of the three color components of a pixel fetched from the cover image yielding a quaternion number. The absolute value of this quaternion number is then transmitted and when received, the original value of the audio sample can be extracted using simple quaternion mathematics.

In addition to increase the level of the complexity, the approach can be applied one of the well-known cryptographic techniques (symmetric or asymmetric).

In [21], a novel approach of data security using video steganography, Huffman code compression, and asymmetric cryptography was proposed. In the proposed system, messages are encrypted with the Rivest–Shamir–Adleman (RSA) algorithm, and encrypted messages are compressed using the Huffman code algorithm. The compressed encrypted messages are hidden using the LSB algorithm. This research brings to light the concept of effectively combining steganography, compression, and asymmetric cryptographic algorithm. The preference for RSA over any other cryptographic algorithm is due to its capability to provide improved security for large file sizes, thereby reducing computational complexity.

3. FLV File Structure

FLV files can be divided into two basic parts: FLV header and stream. FLV header is a record with a length of nine bytes that stores the FLV file type, version, flow information (has audio and video), and header length[22]. Each FLV file consists of a stream of different tags. A tag holds information about the length of a previous tag, the type of the 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 a previous tag. Each tag in an FLV file also consists of two parts: header and data. Tag header determines the type, length, and other information of a tag. Data area can be divided into three types according to tag type: audio data, video data, and metadata. Audio data contain information about the used audio in a tag, such as format, sampling rate, length, and audio. Video tag contains data to determine the frame type of videos, the starting position of tags, tag size, and coded ID. Metadata tag stores general data about FLV files, such as the information about storage, duration, audio data rate, creator or owner, width, player, and creation date. Fig. 1 shows the structure and components of FLV files.

4. Materials and Methods

FLV files are good hosts for hiding secret messages into them. However, an assistant algorithm for distribution or encoding should be utilized to improve the robustness and performance of hiding and extraction processes against sniffing or steganalysis. The main goal in this work is to build an FLV steganography system with simulated annealing (SA) support. The proposed method constructs the minimum path of byte positions at the end of video tags inside cover FLV files by using the SA algorithm. The selected path stores the bytes of secret messages by substituting a byte from the messages with a byte inside FLVs according to corresponding retrieval positions. Several steps are conducted to achieve the objective of this work. The main stages of our system can be summarized through embedding and extraction processes. Each stage has inputs, processing, and outputs; thus, it can be considered a state machine, as shown in Table 1. The next subsections show how embedding and extraction processes utilize multiple supporting algorithms to fulfil the system target.

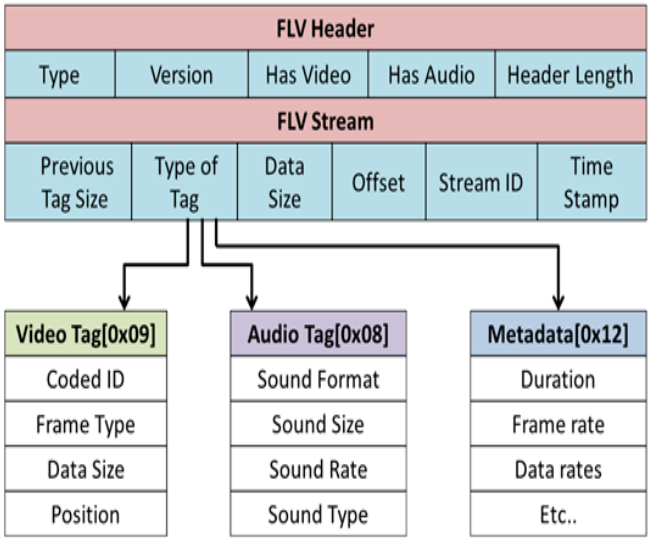

Figure1. Shows the structure and components of FLV files

Table 1. Input, Processing, and Outputs of Embedding and Extraction processes

Stage	Embedding process	Extraction process
Inputs	• FLV file as a host for message (Cover FLV) • Secret message to be embedded • Key for SA to select minimum path • Key for additional bytes	• FLV stego file (Stego-FLV) • Length and extension of secret message • SA Key for path constructing algorithm • Length of used bytes in constructing path
Processing	• Find out the positions of video tags • Find out the length of secret message • Build a list of bytes from FLV file • Build the minimum path from FLV list by using SA • Split the secret message bytes • Embed the message bits inside the FLV file vector	• Build the path from Stego-FLV using path constructing algorithm • Create the secret file • Retrieve bytes from the FLV file • Combine the bytes with each other's to build the secret message file
Outputs	• Stego FLV file • SA key to be used in extraction process	• Secret message file

4.1 Hidden Process

The task of hidden process is needed for numerous phases to accomplish their goals. Fig. 2 shows the framework of the hidden process and how different tasks are distributed among stages. The process starts by fetching FLV files and secret messages to determine the number of FLV video tags and check the size of messages with the number of available tags. After the checking process, a list of bytes that can be used to store message bytes is constructed. The SA procedure determines the suitable vector for embedding the message bytes. Each byte from a message will replace a byte from an FLV file. The complete details of the embedding process will be explained in the next subsections.

4.1.1 Preparing Secret Messages

The system deals with any type of secret messages because it manipulates data as binary one. The secret messages can be text, image, PDF, audio, or video. Secret messages contain important information, such as length and extension, which should be kept by the system user to utilize it in the extraction process.

4.1.2 Video Tag Determination

The cover file employed to hide a secret message must be an FLV file. The system determines the compatibility of a fetching file by checking the header of an input file. FLV file header stores information about the type, version, audio, and video of a file. After proving the file compatibility, the system sends it to the FLV header unit to identify the locations of video tags. The FLV unit returns a result that consists of tag index, starting position, end position, and size of video tag in a form of linked list to the system, as shown in Fig.3.

Tag Index	Starting Position	End Position	Tag Size	Next Pointer

Figure3: Return Linked List of Video tags

4.1.3 Hosting List Construction

In this stage, the system computes the required bytes for embedding the secret message by dividing the message length (ML) over the number of video tags and taking the ceiling value for result, as shown in Equation (1).

$$\text{Required Bytes} = \text{Ceiling}\,(ML/\text{Number of Video Tags}) \quad (1)$$

The system allows users to increase the number of bytes to be employed in selecting the best vector through the SA procedure; thus, the fetching number of bytes from each video tag is determined by adding required bytes to additional ones. The system then determines the number of video tags to construct the hosting list and return it back to the next solving unit, as shown in the procedure in Fig. 4. The structure of the hosting list is shown in Fig. 5.

4.1.4 Minimum Path Construction

After the subprocess of hosting list construction is completed, the execution is translated to the SA procedure. A set of parameters is required to complete the task. The procedure needs two parameter types: essential and supplementary ones. Essential parameters refer to the basic ones that are required for the SA algorithm, such as maximum starting temperature (Max_Temperature), minimum terminating temperature (Absolute_Temperature), and cooling rate (CoolingRate). Supplementary ones are represented by FLV file location, seed key for selecting locations from an FLV list, and length (L) of secret message. After the required values are inserted correctly, the SA procedure initializes linear congruential generator (LCG) seed to utilize it in building the first vector and constructing the next vectors that will be used in computation and

comparison stages. The SA procedure passes through several steps to compute the minimum hidden path (best path) and extraction keys (EKeys), as shown in Fig. 6. First, SA conducts parameter initialization by setting LCG seed, distance, and delta distance. Second, SA builds the current vector, sets it as the best vector, and computes the distance for it from secret messages. Third, several iterations are conducted to compute the best path by comparing the current one with the next computed one according to a fitness function. Finally, SA constructs and returns the final solution with future EKey to the embedding unit. EKey keeps the last values of the LCG seed that is the best solution vector, and it must be retuned back to the user interface to be utilized in the decoding or extraction process. Each stage comprises numerous abstraction levels that will be explained in the following subsections.

Index	*Position*	*Data in Byte*

Figure 5. Hosting List Structure

4.1.4.1 Initial Vector Construction

In this stage, the solver takes the hosting list, LCG seed, hosting file, and ML to build the initial vector. SA path is represented as a vector of random positions, such that all of them are taken from the hosting list. Each location consists of an index, position, and data of selected position, as shown in Fig. 5. Thus, the constructed vector is considered a subset from the hosting list. The LCG method is employed for generating random values to guarantee no redundancy in the selected position. LCG has a set of preconditions that must be satisfied to carry out M random numbers (RDs) in the range of [0, M-1] without any repetition, as will be shown later on. The construction procedure must keep and return the last value for LCG seed to avoid the use of the same seed in another time, which may corrupt or decelerate the embedding process. Each location in the selected path is utilized to hide one byte from a secret message; hence, the path length must be equal to the secret message length. The LCG function builds a random vector that consists of ML RDs in the range of [0, ML-1]. The values inside the random vector represent the index of the selected location from the hosting list. After constructing the current vector, the procedure sets it as the best one and returns it with seed to the SA procedure. The main steps of initial vector construction from the hosting list using LCG are shown in Fig. 7.

4.1.4.2 Fitness Function

The distance between the constructed vector and secret message is computed by comparing each byte with the facing byte in the vector, as shown in Fig. 8. The total distance for each path is utilized in the fitness function used to compare between current and subsequently generated vectors. The minimum total distance vector is considered the best one every time. The new distance is subtracted from the old one (deltaDistance) to compare between next and current vectors. Thus, if the result if less than zero, then the solver must replace the current vector with the next vector and SAKey by OldSeed. Otherwise, the solver generates RD, such that RD ϵ [0,1]. The solver then compares the value of $e^{-deltaDistance\ /\ Max\_Temperature}$ with the RD value. If the former is larger than the latter, then the solver must update the current vector and key, as shown in Fig. 9.

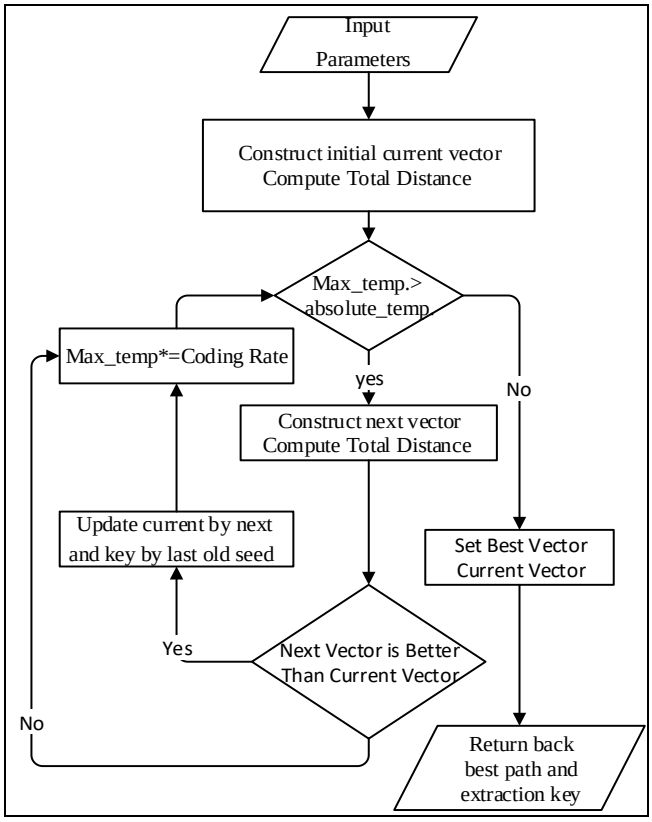

Figure 6. SA Process Framework

Procedure 2. Initial Vector Construction (IVC)

```
Step1: CurrentVector = new VectorList
Step2: For i = 0 To ML-1
         Seed =LCG(Hosting_List.Size, Seed)
         CurrentVector.AddNode(Hosting_List[Seed])
       Next For
Step3: Return CurrentVector, Seed
```

Figure 7. Initial Vector Construction

4.1.4.3 Best Vector Construction

After the input and vector initialization, the SA procedure sets the initial vector as the best one and keeps the constructed seed as retrieved EKey. The solver then constructs new vectors and compares them with the current one to determine the best one. The selection process is accomplished through a loop that starts from Max_Temperature and terminates at Absolute_Temperature. Several steps are conducted inside the loop. Such steps begin by creating the next vector and computing the distance for each location with other neighborhood pixels. Before creating the new vector, our system must keep the old seed inside OldSeed to use it as the new EKey if the next vector is better than the current one. The next vector is constructed in a way similar to the initial vector construction, as shown in Fig. 10.

```
Procedure DC
Step1: Set Distance =0
Step2: For i = 0 To ML-1
        If  Message[i].DataByte <> Vector[i].DataByte Then
            Distance= Distance +1
        End If
       Next For
Step3: Return Distance
```

Figure 8. Distance Computation

$$Current\ Vector = \begin{cases} Next\ Vector & deltaDistanc < 0 \\ Next\ Vector & e^{-Delta\ Distance/max\_Temperature} > RD \\ Current\ Vector & Otherwise \end{cases}$$

Figure 9. Current Vector Replacement

4.1.4.4 Final Solution Construction

The final solution requires a special structure to store the key, total distance, and best vector. The solver accordingly builds a composite structure called StegoList that consists of SAKey, total distance, and a type pointer vector list. After storing data, the SA procedure returns back the result to the hidden procedure to complete their tasks according to the StegoList path, as shown in Fig. 11.

```
Procedure. FSC
Step1: StegoList SV=new StegoList
Step2: SV.SAKey = SAKey
       SV.Solution= CurrentVector
Step3: Return SV
```

Figure 11. Final Solution Construction

4.1.5 LCG Formula and Conditions

LCG is a common random generator that can generate a sequence of RDs over the interval of [0, M-1] without any repetition until completing the cycle M. This random generator has a set of preconditions that must be satisfied to complete a task [15]. The conditions are as follows: 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 is 4. The general formula of LCG is shown in Fig. 12.

$X_{i+1} = (AX_i + C)\ Mod\ M$, *such that*

- X_{i+1} *repsents the next random number*
- X_i *represents the current random number*
- *A represents the multiplier*
- *C represents the non-common factor*
- *M represents the cycle of generator*

Figure 12. LCG Formula

4.1.6 FLV Byte Substitution

The method used to hide a secret message in a cover image is the random substitution technique using SA procedure. This method bases on the principle of updating one byte from the secret message by another byte from a hosting or cover file according to SA-constructed path. The method is fast and simple. However, the distortion in the host file, where data are to be embedded, is noticeable when the number of modified bytes becomes large in the same video tag.

In this work, each byte in the selected position hides a byte for a secret message. Thus, the secret message of length *L* requires *L* from a cover FLV file. The target position for hiding secret message bits is selected sequentially from the solution vector that is returned from the SA procedure. Position locations are selected sequentially, but secret message bytes are scattered randomly inside the FLV file.

The substitution process begins by loading the cover file into a buffer (host) to set or capture bytes from it. The secret message is then opened as a binary file (FS) to read the data byte by byte. The process of embedding secret data is accomplished through a loop that starts from position zero and terminates at the end of binary stream file. Inside the loop, the procedure reads the position from the solution path, reads bytes from the secret message, and replaces it within the FLV file. The embedding process remains working until all bytes of the secret message are hidden in the FLV file buffer, as shown in Fig. 13.

```
Procedure Embedding Process
Step1:Host = FileStream(FLV, File.Write)
Step2:FS = FileStream(SecretMessage , File.Read)
Step3: SV= BuildSASolutionVector( )
Step4:BestPath = SV.Solution.Head
Step5: Counter=1
Step6:While Counter <= FS.Length Do
        W = FS.ReadByte( )
        Host.Postion=BestPath.Position
       Host.Write(W)
       BestPath=BestPath.Next
       Counter=Counter+1
      End While
Step7:Host.Save(StegoFile)
```

Figure 13. Embedding Process

4.2 Extraction Process

The extractor should be aware of numerous important things that represent EKeys to retrieve the secret message from a stego-FLV file. The keys are the number of bytes that are hidden (length of secret message), SAKey for building solution path, and extension of secret message (type of message). Once the earlier keys are available, the extraction process can accomplish and distribute its tasks, as shown in Fig. 14.

SAKey, stego-FLV, and ML play important roles in generating the path where data are hidden. ML represents the cycle of LCG function, and it is used to compute the required values of random generator. SAKey is used as seed for LCG function that generates the vector of SA, where the data are embedded in a stego-FLV file. The length of solution path equals the length of secret message L, as mentioned earlier. Extracted bytes require combination to build the secret message. The extracted bytes are stored in a binary stream file with user-predefined extension to construct the secret message that is hidden. Fig. 15 shows how the extraction process executes and distributes its phases.

```
Procedure . Extraction Process
Step1:Stego = FileStream(FLV, File.Read)
Step2:FS = FileStream(Message_Path , File.Create)
Step3: SV= BuildSASolutionVector(Stego, Ekey,L )
Step4: BestPath = SV.Solution.Head
Step5: Counter=1
Step6:While Counter <= FS.Length Do
    Begin
        Stego.Postion=BestPath.Position
        W = Stego.ReadByte( )
        FS.Write(W)
        BestPath=BestPath.Next
       Counter=Counter+1
   End While
Step7:FS.Save(Message_File)
```

Figure 15. Extraction Process

5. Results and Analysis

The proposed system is tested using a data set of FLV files and secret messages that have different sizes, as shown in Table 2. The maximum size of secret message is computed according to the number of video tags in a cover file. The maximum number of bytes that can be employed to hide message bytes is determined by a visual appearance flicker, which may result from overload bytes at the end of video tags.

Table 2. *Evaluation Data Set*

Secret Message	Size
Secret 1	782B
Secret 2	1667Bs
Secret 3	2257 B

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

The visual appearance of a stego-FLV file is compared with that of the cover one. MR, capacity of cover FLV file, robustness against modification, detecting capability, and security are the main issues that are taken in evaluating the proposed system.

Human eyes and magnifying glasses are used to evaluate the visual appearance of stego-FLV file. The result file is compared with the original one to determine noise, flicker, and irregularity in the final form of the file. The files of secret1, secret2, and secret3 are embedded in cover1 and cover2 at different sizes for hosting list to test the system. The visual appearance of the result stego-FLV file at different cases is similar to that of the cover one without any noise or flicker. The selection of best path by SA, the random distribution of bytes within selected video tags, and the large size of video tags compared with the size of distributed bytes are the main reasons for reducing or eliminating the noise and flicker in the result stego-FLV file.

MR is computed by determining the number of modified bytes, subtracting them from the secret message length and then dividing the result by the total number of bytes from the cover FLV file as shown in Equation 2.

Modification Rate (MR) = (Secret Message Length- Modified Bytes) / Host FLV file Length (2)

The MR value is based on the length of user-selected secret message, length of the cover FLV file, and type of data to be hidden; thus, a short message with a long FLV file provides a low MR. Our system offers a converging MR for different secret messages in the same cover file, as shown in Table 3. Variation in MR results from random and different distributions of bytes within video tags. Fig 16 and Fig. 17 show how the additional bytes provide an improvement to the MR value, because it increases the size of selection random list which improves the randomness selection process and decreases the distance between bytes of secret message and FLV file bytes.

The maximum capacity of a cover FLV file is determined by the number of video tags multiplied by the value of 30. Therefore, the maximum size of secret message that can be embedded in cover2 without any notification is 5100 bytes, which represents 4% of the original file.

Information is considered robust when it is embedded inside a cover file and encounters any modifications. However, the proposed method is considered a robust one because secret data are embedded as a part of encoding information for an FLV file; hence, the secret data are corrupted only when the host FLV file is damaged.

Table 3. MR for Cover1 and Cover 2 with Different Secret Messages

Cover	Message	MR With Required Bytes	MR With Additional Bytes
Cover 1	Secret 1	0.0011689	0.00105436
Cover 1	Secret 2	0.0024789	0.00223661
Cover 1	Secret 3	0.003354	0.00304949
Cover 2	Secret 1	0.0060103	0.00500259
Cover 2	Secret 2	0.0169932	0.01079317
Cover 2	Secret 3	0.1733568	0.14327956

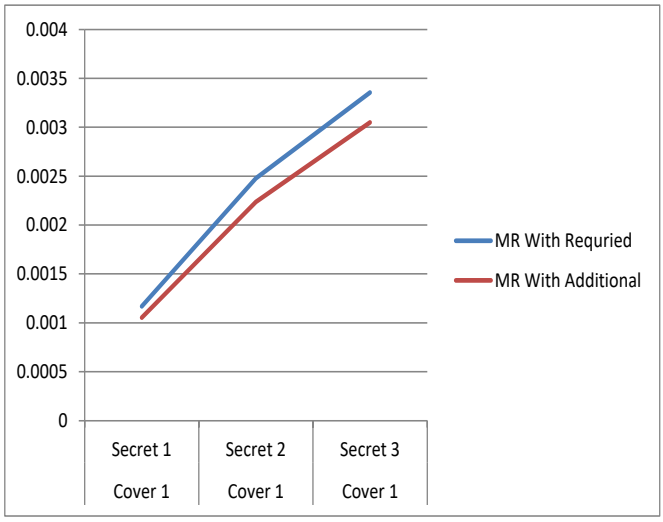

Figure 16. MR for required and additional bytes in Cover1

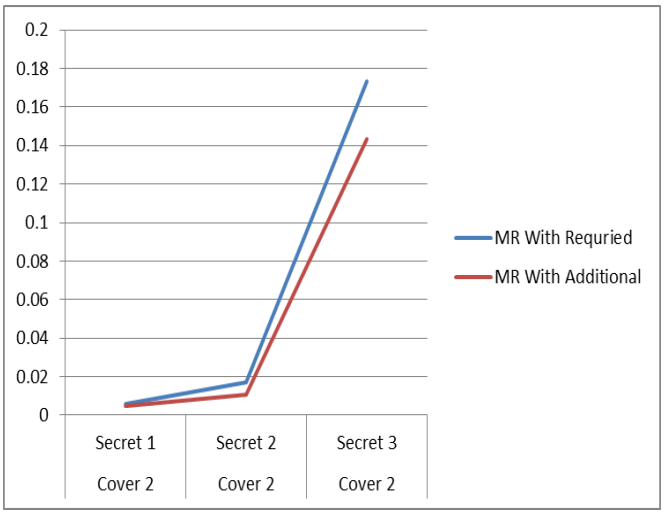

Figure 17. MR for required and additional bytes in Cover2

Undetectability, as a feature, is applied in the proposed method because secret data bytes are distributed randomly within random video tags according to SA best path. Thus, the host FLV files are not doubtable or suspicious for steganalysis.

Complete knowledge about keys is required for attackers to extract hidden messages. The proposed method is therefore considered a secure one because different keys are utilized. The extraction process requires EKey, ML, additional key, and message extension to extract secret messages. The previous description indicates that the extraction process is difficult, complex, and secure.

6. Conclusions and future work

This paper presents a secure video-FLV file stenographic method for information security. The approach bases on the idea of distributing a secret message bytes according to the SA minimum path within a host FLV file. The main goal of this study is to construct a solution that is robust against attacks, is effective in generating stego-FLV files, cannot be predicted through visual appearance, and robust against the sniffing of secret messages. The proposed method satisfies most of the security requirements (visual appearance, security, and undetectability) and exhibits adaptability to FLV files as hosts to hide secret messages.

In near future will determine the minimum path for hiding secret messages in FLV files by using intelligent water drop algorithm and will compare the result with the current approach. The adaptability of other FLV file tags for hiding secret data will also be checked.

7. Acknowledgment

This research is supported by Dept. of Information Technology, Al-huson University College, Al- Balqa Applied University.

References

[1] Bansod, S. and G. Bhure, Data encryption by image steganography. Int. J. Inform. Comput. Technol. Int. Res. Publ. House, Vol. 4, No. p. 453-458, 2014.

[2] Abdelwahab, A.A. and L.A. Hassaan. A discrete wavelet transform based technique for image data hiding. in Radio Science Conference, 2008. NRSC 2008. National. IEEE,pp. 1-9, 2008.

[3] Bertsimas, D. and J. Tsitsiklis, Simulated annealing. Statistical science, Vol. 8, No. 1, p. 10-15, 1993.

[4] Atiea, M.A., Y.B. Mahdy, and A.-R. Hedar, Hiding Data in FLV Video File, in Advances in Computer Science, Engineering & Applications. 2012, Springer. p. 919-925,2012.

[5] Cruz, J.P., N.J. Libatique, and G. Tangonan. Steganography and data hiding in flash video (FLV). in TENCON 2012-2012 IEEE Region 10 Conference. IEEE,pp. 1-6, 2012.

[6] Bawaneh, M.J., A Novel Approach for Image Steganography Using LCG. International Journal of Computer Applications, Vol. 102, No. 10, 2014.

[7] Arraziqi, D. and F. Ferdinandus. Optimalisasi Steganografi Pada File Flv Memanfaatkan Metode Injected At End Of All Video Tag Dengan Penambahan Kompresi. Seminar Nasional Inovasi dalam Desain dan Teknologi,(IDeaTech 2015),pp. 2015.

[8] Dasgupta, K., J. Mandal, and P. Dutta, Hash based least significant bit technique for video steganography (HLSB). International Journal of Security, Privacy and Trust Management (IJSPTM), Vol. 1, No. 2, p. 1-11, 2012.

[9] Alwan, W., Dynamic least significant bit technique for video steganography. Journal of Kerbala University, Vol. 11, No. 4, p. 7-16, 2013.

[10] Kaur, M. and A. Kaur, Improved Security Mechanism of Text in Video using Steganographic Technique. International Journal, Vol. 2, No. 10, 2014.

[11] Shinde, P. and T.B. Rehman, A Novel Video Steganography Technique. International Journal, Vol. 5, No. 12, 2015.

[12] Jókay, M., The design of a steganographic system based on the internal MP4 file structures. development, Vol. 17, No. p. 18, 2011.

[13] Singla, N. and R.B. Kaur, A Modified Data Hiding Approach for Audio and Video Data. International Journal of Advanced Research in Computer Science, Vol. 7, No. 6, 2016.

[14] Sangareswari, S. and M.V. Aruna, Application of image hiding in mp4-video using steganography technique. International Journal of Electrical, Computing Engineering and Communication (IJECC) Vol, Vol. 1, No., 2015.

[15] Danti, A. and G. Manjula, AN INNOVATIVE APPROACH FOR VIDEO STEGANOGRAPHY USING STATISTICAL FEATURES IN ROUND-LSB. International Journal of Engineering Applied Sciences and Technology, Vol. 1, No. 8, p. 194-199, 2016.

[16] Basha, T.G. and S.H. Zabeen, DESIGN AND EVOLUTION OF PERFORMANCE OF SPREAD SPECTRUM TECHNQIUES FOR IMAGE STEGANOGRAPHY USING MIGLS. INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN ELECTRICAL, ELECTRONICS, INSTRUMENTATION AND CONTROL ENGINEERING, Vol. 2, No. 9, 2014.

[17] Obeidat, A.A. and M.J. Bawaneh, A novel FLV steganography approach using secret message segmentation and packets reordering. Int. J. Res. Comput. Applic. Robot, Vol. 4, No. p. 44-54, 2016.

[18] YADAV, V., A SECURE VIDEO STEGANOGRAPHY USING RANDOM ENCODING/DECODING. International Journal of Advance Engineering and Research Development, Vol. 2, No. 7, p. 273-281, 2015.

[19] ElSharkawy, M.I., Integrating and Securing Video, Audio and Text Using Quaternion Fourier Transform. International Journal of Communication Networks and Information Security (IJCNIS), Vol. 9, No. 3, 2017.

[20] Khalil, M., Quaternion-based encryption/decryption of audio signal using digital image as a variable key. International Journal of Communication Networks and Information Security (IJCNIS), Vol. 9, No. 2, p. 216, 2017.

[21] Apau, R., J. Hayfron-Acquah, and F. Twum, Enhancing Data Security using Video Steganography, RSA and Huffman Code Algorithms with LSB Insertion. International Journal of Computer Applications, Vol. 143, No. 4, 2016.

[22] Adobe. ADOBE FLASH VIDEO FILE FORMAT SPECIFICATION, VERSION 10.1 2010 [cited 2017; Available from: Published August 2010, (http://download.macromedia.com/f4v/video_file_format_spec_v10_1.pdf). 2010.

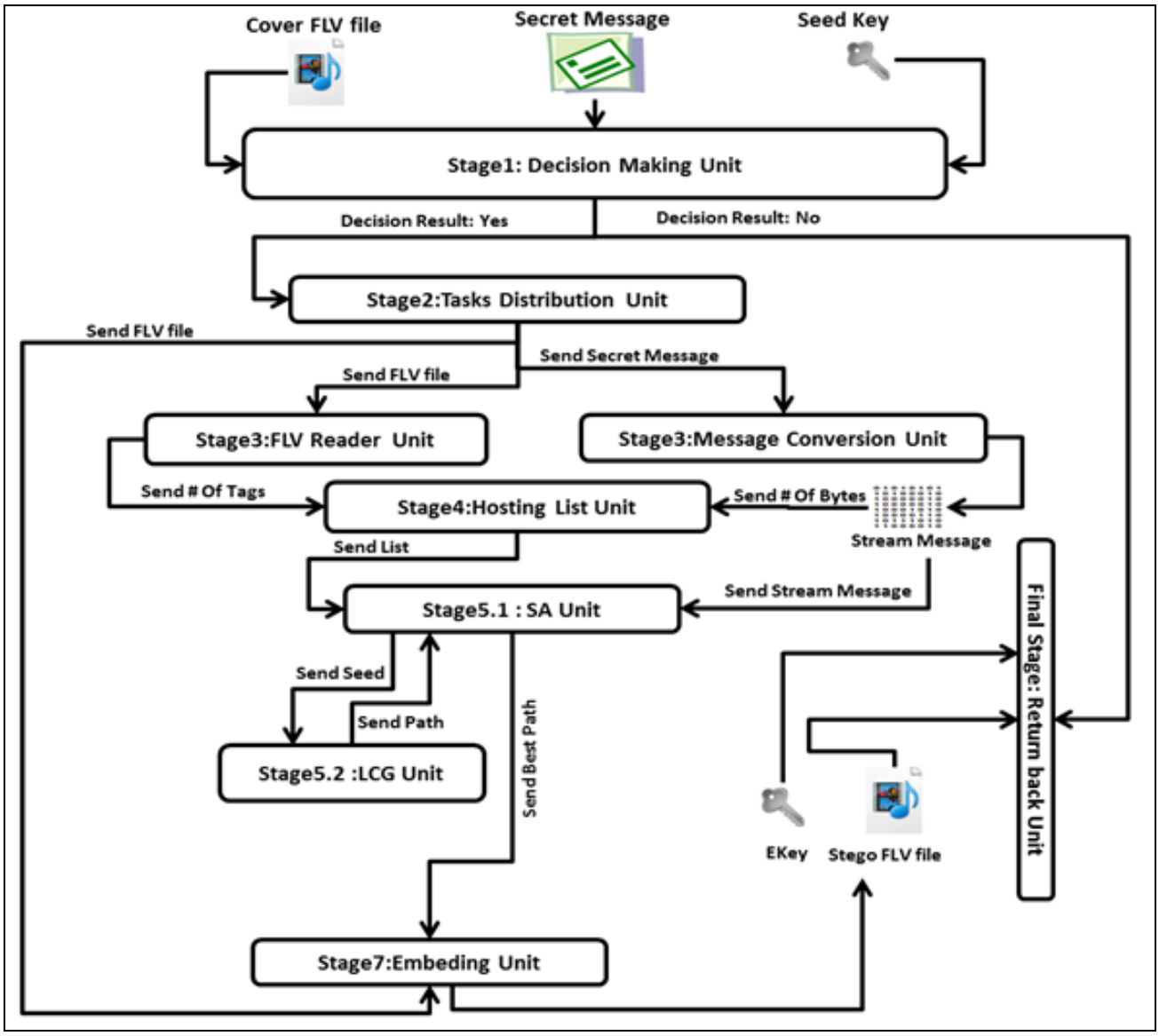

Figure2: Hidden Process Frame Work

Procedure . HCP
Step1: Set Fetching_Bytes= Required_Bytes+ Additional_Bytes *Step2: Set Temp_Tags_Pointer= Head_Tag_Video* *Step3: Set Tags_Counter= GetVideoTagsNumber( )* *Step4: For i = 1 To Tags_Counter* *Set Cover.Position=Temp_Tags_Pointer.Position* *Set Cover.Position= Cover.Position - Fetching_Bytes* *For j=1 To Fetching_Bytes* *P= Cover.Position* *Byte W =Cover.ReadByte( )* *Hosting_List.Add(P, W)* *Next For* *Next For* *Step5: Hosting_List*

Figure 4. Hosting List Construction Procedure(HCP)

Procedure BVC

```
While Max_Temperature > Absolute_Temperature Do
    OldSeed = Seed
    NextVector= GetNextSAVector( )
    NewDistance = GetTotalDistance(NextVector)
    Fitness= e^(-deltaDistance / Max_Temperature)
    deltaDistance = NewDistance - Distance
    If  deltaDistance < 0  Then
        CurrentVector=NextVector
        Distance = deltaDistance + Distance
        SA key = OldSeed
    ElseIf Distance ≥ 0 and Fitness> RD Then
        CurrentVector=NextVector
        Distance = deltaDistance + Distance
        SA key = OldSeed
    End IF
   Max_Temperature *= Cooling_Rate
End While
```

Figure 10. Best Vector Construction

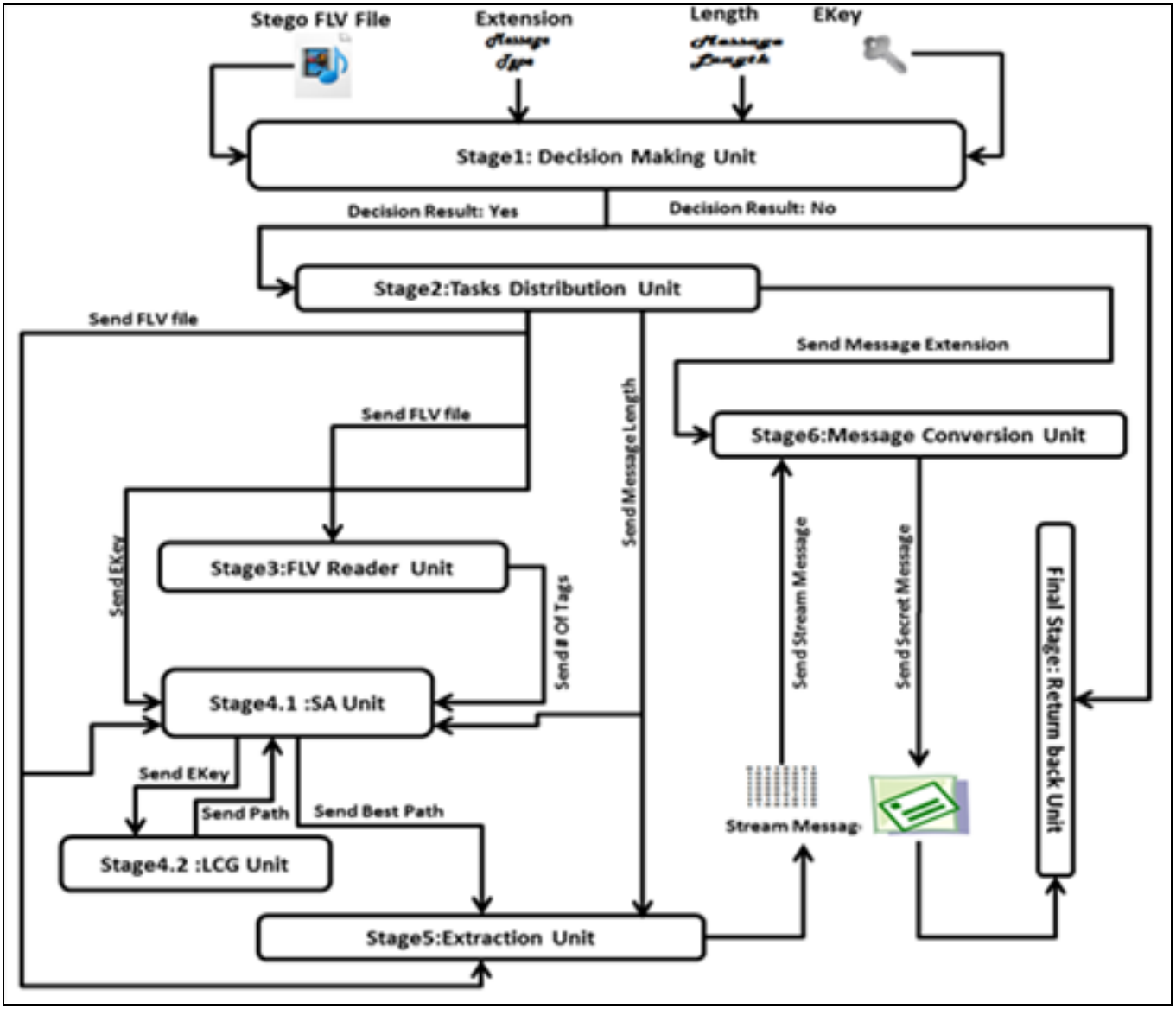

Figure 14. Extraction Process Framework