Imperial Journal of Interdisciplinary Research (IJIR)
Vol-3, Issue-4, 2017
ISSN: 2454-1362, http://www.onlinejournal.in

Video Encryption & Decryption using Compression techniques

Mayur Shah[1], Prakash Choudhary[2] & Abhishek Nijai[3]
[1,2,3] Department Of Computer Engineering, Vidyardhini's College Of Engineering And Technology.

Abstract: *The multimedia technology has a rapid development in various fields like medical, commercial, and defense. Therefore, security and privacy has become an important. When we send any video data over the insecure channel it consumes more time because of the large size of the video file. Therefore, video data should be compressed before sending to the destination. Another important factor during data transfer is security which can be achieved by video encryption. Compression and encryption algorithms can be classified into two main categories: Independent encryption technique and joint compression and encryption technique. Independent encryption techniques can further be classified as heavy weight and light weight encryption algorithms. Joint compression and encryption is employed to enable faster and secured transmission of video data.*

1. Introduction

Video data takes more time for encryption, because of its large size. Since the size of video data is huge in volume, it needs to be compressed and encrypted to avoid security threats and delay.
There are two strategies for this, namely, independent encryption algorithms and joint compression and encryption algorithms.
In independent encryption algorithms, both compression and encryption are done independently as two different steps by employing suitable algorithms. When independent encryption algorithms are employed, overall system performance decreases due to the huge computation overhead involved.
In joint compression and encryption algorithm, both the steps, namely, compression and encryption are integrated together as a single step. There are two approaches for joint compression and encryption algorithm: the first method employs encryption after compression and the second method does encryption before compression.

1.1. Aim and Objective

This project would be focused on the video encryption module where we would perform research on the techniques and methodology to encrypt video and to develop a module for a technique that we prefer to use in this project. This module will hide the text in the video using key and pass it to next module to encrypt the video by using modified zigzag and block scrambling algorithm and then we are performing compression to reduce the size of encrypted video and after compression pass it into the final module in which user sends a compressed encrypted video along with the key to the user to decompress as well as decrypt the video and obtain the hidden text. Video encryption makes the data more secure.

The objectives of the video encryption are:

- To study video encryption technique.
- To perform the encryption on .mpeg, .flv, .avi, etc. video formats.
- To study and perform modified zigzag algorithm and block scrambling method.
- To study compression as well as decryption techniques.

1.2. Scope

This system overcomes most of the disadvantages of existing system. The scope of the project is concerned with the zigzag algorithm based on block scrambling is implemented. Here only one level of scrambling that is inter level scrambling is implemented. This is done to all the frames not only to the i-frame.

We make use of modified zigzag algorithm rather than AES algorithm because encryption of entire data is faster.

2. Video Encryption

Video encryption or video scrambling is a powerful technique for the preventing unwanted interception and viewing of transmitted video, for example from a law enforcement video surveillance being relayed back to a central viewing center.
Video encryption is the easy part. It is the unscrambling that's hard. There are several

techniques of video encryption. However, the human eye is very good at spotting distortions in pictures due to poor video decoding or poor choice of video encryption hardware. So, it is important to choose the right video encryption else your video transmissions may be unsecure or the decoding video unviewable.
The development of encryption systems aims to provide a secure and reliable way for information exchanges.

3. Data Hiding

With the encrypted video, although a data-hider does not know the original video content, he can embed additional message into the video by modifying a small proportion of encrypted video. Firstly, the data-hider segments the encrypted video in to number of frames then each frame convert into a number of non-overlapping blocks sized by BxB. Each block is having the one additional bit for each block, pseudo randomly divide the B2 pixels into two sets B0 and B1 according to a data hiding key. Here, the probability that a pixel belongs to B0 or B1 is 1/2.

4. ZIGZAG ALGORITHM

This section explains the zigzag method in detail below.
This algorithm divides the input bit stream of the video into equal size blocks of size 16x16. Then the partitioned blocks are rearranged based on the zigzag rule. This zigzag rule rearranges the data based on the data itself of the input key. This rearranged data is then swapped in a pseudo-random manner within a block. This algorithm offers high speed of encryption but is not standard codec compliant and it encrypts the whole video stream which is redundant.

4.1. Original Zigzag method

A line or course that proceeds by sharp turns in alternating directions is known as zigzag rule. By this rule first bit of selected bits will be placed in right position. Again, second bit will place in left and third bit will be placed in right position and so on.
Here is an example of zigzag:
Let's select two-bit BIT = "1010010100101011"
BITzz = [Empty]
First bit of BIT = 1
It will be placed in the right position of BITzz.
So, BITzz = "1"
Second bit of BIT = 0
It will be placed in the left position of BITzz.
So, BITzz = "01"
Third bit of BIT=1
It will be placed in the right position of BITzz.
So, BITzz = "011"
Forth bit of BIT= 0
It will be placed in the left position of BITzz.
So, BITzz = "0011"
After placing all bits BIT will become
BITzz = "1000110011000111"

4.2. Modified Zigzag Algorithm

1) Read the given video file using MATLAB video read function.
2) Get the user key from the user of length of 32 characters.
3) Convert the input key into ASCII value.
4) Convert the ASCII value into binary with a resolution of 8 bits.
5) Extract the frames from the given video file.
6) Play the video file.
7) For each frame do the following steps
   a. Resize the frame to a size of 256x256
   b. Segment the frames into blocks of 16x16
   c. For each of the 16x16 blocks do the following
      i. Rearrange the blocks in the form of an array of cell 1x256
      ii. When a '0' appears in the key place the block under consideration before the previously operated block
      iii. When a '1' appears in the key place the block under consideration after the pre-viously operated block
      iv. Rearrange the blocks to form the 256x256 image and save as frame

8) Stop.

5. Intra block scrambling

Even though the scrambling is performed, it was per-formed only to the blocks. But the data in individual blocks are still recognizable and this can solve by further scrambling the data inside the individual blocks. This block size can be varied based on the need, thus the computational load can be kept at an optimal level.

6. Zip Compression

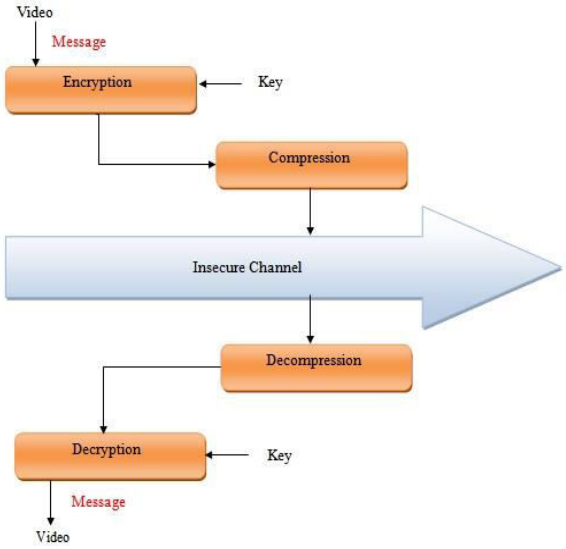
ZIP is an archive file format that supports lossless data compression. A .ZIP file may contain one or more files or directories that may have been compressed. The .ZIP file format permits a number of compression algorithms, though DEFLATE is the most common.
.ZIP files are archives that store multiple files. .ZIP allows contained files to be compressed using many different methods, as well as simply storing a file without compressing it. Each file is stored separately, allowing different files in the same archive to be

compressed using different methods. Because the files in a .ZIP archive are compressed individually it is possible to extract them, or add new ones, without applying compression or decompression to the entire archive. This contrasts with the format of compressed tar files, for which such random-access processing is not easily possible.
A directory is placed at the end of a .ZIP file. This identifies what files are in the .ZIP and identifies where in the .ZIP that file is located. This allows .ZIP readers to load the list of files without reading the entire .ZIP archive. .ZIP archives can also include extra data that is not related to the .ZIP archive. This allows for a .ZIP archive to be made into a self-extracting archive (application that decompresses its contained data), by prepending the program code to a .ZIP archive and marking the file as executable.

7. LZW Compression

LZW compression is the compression of a file into a smaller file using a table-based lookup algorithm invented by Abraham Lempel, Jacob Ziv, and Terry Welch.
A particular LZW compression algorithm takes each input sequence of bits of a given length (for example, 12 bits) and creates an entry in a table (sometimes called a "dictionary" or "codebook") for that particular bit pattern, consisting of the pattern itself and a shorter code. As input is read, any pattern that has been read before results in the substitution of the shorter code, effectively compressing the total amount of input to something smaller.
LZW encoded data consists entirely of 12 bit codes, each referring to one of the entries in the code table. Uncompression is achieved by taking each code from the compressed file, and translating it through the code table to find what character or characters it represents. Codes 0-255 in the code table are always assigned to represent single bytes from the input file. For example, if only these first 256 codes were used, each byte in the original file would be converted into 12 bits in the LZW encoded file, resulting in a 50% larger file size. During uncompression, each 12 bit code would be translated via the code table back into the single bytes. LZW also performs well when presented with extremely redundant data files, such as tabulated numbers, computer source code, and acquired signals.

8. Working of System



9. Proposed Methodology

First Module: Original video
-The video of .mpeg, .avi, etc extension will be accepted as input.

Second Module:
-Get the user key from the user of length of 32 characters.
- Convert the input key into ASCII value.
-Convert the ASCII value into binary with a resolution of 8 bits.

Third Module:
-Extract the frames from the given video file.
- Play the video file.
- Resize the frame to a size of 256x256.
- Segment the frames into blocks of 16x16.
- Apply zigzag algorithm.

Fourth Module:
-Apply Compression to the file.

Fifth Module:
-Apply Decompression to the file.
-Decrypting the video.

10. Hardware and Software Requirements

Hardware Requirements:
System : Pentium IV 2.4 GHz or higher.
Hard Disk : 100 GB or greater.
Monitor : 15 VGA Color.
Ram : 1GB or greater.
Software Requirements:
Operating system :Windows7/8/8.1/10
Programming Language : MATLAB.
Software : MATLAB R2015a.

8. Conclusion

The multimedia technology has a rapid development in various fields like medical, commercial, and defense. Therefore, security and privacy has become an important issue. So as to overcome this issue we have proposed the system, in which we take a video file and hide the message/text and apply various types of security so that the message/text cannot be obtain by the intruders.

9. Acknowledgement

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. We would like to extend my sincere thanks to all of them.

We are highly indebted to our guide for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

We would like to express my gratitude towards my parents & member of Vidyavardhini's College of Engineering And Technology for their kind co-operation and encouragement which help me in completion of this project.

Our thanks and appreciations also go to my colleague in developing the project and people who have willingly helped me out with their abilities.

10. References

1. A. F. M. SuaibAkhter, Saiful Islam, Md. JakirHossain, Rupam Deb, Dr. Md. BasirUddin, "MPEG Encryption by Zigzag, Partitioning and Swapping", IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.5, 116-120, May 2010

2. Fuwen Liu*, Hartmut Koenig, "A survey of video encryption algorithms", Elsevier publications, computers and security, June 2009

3. Ujwala Potdar1, Prof. K.T.Talele, Dr.S.T.Gandhe, "Perceptual Video Encryption for Multimedia Applications", IEEE Second International Conference on Computer Engineering and Applications, 2010.

4. Jayshri Nehete, K. Bhagyalakshmi, M. B. Manjunath, ShashikantChaudhari, T. R. Ramamohan, "A Real-time MPEG Video Encryption Algorithm using AES",

5. M. Eskicioglu, J. Town, and E. J. Delp, ―Security of digital entertainment content from creation to consumption,Signal Processing: Image Communication, vol. 18, pp. 237–262, and 2003.

6. J. Wen, M. Severa, W. Zeng, M. Luttrell, and W. Jin, ―A format compliant configurable Encryption framework for access control of multimedia, in Proc. IEEE Workshop on Multimedia Signal Processing, pp. 435–440, 2001.