VISVESVARAYA TECHNOLOGICAL UNIVERSITY
BELAGAVI – 590 018.

A Project Phase-2
Report on,

"Video Encryption with Message Embedding and motion Vectors"

(KSCST Sponsored)

Submitted in Partial Fulfilment of the Requirement for the Award of the Degree of

BACHELOR OF ENGINEERING

In

COMPUTER SCIENCE & ENGINEERING

Submitted by,

Mr. Balaram Chougale
USN: 2KD19CS024

Mr. Basavaraj Arjunagi
USN: 2KD19CS025

Mr. Basavaraj Sollapur
USN: 2KD19CS026

Mr. Lakkappa Basidoni
USN: 2KD19CS044

Under the Guidance of,
Dr. Bahubali M. Akiwate

Department of Computer Science & Engineering
K.L.E. COLLEGE OF ENGINEERING AND TECHNOLOGY
CHIKODI – 591201.

2022-23

VISVESVARAYA TECHNOLOGICAL UNIVERSITY
BELAGAVI – 590 018.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Certificate of Approval

This is to certify that **Mr. Balaram Chougale** bearing **USN: 2KD19CS024** Students have satisfactorily completed the Project Phase-2 work entitled**"Video Encryption with Message Embedding and Motion Vectors"** for the partial fulfilment of **Bachelor of Engineering in Computer Science and Engineering** prescribed by the Visvesvaraya TechnologicalUniversity, Belagavi for the academic year 2022-23.

The Project Phase-2 report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said Degree.

Guide	HOD	Principal
Dr. Bahubali M. Akiwate	**Prof. Ashwini V. Gavali**	**Dr. Prasad B. Rampure**

Name of the examiners **Signature with date**

1.

2.

A C K N O W L E D G E M E N T

The success and final outcome of this Project work required a lot of guidance and assistance from many people and we are extremely fortunate to have got this all along the completion of our project work. Whatever we have done is only due to such guidance and the assistance.

We owe our profound gratitude to our project guide **Dr. Bahubali M. Akiwate**, who look keen interest in our project work and guided us all along, till the completion of our project work by providing all the necessary information and valuable suggestion for developing a good system.

With deep sense of gratitude, we also acknowledge the encouragement of the Head of the Department **Prof. Ashwini V. Gavali** for her continuous support and permitting to make use of the facilities available in the department.

We extend our gratefulness to our project coordinator **Prof. Vinod B. Biradar**, Department of Computer Science and Engineering, for their great support in coordinating to the needs of us in our endeavour.

We express our sincere gratitude to **Dr. Prasad B. Rampure,** Principal, KLE College of Engineering & Technology, Chikodi for his support and encouragement in this regard.

We are also thankful and fortunate enough to get constant encouragement, support and guidance from entire teaching and non-teaching staff of Department of Computer Science and Engineering who helped us in successful completion of project work.

Last but not the least, we are thankful to our Parents, family members and all our friends for their support and help, extended during this Project work.

Mr. Balaram Chougale (2KD19CS024)

ABSTRACT

The challenges addressed in this work include keeping the encrypted video compliant with standardized decoders, correctly decrypting the video and finally, correctly extracting the message bits. The message embedding is achieved by altering the values of reference picture indices and motion vectors which results in scrambled video. Picture references are used in this work and therefore, combined with alteration of motion vectors, a maximum of six message bits can be embedded per coding unit.

Motion vectors are altered by swapping their x and y components and/or changing their signs.This is achieved with full compliance with the video syntax. To extract message bits, an authorized decoder builds a classification model per video sequence and uses it for predicting the true values of the reference indices and motion vectors. As such, message bits are extracted and the video is correctly reconstructed to its unscrambled state. Coding units that result in misclassification are identified at the encoder and excluded from message embedding.

Keywords: Message embedding, Motion vectors, Video encryption, Video decryption.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

HEVC	High Efficiency Video Coding
AVC	Advance Video Coding
DCT	Discrete Cosine Transform
CABAC	Context-Adaptive Binary Arithmetic Coding
MPEG	Moving Picture Experts Group
CU	Coding Unit
MV	Motion Vector
AES	Advanced Encryption Standard
MP4	MPEG-4 Part 14
MKV	Matroska Video
MOV	Quick Time Movie
AVI	Audio Video Interleave
WMV	Windows Media Video

CHAPTER 1

INTRODUCTION

To authenticate digital video and ensure its confidentiality and integrity, video encryption and data embedding techniques are used [1][2]. Video encryption can be used to increase the difficulty of piracy in digital videos and to protect privacy [3][4]. In all cases, authorized users can restore the encrypted video to its original state [5]. Additionally, with cloud storage becoming feasible and popular, customers might wish to encrypt their videos prior to outsourcing. This is needed as data security is a major obstacle for cloud adoption. For tampering detection, a cloud server manager can embed labelling and authentication data into encrypted video [6]. One approach to protect digital video is through encrypting the entire video standard cipher algorithms such as the Advanced Encryption Standard (AES) [7], however, the encrypted video will be no longer be compliant with standardized decoders which results in prohibiting post processing techniques like video transcoding and watermarking. Thus, video encryption is typically archived by altering selective syntax elements such as the sign of DCT coefficients [8], altering intra prediction modes [9] or altering motion vector difference signs [10].

Video encryption can be combined with data embedding in AVC and HEVC videos as reported in [11] and [12]. The work in proposed a comprehensive solution for encryption and data embedding by altering intra prediction modes, motion vector differences and quantized DCT coefficients of AVC videos. In [13] CABAC bin string substitution is used to embed data in partially encrypted AVC videos. The encryption is performed by altering various syntax elements such that the receiver extract embedded data in the encrypted domain using only the data-hiding key. An improved version of aforementioned solution was reported in [14] where encryption and data embedding did not affect the bitrate and maintained the full bitstream compliance.

Video encryption combined with data embedding is also reported for HEVC videos. A pioneering work was reported in [15] where visual protection of video is achieved by encrypting HEVC-CABAC bin strings whilst maintaining compatibility with standardized decoders and without causing an increase in video bitrate. In [16], motion vector differences, intra modes and quantized DCT coefficients of HEVC videos are altered to achieve these two tasks. An enhanced version of this work was reported in [17] where the data embedding rate is increased without resulting in excessive video bitrate. In this work, we perform partial encryption to HEVC videos by altering picture reference indices and motion vectors. To the Such alteration results in scrambled video by means of encryption.

The major advantage of the proposed work is that we embed up to six message bits in each coding unit of the video. This is made possible through altering the picture reference indices and the motion vectors at the syntax level. As a result, the video becomes decodable by standardized decoders yet scrambled. We

show that by using relevant feature variables, a machine-learning model can be built to unscramble the video and extract message bits.

To the best of our knowledge, video encryption and data embedding by altering picture references and motion vectors, as opposed to motion vector differences, is novel. Likewise, the operations at the decoder's side to extract the embedded data and reconstruct the video by predicting the original values of the picture references and motion vectors using machine learning is also novel.

The wide use of digital images and videos in various applications brings serious attention to security and privacy issues today. There are so many android phones activating each day and through that phone's so many videos are shared to each other. So data encryption is a suitable method to protect the data. Till, now various encryption algorithms have been proposed and widely used like DES, RSA etc most of them are used for text and binary data. It is difficult to use them directly in video encryption as the data in videos maybe of large volumes many a times and it requires real time operations. The sharing and surfing of Real Time Video project is quite similar to YouTube application in some functionality. The source video is uploaded by the user itself which undergoes through various process which takes care of video security. In this system, video will be saved in encrypted form and stored in database. And AES is used to Key generation and that key use to encrypt and decrypt the video form [18].

CHAPTER 2

LITERATURE SURVEY

The below table 2.1 shows the existing techniques used in the domain of video encryption with message embedding and motion vector.

Research Papers	Authors	Published Year	Review
HEVC Video Encryption With High Capacity Message Embedding by Altering the various Picture Reference Indices and Motion Vectors.	Tamer Shanableh	2022	This paper presents an several methods to realize video encryption by using two main modules: Quality segmentation module and feature extraction module.
A Review: Video Encryption Techniques, Advantages and Disadvantages.	Tameem Hameed Obaida, Abeer Salim Jamil	2022	Need to apply the method to a real-time video to know the encryption and decryption methods. This techniques used to encrypt the fully video, which has been used to increase security, but the encryption time has become larger because it focuses on fully encrypting the video.
Video Encryption and Decryption using AES	Tejaswini Sawant,Dipti Patil	2021	He takes different cases to show result of system between classical AES and modified AES in terms of computational complexity and security.
Video Encryption Algorithm Implemented in the Various Stages of Compression	S.Rajgopal	2020	Video data security is very important for multimedia commerce on the internet and real-time video multicast.

Video Encryption Techniques for A Reviews.	Sobia Shafiq, Sundas Hanif	2019	This technique to encrypt a video using Advance Encryption Standard (AES) depending on key size, the data blocks were encrypted in 10, 12 and 14 rounds of 128 bits. Following are the steps to encrypt a 128-bit blocks. 1. First the set of round keys are derived. 2. Plain text is initialized in the state array. 3. First round key is added to the starting state array. 4. Nine rounds of state alteration are performed. 5. After that 10th round of state alteration is performed. 6. Final state array is copied as output cipher text
Video Encryption: A Survey	Jolly shah , Vikas Saxena	2017	The idea of pure AES algorithm is to simply scrambles bytes within a frame of MPEG stream by permutation. Videos are transferred through various types of computer network using various methodologies.

Table 2.1 Existing Techniques

CHAPTER 3

PROBLEM STATEMENT

The purpose of this proposed work is to provide the correct data with security to the users. Our application will give you more security to the data present in the network and there will be able to reduce the loss of data in the network which will be transmitted from the sender to the receiver.

To fulfil security and privacy needs in various applications, encryption of videos is very important to prevent malicious attacks from unauthorized attackers.

CHAPTER 4

OBJECTIVES

The main objectives of our proposed work are as follows:

1) To embed the message by altering the values of reference picture indices and motion vector which results in scrambled video.
2) To avoid the illegal access of the data from unauthorized user and provide reliable data transmission in an effective manner.
3) To ensure the video security between sender and receiver with the fast growth communication and technology, security of sending confidential video information.
4) To protect the confidentiality of digital data stored on computer system or transmitted over the Internet or any other computer network.

CHAPTER 5

OVERVIEW OF THE SYSTEM

In the proposed system, message bits are embedded into a compressed HEVC video by means of altering the reference index of Coding Units (CUs) and their motion vectors. The embedding takes place at the bit stream level and therefore the locally decoded images of the encoder remain intact. However, the generated video bit stream has altered syntax elements and therefore decoding it results in a scrambled video. As such, the output of the proposed encoding process is an encrypted bit stream that embeds message bits. The details of message embedding and the alteration of the reference indices and motion vectors are presented in Section 6.1.

A standardized video decoder will be able to decode the bit stream into a scrambled video. This task is archived without crashing the decoder as the generated bit stream is standard compliant in terms of syntax. On the other hand, in the proposed decoding solution, a classification model is employed to predict the correct values of the reference indices and motion vectors. As such, the embedded message bits are extracted and the video can be reconstructed correctly to its unscrambled state.

Video Encryption is to encrypt the video, you could use a symmetric encryption algorithm like AES (Advanced Encryption Standard) to scramble the video frames. This would make the video unreadable without the correct decryption key. Message Embedding in this step, you could embed a message into the encrypted video frames using a technique called steganography. Steganography involves hiding information within an image or video by modifying the pixel values of certain frames or areas. The message could be a text message, an image, or any other form of data that you want to hide.

Motion vectors are used in video compression to identify areas of the video that have moved between frames. In this project, you could use motion vectors to further obfuscate the message by introducing random noise to the motion vectors. This would make it more difficult for an attacker to identify the areas of the video where the message is hidden. Decryption and Message Extraction to decrypt the video and extract the hidden message, you would need to reverse the encryption and motion vector obfuscation processes. Once the video frames have been decrypted, you could use a steganography tool to extract the hidden message from the frames where it was embedded.

CHAPTER 6

METHODOLOGY

6.1 Message Embedding

To embed message bits in a CU, the following motion information is altered; Vx, Vy and CU reference index (i.e ref_idx). Two bits can be embedded in Vx and Vy and 4 bits can be embedded in ref_idx as follows. For message bits 00, Vx and Vy remain as is. For bits 01, Vx and Vy are swapped and the first vector component is multiplied by −1, which results in (-Vy,Vx). For bits 10, both vector components are negated, which results in (-Vx,-Vy). And lastly, for bits 11, Vx and Vy are swapped and the second vector component is multiplied by −1, which results in (Vy,- Vx). This is arrangement is presented in Equation (1).

$$\text{Altered MV} = \begin{cases} (Vx,Vy), & \text{message bits} = 00 \\ (-Vy,Vx), & \text{message bits} = 01 \\ (-Vx,-Vy), & \text{message bits} = 10 \\ (Vy,-Vx), & \text{message bits} = 11 \end{cases} \quad (1)$$

These altered MV values are chosen to maximize the Euclidean distance between them and at the same time be reproducible at the decoder for message extraction. Additional 4 bits are embedded by altering the CU reference index according to the message bits as shown in Equation (2).

$$\text{Altered Ref\_idx} = \begin{cases} 1, & \text{message bits} = 0000 \\ 2, & \text{message bits} = 0001 \\ & \ldots\ldots\ldots \\ 15, & \text{message bits} = 1110 \\ 16, & \text{message bits} = 1111 \end{cases} \quad (2)$$

Only inter-coded CUs with MVs can embed 6 message bits in this proposed solution. Intra-coded CUs and skipped CUs are not used for message embedding. However, inter-coded CUs with nil MVs can still be used to embed 4 bits in the reference index. The maximum number of message bits per CU type is listed in Table 6.1.

The full arrangement of message embedding is listed in Table 6.1. Notice that in the third column a dash is used in message bits to emphasize that the first 2 bits are a result of MV alteration and the rest of the bits are a result of reference index alternation.

The below table 6.1 shows the maximum number of message bits per CU.

CU Type	Bits in MV	Bits in Reference index	Total Bits
Inter-coded CU with non-nil MV	2	4	6
Inter-coded CU with nil MV	0	4	4
Skipped CU	0	0	0
Intra-coded CU	0	0	0

Table 6.1 Maximum number of message bits per CU

The table 6.2 shows alteration of the MVs and reference indices are carried out as a post process at the encoder. This means that the alerted values are stored in the output bit stream but not used for motion estimation and composition. Consequently, an authorized decoder has three tasks to carry out. Firstly, it will predict the values of the unaltered MVs and unaltered reference indices; secondly, it will extract the embedded bits, and finally it will reconstruct the video.

CU reference indices are used in a technique called CU (coding unit) partitioning, which involves dividing a video frame into smaller rectangular regions. The CU reference index indicates the position of a particular CU within a larger block of CUs, allowing the decoder to reconstruct the full video frame from the compressed data.

MVs, on the other hand, are used to describe the motion of objects within a video frame. They represent the displacement of image blocks between consecutive frames and can be used to predict the location of objects in future frames, allowing for more efficient compression. To alter CU reference indices and MVs for message embedding, one could modify the existing indices and vectors by adding or subtracting small values that encode the message.

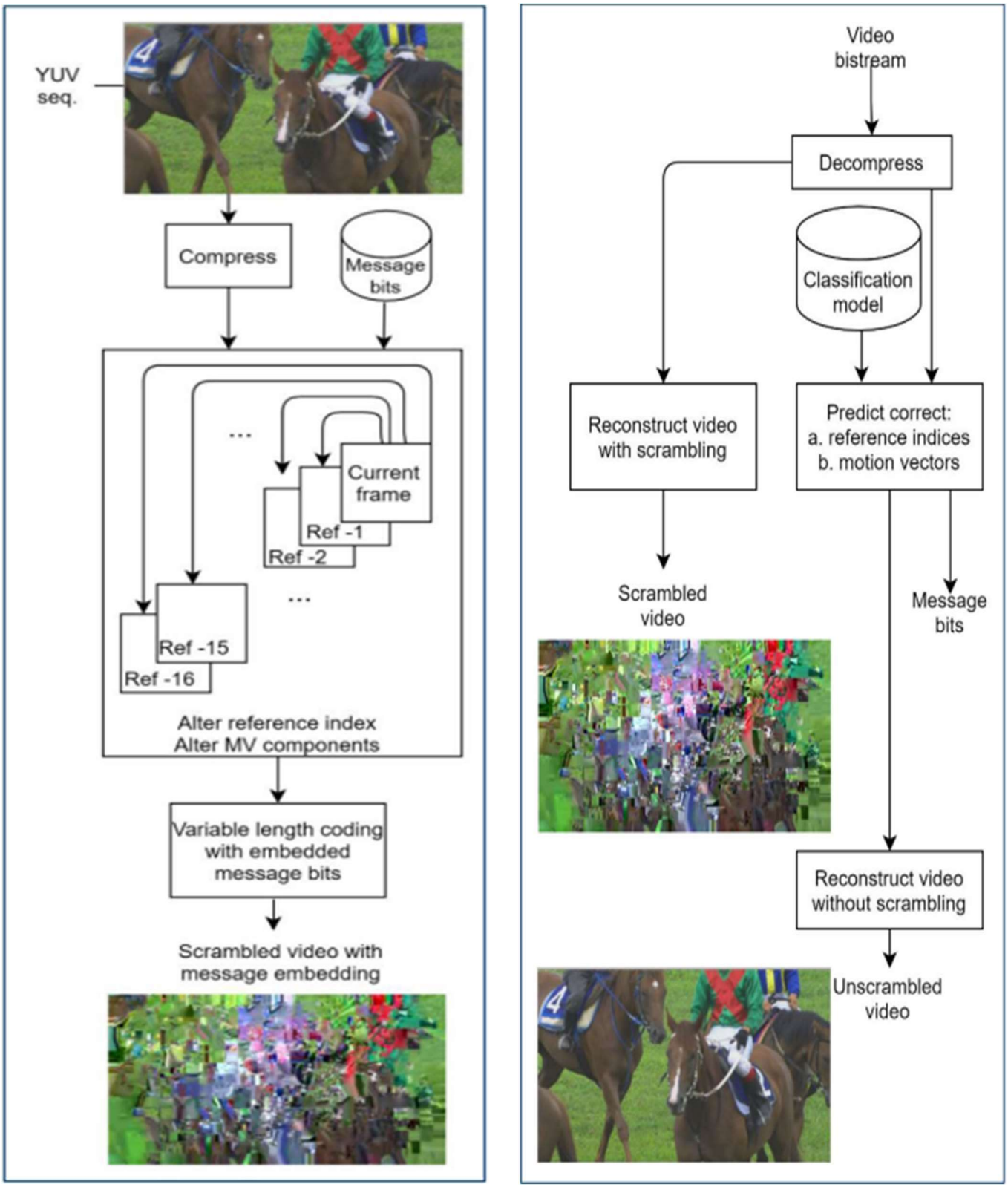
MVs are typically represented using two components: the horizontal component (x) and the vertical component (y). These components indicate the direction and magnitude of the motion between two frames. For example, if a particular block of pixels in frame A has moved one pixel to the right and one pixel down in frame B, the MV for that block would be (1, 1).

The following table 6.2 shows altering CU reference indices and MVs for message embedding.

MV alteration	Reference index alteration	Message bits
(Vx, Vy)	ref_idx = 1	00-0000
	ref_idx = 2	00-0001
	…	…
	ref_idx = 15	00-1110
	ref_idx = 16	00-1111
(-Vy, Vx)	ref_idx = 1	01-0000
	ref_idx = 2	01-0001
	…	…
	ref_idx = 15	01-1110
	ref_idx = 16	01-1111
(-Vx ,-Vy)	ref_idx = 1	10-0000
	ref_idx = 2	10-0001
	…	…
	ref_idx = 15	10-1110
	ref_idx = 16	10-1111
(Vy, -Vx)	ref_idx = 1	11-0000
	ref_idx = 2	11-0001
	…	…
	ref_idx = 15	11-1110
	ref_idx = 16	11-1111

Table 6.2 Altering CU reference indices and MVs for message embedding

The system overview of proposed message embedding and message extracting of video encryption decryption solution is shown in fig 6.2.



a. Encryption **b. Decryption**

Figure 6.1 Encryption and Decryption solution

The message embedding is achieved by altering the reference picture indices and motion vectors which result in scrambled video. To extract message bits, an authorized decoder builds a classification model per video sequence and uses it for predicting the true values of the reference indices and motion vectors. As such, message bits are extracted and the video is correctly reconstructed to its unscrambled state.

The proposed message embedding and message extracting video encryption decryption solution involves a system architecture that consists of several components working together to encrypt and decrypt video files with embedded messages.

Input Video: The input video is the raw video file that needs to be encrypted and has a message embedded into it. This video file can be of any format and resolution.

Message Embedding Module: This module is responsible for embedding the message into the input video. It takes the message as input and embeds it into the video frames using a secure encryption algorithm. The message is hidden within the video frames and is not visible to the naked eye.

Motion Vector Estimation Module: This module is responsible for estimating the motion vectors in the video frames. Motion vectors are used to identify and encode only the changes in the video frames, reducing the size of the encrypted video file.

Video Encryption Module: This module is responsible for encrypting the video frames using a secure encryption algorithm. It takes the motion vectors and the embedded message as inputs and encrypts the frames accordingly.

Encrypted Video Output: The encrypted video output is the result of the encryption process. It is a video file that contains the embedded message and is encrypted with a secure encryption algorithm. The encrypted video file can be of any format and resolution.

Video Decryption Module: This module is responsible for decrypting the encrypted video file. It takes the encrypted video file as input, decrypts it using a secure decryption algorithm, and extracts the embedded message.

Message Extracting Module: This module is responsible for extracting the embedded message from the decrypted video frames. It takes the decrypted frames as input, extracts the message using a secure decryption algorithm, and outputs the original message.

Decrypted Video Output: The decrypted video output is the result of the decryption process. It is a video file that has been decrypted and contains the embedded message. The decrypted video file is of the same format and resolution as the input video file.

6.2 Data Flow Diagram

The data flow diagram in fig 6.2 illustrates the encryption and decryption processing steps.

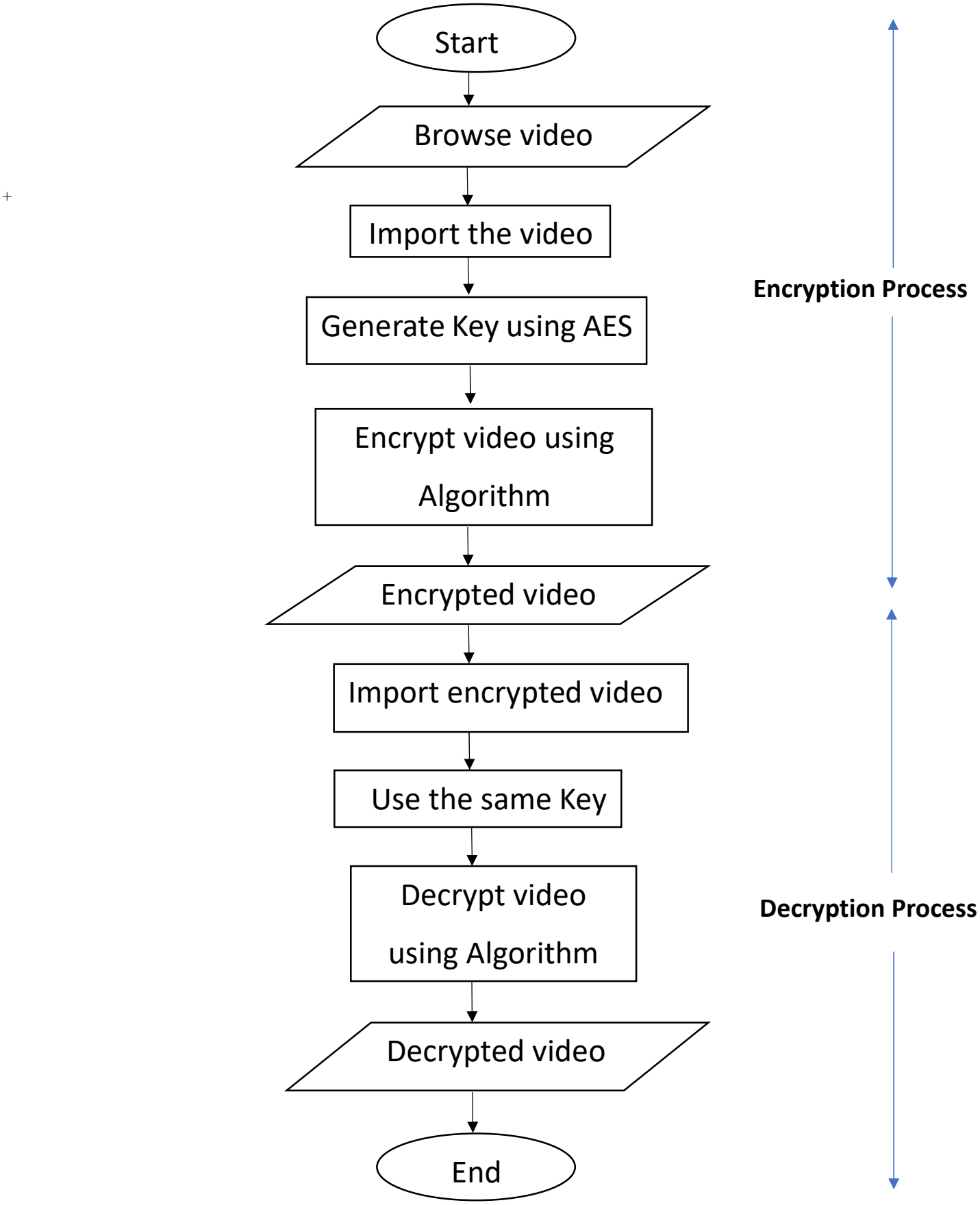


Figure 6.2 Data flow diagram

In this proposed system first browse the video after that the video will be imported to the encryption algorithm. Then the video will be encrypted using algorithm that will give encrypted video as output in the form of scrambled. For the video decryption process import data from encrypted video and next, video will be decrypted using the decryption algorithm. At the last decrypted video is generated.

The levels of DFD with their description are as follows.

Level 0 DFD:

- The video data is input into the encryption system.
- The encryption system receives the message to be embedded in the video.
- The motion vectors are generated based on the video data.
- The encrypted video with message embedding and motion vectors is generated as output.

Level 1 DFD:

- The video data is first preprocessed, including frame segmentation and motion estimation.
- The motion vectors are generated based on the preprocessed video data.
- The message to be embedded is encrypted and prepared for embedding.
- The motion vectors and the encrypted message are embedded into the video frames.
- The encrypted video with message embedding and motion vectors is output as the final product.

Level 2 DFD:

- The frame segmentation process involves breaking the video data into individual frames.
- The motion estimation process calculates the motion vectors for each frame.
- The message encryption process encrypts the message to be embedded.
- The embedding process combines the motion vectors and encrypted message with the video frames to generate the encrypted video with message embedding and motion vectors.

Level 3 DFD:

- The encrypted video with the embedded message is input into the decryption system.
- The decryption system receives the decryption key or password to unlock the encrypted message.
- The encrypted video with message embedding and motion vectors is processed to extract the motion vectors and encrypted message.
- The decryption system decrypts the encrypted message using the decryption key or password.
- The decrypted message is output as the final product.

6.3 AES algorithm

Random key generation is an essential step in using AES (Advanced Encryption Standard) encryption, as the security of the encryption relies on the secrecy and randomness of the key. AES is a symmetric encryption algorithm, which means that the same key is used for both encryption and decryption. Therefore, it is crucial to generate a random and secure key that is long enough to resist brute-force attacks.

The most commonly used key sizes for AES are 128-bit, 192-bit, and 256-bit. The larger the key size, the stronger the encryption, as it makes it harder for an attacker to guess the key through a brute-force attack. To generate a random AES key, a cryptographically secure random number generator should be used. In Python, the os.urandom() function can be used to generate a secure random byte string of the desired length, which can then be used as the AES key. Once a secure and random AES key is generated, it can be used to encrypt and decrypt data using the AES algorithm. The key must be kept secret and secure to maintain the confidentiality of the encrypted data.

Example for Key generation.

i. AES Key generation of 256 bit.
   Key : 0c9bc6aab8357e4f17ed6b93840e16f84b10e94ecb3d2aa12bc00ce6892e6ead
ii. AES Key generation of 192 bit.
   Key : 154231507324064bd1b7d507ad7241e86dff6d4cdd4f8fe6
iii. AES Key generation of 128 bit.
   Key : cce7b4e7427fa287227481791a88dc19

CHAPTER 7

SYSTEM REQUIREMENTS

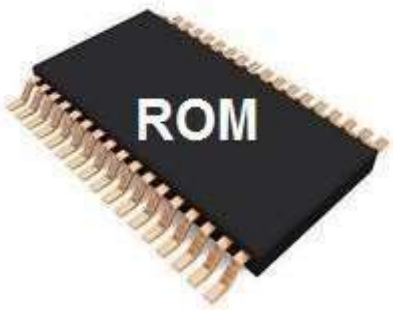
This chapter discusses the software and hardware requirements used in implementation.

7.1 Hardware Requirements

7.1.1 Processor : i3 or above

Processor is known as “Microprocessor“, and it likes as small type of chip that is placed in the computers and another electronics component. Processor can manage all instructions such as arithmetical, logical, input/output (I/O) and other basic instructions, which are created by hardware or operating system. Its main job role is to obtain input from input devices and then produce the accurately result on the output devices.

7.1.2 ROM : 500GB or more



ROM, which stands for read only memory, is a memory device or storage medium that stores information permanently. It is also the primary memory unit of a computer along with the random-access memory (RAM). It is called read only memory as we can only read the programs and data stored on it but cannot write on it. It is restricted to reading words that are permanently stored within the unit. ROM is volatile and loses its contents when power is turned off, ROM retains its data even when power is removed. ROM is typically used to store important system data, such as the computer's BIOS(Basic Input/Output System or firmware, which is necessary for the computer to boot up properly.

7.1.3 RAM : 8GB

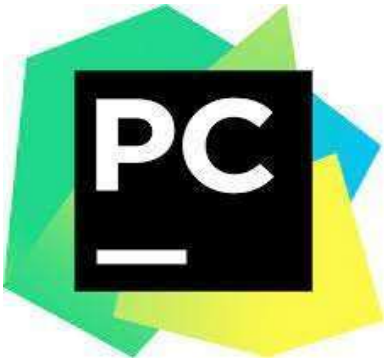
RAM, which stands for Random Access Memory, is a hardware device generally located on the motherboard of a computer and acts as an internal memory of the CPU. It allows CPU store data, program, and program results when you switch on the computer. It is the read and write memory of a computer, which means the information can be written to it as well as read from it.

7.2 Software Requirements

7.2.1 Windows 9 or above

Microsoft Windows is a graphical operating system developed and published by Microsoft. It provides a way to store files, run software, play games, watch videos, and connect to the Internet.

7.2.2 Pycharm IDE

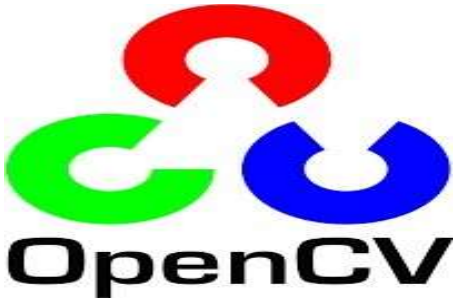


PyCharm is a dedicated Python Integrated Development Environment (IDE) providing a wide range of essential tools for Python developers, tightly integrated to create a convenient environment for productive Python, web, and data science development.

7.2.3 Numpy

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices. It is an open source project and you can use it freely. NumPy stands for Numerical Python.

7.2.4 OpenCV



It is possible to use OpenCV in the development of video encryption software. OpenCV (Open Source Computer Vision) is a library of programming functions primarily aimed at real-time computer vision. These tools can be used to perform various tasks related to video encryption, such as preprocessing the video frames before encryption, postprocessing the encrypted frames after decryption, and so on.

7.2.5 PIL

Python Imaging Library is a free and open-source additional library for the Python programming language that adds support for opening, manipulating, and saving many different image file formats. It is available for Windows, Mac OS X and Linux.

7.2.6 Tkinter Module

It is possible to use Tkinter for video encryption, but it would depend on how you choose to implement the encryption. You could use Tkinter to create a GUI for a video encryption application that allows the user to select the video to be encrypted, choose the encryption algorithm, and enter the encryption key.

7.2.7 Moviepy

It is possible to use Moviepy for video encryption, but it would depend on how you choose to implement the encryption. One way to use Moviepy for video encryption could be to read the video frames using Moviepy, encrypt the frames using a suitable encryption algorithm, and then write the encrypted frames to a new video file.

CHAPTER 8

TESTING

8.1 Introduction to Testing

Testing is a procedure, which uncovers blunders in the program. Programming testing is a basic component of programming quality affirmation and speaks to a definitive audit of determination, outline and coding. The expanding perceive ability of programming as a framework component and chaperon costs related with a product disappointment are propelling variables for we arranged, through testing. Testing is the way toward executing a program with the plan of finding a mistake. The plan of tests for programming and other built as items can be as trying as the underlying outline of the item itself It is the significant quality measure utilized amid programming improvement. Amid testing, the program is executed with an arrangement of experiments and the yield of the program for the experiments is assessed to decide whether the program is executing as it is relied upon to perform.

8.1.1 Unit Testing

- **Unit testing for video encryption:**

  Test that the encryption algorithm is working properly by encrypting a small sample video and verifying that it cannot be played without decryption.

  Test the decryption algorithm by decrypting the encrypted video and verifying that it can be played without any issues.

- **Unit testing for message embedding:**

  Test that the message embedding algorithm is working by embedding a small message into a video and verifying that it can be extracted without any errors.

- **Unit testing for motion vectors:**

  Test that the motion vector algorithm is correctly identifying motion in the video by comparing the original video to the version with motion vectors added.

  Test that the motion vectors can be used to accurately predict the next frame in the video.

- **Integration testing:**

  Test that all the components of the system work together as expected by encrypting a video with an embedded message and motion vectors, and then decrypting it and verifying that the message is intact and the video plays correctly.

8.1.2 System Testing

System testing is a level of software testing where complete and integrated software tested. The purpose of this test is to evaluate the system's compliance with the specified requirements.

8.1.3 Background & Purpose

The increasing use of video-based communication and the availability of high-speed internet has led to a growing demand for secure video communication. To meet this demand, various encryption methods have been developed to protect video data from unauthorized access or tampering. However, most of these methods do not consider the preservation of motion vector information, which is essential for efficient video compression and decompression.

The purpose of video encryption with message embedding and motion vector is to provide a solution that addresses both security and efficiency concerns. This approach allows for the embedding of messages in video data while preserving the motion vector information that is essential for efficient video compression and decompression. The encrypted motion vectors and the embedded message can be transmitted or stored separately, allowing the recipient to decrypt the motion vectors and use them for efficient video compression and decompression while also decrypting the embedded message.

The below table 8.1 shows the various test cases:

Test ID	Test Case Title	Test Condition	System Behaviour	Expected Result
T01	Video Encryption Key	Valid Encryption Key	The encryption key is used to encrypt the video	The video is successfully encrypted and can only be decrypted with the correct key.
T02	Video Decryption Key	Valid Decryption Key	The decryption key is used to decrypt the encrypted video	The decrypted video is the same as the original unencrypted video.
T03	Video Playback	Encrypted Video	The encrypted video is played back using a compatible video player	The video is played back correctly and the message embedded in the video is visible to the user with the correct decryption key.
T04	Message Embedding	Valid Message	The system embeds a message in the video	The embedded message is hidden within the video and can only be retrieved with the correct decryption key.

T05	Encryption Performance	Large Video	The system encrypts a large video file	The encryption process completes within an acceptable amount of time and the encrypted video file is of the expected size.
T06	Decryption Performance	Large Encrypted Video	The system decrypts a large encrypted video file	The decryption process completes within an acceptable amount of time and the decrypted video file is of the expected size.
T07	System Security	Correct Decryption Key	Correct decryption key is used to decrypt the video	The decrypted video is viewable and the message embedded within it is not visible.

Table 8.1 Test Cases

CHAPTER 9

APPLICATIONS

The applications of our implemented model are as follows:

- **Video Conferencing:** Real-time video encryption with message embedding and motion vector can be used to secure video conferencing systems used in business, education, and healthcare. This technology can help protect sensitive information exchanged during virtual meetings and prevent unauthorized access to the video stream.
- **Video Streaming Services:** Real-time video encryption with message embedding and motion vector can also be used in broadcasting and media industries to protect copyrighted content from piracy and illegal distribution. This technology can help ensure that only authorized viewers have access to the video stream.
- **Military and Defense:** Military and defense organizations often use video communication for strategic planning and coordination. Fast and secure real-time video encryption can help to ensure the confidentiality and security of these communications.
- **Medical Imaging:** Medical imaging systems often require encryption and security measures to protect patient data. Video encryption with message embedding and motion vector can be used to ensure that medical imaging data remains confidential and secure.
- **Video Surveillance:** Video surveillance is commonly used for security purposes in various environments. By using encryption with message embedding and motion vectors, video surveillance systems can prevent unauthorized access to the video footage, ensuring that only authorized personnel can access it.
- **Digital Rights Management** (DRM)**:** It can be used to protect copyrighted video content and prevent piracy. By encrypting the video and embedding copyright information, the content owner can ensure that the video is only accessed by authorized users.
- **Online Education:** With the rise of online education, it can be used to protect educational content from unauthorized distribution and piracy. This ensures that the content remains protected and can only be accessed by authorized users.

CHAPTER 10

RESULTS AND DISCUSSION

Fig 10.1 shows the login page where an authenticated user can enter their username and password to gain access to the next page. Once the user successfully login, they will be directed to the home page.

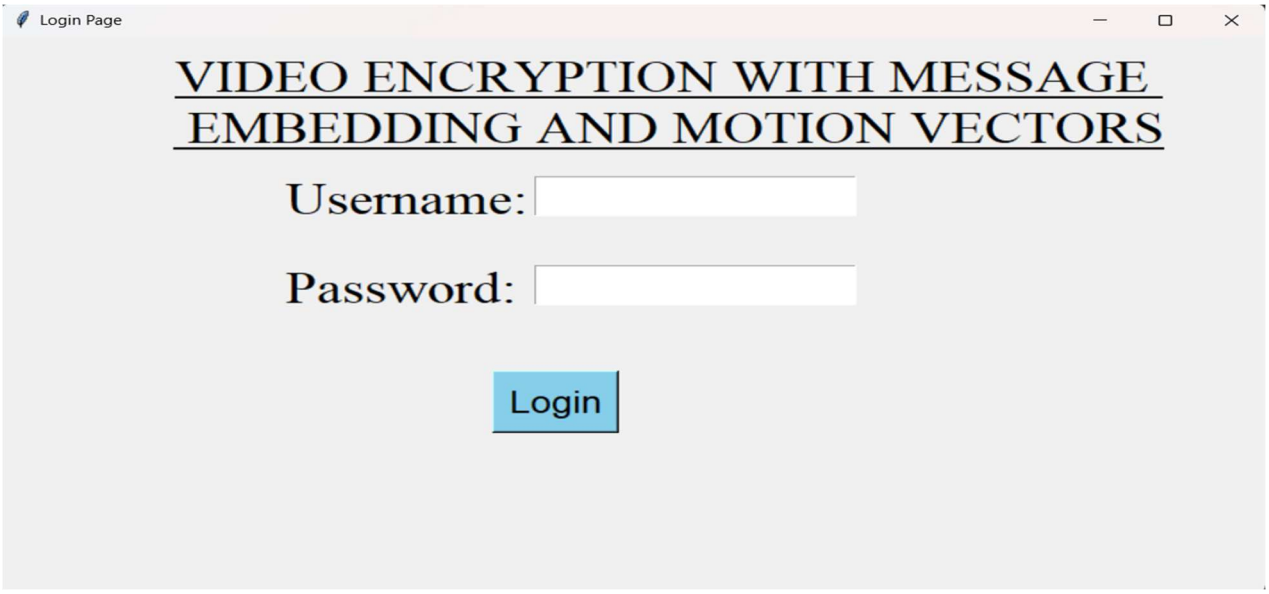


Fig 10.1 Login Page

Once the user login, the home page will be displayed, it is shown in fig 10.2. Home page contains the various buttons such as browse, encrypt, decrypt, reset, generate key, exit. The user can enter any file name they wish to use.

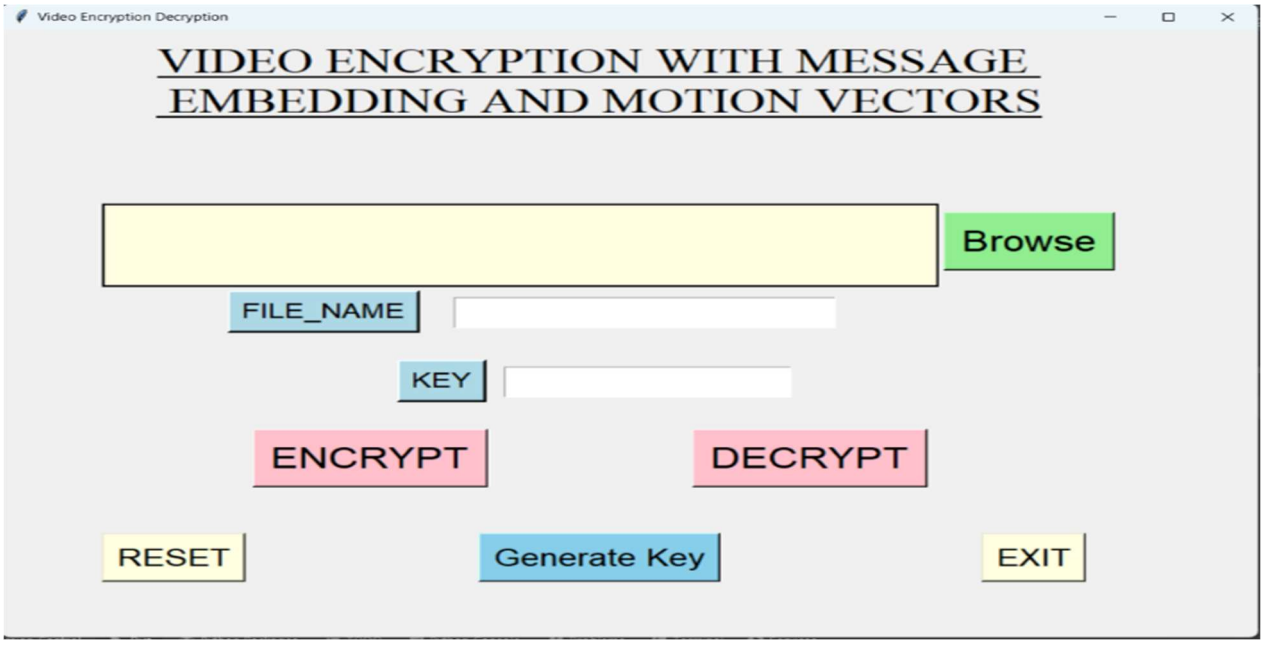


Fig 10.2 Home Page

Fig 10.3 illustrates the browse video feature, where the user can select any format of video from their local machine. The path of the selected video will be displayed in the browse bar.

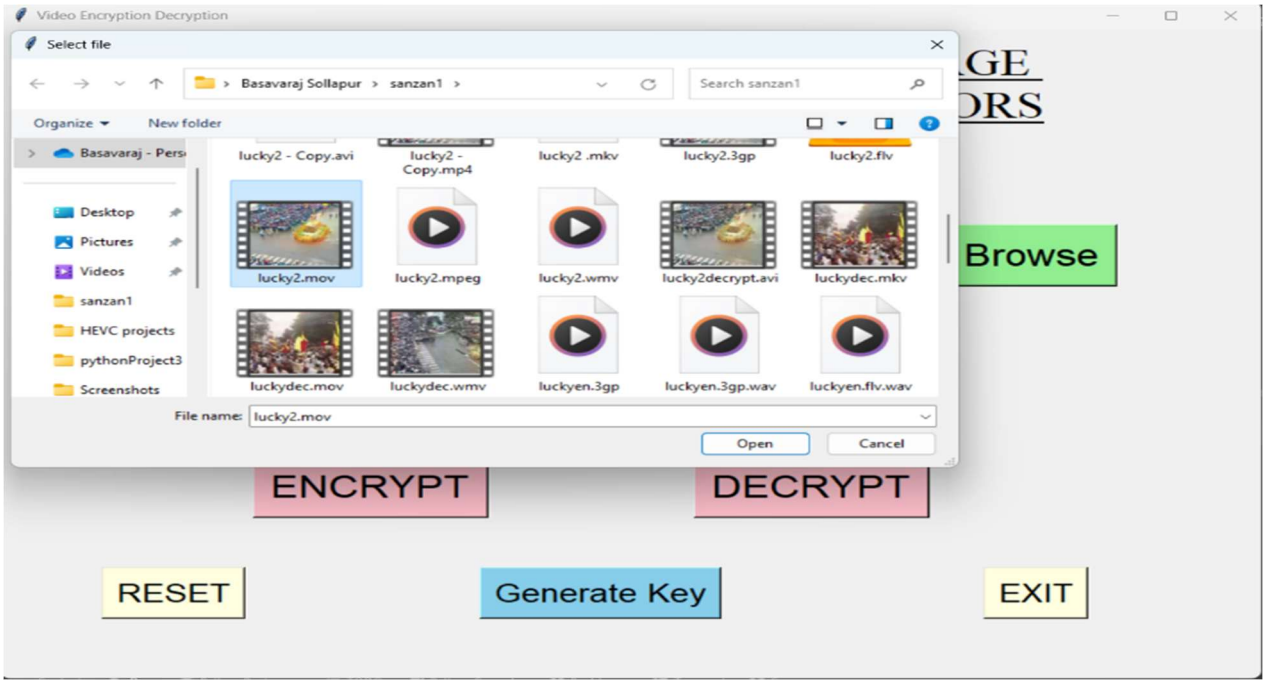


Fig 10.3 Browse Video

Once the video has been selected for browsing, press the generate key button to obtain a randomly generated AES key. This key will be used for both video encryption and decryption purposes. It will shown in fig 10.4.

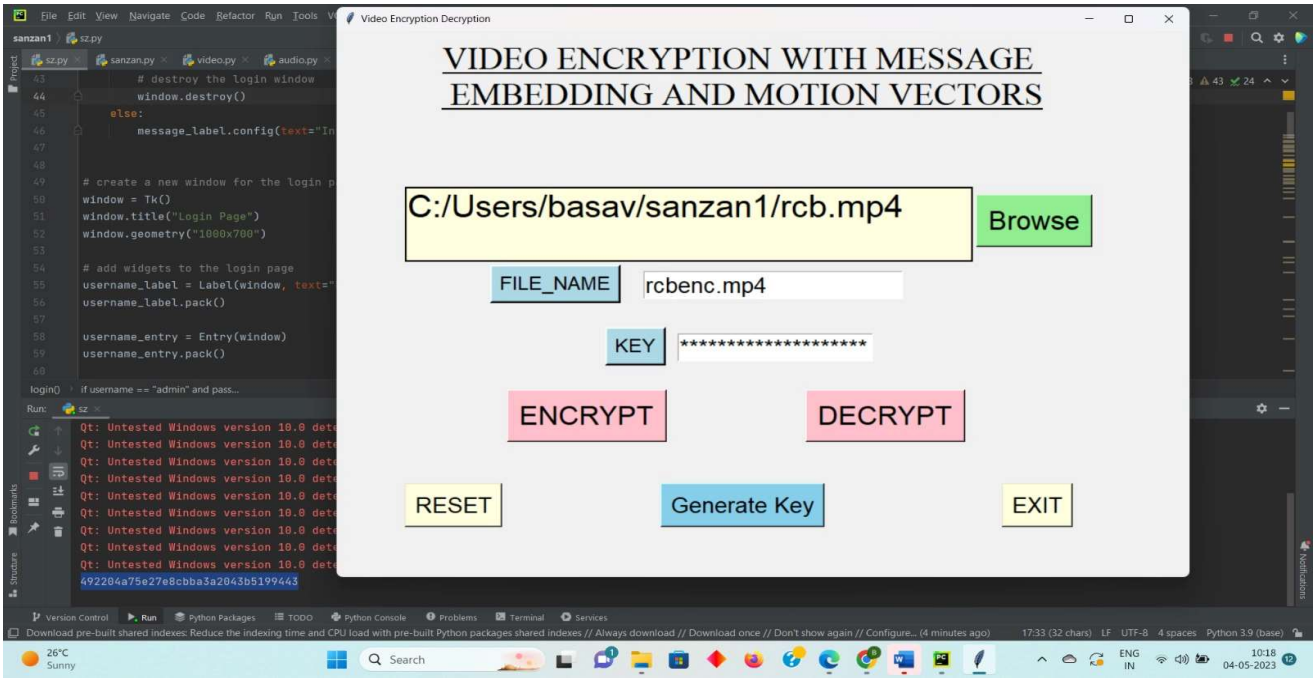
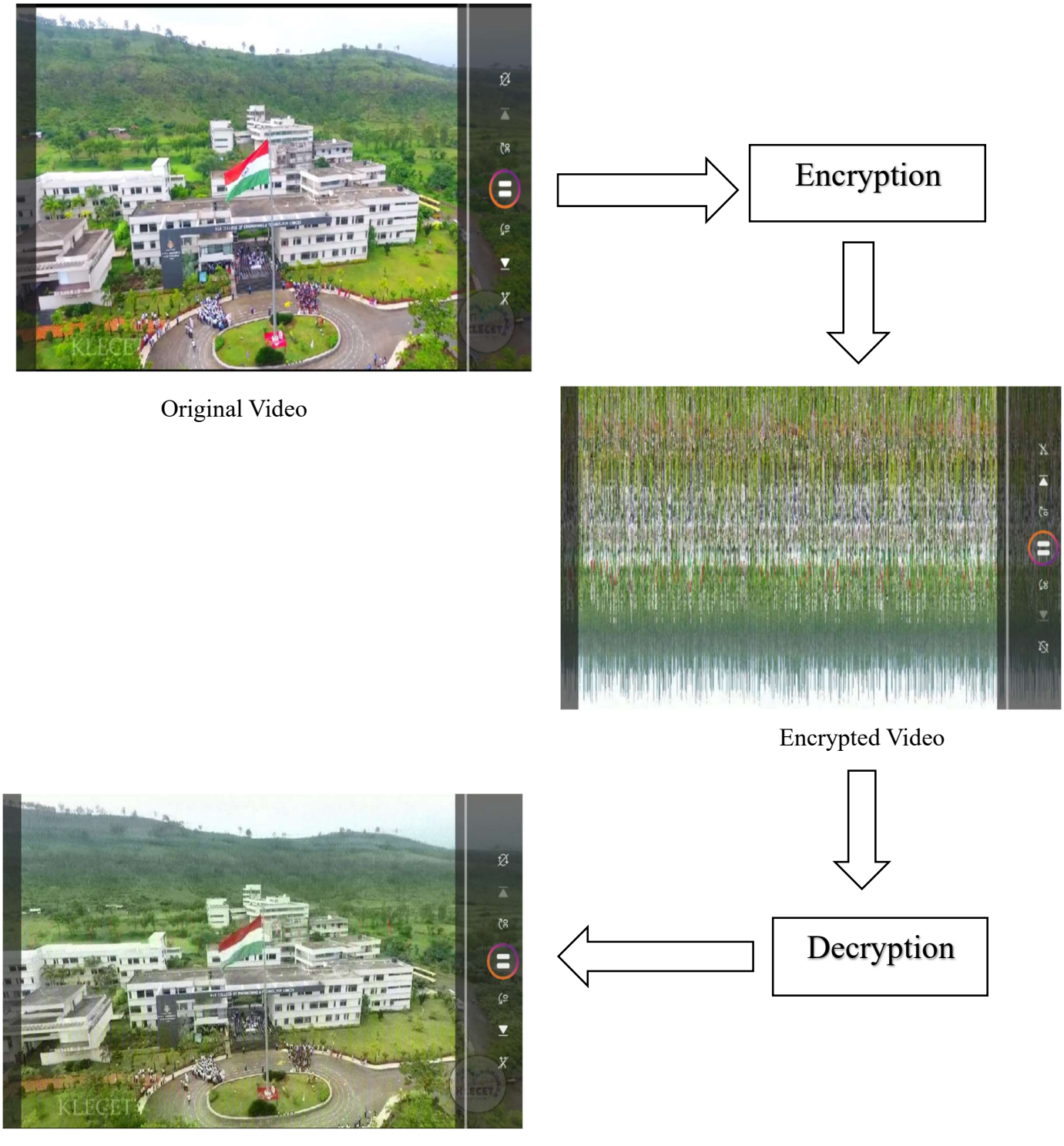


Fig 10.4 Key Generation

In this proposed system, the first step is to browse the video, followed by importing it into the encryption algorithm, which will then generate an encrypted video in the form of scrambled. To decrypt the video, the system imports data from the encrypted video and uses the decryption algorithm to generate the decrypted video as the final output.



Original Video

Encrypted Video

Decrypted Video

Fig 10.5 Video Encryption and Decryption

10.1 Parameters to be considered

10.1.1 File size and Time taken for encryption and decryption

Table 10.1 shows the file size and time taken for encryption and decryption of different types of video formats such as mp4, mkv, mov, avi, wmv etc. Depending on the video extension, the proposed system will produce varying encrypted file sizes, decrypted file sizes, encryption time, and decryption time.

Video Extensions	Original Video size	Encrypted Video size	Decrypted Video Size	Encrypted Time taken	Decrypted Time taken
MP4	7.26 MB	24.5 MB	14.2 MB	17.25Sec	4.25sec
MKV	7.26 MB	24.1 MB	14.2 MB	12.87sec	4.15sec
MOV	7.26 MB	24.3 MB	14.3 MB	12.66sec	0.13sec
AVI	7.26 MB	9.5 MB	3.8 MB	9.35Sec	4.15sec
WMV	7.26 MB	9.7 MB	3.7 MB	9.82sec	4.15sec

Table 10.1 File size and Time taken for encryption and decryption

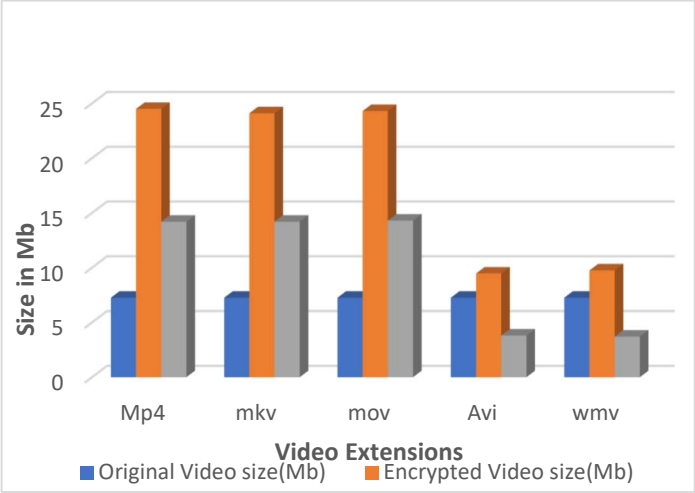


Fig 10.6 Video Extensions v/s Size

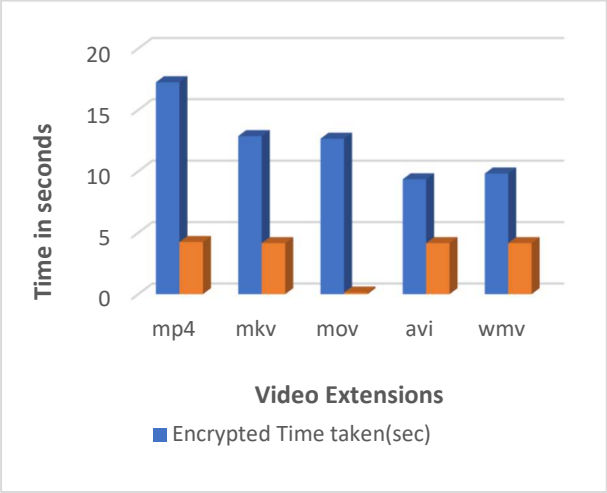


Fig 10.7 Video Extensions v/s Time

10.1.2 PSNR (Peak Signal-to-Noise Ratio)

PSNR values typically range from 0 to 100 decibels (dB), where higher values indicate higher quality and lower values indicate lower quality. In practice, typical PSNR values for video or image compression and reconstruction are usually between 20 dB and 50 dB, depending on the level of compression and the specific application.

Here are some general guidelines for interpreting PSNR values:

- PSNR values between 20 dB and 30 dB indicate a low-quality reconstruction or compression with significant distortion or loss of information.
- PSNR values between 30 dB and 40 dB indicate a moderate-quality reconstruction or compression with some perceptible distortion or loss of information.
- PSNR values between 40 dB and 50 dB indicate a high-quality reconstruction or compression with minimal distortion or loss of information.
- PSNR values above 50 dB indicate an almost perfect reconstruction or compression with no perceptible distortion or loss of information.
- The PSNR (Peak Signal-to-Noise Ratio) formula is used to measure the difference between an original signal and a compressed or degraded version of that signal. It is commonly used in image and video compression to evaluate the quality of the compressed output.

The formula for PSNR is as follows:

PSNR = 10 * log10((MAX^2) / MSE)

Where:

- MAX is the maximum possible pixel value of the image or video frame (e.g. 255 for an 8-bit grayscale image)
- MSE is the mean squared error between the original and compressed versions of the image or video frame, calculated as the average of the squared pixel-by-pixel differences between the two versions.
- PSNR is typically reported in units of decibels (dB), which is a logarithmic scale that represents the ratio between the maximum signal power and the power of the noise or distortion in the signal.
- Higher PSNR values indicate less distortion and better quality. A PSNR value of 30 dB or higher is generally considered good quality, while values below 20 dB are considered poor quality.

Video Extensions	PSNR
MP4	30.04db
MKV	30.01db
MOV	30.02db
AVI	29.31db
WMV	28.62db

Table 10.2 PSNR Values

10.1.3 Bitrate

- Bitrate refers to the amount of data transmitted per unit of time and is typically measured in bits per second (bps). In video encryption, the bitrate determines the level of compression applied to the video file, which in turn affects the quality and size of the encrypted video.
- When encrypting a video with message embedding and motion vectors, the bitrate is used to determine the level of compression that is applied to the video. Higher bitrates generally result in larger video files with better quality, while lower bitrates result in smaller video files with lower quality.
- To ensure that the encrypted video file has a good balance between quality and size, the bitrate needs to be carefully selected based on factors such as the desired level of compression, the resolution of the video, and the available bandwidth for transmitting the video.
- Additionally, the use of motion vectors in video encryption can help to reduce the bitrate by only encoding changes in the video frames rather than re-encoding the entire frame. This can result in significant savings in terms of file size and transmission bandwidth, without compromising the quality of the encrypted video.
- Bitrate refers to the amount of data that is processed per second of playback. It is typically measured in bits per second (bps), kilobits per second (Kbps), or megabits per second (Mbps).
- The formula for bitrate is as follows:

  Bitrate = (file size in bits) / (duration in seconds)
- We first convert the size of the file from megabytes to bits, by multiplying it by 8 (since there are 8 bits in a byte). Then, we divide the size in bits by the duration in seconds to get the bitrate in bits per second.
- For example, if you have a video file that is 7.26 MB in size and has a duration of 29 seconds, you can calculate the bitrate as follows:

  Bitrate = (7.25 MB * 8 bits/byte) / 29 seconds

  Bitrate = 2 Mbps

CONCLUSION

The proposed work is a message embedding solution in encrypted videos. An encrypted video remained compliant with standardized decoders. The message embedding is achieved by altering the values of reference picture indices and motion vectors of Control Units. When altering the values of motion vectors and reference indices, it was shown that a maximum of six message bits can be embedded per CU. Motion vectors were altered by swapping their Vx and Vy components and/or changing their signs. Message bits were extracted using an authorized decoder by generating a classification model and using it for predicting the true values of the reference indices and motion vectors. Consequently, message bits were extracted correctly whilst correctly reconstructing the video to its unscrambled state. Coding units that resulted in misclassification were identified at the encoder and excluded from message embedding. Such an arrangement resulted in lower embedding rates and ensured accurate reconstruction of the video.

REFERENCES

[1] T. Stutz and A. Uhl, ‘‘A survey of H.264 AVC/SVC encryption,’’ IEEE Trans. Circuits Syst. Video Technol., vol. 22, no. 3, pp. 325–339, Mar. 2012.

[2] Y. Tew, K. Wong, R. C.-W. Phan, and K. N. Ngan, ‘‘Separable authentication in encrypted HEVC video,’’ Multimedia Tools Appl., vol. 77, no. 18, pp. 24165–24184, Sep. 2018.

[3] M. Farajallah, W. Hamidouche, O. Déforges, and S. E. Assad, ‘‘ROI encryption for the HEVC coded video contents,’’ in Proc. IEEE Int. Conf. Image Process. (ICIP), Sep. 2015, pp. 3096–3100.

[4] M. A. Taha, N. Sidaty, W. Hamidouche, O. Dforges, J. Vanne, and M. Viitanen, ‘‘End-to-end real-time ROI-based encryption in HEVC videos,’’ in Proc. 26th Eur. Signal Process. Conf. (EUSIPCO), Sep. 2018, pp. 171–175.

[5] P.-C. Su, T.-F. Tsai, and Y.-C. Chien, ‘‘Partial frame content scrambling in H.264/AVC by information hiding,’’ Multimedia Tools Appl., vol. 76, no. 5, pp. 7473–7496, Mar. 2017.

[6] D. Xu and S. Su, ‘‘Reversible data hiding in encrypted images with separability and high embedding capacity,’’ Signal Process., Image Commun., vol. 95, Jul. 2021, Art. no. 116274.

[7] Advanced Encryption Standard (AES), Federal Information Processing Standard 197, NIST, Nov. 2001

[8] Y. Wang, M. O’Neill, and F. Kurugollu, ‘‘A tunable encryption scheme and analysis of fast selective encryption for CAVLC and CABAC in H.264/AVC,’’ IEEE Trans. Circuits Syst. Video Technol., vol. 23, no. 9, pp. 1476–1490, Sep. 2013.

[9] B. Boyadjis, C. Bergeron, B. Pesquet-Popescu, and F. Dufaux, ‘‘Extended selective encryption of H.264/AVC (CABAC)- and HEVC-encoded video streams,’’ IEEE Trans. Circuits Syst. Video Technol., vol. 27, no. 4, pp. 892–906, Apr. 2017.

[10] Y. Wang, M. O’Neill, and F. Kurugollu, ‘‘The improved sign bit encryption of motion vectors for H.264/AVC,’’ in Proc. 20th Eur. Signal Process. Conf. (EUSIPCO), Romania, 2012, pp. 1752–1756.

[11] Y. Yao, W. Zhang, and N. Yu, ‘‘Inter-frame distortion drift analysis for reversible data hiding in encrypted H.264/AVC video bitstreams,’’ Signal Process., vol. 128, pp. 531–545, Nov. 2016.

[12] M. Long, F. Peng, and H.-Y. Li, ‘‘Separable reversible data hiding and encryption for HEVC video,’’ J. Real-Time Image Process., vol. 14, no. 1, pp. 171–182, Jan. 2018.

[13] D. Xu, R. Wang, and Y. Zhu, ‘‘Tunable data hiding in partially encrypted H.264/AVC videos,’’ J. Vis. Commun. Image Represent., vol. 45, pp. 34–45, May 2017.

[14] D. Xu, R. Wang, and Y. Q. Shi, ‘‘An improved scheme for data hiding in encrypted H.264/AVC videos,’’ J. Vis. Commun. Image Represent., vol. 36, pp. 229–242, Apr. 2016.

[15] Z. Shahid and W. Puech, ‘‘Visual protection of HEVC video by selective encryption of CABAC binstrings,’’ IEEE Trans. Multimedia, vol. 16, no. 1, pp. 24–36, Jan. 2014.

[16] D. Xu, ‘‘Commutative encryption and data hiding in HEVC video compression,’’ IEEE Access, vol. 7, pp. 66028–66041, 2019.

[17] B. Guan, D. Xu, and Q. Li, ‘‘An efficient commutative encryption and data hiding scheme for HEVC video,’’ IEEE Access, vol. 8, pp. 60232–60245, 2020.

[18] Tejaswini Sawant , Sagar Soni, Dhwani Tank, Dr. Dipti Patil, “Videozen: Video Encryption and Decryption using AES and Blowfish” © April 2021| IJIRT | Volume 7 Issue 11 | ISSN: 2349-600.

[19] E. Peixoto, T. Shanableh, and E. Izquierdo, ‘‘H.264/AVC to HEVC video transcoder based on dynamic thresholding and content modeling,’’ IEEE Trans. Circuits Syst. Video Technol., vol. 24, no. 1, pp. 99–112, Jan. 2014.

[20] T. Amestoy, A. Mercat, W. Hamidouche, D. Menard, and C. Bergeron, ‘‘Tunable VVC frame partitioning based on lightweight machine learning,’’ IEEE Trans. Image Process., vol. 29, pp. 1313–1328, 2020.

[21] I.-K. Kim, K. D. McCann, K. Sugimoto, B. Bross, W.-J. Han, and G. J. Sullivan, “High Efficiency Video Coding (HEVC) Test Model 13 (HM13) Encoder Description,” document JCT-VC O1002, 15th Meeting Joint Collaborative Team Video Coding ITU-T SG 16 WP 3 ISO/IEC JTC 1, Nov. 2013.

[22] G. Bjentegaard, “Calculation of Average PSNR Differences Between RDCurves (VCEG-M33),” document ITU-T SG16 Q.6, VCEG Meeting, Apr. 2001.

[23] D. Xu, ‘‘Data hiding in partially encrypted HEVC video,’’ ETRI J., vol. 42, no. 3, pp. 446–458, Mar. 2020.

[24] I. Agi and L. Gong. “An empirical study of MPEG video transmission”. In Proc. of the Internet Society Symposium on Network and Distributed Systems Security, pages 137–144,1996.

[25] A. Biryukov and E. Kushilevitz. From differential cryptanalysis to ciphertext-only attacks. Lecture Notes in Computer Science, Advances in Cryptology – Proceedings of CRYPTO’98, 1462:72–88, 1998.

[26] A. Biryukov and E. Kushilevitz. Improved cryptanalysis of RC5. Lecture Notes in Computer Science, 1403:85–100,1998.

[27] Z. Chen, Z. Xiong, and L. Tang. A novel scrambling scheme for digital video encryption. In Proc. of Pacific-Rim Symposium on Image and Video Technology (PSIVT), pages 997–1006, 2006.

[28] E. Choo, L. Jehyun, L. Heejo, and N. Giwon. SRMT: A lightweight encryption scheme for secure real-time multimedia transmission. In Multimedia and Ubiquitous Engineering, pages 60–65, 2007.

[29] L. S. Choon, A. Samsudin, and R. Budiarto. Lightweight and cost-effective MPEG video encryption. In Proc. of Information and Communication Technologies: From Theory to Applications, pages 525–526, 2004.

[30] T. B. Maples and G. A. Spanos. Performance Study of a Selective Encryption Scheme for the Security of Networked, Real-Time Video. In Proc. of Fourth International Workshop on Multimedia Software Development ’96), 1995.

[31] L. Qiao and K. Nahrstedt. A new algorithm for MPEG video encryption. In Proc. of First International Conference on Imaging Science System and Technology, pages 21–29,1997.

[32] R. L. Rivest. The RC5 encryption algorithm. In Proc. of the Second International Workshop on Fast Software Encryption (FSE), pages 86–96, 1994.

[33] B. Schneier. “Applied Cryptography Second Edition Protocols Algorithms and Source Codes” in C, 1996.

[34] L. Tang. Methods for encrypting and decrypting MPEG video data efficiently. In Proc. of ACM Multimedia, pages 219–229, 1996.

[35] W. Tsang, B. Smith, S. Mukhopadhyay, H. H. Chan, S. Weiss, M. Chiu, and J. Song. The DALI multimedia software library. In IEEE 2nd Workshop on Multimedia Signal Processing, 1999.

[36] P. N. Tudor. MPEG-2 video compression. In Electronics and Communication Engineering Journal, December - 1995.

[37] T. Uehara and R. Safavi-Naini. Recovering DC coefficients in block-based DCT. In Proc. of IEEE Transactions on Image Processing, volume II, pages 3592–3596, 2006.

[38] W. Zeng and S. Lei. Efficient frequency domain selective scrambling of digital video. In Proc. of the IEEE Transactions on Multimedia, pages 118–129, 2002.

[39] Ian Sommerville: “Software Engineering”, 9th Edition, Pearson Education, 2012.

[40] Michael Blaha, James Rumbaugh: “Object Oriented Modelling and Design with UML,”2nd Edition, Pearson Education, 2019.