Video Steganography: Storing Data in the Transition of Frames

Tejas Paranjape, Vineet Paradhi, Kewal Shah, and Varsha Hole

Abstract In today's fast-paced world, security has attained an extremely important place in the technological sphere. In the last couple of decades, a huge boom was seen in the number of techniques of steganography that were developed. Steganography is the science of concealing data in an image, an audio, or a video file, such that the data is not at all discernible to the human senses. This paper intends to delve into the relatively new technique of video steganography. New algorithms are explored to hide information not in the individual frames of the video, but in the transition of the frames. The least significant bit technique is used in our algorithm. This method will provide a more secure method as compared to the existing techniques, as it is a novel approach to hiding data in plain sight.

Keywords Video steganography · Transition · Least significant bit

1 Introduction

To protect our data from hackers, eavesdroppers, and other third parties that may be interested in the data that is sent and received across two nodes in a medium, the data needs to be hidden in some way or the other. Mostly, two major methods are implemented today, namely, cryptography, and steganography. Cryptography [1] is the method that enables the encryption of the data so that the data cannot be interpreted by any third party, even if it is intercepted by them because it is encrypted. It can only be read if the key that is needed to translate the encrypted data to its original form is known. That is also considered as a disadvantage of cryptography, that if the key used is disclosed, then the data can easily be recovered by the third party, and misused.

T. Paranjape (✉) · V. Paradhi · K. Shah · V. Hole
Department of Information Technology, Sardar Patel Institute of Technology, Mumbai, India
e-mail: varsha_hole@spit.ac.in

S. Smys et al. (eds.), *Computer Networks and Inventive Communication Technologies*, Lecture Notes on Data Engineering and Communications Technologies 58,
https://doi.org/10.1007/978-981-15-9647-6_63

On the other hand, steganography [2] focuses on hiding the data in plain sight, so that the interested third party does not know that data is being sent in the first place. The methods of implementing steganography are image, audio, and video steganography. Image steganography is an area that was developed in the last decade and has been extensively researched upon. On the other hand, audio and video steganography are relatively newer domains of steganography which are still a subject of good research. A video is a continuous stream of frames, and a lot more data can be encrypted in a video as compared to the amount of data that could be stored in an image.

In all the papers that have been surveyed in the next section, it is observed that the papers have focused on embedding the information as bits of data in individual blocks of the images. This means that the entire information that is to be concealed has to be put in individual frames. But, all of the papers focus on storing information in the frames themselves, without giving any importance to the fact that a video has multiple frames, and hence can be used to store the secret information in the transition between the frames. This research paper is more focused on storing information in the transition of one frame of the video to another.

The input includes a secret or an important text file or text. The sender who wishes to send important data enters the location of that text file and the system will then encrypt and encode its contents in a video whose location will again be entered by the user. The receiver, after getting the video from sender, will download it and again send its location to the system. The system will then decode, decrypt and show the important data to receiver. To encrypt, a password (key) will be entered as input. While decrypting as well, a password will be asked. According to the password that the receiver enters, the decryption will be carried out on the decoded text.

2 Literature Review

In the vast field of video steganography, the number of techniques that use the least significant bit limits their scope to the individual frame themselves. There is almost no technique that indulges in the application of steganography across multiple frames to store just one bit of significant data. In one paper [3], the authors have created a combination of cryptography and steganography. The text that is to be embedded in the given video is first encrypted and then it is stored in the video using steganography. The model explained in this paper also does this, but changes the technique of both cryptography and steganography. In another paper by Yadav et al. [4], sequential encoding has been done on the least significant bit in a sequential manner. The disadvantage of this technique is that they have not used any cryptography on their data which makes the information stored susceptible to access by a third party if they are able to figure out the technique used. But, the PSNR and the RMSE of this paper is comparable to our results, which makes this technique highly imperceptible. The technique of encoding information in the least significant bit instead of the most significant bit is shown in paper [5]. As is demonstrated in the results, the pictures where the most significant bit was changed, were very distorted and it was very obvious that there was information inside them. On the other hand, the technique that used the least significant bit method had a lot less distortion and hence it is decided to stick to the least significant bit method in our paper for the least distortion of the video.

One more paper that has been reviewed, by Juneja et al. [6] looked at a technique of steganography where a 8-bit bitmap was created of the 24 bits of a pixel and then information was stored in this bitmap. While this technique leads to the reduction of space, the picture becomes very distorted as it is a lossy technique. Hence, it is decided against including this in our paper. Although it is difficult to tell that a change in the image is because of compression or hidden data, any persistent attacker might figure it out.

As for the encryption technique, many papers with different algorithms for encrypting the given data are analyzed. One such paper, Rani et al. [7] uses a modified DES algorithm for encrypting the data before embedding it into the video. But, the AES is much better than DES mainly due to the variable key length in AES encrypting [8]. The time taken is also very less to encrypt and decrypt as compared to DES. Hence, the AES algorithm is used to encrypt our data.

3 Proposed Solution

3.1 Encrypting and Encoding

3.1.1 Symmetric Key Encryption

Symmetric key encryption uses a single key to convert the data into some other form. While decryption the same key is used to decrypt the actual message from encrypted text. The symmetric key algorithm that used here is AES (Advanced Encryption Standard) algorithm. Refer paper [9], Deshpande et al. to get more information on AES Encryption algorithm. Normally a random 256-bit is generated to be the key for AES Encryption algorithm. Instead of doing this, a string is taken from the user. This password must contain at least 8 letters, 1 uppercase, 1 numeric and 1 special character. Let's consider it as a password, and then from this password, a key is generated using SHA-256 hashing algorithm [10].

Password: abc
SHA-256: ba7816bf8f01cfea414140de5dae2223b00361a39 6177a9cb410ff61f20015ad

3.1.2 LSB Encoding Technique

Steganography is the technique to hide a file, that is, the important data which is to be concealed, into a cover video. The technique that used here is the LSB (Least

Significant bit) technique to encode data into the frames of a video. The data which is in a text format will have characters and these characters will have ASCII values which are of 8 bits. Also, the video has multiple images, each image has pixels and each pixel has RGB values. The proposed solution suggests embedding the ASCII values of the characters in an image frame(of a video) in a unique way (Figs. 1 and 2).

ASCII values are of 8 bits. One pixel contains three values R, G and B. So now if three pixels are clubbed together a total of nine values are obtained. If a character has to be embedded inside these pixels then the first eight values of a pixel will be required to fill the ASCII value of a character and the ninth pixel value will suggest whether more information or data follows or not. So in this way, three pixels are required to store a character's ASCII value.

Data will be encoded using two frames of the video simultaneously. At first the data will be encoded in frames 1 and 2. If the pixels in frame 1 and 2 are depleted, then frame 2 and 3 will be used. If the pixels in frames 2 and 3 are depleted too, then frames 3 and 4 will be used, and so on. Now a typical video frame might have 720 × 1280 pixels which totals up to 921,600 pixels. Each character uses 3 pixels. So 921,600/3 = 307,200 characters can be embedded in frame 1 and 2. If characters are still left then the next pair of frames are to be used. If they are depleted, then their subsequent frames will be used.

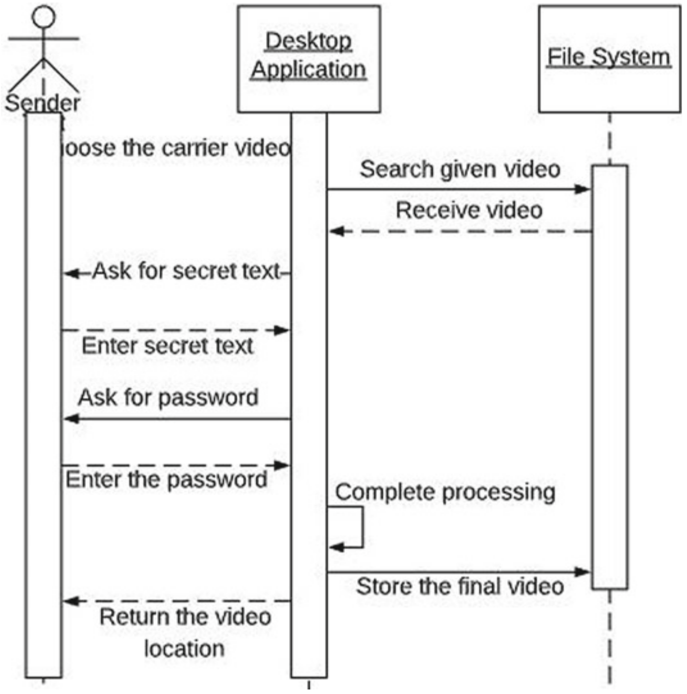


Fig. 1 Sequence diagram of the sender

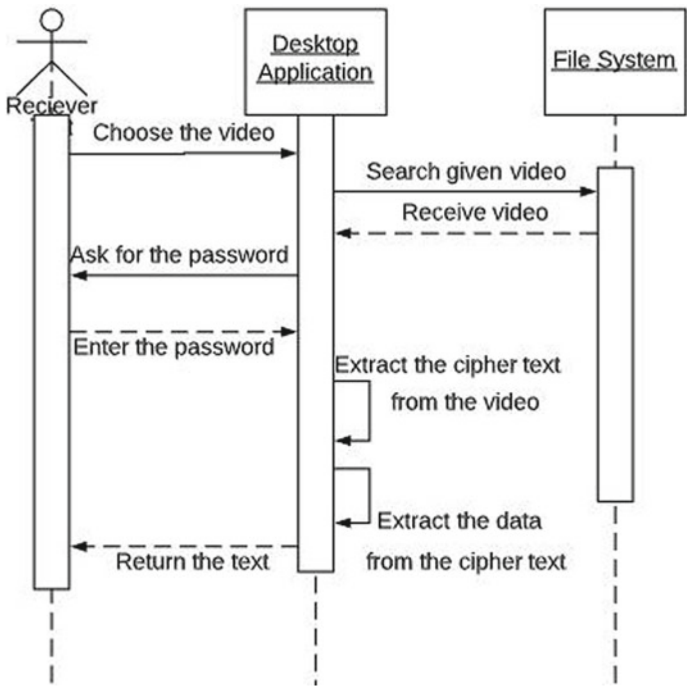


Fig. 2 Sequence diagram of the receiver

The conditions to encode are:

- If the ASCII bit of a character is 0, the modular difference between the units place of either the R or G or B values of two corresponding frames should be even. If it is odd then the least significant bit (LSB) has to be flipped.
- If the ASCII bit of a character is 1, the modular difference between the units place of either R or G or B values of two corresponding frames should be odd. If it is even then the LSB has to flipped.

For the B value of every third pixel, the following process is performed

- If more characters follow a particular character, that is, there is still more data to be encoded, then the modular difference between the B value of the corresponding frames should be even. If it is odd then the LSB has to be flipped.
- If more characters do not follow a particular character, that is, there is no more data to be encoded, then the modular difference between the B value of the corresponding frames should be odd. If it is even then the LSB has to be flipped.

Here flipped means that RGB value is converted to binary and its Least Significant Bit is flipped. So if LSB is 1(meaning number is odd) it is flipped to 0 (converted to even) and vice versa.

The procedure for encoding A(01000001) has been shown below (Table 1).

Table 1 Example of encoding the data into frames

Pixel (1, 2, 3) of Frame 1	Pixel (1, 2, 3) of Frame 2	Comments
Initial		
(223, 312, 127)	(129, 117, 119)	Actual frames. It encodes 'A(01000001)' here
(113, 119, 213)	(122, 113, 156)	
(118, 210, 115)	(126, 135, 157)	
Final		
(223, 312, 127)	(129, 117, 119)	Frame 1 will remain the same.
(113, 119, 213)	(126, 135, 157)	Frame 2: 0:mod(3–9) is even and data to be encoded is 0.
(118, 210, 115)	(122, 113, 156)	Thus there will be no change
		1:mod(2–7) is odd and data to be encoded is 1. Thus there will be no change
		0:mod(7–9) is even and data to be encoded is 0. Thus there will be no change
		0:mod(3-2) is odd and data to be encoded is 0. Thus there will be a change in LSB of frame 2(122 i.e 1111010) and 0 will be flipped to 1 i.e 1111011 and value will become 123.
		0:mod(9-3) is even and data to be encoded is 0. Thus there will be no change
		0:mod(3–6) is odd and data to be encoded is 0. Thus there will be a change in LSB of frame 2(156 i.e 10011100) and 0 will be flipped to 1 i.e 10011101 and value will become 157
		0:mod(8-6) is even and data to be encoded is 0. Thus there will be no change
		1:mod(1–5) is even and data to be encoded is 1. Thus there will be a change in LSB of frame 2(135 i.e 10000111) and 1 will be flipped to 0 i.e 10000110 and value will become 134
		Now, if there is more data then mod(5–7) i.e even, no need of change but if there is no more data then 157(10011101) will become 156(10011100). In this example lets assume only 'A' is to be stored and no more data is left

Now an interesting thing to be noted here is that whenever the pixel value was even and it had to be flipped, the value always decreased by one whereas if the pixel value was odd and it had to be flipped, then the value always increased by one.

3.2 Decoding and Decrypting

3.2.1 LSB Decoding Technique

The video that is encoded with the data is received by the receiver. That video has to be decoded in order to retrieve the secret information stored in that video. The data has been encoded using two consecutive frames of a video. So, in order to decode it, two consecutive frames have to be extracted. From those consecutive frames, their corresponding pixel values (RGB) have to be extracted. One character has been encoded in 3 pixels and hence 3 pixels are taken at once to get the character using the following conditions. Conditions to decode are:

- If the modular difference between the R, G, and B values of two corresponding frames are even (individually), then the equivalent ASCII bit is 0.
- If the modular difference between the R, G and B values of two corresponding frames are odd (individually), then the equivalent ASCII bit is 1.
- If the modular difference between the B value of every third pixel in the corresponding frames is even, then more data is present
- If the modular difference between the B value of every third pixel in the corresponding frames is odd then more data is not there.

In the encoding part, the encoding of A(01000001) in the following pixels was done: (223, 312, 127) (113, 119, 213) (118, 211, 115) of frame 1 and (129, 117, 119) (122, 113, 156) (126, 135, 157) of frame 2. After encoding A(01000001), the pixels of frame 2 were changed to (129, 117, 119) (123, 113, 157) (126, 134, 156). Now using the above conditions it can decode and find the data that was encoded (Table 2).

In this way, the message will be decoded, which was initially encoded in the video. If frames 1 and 2 get depleted and still more data is left to encode, then frames 2 and 3 will be used for decoding. If frames 2 and 3 do not suffice, then the subsequent pair of frames will be used for decoding and so on until the modular difference between the B value of the last pixel in the group of 3, is obtained as odd.

So, the flow of the process right from sender sending the message to receiver receiving it is given below:

- User sends the important data(secret text), cover video, and password along with it to the offline application.

Table 2 Example of decoding the data from the frames

Pixel (1, 2, 3) of Frame 1	Pixel (1, 2, 3) of Frame 2	Comments
Initial		
(223, 312, 127)	(129, 117, 119)	
(113, 119, 213)	(123, 113, 157)	
(118, 211, 115)	(126, 134, 156)	
Final		
(223, 312, 127)	(129, 117, 119)	mod(3 - 9) is 6. Which is even. Thus data is 0
(113, 119, 213)	(118, 211, 115)	mod(2 - 7) is 5. Which is odd. Thus data is 1
(123, 113, 157)	(126, 134, 156)	mod(7 - 9) is 2. Which is even. Thus data is 0
		mod(3 - 3) is 0. Which is even. Thus data is 0
		od(9 - 3) is 6. Which is even. Thus data is 0
		mod(3 - 7) is 4. Which is even. Thus data is 0
		mod(8 - 6) is 2. Which is even. Thus data is 0
		mod(1 - 4) is 3. Which is odd. Thus data is 1
		mod(5 - 6) is 1. Which is odd. This means that there is no more data, if it would have been even, then more data would be there

- The data will then be encrypted using symmetric key encryption (AES technique) [9] using the SHA-256 hash value [10] of the password entered by the sender as its symmetric key. This password has to meet all the standards of a good password, like having at least 8 characters, 1 uppercase, 1 special character, and 1 numeric
- Then that encrypted data is hidden using video steganography using the LSB technique mentioned above.
- The sender then sends the video and the password via a pen drive or via any medium. He or she needs to ensure that the medium will not use any lossy compression technique on the video or the data might get corrupted.
- Now the receiver who gets the video will again upload on our application.
- Receiver will enter the password he received from the sender and its SHA-256 hash value will be used for decryption of data and the actual data is displayed to the receiver.

3.3 Results

The time complexity of the proposed solution is O(n). That is, our solution has a linear time complexity (asymptotically). The time complexity remains the same for both the encrypting and the decrypting algorithms.

To test the results of the above-mentioned technique, a desktop application is created based on python with two libraries named Tkinter and Opencv2 for giving a graphical user interface and to fiddle around videos for changing their pixel values respectively.

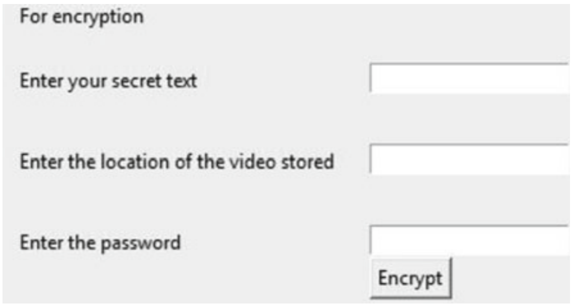
The efficiency and accuracy of the result can be calculated using two parameters called RMSE (Root mean square error) [12] and PSNR (Peak Signal to Noise Ratio) [13]. These similarity measures are used to identify the difference between the original and the final video.

Root Mean Square error (RMSE) is the standard deviation of the approximation values from original values. That is, RMSE is a measure of the difference between the original image and the final image. The square root of the square of difference between the data and its approximation, divided by the number of elements. In this case, the RMSE value is calculated between the original frame and then the frame after encoding. The low value indicates that there is no significant difference between original video and encoded video.

Peak Signal to Noise Ratio (PSNR) is defined as the ratio between the maximum possible power of a signal and the power of the corrupting signal. Its ideal value is infinity. But a value above 30–50 is considered acceptable [11]. In paper [4], the PSNR value that they recorded is 33.75 DB. On the other hand, a PSNR value of 66.83DB is recorded (Figs. 3, 4, 5, 6, 7 and 8; Table 3).

3.4 Conclusion and Future Work

This paper has explored a never before explored space in video steganography. All video steganography papers and implementations split the entire secret text into

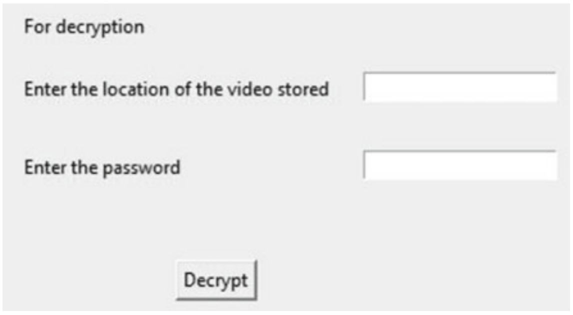

For encryption

Enter your secret text

Enter the location of the video stored

Enter the password

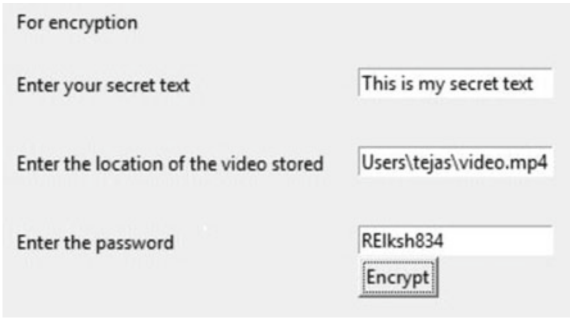
Fig. 3 Screenshot of the encryption


For decryption

Enter the location of the video stored

Enter the password

Fig. 4 Screenshot of the decryption

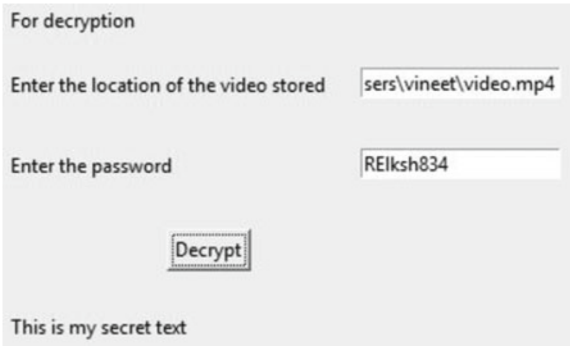

For encryption

Enter your secret text: This is my secret text

Enter the location of the video stored: Users\tejas\video.mp4

Enter the password: REIksh834

Fig. 5 Screenshot of the encryption


For decryption

Enter the location of the video stored: sers\vineet\video.mp4

Enter the password: REIksh834

This is my secret text

Fig. 6 Screenshot of the decryption

blocks and then each block is stored individually in one frame of the video using different techniques. This method is used extensively in all the papers that are surveyed. The proposed work has stored the data blocks in the difference between the pixels when it is on transition from one frame to another. In the results that are obtained after the implementation, it is noticed that any change was completely imperceptible, not just to the human eye, but the very small RMSE value means that even a computer would likely reject the discrepancy in the original and the final video as a mere transmission error. In case if rigorous steganalysis is applied, taking out the cipher text in the video is highly difficult since the difference between pixels from

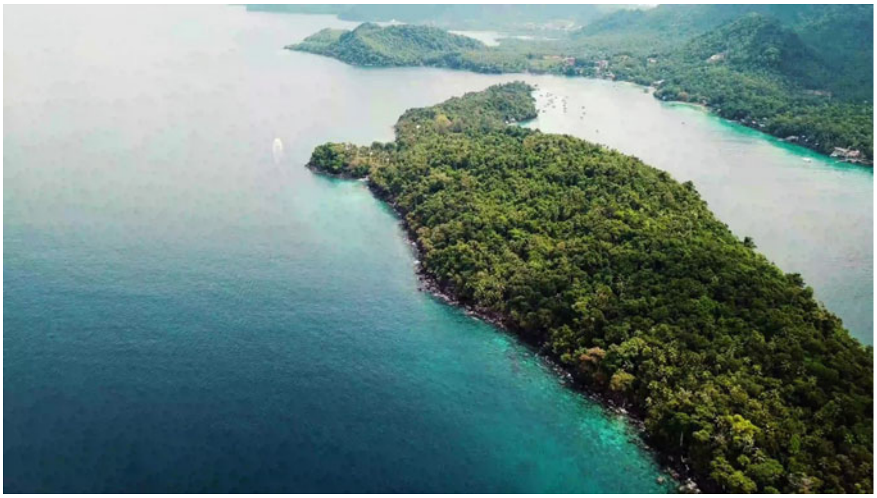


Fig. 7 The second frame before encryption of secret data

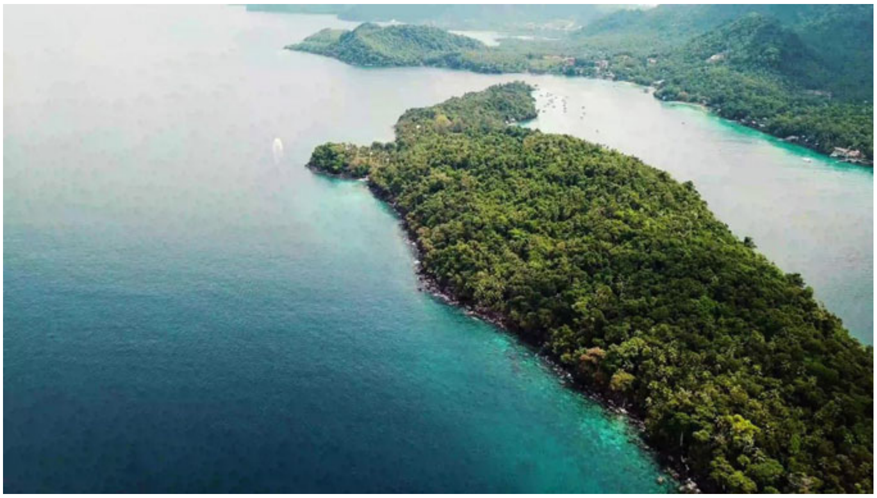


Fig. 8 The second frame after encryption of secret data

across two frames are used to embed the data. Even if the steganalysis is successful, the hacker will get an AES encrypted cipher text, which is very difficult to crack. This makes the technique explained in this paper very robust to external attacks. This technique can be used as of today for IPR purposes. An artist can easily add his or her name in the picture that they create. This will be useful in copyright cases. This technique can also be used by businesses to hide their technological innovations or chemical formulae. Additionally, in the future, instead of embedding the information

Table 3 Comparative results

Parameters	Values	Ideal	Acceptable
PSNR	66.83 dB	Infinite	More than 30 dB
RMSE	0.201	Zero	As low as possible

in frames sequentially, this can be randomised according to the password hash generated by the password given by the sender. Also, it is observed at splitting the text into blocks such that one block can be embedded in two frames. Then, this process will be parallelized to ensure that the least amount of time is taken to process the data hiding. Furthermore, it has observed the modular difference between two frames. But, this can also be done by taking the modular difference between the pixels of three or more frames. Hence, even if a single bit of data is modified, the entire data will get corrupted. This way the integrity of the messages can be maintained even better.

References

1. Kessler GC (1998) An overview of cryptography. Auerbach, The Handbook on Local Area Networks
2. Kumar A, Pooja K (2010) Steganography-A data hiding technique. Int J Comput Appl 9(7):19–23
3. Mustafa MF, Beh dmy. (2018) Secure data transmission by using video steganography
4. Yadav P, Mishra N, Sharma S (2013) A secure video steganography with encryption based on LSB technique. In: 2013 IEEE international conference on computational intelligence and computing research. IEEE
5. Thangadurai K, Sudha Devi G (2014) An analysis of LSB based image steganography techniques. In: 2014 International conference on computer communication and informatics, Coimbatore, pp 1–4
6. Juneja M, Sandhu PS, Walia E (2009) Application of LSB based steganographic technique for 8-bit color images. World Acad Sci Eng Technol 50:423–425
7. Rani S (2013) LSB and MSB based steganography for embedding modified DES encrypted text
8. Mahajan P, Sachdeva A (2020) A study of encryption algorithms AES, DES and RSA for security. Glob J Comput Sci Technol [Online] (0): n. pag. Web. 30 Mar. 2020
9. Deshpande AM, Deshpande MS, Kayatanavar DN (2009) FPGA implementation of AES encryption and decryption. In: International conference on control, automation, communication and energy conservation, Perundurai, Tamilnadu, pp 1–6
10. Gilbert H, Handschuh H (2004) Security aAnalysis of SHA-256 and Sisters. In: Matsui M, Zuccherato RJ (eds) Selected areas in cryptography. SAC (2003). Lecture Notes in Computer Science, vol 3006. Springer, Berlin, Heidelberg
11. Salomon D (2007) Data compression: the complete reference, 4th edn, p 81. Springer. ISBN 978-1846286025. Retrieved 26 July 2012
12. Shcherbakov MV et al (2013) A survey of forecast error measures. World Appl Sci J 24(2):171–176
13. Huynh-Thu Q, Ghanbari M (2008) Scope of validity of PSNR in image/video quality assessment. Electron Lett 44(13):800–801