This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3016211, IEEE Access

IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2020.DOI

Preserving Chain-of-Evidence in Surveillance Videos for Authentication and Trust-Enabled Sharing

NADIA KANWAL[1,2], (Senior Member, IEEE), MAMOONA N. ASGHAR[1, 3], MOHAMMAD SAMAR ANSARI [1,4],(Senior Member, IEEE), MARTIN FLEURY [5], BRIAN LEE [1], MARCO HERBST [6] and YUANSONG QIAO [1]

[1]Software Research Institute, Athlone Institute of Technology, Ireland (nkanwal@ait.ie, blee@ait.ie, ysqiao@research.ait.ie)
[2]Lahore College for Women University, Lahore, Pakistan. 54000 (nadia.kanwal@lcwu.edu.pk)
[3]The Islamia University of Bahawalpur, Punjab, Pakistan. 63100 (masghar@ait.ie)
[4]Aligarh Muslim University, Aligarh, India (mansari@ait.ie)
[5]School of EAST, University of Suffolk, Ipswich, UK (fleury.martin55@gmail.com)
[6]Evercam Pvt. Ltd., Dublin, Ireland. (marco@evercam.io)

Corresponding author: N. Kanwal (e-mail:nkanwal@ait.ie).

This paper is part of the Marie Skłodowska-Curie Career-FIT Postdoc Fellowship programme under project ID: MF-2018-0058 funded by the European Union’s Horizon2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 713654 and Science Foundation Ireland (SFI) under Grant Number SFI 16/RC/3918 and the European Regional Development Fund

ABSTRACT Surveillance video recording is a powerful method of deterring unlawful activities. A robust data protection-by-design solution can be helpful in terms of making a captured video immutable, as such recordings cannot become a piece of evidence until proven to be unaltered. Similarly, video sharing from closed-circuit television video recording or in social media interaction requires self-authentication for responsible and reliable data sharing. This paper presents a computationally inexpensive method of preserving a chain-of-evidence in surveillance videos by means of hashing and steganography. The method conforms to the data protection regulations, which are increasingly adopted by governments, and is applicable to network edge storage. Encryption keys are stored in a hardware wallet independently of the video capture device itself, while evidential information is stored steganographically within video frames themselves, independently of the content. Added protection is provided by hiding information within the two least-valued of pixel bitplanes, using a newly introduced technique that randomizes the pixel storage locations on a per video frame and video-capture device basis. Overall, the proposed method has turned out to not only preserve the integrity of stored video data but also results in minimal degradation of the video data resulting from steganography. Despite the inclusion of hidden information, video frames will still be available for common image-processing tasks such as tracking and classification, as their objective video quality is almost unchanged.

INDEX TERMS Video Security, Video Surveillance, Steganography, Hashing, Information Sharing

I. INTRODUCTION

It is no longer the case that only special places, such as international airports, are kept under surveillance for security and safety purposes. In fact, surveillance is rapidly becoming a requirement for almost every house, office, and public place. In those environments smart tracking can be applied [1], along with the classification of objects within the video. This development is further fuelled by the availability of low-cost and small-sized surveillance cameras. These cameras potentially help law-enforcement agencies to utilize the resulting video recordings as proof of a crime or illicit activity. However, due to equally rapid advances in the field of image processing, surveillance video data can be easily tampered with. Examples of tampering include [2]: regional alteration of intra-frames, through cut-and-paste; and inter-frame forgery, by the insertion of video frames. Furthermore, transmission errors also contribute to the alteration of videos, if they are not suitably protected with forward error correction or other error protection methods. Consequently, such recordings are not directly admissible in a court of law as an item of evidence until they are proven to be authentic through forensic analysis [3]. Unfortunately, applying forensic analysis is not a cheap operation in terms of time and expense. Thus, an important objective for a Closed Circuit TeleVision (CCTV) systems is that it should provide a chain-of-evidence, according to stated rules. In that way, the videos can prove themselves to be an authentic piece of information, avoiding or reducing the need for forensic analysis. In other words the videos themselves can self-authenticate.

A driver towards self-authentication is the need for storage and analysis of huge amounts of multimedia data, which has propelled companies towards cloud storage for efficient access [4]. However, following on from the European Union's ratification of the General Data Protection Regulation (GDPR) in May 2018 , cloud storage may be unacceptable, due to the potential for unauthorized access to video data by third parties. Other similar data privacy regulations to GDPR now exist in other countries. GDPR itself focuses on reversible data protection via encryption [5] as a data-protection safeguard [6], which encryption is also employed herein. For example, in [7], a chain-of-evidence is kept for forensics purposes but a block chain is employed to store hashes of information gathered, which may add to the complexity of the method. On the other hand, this article presents a multifaceted solution: to the problem of self-authentication; to the requirements of data protection regulations; and in respect to non-cloud storage.

The proposed solution works as follows:

1) Every frame holds the hash of its previous two frames and, therefore, a chain-of-evidence is created.
2) Each video frame is also protected by calculation and storage of its hash, after the insertion of the hash of step 1.
3) These two hashes are stored inside each video frame as hidden information.
4) Information is hidden using a modified version of the already efficient Least Significant Bit (LSB) steganography algorithm. This is achieved by unpredictable placement of the information, according to a technique introduced in this paper. Information is stored in the LSBs of the three color component (RGB) values.
5) The technique introduced is capable of generating random positions for placement of information in each video frame. Furthermore, it will be shown that the generation of randomized locations for information concealment may also be performed in two additional ways. This can be firstly through identification information from the capturing device (e.g. the MAC address of the camera) and secondly through the video frame information, as mentioned above. This ensures that the identity information of the capturing device also plays a role in the generation of the randomized path for information storage.
6) The path to the random positions of those pixels that carry the hidden information is stored in the second lowest bitplane (second least significant bits in the RGB values).
7) All video frames are encrypted and stored on the network edge, i.e. close to the video capture device itself, without being kept in cloud storage or transmitted over a wide-area network.
8) To further enhance the security, encryption keys are stored in a hardware wallet, separately from the storage device holding the actual video.

The strength of the proposed method is that it hides evidential data inside the video frames themselves. That is achieved with limited effect upon the image content, because the hidden data is negligible in quantity and exists only in the lowest two bits of pixels. Selecting a modified LSB steganography makes for video rate insertion of hidden information. This rapid insertion rate can be compared with (say) Bit-plane Complexity Segmentation (BPCS) [8], which needs to search for noisy blocks within bit-planes before hidden information can be embedded. Furthermore, the proposed method makes it easier to maintain, synchronize, and prevent loss of evidence. Such a chain-of-evidence is desirable not only in the case of surveillance videos but could also be useful in ascertaining the originality and authenticity of videos uploaded to social media sites.

The rest of this paper is organized as follows. Section II establishes the main components of the chain-of-evidence method. Then Section III describes and analyses that method in respect to use of steganography. However, the usage of encryption in the methodology is treated separately in Section III-E. Following on, Section IV is an evaluation of the proposed chain-of-evidence method, suitable for GDPR usage, while Section V summarizes the paper's contribution.

II. BACKGROUND

As is well known, steganography refers to the hiding of information (text, audio, image, video) in another carrier media (usually referred to as the 'cover'). There are four essential properties of a good steganographic system [9], *viz.* 1) imperceptibility, 2) security, 3) information hiding capacity, and 4) robustness. There have been several different approaches to achieve the goal of information hiding in the cover media. Thus, an effective approach works by manipulating LSBs of the three color channels (Red, Green, Blue (RGB)), in the light of the fact that the LSBs carry minimal information. This spatial approach ensures that the overall visual aesthetic of the image is not significantly altered. Initial research into LSB steganography concentrated on designing the system to increase the payload capacity by utilizing most of the cover-image pixels. [10]–[18]. However, steganalysis techniques soon became strong enough to break such systems using statistical analysis, which was developed to identify the regular patterns by which data were stored inside the cover [19]–[21]. Therefore, there was a need for robust LSB techniques based on cryptography-steganography, which can evade such steganalysis attacks. The subsequent research into the Stego Color Cycle (SCC) approach [22], Magic LSB technique [16], hash-LSB [17] and nearest-centroid clustering (LSB-M) [18] are comparatively recent such additions to the field of steganography. In this paper, as described in detail in

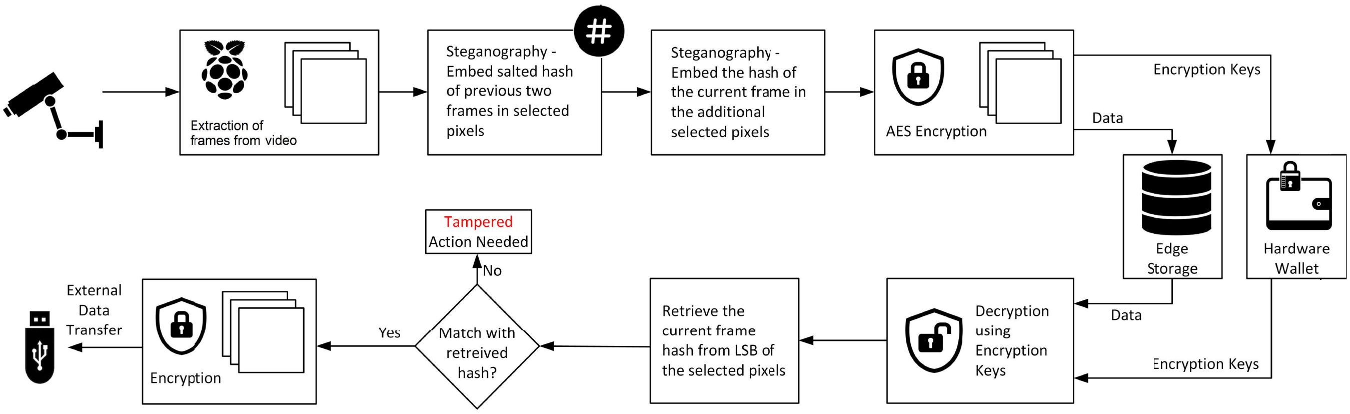

Extraction of frames from video
Steganography - Embed salted hash of previous two frames in selected pixels
#
Steganography - Embed the hash of the current frame in the additional selected pixels
AES Encryption
Encryption Keys
Data
Edge Storage
Hardware Wallet
Data
Decryption using Encryption Keys
Encryption Keys
Retrieve the current frame hash from LSB of the selected pixels
Match with retreived hash?
No
Tampered Action Needed
Yes
Encryption
External Data Transfer

FIGURE 1: Life cycle from video capture, embedding evidential data and current frame hash, and edge storage to decryption, validation of evidential data and current frame hash, and onward video transfer

Section III, after per video frame identification of a starting pixel or seed, random pixel selection for embedding the data within LSBs takes place within the neighborhood of the seed. Relatively few of a cover image's pixels are chosen for data hiding, avoiding another problem with early usage of LSB steganography.

The principal advantage of the LSB technique is the inherent simplicity of embedding and decoding process, making video rate operation possible. However, the decoding of such stego-images is affected by the levels of noise in the communication channel. Nonetheless, in the envisaged applications, surveillance video frames after capture are stored at the network edge, without transmission over a communication channel. Data transfer is by portable Universal Serial Bus (USB) drive (see Section III). Storage at the network edge is anyway advisable because of the risks arising from third-party intervention during cloud storage.

For encryption there exists a number of block-based, symmetric encryption techniques, such as the Blowfish, RC5, and Advanced Encryption Standard (AES) algorithms [23]. All of the latter algorithms avoid the need for a Public Key Infrastructure in the case of asymmetric cryptography. Even so, in terms of computational overhead, their calculations still involve considerable processing. This is especially so if real-time operation on resource-constrained edge devices is required. However, considering the sensitivity of the surveillance data, the industry-standard AES algorithm was selected. As AES is a symmetric block cipher so a single key is used for the encryption and decryption processes.

AES (also known as Rijndael) has been widely deployed as an encryption standard since 2000 [24]. Indeed, AES is considered a robust industry standard cipher and, hence, is extensively utilised to provide confidentiality in cyber-physical systems [25]. Overall, AES is widely employed because of its ease of implementation, defences against threats and attacks, as well as flexibility in the cases of encryption/decryption modes and keying material. In terms of keys, AES is a symmetric-key block cipher, which supports either a 128-bit key for 10 rounds, a 192-bit key for 12 rounds, or a 256-bit key for 14 rounds of operation. Further information about the communication mode of AES utilized herein can be found in Section III.E.

AES acts upon a 4 × 4 byte matrix, which is called the algorithm's state. After initiation of the algorithm, every round comprises of four stages/phases: (1) Byte-substitution, (2) Shift Rows, (3) Mix Columns, and (4) Add Round Key. Stage 1 provides non-linear substitution by substituting each byte in the state with a byte from a lookup table or S-Box. In that way, the substitution part of a classic substitution-permutation cipher takes place. Permutation occurs in stage 2 through cyclic left-shifts of the state's rows. In stage 3, the four bytes of each column of the state undergo a linear transformation. Finally, stage 4 derives a sub-key from the main-key. The sub-key has the same number of bytes as the state and, hence, can be added to it. Because addition is defined as an Exclusive-OR (XOR) operation in stages 3 and 4, again a non-linearity is introduced into the processing. Likewise, modulo arithmetic in stage four provides multiplication, introducing further non-linearity. Given that substitution provides confusion and stages 2 and 3 supply diffusion, the whole algorithm meets the need to contain confusion and diffusion in any acceptable cipher.

Turning to the hashing functions used in this paper to detect tampering of video files, these functions work by extracting a unique fixed-length bit-string from a given message (text, image, video). However, by 2015 one of the principal such functions, Secure Hash Algorithm 1 (SHA-1), was found by the software industry to be suspect because of likely collisions, i.e. the same hash produced by different messages. Therefore, in this paper, the SHA-256 hashing algorithm [26] has been used in this work, as, differing in design from SHA-1, it is currently thought to be collision free. SHA-256, operating with 32-byte words is one of the two members of the SHA-2 family of hash functions, the other being SHA-512 with 512 byte words. There is another later family of hash functions known as SHA-3. However, industry opinion is that for software implementations SHA-3 is slower than the SHA-2 family. To avoid that risk on constrained CCTV devices processing video rate data, this paper's implementation retained SHA-256 from the SHA-2 family of hashes. However, given the ongoing development of attacks, it is wise to keep the situation under review, with a view to possible eventual implementation of a SHA-3 hash function. In general, altering even a single bit of the hash input changes the output hash completely. For the purposes of the forensic use of video surveillance, no image processing is permitted, which is why this article retains the use of SHA-256. However, for the video sharing applications mentioned in Section I, such multimedia data may go through various manipulations such as cropping, scaling, enhancement, and compression. In that case, possible incorporation of one of the robust image hashes such as [27] is possible.

III. ANALYSIS OF THE METHOD

The following, after providing a system overview, describes and motivates the various aspects of the method, before giving a detailed step-by-step analysis of the method employed.

A. OVERVIEW

As illustrated in the overview of Fig. 1, a chain-of-evidence is stored along with the originally captured visual data so that the authenticity of the video content can be established afterwards. Firstly, individual frames are extracted from the captured surveillance video. Next, salted hashes are embedded through steganography in each video frame (see Section III-C). A hash of each video frame is also taken and embedded through steganography. These latter hashes are needed as a way to verify that the video frames have not been tampered with. Following encryption with the industry-standard Advanced Encryption Standard (AES) [23], the video data are stored in close proximity to the video capture device at the network edge, without transmission over a network. Symmetric encryption keys are stored in a hardware wallet, which may be accessed by a combination of a smart card and a biometric identifier. When a sequence of video frames is retrieved from storage, after decryption, each frame's image data and its embedded salted hashes are checked by comparison with the additional embedded hash, after it has been extracted. A detailed description of this procedure is given in Section III-F.

Notice that hardware wallets are thought to be [28] the most secure option for storing public-private, asymmetric keys needed in access of cryptocurrencies, though herein symmetric keys are employed. The process is reversed when a video frame is checked for tampering before transfer in encrypted form, as required by GDPR.

B. STORAGE OF HASHES

The Secure Hashing Algorithm (SHA)-256 has been used in this paper, as, differing in design from the deprecated SHA-1 (since 2015), it is currently thought to be collision free. In order to create a chain-of-evidence by means of hashes, a salted hash is first created by concatenating half of the bytes of the hashes of each of the previous two video frames, as described by equation (1). Therefore, each frame will store a salted hash of data derived from the previous two frames (256 bits in all). Other optional data such as the camera identity, with the date and time of video capture, and the Global Positioning System (GPS) location, could also be included in the salted hash. However, there is a trade-off according to the amount of optional data included because currently, relatively few of a cover image's pixels are chosen for data hiding, avoiding the excessive hidden bits of early usage of LSB steganography [10].

$$S_{\#} = \frac{1}{2}\#(F_{i-1}) + \frac{1}{2}\#(F_{i-2}) \tag{1}$$

Subsequently, the frame's own hash (256 bits), after the insertion of the evidential data, is calculated and also inserted. The essential size of the data to be stored (without optional data) becomes 512 bits per frame. These data provide authentication or guarantee of the integrity of the stored video frames, to guard against the possibility of tampering.

In fact, various ways exist to store such data, including: a metadata stream [29]; using a frame's subtitles (as available in current surveillance systems) [30]; or through steganography. However, a metadata stream is not used in this paper's method because storing data in a separate channel may lead to loss of evidence, misalignment in the retrieval module, and increasing management costs. Similarly, saving data in subtitles directly threatens the secrecy of data and, more importantly, disrupts the actual image content, which may be important to data controllers when performing a query-based search. Therefore steganography is selected as a means of storage, as it avoids the delays of side-channels and the insecurity of subtitle storage.

C. MODIFIED LSB STEGANOGRAPHY

LSB steganography has the advantage that its impact on the measured PSNR is reduced compared to some other forms of steganography [31]. However, even though the cover image quality is relatively improved, if an attacker suspects that an image holds steganographic information then extracting data hidden by LSB-based steganographic is relatively easy. This is because methods of statistical analysis have been developed to identify the regular patterns by which data used to be stored inside a cover [19].

To counter such attacks, herein we introduce random pixel selection starting from a seed pixel's neighbourhood, storing the path to the random pixel locations as steganographic data. Further, the seed pixel is also now selected using a procedure dependent on the current video frame and (optionally) the video capture device identity. The selection of the seed pixel, the generation of random locations for information storage, and the actual storage of the information are explained next.

Seed Pixel: The seed pixel refers to a randomly identified starting pixel. The seed pixel is the start of the random path containing the pixels identified for data storage. The

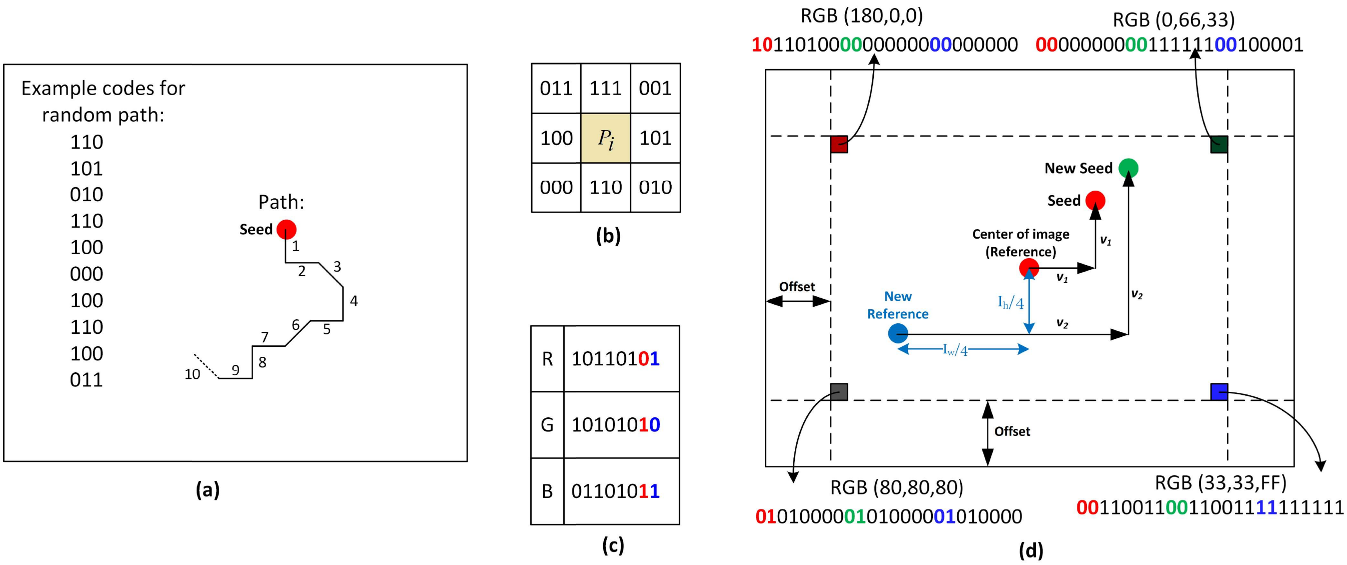


FIGURE 2: (a) A randomly identified seed pixel (red) and a random path obtained with the codes listed alongside; (b) codes for random selection of neighbourhood pixels for a given pixel; (c) storage of the path (red) and the data (blue) bits in the pixel RGB values; and (d) detailed seed pixel selection process.

process of seed selection, and pertinent issues are discussed in Section-III-D. Notice that the red pixel in Fig 2(a) is an example of a randomly-chosen seed pixel.

Path Selection: Any pixel under consideration (except the border pixels) has eight 'neighboring' pixels. For path selection, three bits are needed to generate eight different codes for eight neighbourhoods of a pixel, as shown in Fig. 2(b). These three bits are stored in the second-fro-last LSB in each of the three color channels for the chosen pixel, while the last LSB of each color channel represents the data bit, as shown in Fig. 2(c). In this way each selected pixel will store 3 bits of the data, together with the 3-bit code of the path to the next storage pixel. Notice that the distance between two randomly selected pixels is set in this paper's experiments at five, though this value can also be customized. Thus, the codes shown in Fig. 2(b) only denote the *directions* of the pixels to be selected (not the gap between them). Only denoting the directions helps to ensure that no two pixels have overlapping neighbourhoods. It also avoids any path reversals. A sample random path generated from a dummy random sequence of codes is also presented in Fig. 2(a).

Storage: For storage, two bits per color channel of a pixel are used for data (1 bit) and path (1 bit) storage. This allows for 6 bits of a given pixel encoded in the standard RGB format. Thus, the LSB technique is modified to possibly change the value of pixels in the two lowest bit-planes rather than just the lowest bit-plane. By comparison, BPCS steganography also selects bits from differing bit-planes, though it is sensible to avoid the higher bit planes [8] because of their greater correlations between neighboring bits. Using this paper's method, 171 pixels will be required to store 512 bits of data that consist of 256 bits of salted hash as evidential data and 256 bits of a frame's own hash along with the path to be followed for the storage and retrieval of hashes.

D. SEED IDENTIFICATION

The process of seed selection to identify the starting point of the random path is illustrated in Fig. 2(d). A rectangle is selected from each video frame by setting an offset value from the edge of the image, as shown in Fig. 2(d). In Section IV the offset is set to 50 pixels for all video frames for testing purposes. However, this rectangle size can be varied on a per capture-device basis and varied for each video frame, bearing in mind that the Media Access Control (MAC) address of each device is one way to start with a unique number from which the rectangle's sizes can be generated.

Subsequently, the corner pixels of the image rectangle inside this border region are then identified. The first two bits of the each of the RGB components of each of these corner pixels are concatenated and converted to a denary or "decimal" value. Thereafter, the four decimal values obtained are averaged to get a single number (rounded to the nearest integer), which is used to point to the seed pixel in the image. Because the value of the corner pixels will vary from frame to frame, this procedure will normally result in different seed locations within the rectangle on a per video frame basis. As shown in Fig. 2(d), this average decimal value is referred to as v_1 pixels, and the seed pixel is identified in this paper as v_1 pixels horizontally to the right of the centre of the image (the reference point), and then v_1 pixels vertically after that. Notice that because the value of the corner pixels contains the location of the storage starting point or seed pixel these pixels should not be altered in the steganographic process.

As an example, consider the corner pixel values from

Fig. 2(d). The concatenated binary numbers obtained from the first 2 bits of the RGB components of each pixel are:

Binary: 100000, 000000, 010101, 000011

Equivalent decimal: 32, 0, 21, 3

Average (rounded to an integer): 14, which could be used for $v1$ or $v2$ (if average is large) in Fig. 2(d).

This method of seed location, however, is prone to a pitfall, in that the average value calculated could turn out to be too large for a given rectangle, causing the seed pixel to lie outside the rectangle. To circumvent this issue, the reference point may be shifted by $(I_h/4, I_w/4)$, as also shown in Fig. 2(d), and then the seed pixel located relative to this new reference, when using larger calculated value, $v2$. Here, (I_h, I_w) are the image height and width values respectively in units of pixels. Should this procedure still result in a seed location outside the rectangle then the reference point can be further adjusted in a similar fashion. The procedure is illustrated by the blue text in Fig. 2(d). Notice that the maximum value of the calculated value for $v1$ (or $v2$) is 64 pixels, which implies that this hazard will only occur for particularly small rectangles. In fact, setting horizontal and vertical minima to the rectangle's size is another way to avoid the need for recalculation of the reference point.

A random path is started from a seed pixel's location. That path is constructed is such a way that if the path reaches the edge of the image region under consideration, the path is continued from the opposite edge of the image (as in the classic `PacMan` video game). If the random path touches any corner pixel then it should be modified to avoid changing that pixel's value. This is because corner pixels are used to retrieve the seed pixel's location.

Fig. 3 shows the pixels in which data are stored, marked in blue. The two video clips illustrated are (upper) VISOR1, available from [32], resolution 640 × 480 pixels/frame, surveillance video captured at 7 fps, and Duval_street_cam, available from [33], street scene, resolution 946 × 360, captured at 25 fps. It can be seen that every frame has a different distribution of pixels to store this data. Furthermore, the limited data are stored uniformly in RGB color channels and use only two bits per channel for each pixel selected. Notice that the proposed approach is specially suitable for an application to surveillance video, when no image processing is allowed on the data captured, and when all modifications of the video data need to be identifiable so as to ascertain the authenticity of the video.

E. ENCRYPTION PROCEDURE

AES has multiple operational modes available [34]. The mode used herein is Output Feedback (OFB) mode. One reason for utilizing OFB mode is because the same program code serves for both encryption and decryption, thus saving on the coding space within an embedded device. Another reason for choosing OFB mode is that, consequently, AES then operates as a stream cipher (rather than a block cipher), in which a continuous stream of bytes are encrypted. Any modifications to a plaintext block P_t are reflected in the corresponding ciphered block C_t, where $t = 1, 2, 3, \ldots n$ with n the number of plaintext blocks. However, other ciphered blocks remain unaffected. Thus, a significant reason for choosing OFB mode is that it provides a degree of transmission error resilience. Moreover, if any modification/error occurs during the transmission, that error is not propagated. (Also see the remark on OFB's avoidance of padding at the start of Section IV.) However, OFB lacks self-synchronization, being independent of the previous ciphertext. Therefore, if synchronization is lost during transmission then a new IV needs to be established to enforce explicit re-synchronization.

In OFB, suppose X_{t-1} is an input block from the $t-1$ stage, which has been AES encrypted, using key K_e. Then X_{t-1} is again AES encrypted using key K_e to produce X_t. After that X_t and the next plaintext block P_t are XORed together to output encrypted block C_t. For encryption of the following plaintext block, AES encryption with K_e is again performed on the X_t of the previous stage to produce X_{t+1}, after which XORing is again performed with the plaintext P_{t+1} to output C_{t+1} and so on. Moreover, OFB normally generates different output C_t for the identical input P_t because the process is inititated with a random Initialization Vector (IV) [35]. The following equations (2) and (3) represent the encryption and decryption processes in OFB mode, respectively.

$$C_t = P_t \, XOR \, X_t \tag{2}$$

$$P_t = C_t \, XOR \, X_t, \tag{3}$$

where $t = 1, 2, 3, \ldots\ldots n$, for n stages of block encryption, and $X_t = Encrypt(K_e(X_{t-1}))$

Symmetric encryption keys are generated at run-time for each protected video, by using a pseudo-random function (PRF). To keep the procedure simple, encryption keys are generated on a per video sequence basis. They are then stored separately in a Hardware Wallet as shown in Fig. 1. Key security can be enhanced by using any standardized key-management scheme [36]. This would be the case for real-time key distribution during transmission of CCTV videos over a network. However, in this paper we only assume the secure physical transmission of data via a Universal Serial Bus (USB) connection directly to and solely to an authorized party. Additionally, an 128-bit key is secure enough because, for current computing powers, a key space greater than 2^{100} is considered resilient to key guessing or brute force attacks upon keys [37]. It should be mentioned that the person responsible for key management in any well-managed organization should not also be the person with the authority to alter data in any way. This is an application of the well-established security principle of the separation of roles.

F. DETAILED DESCRIPTION

A detailed description of the steps in the method is provided next, with video frames indexed by i:

1) Generate the current storage rectangle's 'offset' value (see Fig 2(d)) by means of a unique identifier of the video capture device.
2) Create a 256-bit salted hash ($S_\# = F_{i-1}\# + F_{i-2}\#$) from the hashes of the previous two frames. This is a protection against video-frame insertion forgery.
3) Generate the seed pixel's position using the four corner pixels of the current frame's rectangle.
4) Generate a random sequence of 170 directions starting from the seed pixel. The path must avoid the four corner pixels used in the calculation of the seed position and may wraparound if the rectangle's edge is reached.
5) Embed the evidential data $S_\#$ in 85 pixels according to the random sequence of directions in the current video frame, F_i. The directions to the next pixel in the path are also stored in each pixel of the current video frame.
6) Embed zeros in a further 86 pixels' data bits to act as a container for a hash of the current frame. The directions to each pixel in the path are also stored in these further pixels, with a zero direction to terminate the path.
7) Calculate the hash of the current frame, $C_\#$.
8) Insert $C_\#$ in F_i's reserved bits (from Step-6) by replacing the zeros previously set.
9) Check whether a video session has finished and if not return to step one.
10) Encrypt a session of video frames by means of AES and storage of the symmetric key (K_p) in a hardware wallet, along with the corresponding session id.

As mentioned above in step five, the evidential data, i.e. the salted hash created from the previous two frames, is stegnographically embedded in each frame. It is convenient to use the two previous frames after the additional hash of each frame, $C_\#$, has been inserted. Otherwise, it would be necessary to remove the additional hashes from the two previous frames before creation of the salted hash for the current frame. It is also important to mention that for the first two frames the salted hash is created by generating two AES keys using a key management module. However, these keys do not need to be stored or sent to the receiver because they are only employed to generate a salted hash. That salted hash is stored in the initial two video frames for their own hash calculation, i.e. creation of each of their $C_\#$.

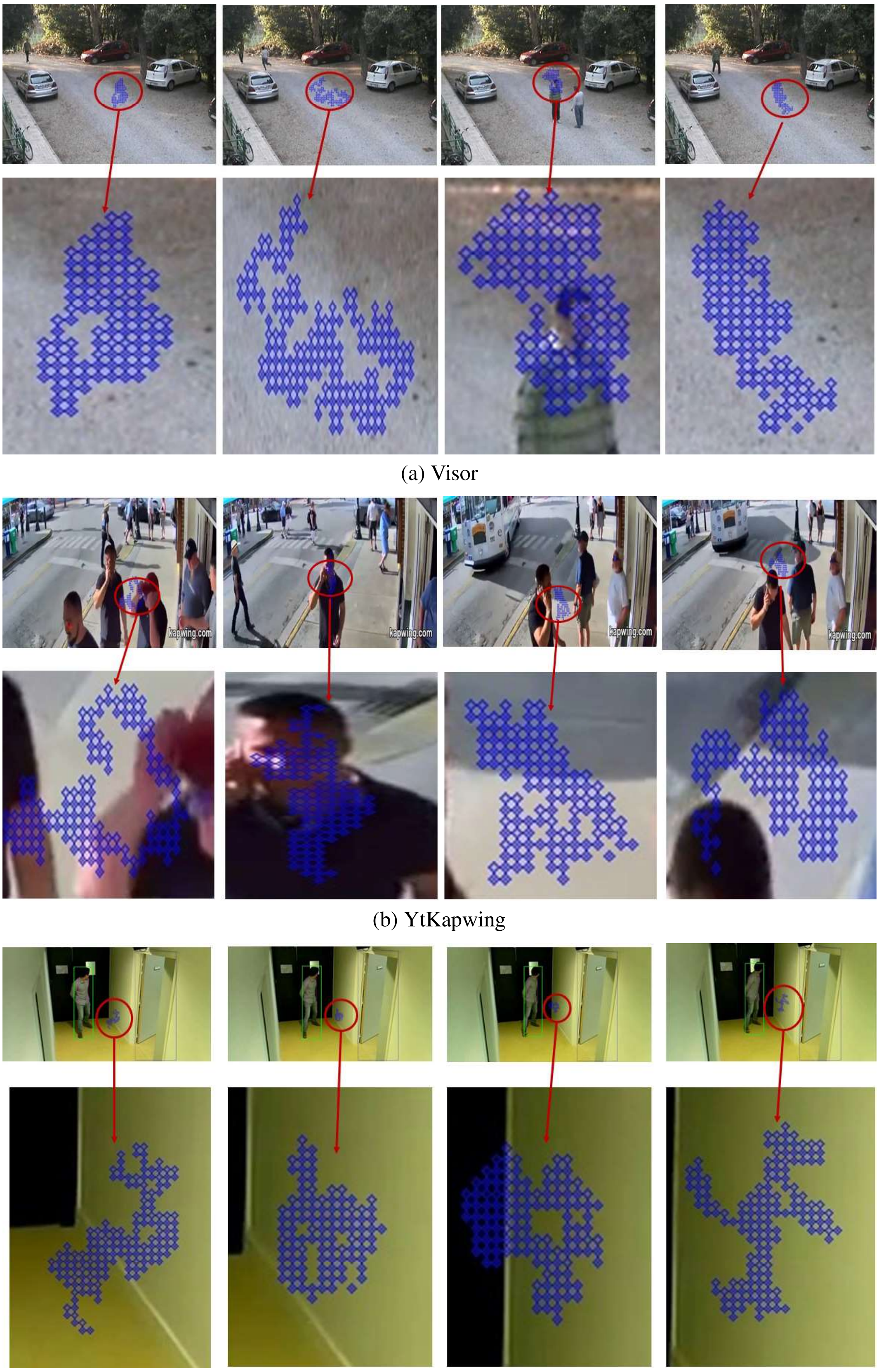
The retrieval procedure works as follows:

1) Retrieve the symmetric key (K_p) using its corresponding session id. from the hardware wallet for the video session. Then decrypt the video frames.
2) For each video frame, re-generate the current storage rectangle's offset value (see Fig. 2(d)) by means of a unique identifier of the video capture device.
3) Re-generate the seed pixel's position using the four corner pixels of the current frame's rectangle.
4) Retrieve the current frame's hash $C_\#$, changing its storage bits back to zero.
5) Recalculate the hash of the current frame, now containing the evidential data (in the form of hash $S_\#$) and the zeroized bits reset in the previous step. Call this new hash $C'_\#$.
6) Attempt a match between the $C'_\#$ with the corresponding hash, $C_\#$, retrieved from steganographic storage in the current frame. If there is a match, the video frame is proven, with high certitude according to the strength of the hash, to be authentic. In the case of a mismatch, the video data have been compromised and cannot be validated. If a frame cannot be validated leave the procedure, as a chain-of-evidence cannot be established.
7) Check whether the video session's frames have all been validated and if not return to step two.

IV. RESULTS

The proposed security system has been developed for resource-constrained devices and, therefore, has been tested on a Raspberry Pi-4 board with 4 GB RAM and a 64-GB Secure Digital (SD) memory-card for data storage. The Pi was loaded with Python and OpenCV modules to process the videos, along with the Crypto cipher module for AES encryption. The AES block size was 64-bits, with AES operating as a stream cipher in OFB mode. The OFB mode avoids the overhead of padding and, if USB storage errors occur, these are not necessarily propagated, allowing partial stream recovery. The key size was 128 bits rather than 256 bits, in order to reduce the computational overhead.

Pi camera videos were downloaded from a project's source pages [39], which were captured with object detection in mind. The videos were recorded in an indoor multi-space environment, using six Raspberry Pi 3 model B1s with the camera module v2.12 [38]. For performance evaluation, processing time and video quality metrics were determined for six videos mentioned above and compared with related methods as shown in Table 1. Previously proposed techniques demonstrated their steganography results on cover images and therefore, cannot be directly compared with videos. However, we have averaged the quality measures over 100 frames to be compared with these methods. Magic LSB uses a more complex method of converting an RGB image into an equivalent Hue, Saturation, Intensity (HSI) color space and then rotating the I component at four different angles. Each rotated I component of the image stores one block of data (divided into four blocks) and then an RGB image is reconstructed from the HSI image. Although complex and difficult to break, the scheme is most likely to be too costly for constrained devices. In fact, that computational overhead motivated the development of the proposed technique to modify LSB steganography. Similarly, LSB-M [18] first identifies the intensity clusters in the image and then hides data inside those clusters, again a time-complex process, which is not suitable for video steganography. Other techniques including

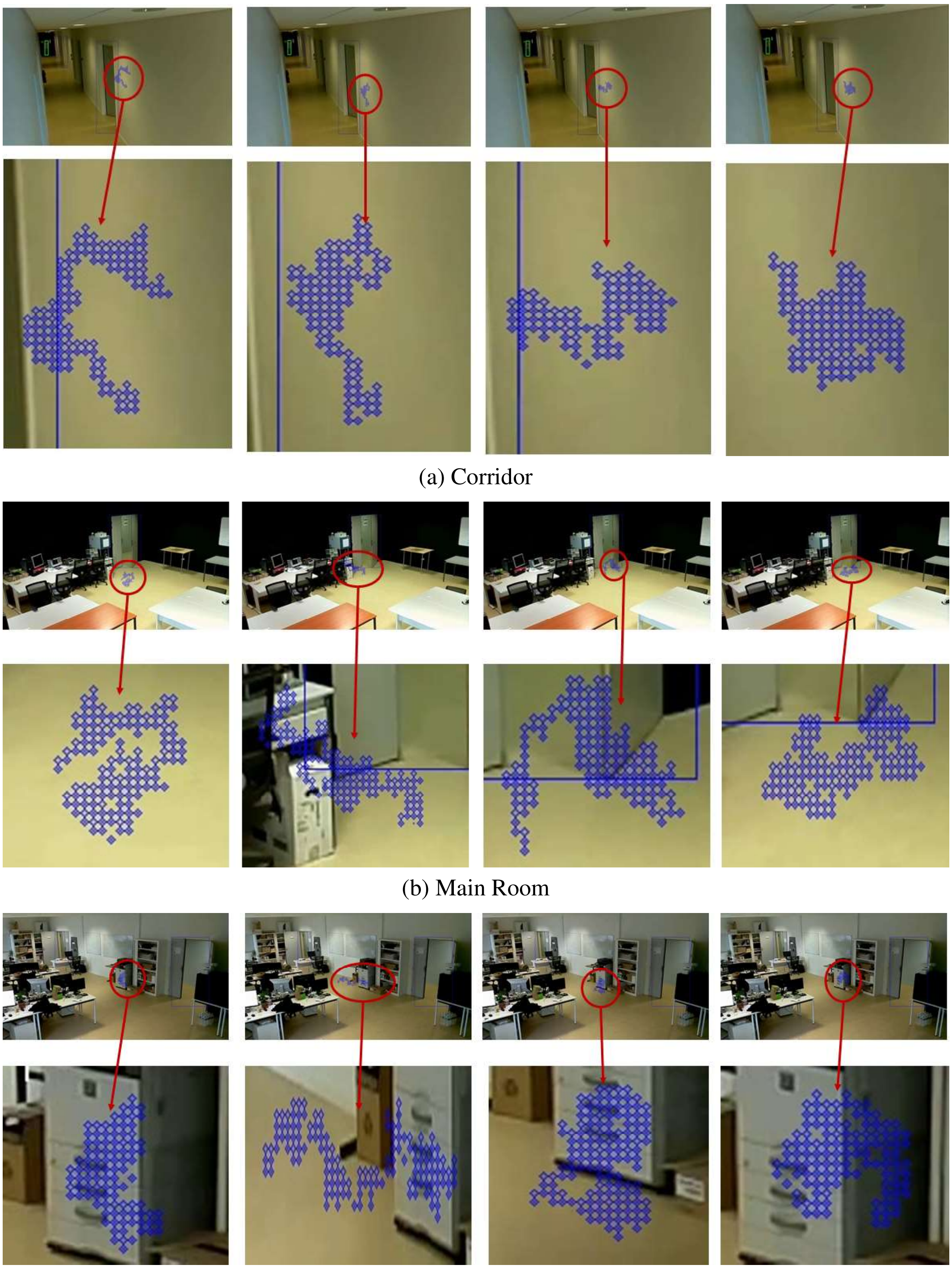


(a) Visor

(b) YtKapwing

(c) Entry

FIGURE 3: Random distribution of pixels selected to store chain-of-evidence using LSB Steganography in video imagery, with results for two different videos shown (see text). The blue markings of the location paths are only for illustrative purposes.



(a) Corridor

(b) Main Room

(c) PhD-room

FIGURE 4: Random distribution of pixels selected to store a chain-of-evidence using LSB Steganography in Pi captured videos [38], [39]. The blue markings of the location paths are only for illustrative purposes.

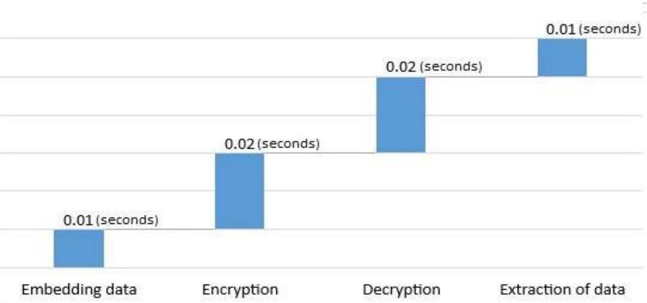


FIGURE 5: Averaged computational cost of the proposed algorithm over tested videos

a DWT-haar wavelet-based method [13], hash-LSB [17] and secret key LSB [14] suffer the same problems of not being able to run fast enough to match the video speed.

A. TIMINGS

The timings for the processing of a video stream across the four main security steps has been measured. The chain-of-evidence and steganographic processing consumes relatively less time compared to the encryption/decryption processes, i.e. 0.01 seconds and 0.02 seconds respectively (for 100 frames) using the Raspberry Pi-4 board as shown in Fig. 5. Overall, video processing on the Raspberry Pi-4, including steganography and encryption, with the previously given pixel resolution, supported a frame rate of 25 fps, which is acceptable for surveillance videos.

B. STEGANOGRAPHIC CONDITIONS AND VIDEO QUALITY METRICS

The effect of the storing of the path and data information in the original image on the quality of the image is typically represented by the Peak Signal to Noise Ratio (PSNR) [40]. Toward that end, first the Mean Square Error (MSE) is calculated which for a monochrome image is given by (4). For RGB images, such as those under consideration in this work, the expression for MSE changes to (5).

$$MSE_{Mono} = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{i=0}^{n-1}[I(i,j)-K(i,j)]^2 \tag{4}$$

$$MSE_{RGB} = \frac{1}{3mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\sum_{k=0}^{2}[I(i,j,k)-K(i,j,k)]^2 \tag{5}$$

Using the MSE_{RGB} from (5), the PSNR (in dB) is defined as:

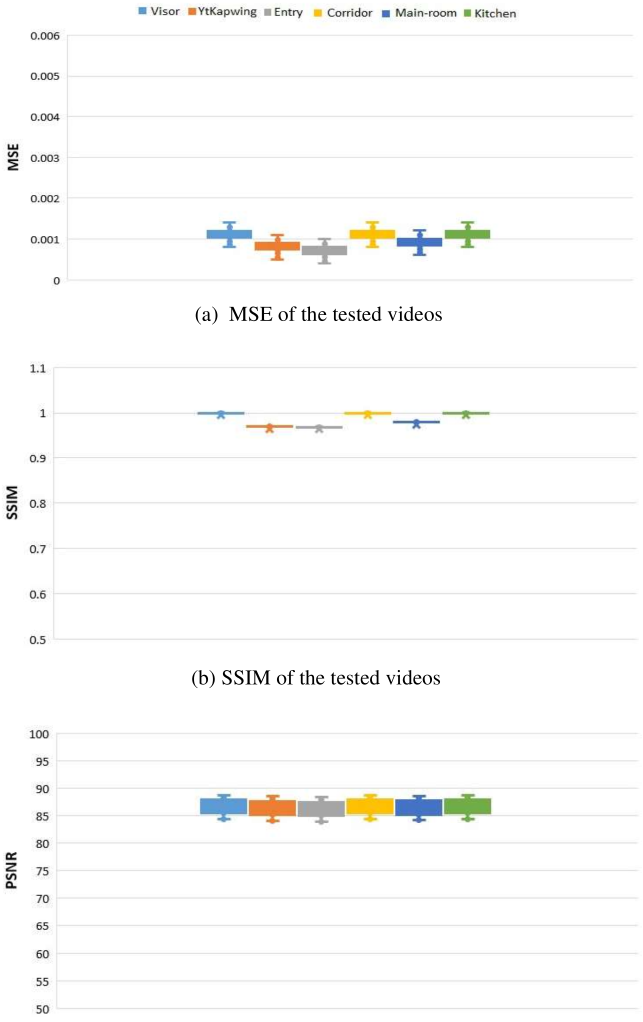
$$PSNR = 10 \cdot \log_{10}\left(\frac{Max_i^2}{MSE_{RGB}}\right)$$

which can be simplified to:

$$PSNR = 20 \cdot \log_{10}(Max_i) - 10 \cdot \log_{10}(MSE_{RGB}) \tag{6}$$

where, Max_i is the maximum possible pixel value of the image.

$$SSIM(i,j) = [l(i,j)^{\alpha}.c(i,j)^{\beta}.s(i,j)^{\gamma}] \tag{7}$$



(a) MSE of the tested videos

(b) SSIM of the tested videos

(c) PSNR of the tested videos

FIGURE 6: Video quality metrics for the Pi videos. These videos belongs to publicly-available datasets for object detection at [38]. The box and whisker plots show average measures for 200 frames per video.

Similarly, the Structural Similarity (SSIM) index [41] is a metric which estimates the structural similarity between original and reconstructed video frames, having a range generally from 0 to 1 (eq. 7). Values of SSIM nearer to 0 means less structural similarity between the plaintext and the reconstructed encrypted video frame, which means higher distortion has occurred. Values nearer to 1 means more structural similarity. The SSIM formula is based on three comparison measurements between the samples of i and j (two compared image windows n*n): luminance (l), contrast (c) and structure (s). SSIM is then computed as weighted combination of these three comparative measures. Where α, β, γ are set to 1. These video quality metrics have been used to evaluate the performance of proposed algorithm for standard steganographic conditions given below.

A reliable technique for steganography needs to fulfil the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3016211, IEEE Access

following conditions [42]:

- *Invisibility*: The data concealed in randomly selected pixels of each frame is not visible to the human eye because it is stored in the two lower bit planes of only 171 pixels and a human eye cannot normally identify such abnormalities at Pi-camera resolution. Thus, the Structural Similarity (SSIM) index (output range 0 to 1) is a well-established metric, which measures the response of the human visual system to changes between an original and modified image. For a surveillance video captured by us, the SSIM value is very high for the proposed method, as can be seen in Fig. 6b, implying that the original video and the video with data inserted are very similar. The average SSIM index was 1.00 using the proposed method, confirming that the hidden evidential data are visually unidentifiable.
- *Payload capacity*: The payload of the proposed method is 2 bits per color component per pixel and the data stored only uses 171 pixels of the whole image, which is particularly small. Although the payload of Magic LSB and LSB-M is also small with the use of only one bit per pixel, due to its complexity and processing requirements, it is unsuitable for resource-constrained devices.
- *Robustness against statistical attacks*: Common methods used to detect steganography are through taking the SSIM index, the Peak Signal-to-Noise Ratio (PSNR), and the Mean Squared Error (MSE). For these metrics, with the proposed method, the modified video is always more similar to the original video in the comparisons.
- *Non-suspicious files*: The file formats and file sizes remain the same during storage, whether there is embedded information or not.
- *Steganographic quality*: As previously mentioned, the presence of steganography is commonly detected by finding the changes in video quality, which is demonstrated in Fig. 6c and 6a. The average PSNR after steganography is around 88.0 dB, indicating high similarity (maximum PSNR = 96.33 dB for 16 bit depth image). This high value is attributed to the small amount of data embedded in the image (1026 bits), also resulting in an average MSE of 0.002, implying that it will be very difficult to detect the presence of steganography by such video metrics.

C. COMPARATIVE ANALYSIS

The proposed method has been compared with existing schemes published in the last decade or so for the above mentioned performance metrics and other prominent features as shown in Table 1. Almost all methods have been developed and tested for images and therefore, cannot be compared directly with our method, however, we took average PSNR, SSIM and MSE of 200 video frames to compare with these methods. The results depict that our method shown significantly better performance as compared to others with highest PSNR and lowest MSE value. Furthermore, SSIM of 1.00 also shows that the hidden data made no visual change in the image that can be identified by a naked eye.

Lastly, the proposed solution, including features from [5], has also been compared with some well-known products in the marketplace, when it can be seen that the solution is competitive with these products, as is evident from Table 2.

V. CONCLUSION

Provision of a chain-of-evidence in surveillance videos is not only a computer security application but a legal requirement for current surveillance systems, due to the ratification of the EU's GDPR and privacy laws in other countries. The proposed method provides a comprehensive solution by storing such evidence steganographically embedded alongside the video content, with overall encryption. Though relatively simple, the method is not only a GDPR protection-by-design aimed at surveillance video but is also capable of being implemented on resource-constrained devices such as the Raspberry-Pi processor. Moreover, the video data can be marked with unique identification at source, using salted hashes that can later serve to verify the originality of the shared video content, when they are exchanged within social networks. It needs to be pointed out that the integrity of the physical recording device (camera) is assumed to be guaranteed in the current paper. In fact, making a recording device physically tamper-proof is beyond the scope of this paper, though we pinpoint this as an issue for future research. Future work will also consider the best mechanism to generate each rectangle's location from the video capture device's unique identifier.

REFERENCES

[1] A. Brunetti, D. Buongiorno, G. F. Trotta, and V. Bevilacqua, "Computer vision and deep learning techniques for pedestrian detection and tracking: A survey," *Neurocomputing*, vol. 300, pp. 17–33, 2018.

[2] R. D. Singh and N. Aggarwal, "Video content authentication techniques: A comprehensive survey," *Mult. Syst.*, vol. 24, no. 2, pp. 211–240, 2018.

[3] K. Sitara and B. M. Mehtre, "Digital video tampering detection: An overview of passive techniques," *Digital Investigation*, vol. 18, pp. 8–22, 2016.

[4] D. A. Rodríguez-Silva, L. Adkinson-Orellana, F. Gonz'lez-Castano, I. Armino-Franco, and D. Gonz'lez-Martinez, "Video surveillance based on cloud storage," in *IEEE 5th Int. Conf. on Cloud Comput.*, 2012, pp. 991–992.

[5] M. N. Asghar, N. Kanwal, B. Lee, M. Fleury, M. Herbst, and Y. Qiao, "Visual surveillance within the EU General Data Protection Regulation: A technology perspective," *IEEE Access*, vol. 7, pp. 111 709–111 726, 2019.

[6] J. Rajamäki, "Design science research towards privacy by design in maritime surveillance ICT systems," *Info. & Security*, vol. 43, no. 2, pp. 196–214, 2019.

[7] S. Li, T. Qin, and G. Min, "Blockchain-based digital forensics investigation framework in the Internet of Things and social systems," *IEEE Trans. on Computational Social Syst.*, vol. 6, no. 6, pp. 1433–1441, 2019.

[8] S. Sun, "A new information hiding method based on improved BPCS steganography," *Advances in Multi.*, vol. 2015, 2015, article ID 698492.

[9] I. Cox, M. Miller, J. Bloom, J. Fridrich, and T. Kalker, *Digital watermarking and steganography*. Amsterdam: Morgan Kaufmann, 2007.

[10] R. Chandramouli and N. Memon, "Analysis of LSB based image steganography techniques," in *Proceedings 2001 International Conference on Image Processing*, vol. 3, 2001, pp. 1019–1022.

TABLE 1: Comparison of Quality metrics and the features of the proposed algorithm with prior art

Ref.	Year	Image/ Video	PSNR	SSIM	MSE	Embeded data	Encryption	Hidden Data
[13]	2011	Image	62.1	–	–	Wavelet coefficient	Filter bank cipher	Secret Message
[14]	2013	Image	56.72	0.99	–	RGB matrix green or blue components	XOR	Secret Message
[15]	2014	Image	74.39	–	0.023	RGB color compunent	AES	Secret Message
[16]	2016	Image	62.67	0.99	–	Achromatic component (I-plane) of HIS	Multilevel encryption (MLE)	Secret Message
[17]	2017	Image	63.00	–	0.030	RGB pixels	RC4 and Pixel Shuffling	Secret Message
[18]	2018	Image	68.90	1.00	0.027	RGB pixels + nearest neighbor clustering	AES	Secret Message
Our Method	2020	Video	μ PSNR of 200 video frames **88.77**	μ SSIM of 200 video frames **1.00**	μ MSE of 200 video frames **0.002**	RGB pixels	AES-OFB	Chain of Evidence

TABLE 2: Prominent features of surveillance products and the proposed solution

Features	Hikvision Smart Camera [43]	Swann Smart Video Analytics Camera [44]	Sony Intelligent Camera [45]	Proposed Solution
Object Detection	✓	✓	✓	✓
Motion Detection	✓	✓	✓	✓
Face Detection	✓	✓	✓	✓
Activity Detection	X	X	X	✓
Chain-of-Evidence	X	X	X	✓
Storage	Cloud	Cloud	Cloud	Edge

[11] W. S. Sari, E. H. Rachmawanto, C. A. Sari *et al.*, "A good performance OTP encryption image based on DCT-DWT steganography," *Telkomnika*, vol. 15, no. 4, pp. 1987–1995, 2017.

[12] M. Sutaone and M. Khandare, "Image based steganography using LSB insertion," in *IET Conference on Wireless, Mobile and Multimedia Networks*. Stevenage, UK: IET, 2008, pp. 146–151.

[13] S. M. Karim, M. S. Rahman, and M. I. Hossain, "A new approach for LSB based image steganography using secret key," in *14th International Conference on Computer and Information Technology (ICCIT 2011)*. IEEE, 2011, pp. 286–291.

[14] S. Saraireh, "A secure data communication system using cryptography and steganography," *International Journal of Computer Networks & Communications*, vol. 5, no. 3, p. 125, 2013.

[15] M. R. Islam, A. Siddiqa, M. P. Uddin, A. K. Mandal, and M. D. Hossain, "An efficient filtering based approach improving LSB image steganography using status bit along with AES cryptography," in *2014 International Conference on Informatics, Electronics & Vision (ICIEV)*. IEEE, 2014, pp. 1–6.

[16] K. Muhammad, M. Sajjad, I. Mehmood, S. Rho, and S. W. Baik, "A novel magic LSB substitution method (M-LSB-SM) using multi-level encryption and achromatic component of an image," *Mult. Tools and Apps.*, vol. 75, no. 22, pp. 14 867–14 893, 2016.

[17] M. H. Abood, "An efficient image cryptography using hash-LSB steganography with rc4 and pixel shuffling encryption algorithms," in *2017 Annual Conference on New Trends in Information & Communications Technology Applications (NTICT)*. IEEE, 2017, pp. 86–90.

[18] A. Shifa, M. S. Afgan, M. N. Asghar, M. Fleury, I. Memon, S. Abdullah, and N. Rasheed, "Joint crypto-stego scheme for enhanced image protection with nearest-centroid clustering," *IEEE Access*, vol. 6, pp. 16 189–16 206, 2018.

[19] S. Rajendran and M. Doraipandian, "Chaotic map based random image steganography using LSB technique." *IJ Network Security*, vol. 19, no. 4, pp. 593–598, 2017.

[20] N. Patel and S. Meena, "LSB based image steganography using dynamic key cryptography," in *IEEE Int. Conf. on Emerging Trends in Commun. Technol.*, 2016, pp. 1–5.

[21] X. Zhou, W. Gong, W. Fu, and L. Jin, "An improved method for LSB based color image steganography combined with cryptography," in *IEEE/ACIS 15th Int. Conf. on Computer and Info. Sci.*, 2016, pp. 1–4.

[22] K. Muhammad, J. Ahmad, N. u. Rehman, Z. Jan, and R. J. Qureshi, "A secure cyclic steganographic technique for color images using randomization," *Tech. J. (Taxila)*, vol. 19, no. 3, pp. 57–64, 2014.

[23] C. Paar and J. Pelzl, *Understanding cryptography: A textbook for students and practitioners*. Sci. & Business Media, 2009.

[24] N.-F. Standard, "Announcing the Advanced Encryption Standard (AES)," *Federal Information Processing Standards Publication*, vol. 197, no. 1-51, pp. 3–3, 2001.

[25] C. Saifurrab and S. Mirza, "AES algorithm using advance key implementation in MATLAB," *International Research Journal of Engineering and Technology*, vol. 3, no. 09, 2016.

[26] H. Gilbert and H. Handschuh, "Security analysis of SHA-256 and sisters," in *ACM Symp. on Applied Comput.*, 2003, pp. 175–193.

[27] Z. Tang, X. Zhang, X. Li, and S. Zhang, "Robust image hashing with ring partition and invariant vector distance," *IEEE Trans. on Forensics and Security*, vol. 11, no. 1, pp. 200–214, 2016.

[28] H. Rezaeighaleh and C. Zou, "New secure approach to backup cryptocurrency wallets," University of Central Florida, Tech. Rep., 2019.

[29] Y.-T. Tseng, P. Jurczyk, S. Watson, and M. Dalcin, "Merged video streaming, authorization, and metadata requests," Oct. 2 2018, US Patent 10,091,192.

[30] D. Stanescu, M. Stratulat, B. Ciubotaru, D. Chiciudean, R. Cioarga, and M. Micea, "Embedding data in video stream using steganography," in *24th Int. Symp. on Applied Computational Intelligence and Informatics*, 2007, pp. 241–244.

[31] N. Akhtar, S. Khan, and P. Johri, "An improved inverted LSB image steganography," in *IEEE Int. Conf. on Issues and Challenges in Intellig. Comput. Techniques*, 2014, pp. 749–755.

[32] "Visor1," https://aimagelab.ing.unimore.it/imagelab/ datasets.asp, accessed: 01-05-2020.

[33] "Dual street cam," https://www.youtube.com/watch?v=lwENY3mCdeg&list=PLAJCo2LcrwyQ0XOtOqdqI-HmGPzzjLfP0&index=7&t=0s, accessed: 01-05-2020.

[34] D. Jayasinghe, R. Ragel, J. A. Ambrose, A. Ignjatovic, and S. Parameswaran, "Advanced modes in AES: Are they safe from power analysis based side channel attacks?" in *2014 IEEE 32nd International Conference on Computer Design (ICCD)*. IEEE, 2014, pp. 173–180.

[35] C. Paar and J. Pelzl, "The Advanced Encryption Standard (AES)," in *Understanding cryptography*. Springer, 2010, pp. 87–121.

[36] S. M. Bellovin and R. Housley, "Guidelines for cryptographic key management," in *Symp. on Research in Security and Privacy*, 2005.

[37] J. S. Khan and J. Ahmad, "Chaos based efficient selective image encryption," *Multidimensional Systems and Signal Processing*, vol. 30, no. 2, pp. 943–961, 2019.

[38] R. Marroquin, J. Dubois, and C. Nicolle, "Method and system of securing wearable equipment," 2019, last accessed:July 8, 2020. [Online]. Available: https://doi.org/10.4121/uuid:c1fb5962-e939-4c51-bfd5-eac6f2935d44

[39] ——, "Wisenet: An indoor multi-camera multi-space dataset with contextual information and annotations for people detection and tracking," *Data in brief*, vol. 27, p. 104654, 2019.

[40] Q. Huynh-Thu and M. Ghanbari, "The accuracy of psnr in predicting video quality for different video scenes and frame rates," *Telecommunication Systems*, vol. 49, no. 1, pp. 35–48, 2012.

[41] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[42] M. Douglas, K. Bailey, M. Leeney, and K. Curran, "An overview of steganography techniques applied to the protection of biometric data," *Mult. Tools and Appls.*, vol. 77, no. 13, pp. 17 333–17 373, 2018.

[43] "Smart camera, Hikvision Digital Technology Co." https://www.hikvision.com/, accessed: 01-05-2020.

[44] "Smart video analytics camera, Swann Communications Pty. Ltd." https://www.swann.com/us/swwhd-intcam, accessed: 01-05-2020.

[45] "Intelligent camera, Sony Europe B. V." https://pro.sony/en_CL/products/ip-cameras/video-security-g6-intelligent-surveillance, accessed: 01-05-2020.