**Report**

# Video Steganography (Encryption and Decryption)


## Abstract

In this report, video steganography with frame number as a key, is to be achieved, along with encryption of the data to prevent basic brute force attacks on the hidden data. Video steganography is becoming an important research area in various data hiding technologies, becoming a promising tool because not only the security requirement of secret message transmission is becoming stricter, since videos are bigger in format size, it is more favourable to transfer data through videos.

This paper aims to create a program which takes in two things: A Text Data, Image File and embeds the data in the image file, so that it does minimal change in the image file's data. And, extracts data from the Embedded file. This will be done using an encryption and embedding algorithm. It is easier to decode the security system in an image embedded system, Visual quality might be affected, In Sound steganography waveforms can be easily detected.

In video steganography these issues are solved because Video files are large enough to contain sufficient data, Video files can have embedded data at any frame among thousands of frame and Any distortion in a single frame can be completely hidden because of high FPS



Keywords: Video Steganography, Encryption, Data hiding


## I. Introduction

Steganography is a method of sharing secret information by making it inconspicuous to non authenticated users. Steganography has been originated from Greek word Steganos and graphics .Digital Steganography provides capability to protect private communication that has become necessity in today's Internet era. Generally sharing of information takes place in the form of text, image, audio and video. Steganography uses image, text, video and audio to disguise secret information. In video steganography secret information is enveloped inside a video to make it safe from intruders.

In recent times, Video steganography is becoming an important research area in various data hiding technologies, which has become a promising tool because not only the security requirement of secret message transmission is becoming stricter but also video is more favored.

We will be making a program to encode data to a video using steganography, and decode data from a video which has encoded data hidden in it. Data is encoded in a video file, and is unseen when the video is played. The project is to be made in Python.

**This paper is organized as follows: Current section gives introduction about importance of steganography in video files, and data hiding Section II shows the related work. Section III describes about the traditional image steganography systems and their limitations. Section IV elaborates the algorithm and workflow for the proposed video steganography system. Finally concludes the paper.**

## II. Related Works

Prof. Dr. P. R. Deshmukh,Bhagyashri Rahangdale said that this technique of hiding a particular text file in the video hide it securely with minimum mean square error and hence gives maximum PSNR i.e peak signal to noise ratio.So it help to transmit data securely by embedding it in a video file and without disclosing to the unintended receiver and without any alternation in secret message.

Rajesh Kumar, A.J. Singh In this research paper we talk about the technique crossbreeding algorithm used with steganography so that we can get more security to send the files even over the open networks.Steganography is more secure medium than cryptography. In steganography the transmitting medium is any media file such as Image, Audio, Video or animation so this is very often that nobody is bothered about the message behind the media but in case of cryptography data is in encrypted form so the hackers are more aware of it.

Vipula Madhukar Wajgade, Dr. Suresh Kumar said that Steganography, Cryptography and Digital Watermarking techniques can be used to obtain security and privacy of data.The separation of video into audio and images or frames results in the efficient method for data hiding.The use of video files as a carrier medium for steganography is more eligible as compared to other techniques.

Kousik Dasgupta1, J.K. Mandal and Paramartha Dutta proposed Hash based Least Significant Bit (HLSB) technique for Video Steganography.This technique utilizes cover video files in spatial domain to conceal the presence of sensitive data regardless of its format. Performance analysis of the proposed technique after comparison with LSB technique is quite encouraging.

Ramadhan J. Mstafa &Khaled M. Elleithy gave a comprehensive review and analysis of video steganography methods in both compressed and raw domains.A large number of methods describing the embedding and encoding process were discussed and compared. Steganography was also compared with other methods of data hiding, such as cryptography and watermarking

## III. Traditional Steganography System and its Limitation

Image Steganography is the process of hiding information which can be text, image or video inside a cover image. The secret information is hidden in a way that it not visible to the human eyes.

The idea behind image-based Steganography is very simple. Images are composed of digital data (pixels), which describes what's inside the picture, usually the colors of all the pixels. Since we know every image is made up of pixels and every pixel contains 3-values (red, green, blue).

Every byte of data is converted to its 8-bit binary code using ASCII values. Now pixels are read from left to right in a group of 3 containing a total of 9 values. The first 8-values are used to store binary data. The value is made odd if 1 occurs and even if 0 occurs.

To decode, three pixels are read at a time, till the last value is odd, which means the message is over. Every 3-pixels contain a binary data, which can be extracted by the same encoding logic. If the value if odd the binary bit is 1 else 0.

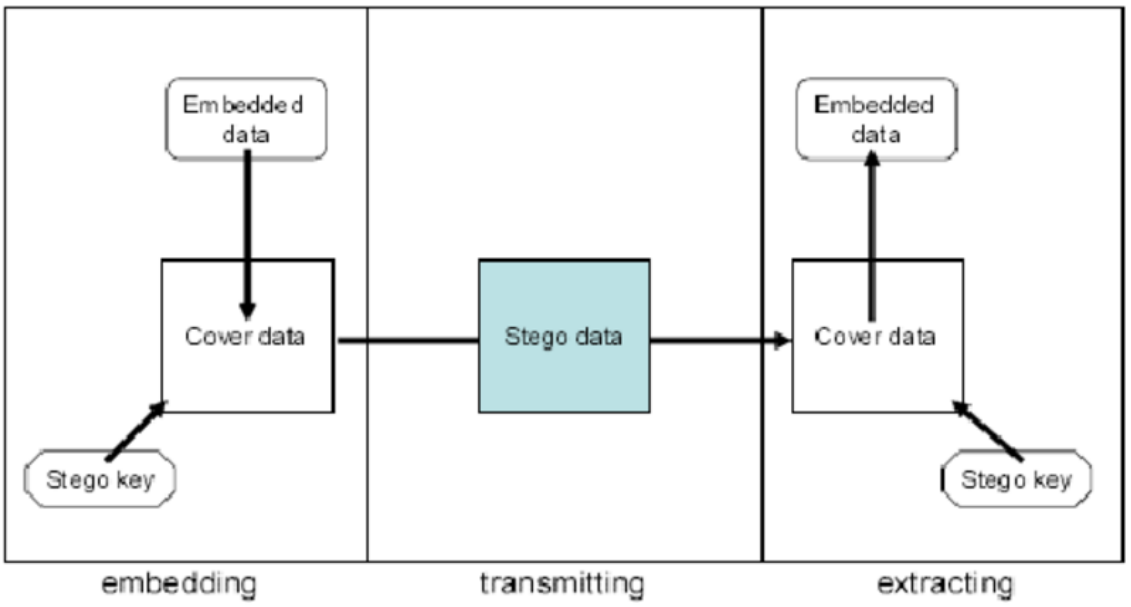



**Fig. 1 : Block diagram of Traditional Steganography System**

### Limitations

Though the steganography is able to hide messages in cover messages, there are at least two problems in the framework of steganography. First, to hide an image using existing

steganographic approaches is very difficult unless the size of the secret image is much smaller than that of the cover image. For example, a 1024×768 color image, with 3 bytes per pixel, has the potential to hide 294,912 bytes of information, if 3 bits are used for each pixel. In this case, the size of the secret image should be smaller than or equal to 1/8 of the size of the cover image. Consequently, in the framework of steganography, it is a challenge to embed a secret image into a cover image when both images of same size. Another problem in existing steganographic approaches is that partial hiding of messages is not allowed. However, there are cases in which partial hiding is required.

A scenario might be doctor's co-examination on medical images. In this case, one doctor may hide the patient's personal information (e.g., the patient's face image) while keeping the sickness information (e.g., face color) "readable" to other doctors. In the conventional steganographic approaches, the image data of the patient must be hidden completely in the cover data, and be recovered completely when the recipients want to see the data.

## IV. Proposed Video Steganography System

Steganography and cryptography are two different data hiding techniques. Cryptography, obscures the content of the message.

By using steganography secure data transmission over internet could be achieved. Video files are generally a batch of still images. So, most of the techniques used for mages and audio can also be applied in video steganography. The main advantage of video stream is amount of data they carry that cannot be easily detectable. Above all, it is a moving sequence of still pictures. With the growth of data communication over computer network, the security of information has become a major issue.

Our system reads a video file, and the data to hide. Performs multiple encoding algorithms on the text. After that, the key is taken as an input, which is the frame number where the data is to be embedded. This data is embedded in the frame and the frames are combined together to form a final video file. While decoding, the key entered is important, as if it is entered wrong, the data can only be decoded if the key entered is correct. The data is then decoded by performing the decoding algorithms, and then finally, the data is decoded, and shown on screen.

**Algorithm:**

Encoding:

Step 1: Enter video file name

Step 2: Check if the video file exists and is a valid file

Step 3: Query if the video has audio embedded

Step 4: Read the input for the data to be embedded in the video

Step 5: Encode the data using BASE64 encoding

Step 6: Encrypt the data using ROT13 encryption

Step 7: Separate the frames from the video file, and save temporarily

Step 8: If the video has audio, the audio is saved temporarily

Step 9: Read frame number from user, which acts as key.

Step 10: Encode the Encoded & encrypted data in the extracted frames.

Step 11: Combine the frames together to make the encoded video.

Decoding:

Step 1: Read video file

Step 2: check validity of video file. If invalid, abort.

Step 3: Read the key (Frame number) from the user

Step 4: If key frame is invalid, abort

Step 5: Decode the Text from the frame

Step 6: Decrypt the data using ROT13 decryption

Step 7: Decode the data using BASE64 decoding

Step 8: Show the decoded data

**Block Diagram:**

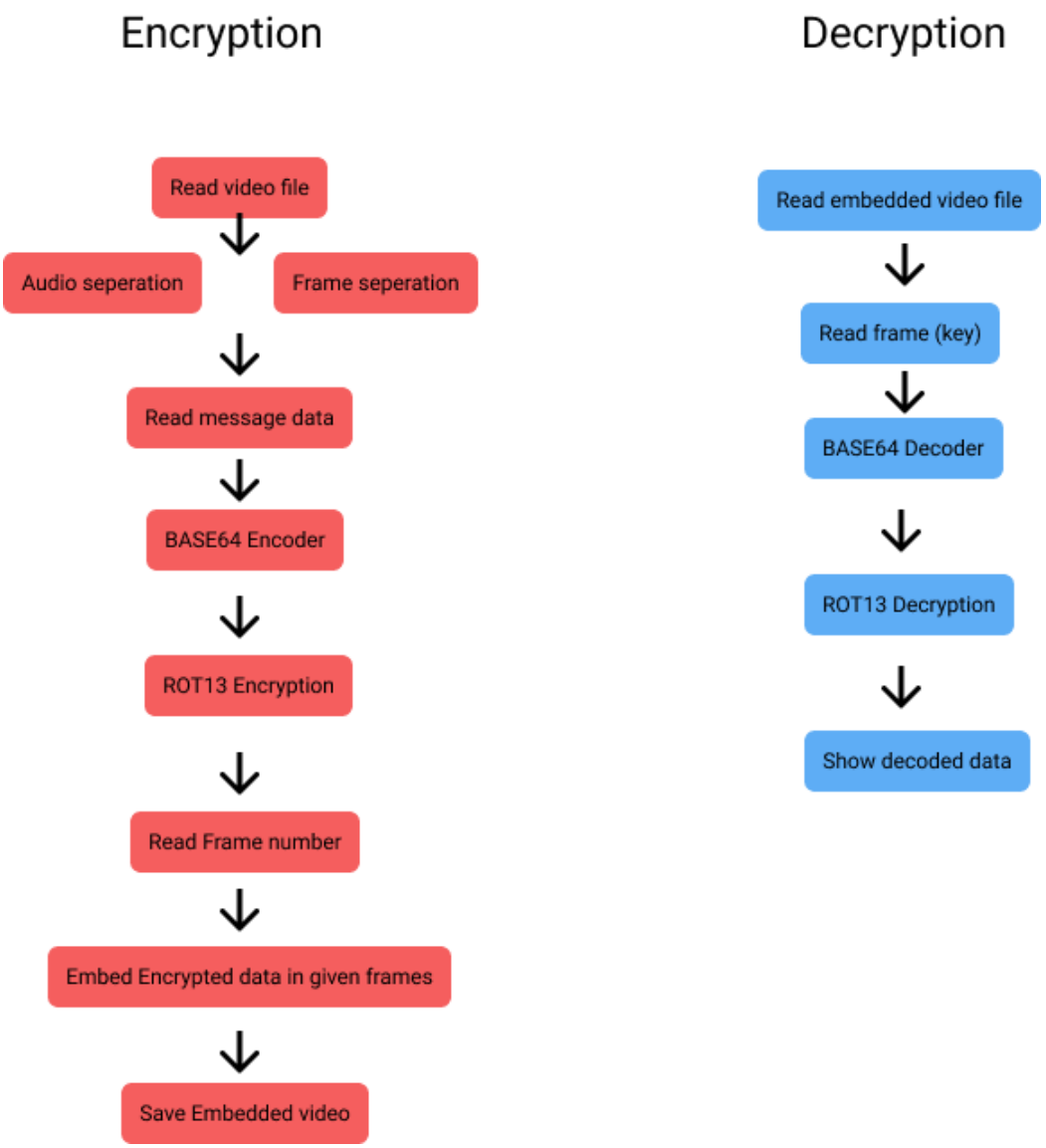



**Fig. 2 : Block diagram of Proposed Steganography System**

## Components of the project:

1.main.py: Main python file, which is to be run to perform both encryption and decryption.

2.Video File: The clean video file in which the data is to be embedded

3.Input Data: Any kind of data, which will go through two sets of encryptions and be embedded in a certain frame, as indicated.

**Software and Libraries used:**

1. **opencv**

OpenCV is a huge open-source library for computer vision, machine learning, and image processing. OpenCV supports a wide variety of programming languages like Python, C++, Java, etc. It can process images and videos to identify objects, faces, or even the handwriting of a person.

2. **os**

The OS module in Python provides functions for creating and removing a directory (folder), fetching its contents, changing and identifying the current directory, etc.

3. **subprocess**

A subprocess in Python is a task that a python script delegates to the Operative system (OS). The subprocess library allows us to execute and manage subprocesses directly from Python.

4. **glob**

The glob module is a useful part of the Python standard library. glob (short for global) is used to return all file paths that match a specific pattern.

5. **PIL**

The Python Imaging Library adds image processing capabilities to your Python interpreter. This library provides extensive file format support, an efficient internal representation, and fairly powerful image processing capabilities.

6. **stepic**

Stepic is a Python module and command line tool for hiding arbitrary data within images by slightly modifying the colors

7. **base64**

Python's Base64 module provides functions to encode binary data to Base64 encoded format and decode such encodings back to binary data.

8. **python**

Python is used for running the code.

9. **VSCode**

Text Editor and IDE

**Advantages:**

1. Being able to hide data in a frame in a video file makes it difficult to find the frame where the data is embedded.
2. Additional BASE64 ROT13 encodings make the data more hidden, and is scalable by using RSA, SHA or more advanced encryption standards.
3. System is quick for small video files, and is difficult to find the frames where the data is hidden.
4. It is better than image steganography since the frames are high.
5. Embedded video is indistinguishable from original video.

**Code:**

main.py > ...

```python

# Importing all the required modules
import cv2              # Module for video processing
import os               # Module for executing system commands
import subprocess       # Module for executing system commands
import glob             # Module to collect specific files recursively
from PIL import Image   # Module for image processing
import stepic           # Module for image steganography
import base64 as b64    # Module for base64 data processing

# ANSI codes for font colors
class colors:
    GREEN = '\033[92m'  #GREEN COLOR
    YELLOW = '\033[93m' #YELLOW COLOR
    RED = '\033[91m'    #RED COLOR
    RESET = '\033[0m'   #RESET COLOR

# Function that implements the rot13 cipher
def rot13(input_text):
    return input_text.translate(str.maketrans("ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz","NOPQRSTUVWXYZABCDEFGHIJKLMnopq

# Function that implements video steganography
def steg_video(video_name, audio):
    # Prompting user input for the secret message
    message = input(f"Enter the message you want to encode in the video:\n{colors.YELLOW}")
    print(f"{colors.RESET}")

    # Encoding the secret message into base64 format and encrypting it using rot13 cipher
    message = b64.b64encode(message.encode())
    print(f"Base64 encoded Message: {colors.YELLOW}{message}{colors.RESET}")
    message = rot13(message.decode())
    print(f"ROT13 Encrypted Message: {colors.YELLOW}{message}{colors.RESET}")
```

```
    # Creating a video file object from the selected video file
    video = cv2.VideoCapture(video_name)

    # Creating ./temp directory to store the frames and audio extracted
    no_of_frames = 1
    os.system('mkdir temp && mkdir ./temp/frames/')
    frames_folder = './temp/frames/'

    # Extracting audio from the video
    if(audio == 'y'):
        subprocess.call(["ffmpeg","-i",video_name,"-q:a","0","-map","a","./temp/audio.mp3","-y"],stdout=open(os.devnull,"w"),stderr=

    # Dividing the video into frames
    while True:
        success, image1 = video.read()
        if not success:
            break
        cv2.imwrite(os.path.join(frames_folder, f"{no_of_frames}.png"),image1)
        no_of_frames += 1

    # Collecting all the images in the ./temp/frames/ directory into a list
    frames = glob.glob('./temp/frames/*.png')

    # Sorting the image file names according to the creation time
    frames.sort(key=os.path.getmtime)

    # Prompting user input for the frame no. in which the message has to be embedded
    key_frame_no = input(f"\nEnter the frame no.(between 1-{len(frames)}) where you want to embed the message: ")
    key_frame_no = int(key_frame_no)
    if(key_frame_no > len(frames)):
        print(f"\n{colors.RED}Frame no. {key_frame_no} doesn't exist! There are only {len(frames)} frames!{colors.RESET}\n")
        os.system('rm -rf ./temp/')
        exit()
```

```
    key_frame_name = frames[key_frame_no-1] # Getting the image file name at the specified position
    key_frame = Image.open(frames[key_frame_no-1]) # Creating an image file object of the selected image
    encoded_frame = stepic.encode(key_frame, message.encode()) # Creating a file with the embedded secret message

    os.remove(key_frame_name) # Deleting the already existing frame or image with the same name

    encoded_frame.save(key_frame_name) # Saving the new frame of image file with the same name

    # Combining all the images inside the ./temp/frames/ directory to create a new video
    frames_array = []
    for frame in frames:
        image2 = cv2.imread(frame)
        height, width, layers = image2.shape
        size = (width, height)
        frames_array.append(image2)

    temp_video_name = "./temp_" + video_name.split(".")[0] + ".avi"
    new_video = cv2.VideoWriter(temp_video_name, cv2.VideoWriter_fourcc(*'DIVX'), 30, size)

    for frame in range(no_of_frames-1):
        new_video.write(frames_array[frame])
    new_video.release()

    new_video_name = "./new_" + video_name.split(".")[0] + ".avi"

    print(f"\n{colors.GREEN}Video created!{colors.RESET}\n")
```

```
    # Prompting the user for extracting the secret message
    choice = input("Do you want to decode the message in the video? (y/n): ")
    if(choice == 'y'):
        # Prompting the user to enter the same frame no. where the message is embedded
        key_frame_no2 = input("Enter the key frame no.(where the image is hidden): ")
        key_frame_no2 = int(key_frame_no2)

        if(key_frame_no == key_frame_no2):
            # Extracting and decoding the secret message to its original form
            decoded_message = stepic.decode(encoded_frame)
            print(f"\nExtracted Message: {colors.YELLOW}{decoded_message}{colors.RESET}")

            decoded_message = rot13(decoded_message)
            print(f"ROT13 Decrypted Message: {colors.YELLOW}{decoded_message}{colors.RESET}")

            decoded_message = b64.b64decode(decoded_message)
            decoded_message = decoded_message.decode()
            print(f"Base64 Decoded Message: {colors.YELLOW}{decoded_message}{colors.RESET}")
        else:
            print(f"\n{colors.RED}Incorrect key entered!{colors.RESET}\n") # Notifying the user if the entered key or f

    if(audio == 'y'):
        subprocess.call(["ffmpeg","-i",temp_video_name,"-i","./temp/audio.mp3","-codec","copy",new_video_name,"-y"],std
        os.system('rm -rf ./temp/ && rm ' + temp_video_name)
    else:
        os.system('rm -rf ./temp/ && mv ' + temp_video_name + ' ' + new_video_name)
    print(f"\n{colors.GREEN}Thank you!{colors.RESET}\n")
```

```
# Function to check the file existence and file format
def check_file(video_name):
    flag = 0
    if((";" in video_name) or ("&&" in video_name)):
        print(f"\n{colors.RED}Invalid characters not allowed!{colors.RESET}\n")
    else:
        if(len(video_name.split(".")) == 2):
            cmd = "file " + video_name
            filetype = subprocess.check_output(cmd, shell=True)
            if("No such file or directory" in filetype.decode()[:-1]):
                print(f"{colors.RED}No video file exists with the name {video_name}!{colors.RESET}")
            elif(("MP4" in filetype.decode()[:-1]) or ("AVI" in filetype.decode()[:-1])):
                flag = 1
            else:
                print(f"{colors.RED}Invalid file format!{colors.RESET}")
        else:
            print(f"{colors.RED}No video file exists with the name {video_name}! (If exists then it should be in the present dir

    if(flag == 1):
        return True
    else:
        return False

# Introduction Banner
print("--------------------------------------------------------------------------")
print(f"------------------------ {colors.GREEN}VIDEO STEGANOGRAPHY{colors.RESET} ------------------------")
print("--------------------------------------------------------------------------\n")

video_name = input('Enter the name of the video file: ')

if(check_file(video_name)):
    audio = input('Does this video have an audio? (y/n): ')
    steg_video(video_name, audio)
```

**Outputs:**

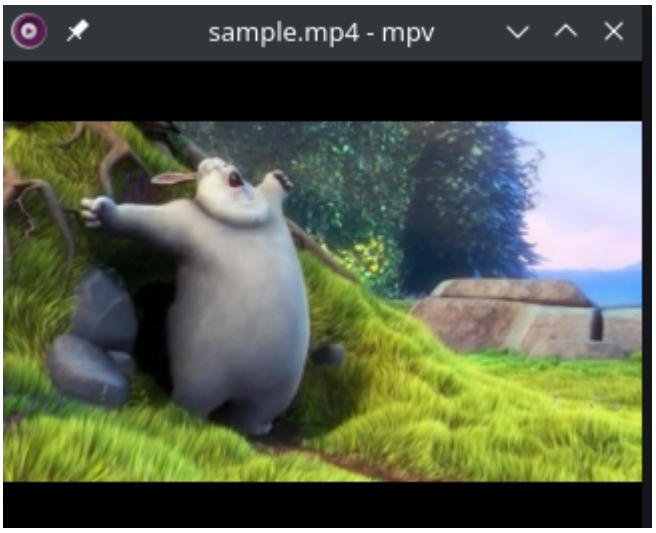
- ENCODING

```
[sour@sourlinux third]$ /bin/python /home/sour/Downloads/ISA/third/main.py
------------------------------------------------------------------------
------------------------ VIDEO STEGANOGRAPHY ---------------------------
------------------------------------------------------------------------

Enter the name of the video file: sample.mp4
Does this video have an audio? (y/n): y
Enter the message you want to encode in the video:
asdfgh

Base64 encoded Message: b'YXNkZmdo'
ROT13 Encrypted Message: LKAxMzqb

Enter the frame no.(between 1-205) where you want to embed the message: 10

Video created!
```

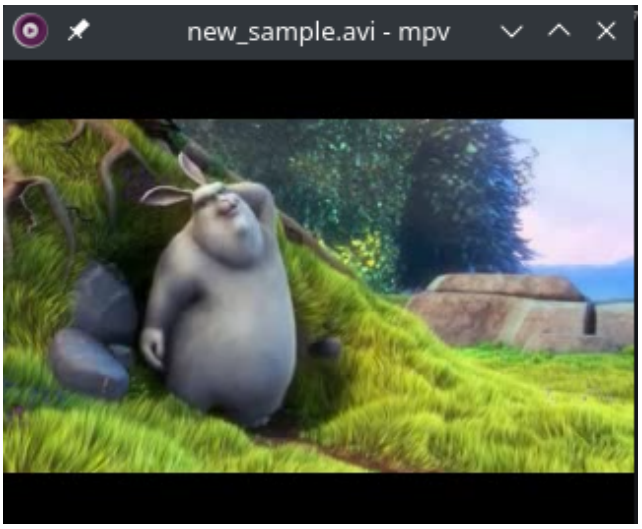
- DECODING

```
Do you want to decode the message in the video? (y/n): y
Enter the key frame no.(where the image is hidden): 10

Extracted Message: LKAxMzqb
ROT13 Decrypted Message: YXNkZmdo
Base64 Decoded Message: asdfgh

Thank you!
```

- OLD VIDEO



- DECODED VIDEO




**References**


[1] Deshmukh, P. R., and Bhagyashri Rahangdale. "Data Hiding using Video Steganography." International Journal of Engineering Research & Technology 3.4 (2014): 856-860.

[2] Kumar, Rajesh, and A. J. Singh. "Understanding steganography over cryptography and various steganography techniques." International Journal of Computer Science and Mobile Computing 4.3 (2015): 253-258.

[3] Wajgade, Vipula Madhukar, and Dr Suresh Kumar. "Enhancing data security using video steganography." International Journal of Emerging Technology and Advanced Engineering 3.4 (2013): 549-552.

[4] Deshmukh, P. R., and Bhagyashri Rahangdale. "Hash based least significant bit technique for video steganography." Int. Journal of Engineering Research and Applications 4.1 (2014): 44-49.

[5] Mstafa, Ramadhan J., and Khaled M. Elleithy. "Compressed and raw video steganography techniques: a comprehensive survey and analysis." Multimedia Tools and Applications 76.20 (2017): 21749-21786.

[6] Masoud Nosrati, Ronak Karimi, Mehdi Hariri, An introduction to steganography methods, World Applied Programming, Vol (1), No (3), August 2011. 191-195.

[7] Shawkat, Shihab Ahmed. Enhancing Steganography Techniques in Digital Images. Diss. Faculty of Computers and Information, Mansoura University Egypt-2016, 2007.

[8] Liu, Yunxia, et al. "Video steganography: A review." Neurocomputing 335 (2019): 238-250.

[9] Anderson, Ross J., and Fabien AP Petitcolas. "On the limits of steganography." IEEE Journal on selected areas in communications 16.4 (1998): 474-481.

[10] Johnson, Neil F., and Sushil Jajodia. "Exploring steganography: Seeing the unseen." Computer 31.2 (1998): 26-34.