



# Using Blockchain for Improved Video Integrity Verification

Sarala Ghimire, Jae Young Choi , *Member, IEEE*, and Bumshik Lee , *Member, IEEE*

**Abstract**—A video record plays a crucial role in providing evidence for crime scenes or road accidents. However, the main problem with the video record is that it is often vulnerable to various video tampering attacks. Although visual evidence is required to conduct an integrity verification before investigations, it is still difficult for human vision to detect a forgery. In this paper, we propose a novel video integrity verification method (IVM) that takes advantage of a blockchain framework. The proposed method employs an effective blockchain model in centralized video data, by combining a hash-based message authentication code and elliptic curve cryptography to verify the integrity of a video. In our method, video content with a predetermined size (segments) is key-hashed in a real-time manner and stored in a chronologically chained fashion, thus establishing an irrefutable database. The verification process applies the same procedure to the video segment and generates a hash value that can be compared with the hash in the blockchain. The proposed IVM is implemented on a PC environment, as well as on an accident data recorder-embedded system for verification. The experimental results show that the proposed method has better detection capabilities and robustness toward various kinds of tampering, such as copy-move, insert, and delete, as compared to other state-of-the-art methods. An analysis based on execution time along with an increase in the number of blocks within the blockchain shows a minimal overhead in the proposed method.

**Index Terms**—Blockchain, video integrity, elliptic curve cryptography (ECC), HMAC.

## I. INTRODUCTION

IN RECENT years, advancements in technology have increasingly used surveillance cameras, such as in Closed Circuit Television (CCTV) systems or accident data recorder (ADR) systems, i.e., the so-called vehicle “black box”. As the video recorded by the surveillance cameras can provide critical visual information and act as a witness, it plays a key role in providing evidence in criminal investigations or dispute examinations [1]. However, the access to the video data is substantially easier owing to the openness of the networks [2]. The shared and open videos may incur duplication of videos that may infringe the copyright [3] or illegal distribution [4]. In addition, the

publicly-available media data can easily be manipulated or tampered with video editing tools utilized with malicious intentions, and leaving no visible clues [5]. In such a scenario, distinguishing an authentic video from a tampered one is a challenging problem, and may lead to misguided investigation proceedings. Therefore, the video integrity must be verified when using it as evidence.

Conventionally, the integrity and authenticity of the video can be certified through two types of approaches: active and passive.

The active approaches include embedding digital watermarking, fingerprints, and digital signatures, which require prior information such as hash values or signatures as a proof [6]. In contrast, the passive approach is known as a blind method of tampering detection without any reference to original contents, and is generally based on compression artifacts and noise residue in the manipulated video. As most of video cameras and surveillance systems use built-in video compression algorithms, video forgery is usually performed by decompressing the compressed video, tampering with it, and recompressing it again [7].

A number of passive video tampering detection schemes based on compression artifacts were proposed in [7]–[14]. For detection of double compression artifacts, [8] exploits the relations between video frame tampering (delete, copy, and move) and coding-type changes, e.g., in intra- and inter- coding, and uni- and bi-directional coding. In addition, the different influences of quantization on different frame coding types are qualitatively analyzed. However, if the number of deleted frames is equal to the group of picture (GOP) size, it is difficult to detect the tampering, because the detection of the tampering is highly dependent on the changes in the coding types. In [9], Markov modeling of a difference of discrete cosine transform (DCT) coefficients is used to detect double compression artifacts. The method has been limited to the use of scaling factors (quantization parameters) for first and second compressions. It is reported in [9] that the system fails to detect tampering if the scaling factor in the first compression is an odd multiple of the scaling factor of the second compression. The block-level correlation of noise residues is explored in [10]. The correlations between the noise residues of temporally neighboring blocks (i.e., blocks in the same position belonging to two adjacent frames) are evaluated. However, the method is likely to miss detection of a forgery if a calculation error of noise residues occurs. In [11] artifacts from the distribution of DCT coefficients of *I*-frames and prediction errors of *P*-frames are utilized to classify a video as a double-compressed video, and accordingly to detect the occurrence of forgery in the video. The principle of estimation

Manuscript received March 15, 2019; revised May 30, 2019; accepted June 22, 2019. Date of publication July 1, 2019; date of current version December 31, 2019. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Xin Geng. (*Corresponding author: Bumshik Lee.*)

S. Ghimire and B. Lee are with the Chosun University, Gwangju 61452, South Korea (e-mail: srlghm@chosun.kr; bslee@chosun.ac.kr).

J. Y. Choi is with the Hankuk University of Foreign Study, Yongin 17035, South Korea (e-mail: jychoi@hufs.ac.kr).

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TMM.2019.2925961

theory is used in [12] to detect double compression artifacts by double quantization. Each pixel of a given frame is estimated from a spatially co-located pixel of all the other frames in a GOP. The error between original and estimated pixel values is subjected to a threshold to identify the double-compressed frames in a GOP. In [13], a segmentation of background regions in each frame is obtained based on the motion vector field which is used to calculate the prediction residuals in each frame with adaptive parameters. The artifacts of double compression are detected by temporal periodic analysis of the feature sequence generated from the post-processed prediction residual. The detection method in [14] is based on a variation of macroblock prediction types in a re-encoded P-frame. Similarly, the compression noise characteristics in a frame are analyzed in [7], where the method fails to detect tampering if a quantization step size in the second compression is the same as the one used in the first compression. The aforementioned previous methods [7]–[14] are based on artifacts generated by the double compression. The main drawback of these methods is that the forgery detection rate is significantly reduced if parameters such as GOP size, quantization step size, or scaling factors are not identical between the first and second compressions.

Content-based video hashing methods in [15] and [16] ensures the authenticity of the video feature. However, they are more focused on video retrieval and identification. The work in [15] is used for near-duplicate video retrieval and video feature representation, whereas the method in [16] uses normalized vector representation of the video for identification of the video that includes the spatial resizing and temporal sampling of the content of the video. Moreover, such methods require sensitivity towards the content changing manipulations, but they must be robust against content-preserving manipulations, ignoring the non-malicious changes.

File system-based integrity verification methods (IVMs) are proposed in [1] and [17]. The method in [17] exploits video frames remaining in the slack space of media storage, whereas [1] utilizes the structure of the video content in a video event data recorder (VEDR), and identifies a difference in a frame index field between forged and original files. However, these methods may not work for all video file systems with different file structures, and is not promising for detection of integrity violations such as frame replacement, insertion, and image editing. Song *et al.* [18] proposed a video IVM based on the file structure of manipulated video content. Because the file structure generated by a video editing tool is stored as a signature in a database, it can only detect tampered files whose structures are stored in the database. Thus, a video manipulated by using a proprietary method instead of using a standard editing tool is not detected by the system.

In [19], chained hash and symmetric-key encryption are used to check for data forgery. Similarly, a storage integrity guaranteeing mechanism against tampering attempts (SIGMATA) using cryptographic algorithms is proposed in [20], where integrity assurance values of every frame are generated by the hashing algorithms MD5 [21], RIPEMD-128 [22], and SHA-1 [23], and are used for verifying the integrity. However, as the hash algorithms are applied for every individual frame, the encoding and

verification processes of these methods consume large amounts of time and memory.

Robert *et al.* [24] proposed a signature-based image authentication method for the JPEG image using blockchain in the decentralized network. The scheme utilizes the quantized coefficients of each  $8 \times 8$  block of the image as a signature, which is encrypted with Advanced Encryption Standard (AES) [25] encryption and stored in blockchain. However, the video consisting of large number of image frames requires significantly higher computational complexity for signature generation, compared to the case of using still images for this method. Moreover, it is required to parse the bitstream to obtain the quantized coefficients for signature generation, which also results in high complexity.

Furthermore, [26] and [27] developed an architecture of integrity verification method for ADR and analyzed the method with different configurations. The best method with hash-based message authentication code (HMAC) and elliptic curve cryptography (ECC) is recommended for the implementation.

To overcome the problems in the previously-developed methods, we propose an active video forgery detection scheme based on *blockchain* [28] that ensures the integrity of the video data against forgery attacks. Because the proposed method is categorized as an active approach, it generates an integrity (level) value for every video segment, and the integrity is validated based on such values. Thus, it is not necessary to find inconsistent periodic artifacts and noisy residues in the video for tampering detection in the proposed method, resulting in a more robust integrity verification against inherent manipulations or tampering.

The surveillance videos are recorded in small video segments with a few minutes' time intervals, such as one or five minutes. Considering the property that video surveillance systems generate and store each complete bitstream encoded by the video codec in every few minutes, the proposed method do not need to take the interdependencies [29] of video frame between *I*, *P*, and *B* frames into consideration. Furthermore, since we utilize the 'already compressed video segments' by any types of video codecs, there is no parsing process with complicated decoding for video integrity check to use the coding parameters such as motion vectors, quantization parameters, and quantized coefficients that can be obtained with parsing process. Instead, the proposed method generates hash values of the video segments after compression, resulting in high accuracy of forgery detection without parsing and interdependencies between video frames. In the proposed method, each encoded video segment is then hashed and chained with the hash of the previous segment. Thus, the integrity values (hash values) of video segments are connected each other in such a way that an unbreakable chain of hash is created. Moreover, a HMAC algorithm is used for hashing the video segment instead of a standalone hash function, which can increase the level of security. The HMAC is a shared-key cryptographic algorithm that is used to verify the data integrity and authenticity using an underlying hash function [30]. In addition, the segment-wise hashing increases the efficiency in terms of memory consumption.

The contributions of the proposed method are summarized as follows:

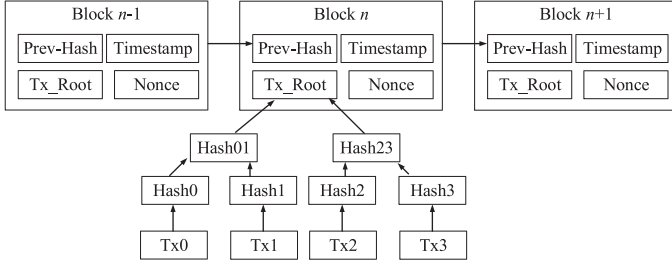


Fig. 1. General concept of blockchain.

- 1) A blockchain concept is introduced in the context of centralized video data. To the best of our knowledge, this is the first time that blockchain has been used for video integrity in centralized video data.
- 2) Our segment-wise integrity check can achieve faster processing and less memory consumption for video integrity verification.
- 3) A number of security issues and attacks are systematically analyzed, by which the IVM is verified with a high level of security.
- 4) The efficacy of our method is well-tested on a real-time ADR system as well as on a computer simulation, ensuring the applicability in real-world applications.

The rest of the paper is organized as follows. In Section II, we introduce the basic concept of blockchain and ECC. The proposed method is presented in Section III. The experimental results and analysis are demonstrated in Section IV and finally, the paper is concluded in Section V.

## II. BACKGROUND

### A. Blockchain

A blockchain is known as a distributed open ledger containing a block of transactions executed in a network, and is maintained by a node itself [31]. Fig. 1 shows a general concept of blockchain. A block is added to the chain, and the hash of the block is included in the next block. Thus, chronological chain of data guaranteed by the sequential nested blocks states that the data could not be changed without changing its block and the following blocks [32]. The individual block is recognized by the cryptographic hash on the header of the block, which is a hash of its parent block. Every block containing the hash of its parent links the sequence of blocks to create a chain [33]. The body part of the block contains batches of valid transactions that are hashed and encoded into a Merkle tree, as shown in Fig. 1, by  $Tx\_Root$ . In addition, a timestamp and a nonce value are added to the block, where the nonce is a random integer number that is repeatedly discovered until the hash of the block will contain a run of leading zeros that makes the block qualified to be added to the blockchain. As this iterative process requires time and resources, the calculation of correct nonce constitute proof of work in blockchain [34]. Hence, the integrity of blockchain is based on chained cryptography that makes it quite difficult to break.

The blockchain technology, which was originally designed for a financial ledger in a decentralization concept, can be extended to other frameworks and in other contexts, such as medical

data access and permission management [35]. However, a full-principle implementation of blockchain requires the system to be decentralized, and a proof of work algorithm to be implemented. In some cases, such as where there is lack of infrastructure, highly sensitive data, or low power devices, integrating a full-principle implementation can be an overwhelming problem.

In this study, we develop a novel way of incorporating the above-mentioned properties of the blockchain into a centralized database for video integrity verification systems. Moreover, the likelihood of an attacker being able to change the entire database system by recalculating the hash value is resolved by applying a keyed hash algorithm with a unique private key to individual blocks.

### B. Elliptic Curve Cryptography

Cryptography is the key for securing data from an intruder, and plays a major role in storing data securely. ECC [36] is a public key cryptography based on solving the elliptic curve discrete logarithm problem (ECDLP) [36]. It provides a high level of security with a smaller key size as compared to other cryptographic schemes [36]. An elliptic curve  $E$  over the finite field (or Galois Field)  $GF$  is defined by (1), known as the *Weierstrass* equation for elliptic curves in non-homogeneous form [37]:

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \quad (1)$$

where  $x$  and  $y$  are variables in affine coordinates of the elliptic curves and  $a_1, \dots, a_6$  are the set of elements in Galois field ( $GF$ ) [38], which is a field that contains set of finite number of elements and needs to have well defined operations of multiplication, addition, subtraction and divisions with certain basic rules [38]. The discriminant  $\Delta$  of  $E$  is not equal to zero ( $\Delta \neq 0$ ), and is calculated as in (2).

$$\Delta = -d_2^2d_8 - 8d_4^3 - 27d_6^2 + 9d_2d_4d_6 \quad (2)$$

where  $d_2, d_4, d_6$ , and  $d_8$  are calculated using the elements  $a_1, \dots, a_6$  from finite field  $GF$  as in (3) and (4).

$$d_2 = a_1^2 + 4a_2, \quad d_4 = 2a_4 + a_1a_3, \quad d_6 = a_3^2 + 4a_6 \quad (3)$$

and (4) can finally be obtained as:

$$d_8 = a_1^2a_6 + 4a_2a_6 - a_1a_3a_4 + a_2a_3^2 - a_4^2 \quad (4)$$

The condition  $\Delta \neq 0$  assures that the curve is non-singular, and that there are no curve points with two or more different tangent lines. The projective form of (1) of an elliptic curve  $E$  defined over  $GF$  is obtained by replacing  $x$  by  $X/Z^c$ , and  $y$  by  $Y/Z^d$ , and clearing denominators. Here  $X, Y$ , and  $Z$  are projective coordinates of elliptic curve and  $c$  and  $d$  are positive integer values. Thus, the homogeneous form of (1) can be written as:

$$Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3 \quad (5)$$

where  $X = xZ^c$ ,  $Y = yZ^d$ ,  $c = 1$ , and  $d = 1$ . This entails the existence of a special point, known as the identity element, that characterizes the elliptic curve. When the same point is added several times to itself in an elliptic curve, the addition operator is transformed into the scalar multiplication, which in practice



allows to multiply an elliptic curve point  $P$  by a positive integer  $n$  in order to produce another elliptic curve point,  $S = n \cdot P$ .

Moreover, for the ECC computation, two types of finite fields  $GF(q)$ , prime and binary finite fields, with  $q = p^m$  elements are used, where  $p$  is a prime number called the characteristic of finite field, and  $m$  is a positive integer. The prime finite field is denoted by  $GF(p)$  where  $p$  is any odd prime number with  $m = 1$ . The binary finite field which is also known as characteristic-two finite field  $GF(2^m)$  is a finite field with  $p = 2$  and  $m$  having any integer value greater than 1. By changing the variables for finite fields, the *Weierstrass* equation (1) can be simplified [39]. Thus, if the characteristics of the finite field is neither two nor three, the equation (1) can be reduced to the form (6):

$$y^2 = x^3 + ax + b \quad (6)$$

This curve equation is used in ECC, where the discriminant is defined as:

$$\Delta = -16(4a^3 + 27b^2) \quad (7)$$

Similarly,  $a$  and  $b$  are constants with the constraint of  $4a^3 + 27b^2 \neq 0$ . The coordinate points of an elliptic curve are used for a cryptographic operation [40]. The equation for the elliptic curve over a finite field is written as (8):

$$y^2 = \{x^3 + ax + b\} \bmod \{p\} \quad (8)$$

where *mod* is a modular operator. The set of parameters that defines the curve used in the ECC implementation depends on the underlying finite field, i.e., if the finite field is  $GF(p)$  then, the set of parameters are  $(p, a, b, G, n, h)$ . Here  $a$  and  $b$  are field elements that specify the equation of EC,  $p$  is a prime number that characterizes the finite field  $GF(p)$ ,  $n$  is the prime number whose value represents the order of the point  $G$  (i.e.,  $n \cdot G = O$ ; the point in infinity). Similarly,  $h$  is co-factor, and  $G$  is a point on the curve that is used as a generator of the points representing public keys. Some of the mathematical operations that are used for implementation of ECC are shown in Fig. 2.

If two parties  $A$  and  $B$  require secure communication between them, then they should agree upon a common elliptic curve equation and a generator  $G$ . Assuming that the private keys of  $A$  and  $B$  are denoted as  $K_A$  and  $K_B$ , respectively, then their public keys are derived from  $P_A = K_A G$  and  $P_B = K_B G$ . These public keys are distributed publicly. If  $A$  wants to communicate with  $B$  and wants to send a message, then  $A$  encrypts the message with the public key of  $B$  ( $P_B$ ) and generate the cipher text  $P_C$  as (9)

$$P_C = \{kG, P_m + kP_B\} \quad (9)$$

where  $k$  is any random number,  $P_m$  is a message to send and  $P_C$  is the cipher text. For the decryption of the message,  $B$  decrypts the message as (10).

$$P_m = \{P_m + kP_B - K_B kG\} \quad (10)$$

As  $k$  is a random number that is generated unique every time, a different cipher text is generated for the same message each time. This makes it difficult for anyone who tries to illegally decrypt the message. Thus, the message can only be decrypted with the agreed-upon secret key. Furthermore, the computational

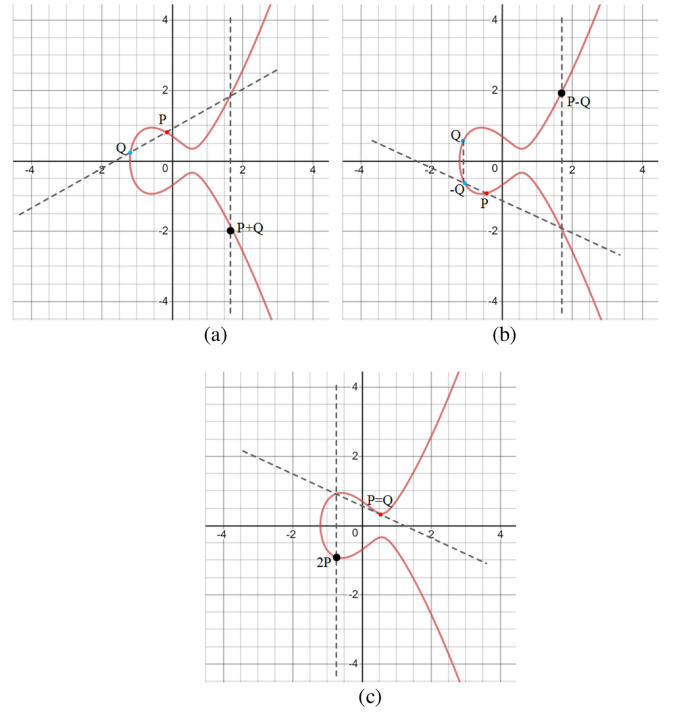


Fig. 2. (a) Point addition. (b) Point subtraction. (c) Point doubling.

complexity required to break the encryption of the ECC algorithm is very high, making it more secure than other asymmetric encryption algorithms [41]. Hence, to achieve a higher level of security, we employed the ECC encryption algorithm for key encryption in the proposed method. Moreover, a performance comparison between ECC and Rivest–Shamir–Adleman (RSA) is described in Section IV.

### III. PROPOSED METHOD

We propose a so-called video IVM based on a concept of a blockchain with ECC, storing the hash of video segments in a chronological order while recording videos captured using CCTV or ADR, etc. It is noted that videos in the surveillance video system are conventionally recorded in intervals of every few minutes. For example, the camera captures a scene every one minute for an ADR system, whereas CCTV usually records every few minutes according to the user's setting [42]. In our proposed method, the video segments are defined as the video clips recorded every few minutes.

Fig. 3 shows an overall framework of the proposed method for video integrity verification. As shown in Fig. 3, the proposed method can be divided into four components: a data hash section, key encryption section, block key generation section, and blockchain storage section. The data hash section generates a hash and message authentication code (MAC) with the HMAC algorithm [43] for each video segment. The video segments are hashed by a secure hash algorithm (SHA-256) [44], followed by the HMAC algorithm with a data HMAC key (denoted as  $dK$ ), ensuring the confidentiality and authenticity of the video. Moreover, the key  $dK$  is randomly generated, and has unique values for every video segment. It must be secure enough to

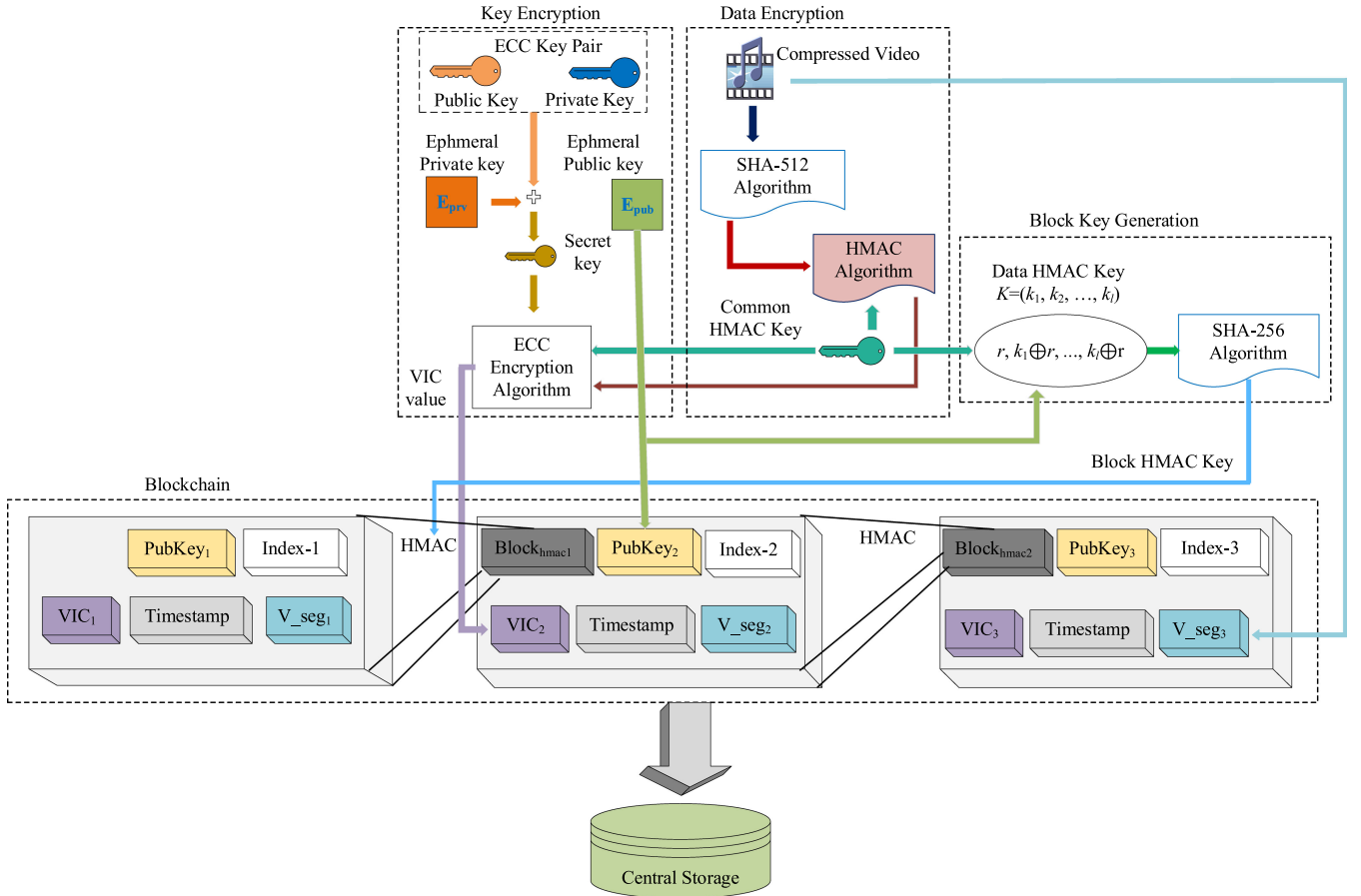


Fig. 3. A schematic block diagram of the proposed method.

keep the data safe. If the key is easily accessible to an attacker, one could tamper with the video and replace its original hash value without leaving any clue of modification. Nevertheless, the video integrity can still be valid, due to the lack of evidence of modification. The second component includes the key encryption section, where the key is encrypted using ECC. To make the process more secure and to increase the difficulty of tampering, the MAC of the video is encrypted along with the key  $dK$ . The output of this encryption is defined as the video integrity code (VIC) in the proposed method. As the number of keys increases with the number of video segments, resulting in an increase in memory consumption, an ECC that provides higher security with small key sizes [39] is used in our proposed method. For the key encryption process, we employ an elliptic curve integrated encryption scheme (ECIES), an extended form of ECC. The ECIES uses the elliptic curve Diffie-Hellman algorithm for the key agreement function, and a symmetric encryption algorithm for encryption of the key. In the block key generation component, the key for the HMAC algorithm used in block hashing is generated. A block HMAC key ( $bK$ ) is generated by applying a randomized hashing to the key  $dK$ . In the last component of the block diagram in Fig. 3, denoted as “Blockchain”, the outputs from the key and data encryption sections are stored in the block. The block is linked with prior and subsequent blocks as a blockchain. It is noted that in the proposed method, a single block contains the path of the video segment, the VIC value of

the current video segment, the ephemeral public key of ECC, a timestamp, and the HMAC of the previous block. Moreover, attaching a timestamp to any document or file to verify the integrity is well-known [45], and provides an authentic time and evidence of the file.

#### A. Key Encryption Using ECIES

Fig. 4 shows an overall pipeline of the ECIES. The ECIES is a hybrid public-key encryption scheme that is based on the elliptic curve equation, and its domain parameters over a finite field are described in Section II.B. The private key is randomly chosen, and is used to calculate the public key based on the same field and parameters as follows:

$$Kb = KvG \quad (11)$$

where  $Kb$  is a public key,  $Kv$  is a private key, and  $G$  is the generator of the points on the curve. The public key is distributed publicly, while the private key is kept secret. To derive an ECC key, the ephemeral public-private key pair is generated using the same agreed-upon curve parameters as depicted in Fig. 4. The ephemeral private key generated randomly is used to calculate the public key using (12):

$$k_{ue} = k_e G \quad (12)$$

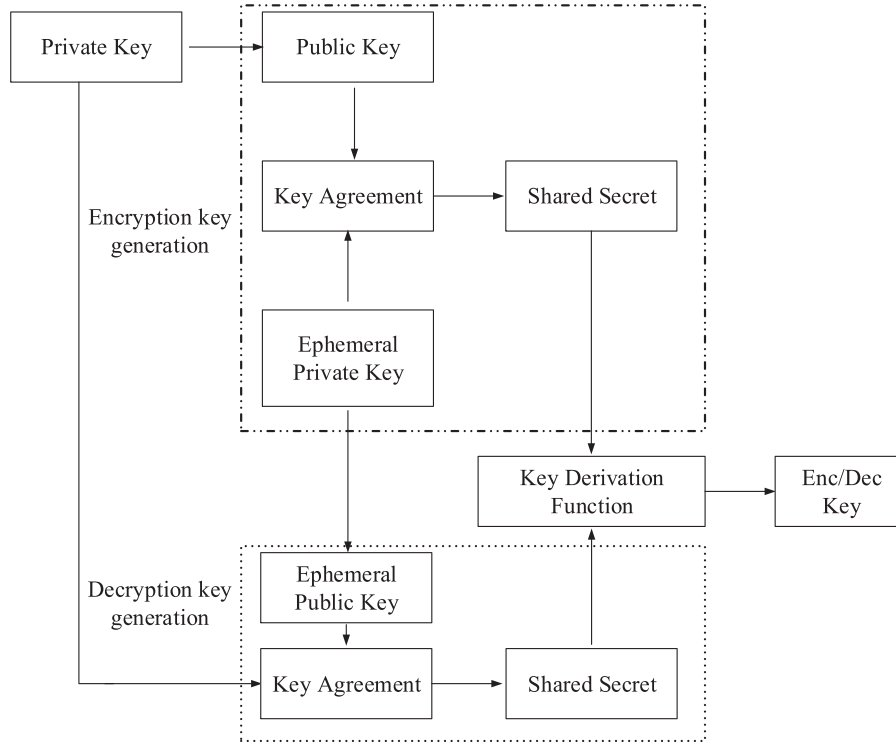


Fig. 4. Elliptic curve cryptography (ECC) functional diagram.

where  $k_e$  is an ephemeral private key,  $k_{ue}$  is an ephemeral public key, and  $G$  is the generator of the points on the curve. The secret key is derived by using the ephemeral private key and the main public key as defined in (13), and then the ephemeral private key is destroyed.

$$SK_E = k_e K b \quad (13)$$

A key derivation function based on the hash algorithm SHA-256 is applied to the secret key to generate the ECC encryption key, as in (14).

$$E_k = \text{Hash}(SK_E, \text{info}, i) \quad (14)$$

In (14),  $E_k$  is the output ECC key,  $\text{info}$  is an optional extra shared value, and  $i$  is the number of iterations of the hash function. The encryption key  $E_k$  is used to encrypt the key  $dK$  and the HMAC of the video  $H_m$  using the AES encryption algorithm, as:

$$\text{VIC} = \text{Encrypt}(E_k, H_m || dK) \quad (15)$$

where  $||$  denotes the concatenation operator, and VIC is the encryption output. In terms of the decryption side, the secret key is calculated by using the ephemeral public key and the main private key as:

$$SK_D = k_{ue} K v \quad (16)$$

Similarly, the ECC decryption key is obtained from a key derivation function similar to the encryption key generation.

$$D_k = \text{Hash}(SK_D, \text{info}, i) \quad (17)$$

In the overall process, the public key is open to everyone, and anyone can encrypt, whereas only an authorized person has the (static) private key and is able to decrypt. In addition, the ephemeral private key is randomly generated for the ECC key, and is subsequently destroyed immediately. A single public key can be used for more than one encryption. Hence, if multiple encryptions are required, the same public key is used without keeping it secret. This indicates that the computational cost is same regardless of the number of encryptions. In addition, it is more beneficial than the conventional symmetric schemes, in which the management of multiple keys on both encryption and decryption sides is required for multiple encryptions.

### B. Proposed Blockchain Generation

The general blockchain is defined as the chain of blocks that contains the following: transaction information, Merkle root, proof of work, timestamp, and reference to the hash of the previous block, which is publicly decentralized. The previous hash is the hash value of the previous block, and the Merkle root is used to verify transaction integrity. Similarly, the timestamp represents the time at which the block was generated, and the proof of work designates the difficulty of finding an eligible hash value. Based on this concept, the proposed blockchain-based scheme generates an unbreakable chain of video segments in chronological order in a real-time recording of centralized video data. The memory size of a block element in the proposed method is shown in Table I. The total memory size of a single block is 114 bytes.

In Table I,  $ID$  is the index of the block in the blockchain,  $Prev\text{-}HMAC$  is the HMAC value of the previous block, and VIC is the

TABLE I  
BLOCK STRUCTURE OF THE PROPOSED BLOCKCHAIN IN BYTES

ID	Prev-HMAC	VIC	Time-stamp	Pub-Key	v-fname
4	32	32	4	32	16

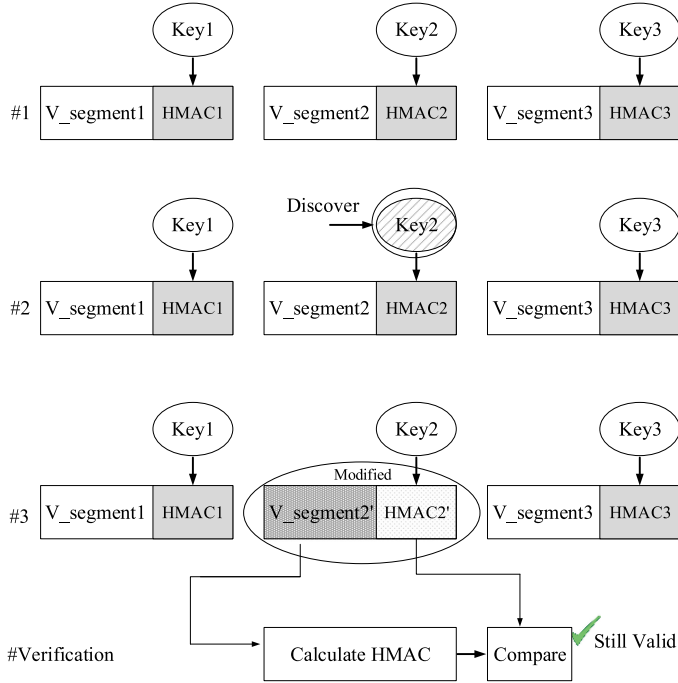


Fig. 5. A forgery and validation process in video segments without using blockchain in four steps. Key1, Key2, and Key3 are the keys for the hash-based message authentication code (HMAC) algorithm applied to the corresponding video segments V\_segment1, V\_segment2, and V\_segment3 to generate HMAC1, HMAC2, and HMAC3 as output, respectively. Key2 in step #2 (diagonal hatching) indicates the determination of the key by the attacker. V\_segment2' and HMAC2' in step #3 indicate the modification and replacement of the corresponding value with the original one.

value obtained by encrypting the key  $dK$  concatenated with the HMAC value of the current video segment. Similarly, timestamp indicates the time when the block was generated, and *Pub-Key* is the ephemeral public key. *v-fname* is the file name of the current video segment. The video segments captured from the camera are compressed and then hashed by a hashing algorithm named SHA-512, and the output is applied to an HMAC algorithm that produces the MAC value using  $dK$ . To protect the  $dK$  and data from the attacker, as well as to verify the integrity, the VIC value is generated by encrypting the key and HMAC value using ECC.

To show the effectiveness of the blockchain in the proposed method, two block diagrams are compared in Fig. 5 and Fig. 6. Note that only three video segments are considered in the illustrations, as a simple example. In the first step, three video segments are securely stored after applying hashing algorithms. Assuming that the second segment is suspicious to the attacker and Key2 is discovered in the second step, the data is modified and replaced in the following step. The integrity validation is performed in the final step.

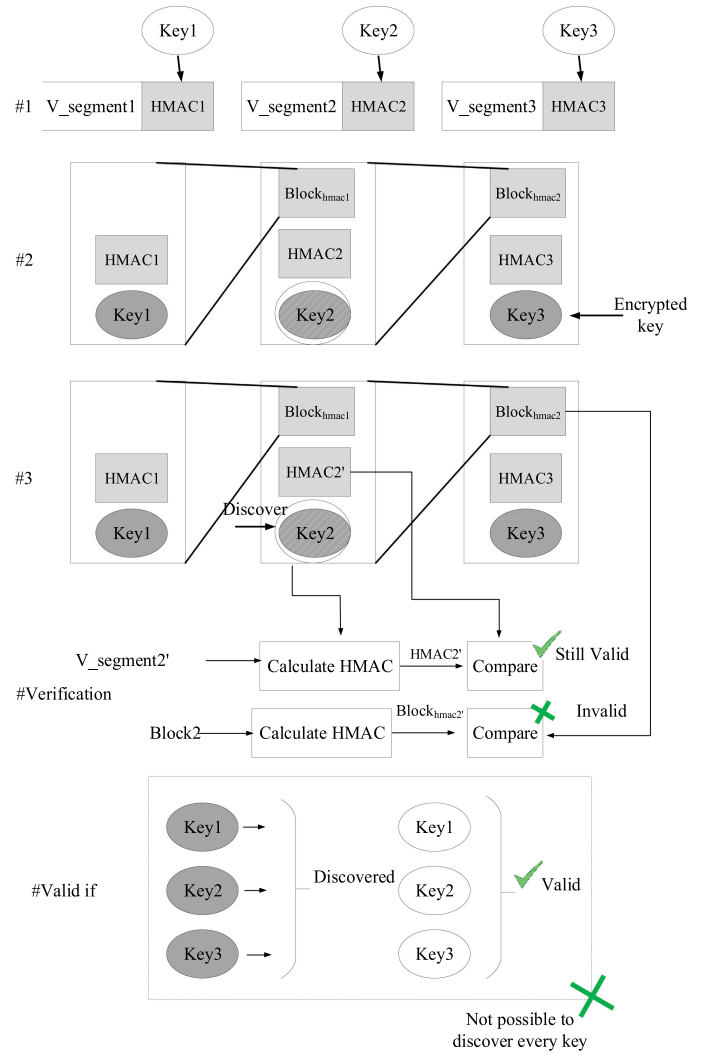


Fig. 6. A forgery and validation process in video segments using blockchain in four steps. Key1, Key2, and Key3 are the key for HMAC algorithm applied to the corresponding video segments V\_segment1, V\_segment2, and V\_segment3 that generate HMAC1, HMAC2 and HMAC3 as output, respectively. Every segment and their corresponding values are stored in the block, which are connected with each other by applying HMAC to the previous block, denoted as Block<sub>hmac1</sub> and Block<sub>hmac2</sub>. Key2 in the step #2 (diagonal hatching) indicates the determination of the key by the attacker. HMAC2' and V\_segment2' in the step #3 indicate the modification and replacement of the corresponding value with the original one.

Fig. 5 shows the forgery and validation process in video segments when the blockchain is not used for video integrity verification. If suspicious frames or images are found in a video segment, a suspect may try to discover a key for that segment. After discovering the key, the attacker can manipulate the video and generate an HMAC value for that segment. He then replaces the original video and its HMAC value with the manipulated ones without leaving any evidence of manipulation, as shown in Fig. 5. In contrast, a blockchain that creates a hash of a current block chained with a next block can identify and control this violation. Fig. 6 shows the process of forgery and validation in video segments when the blockchain is used.

As shown in Fig. 6, one can track the single key and change the single segment, but cannot track the entire key to change the corresponding sequence of the chain. Conversely, the single



modification is traced, and any alteration is invalid. Furthermore, as the proposed method is in a centralized architecture, if the key of any suspicious block is found, it is likely that the attacker changed the entire system, by recalculating the HMAC value for the block and replacing all the previous hash values within the block in a loop. This violation of integrity, i.e., by running a loop in the entire system, can be solved by calculating the HMAC value of the block.

Thus, the previous hash value for the blockchain of the proposed method is the HMAC value of the previous block. The key for the block operation  $bK$  is generated by applying randomized hashing to the key  $dK$ , and the ephemeral public key of ECC is used as a seed value. The randomized hashing on  $dK$  can protect the key  $bK$  from the security threat of a collision attack, to obtain the original key  $dK$  [46]. The dual use of these random values makes the system more feasible in memory-limited applications. In addition, the hash algorithm, as followed by the HMAC and encryption processes in the proposed method, increases the level of difficulty for the suspect to alter the block.

The overall process of blockchain generation is described as in the following steps:

**Hash** ( $V_{seg}$ )  $\rightarrow$  ( $H_v$ ), compute hash algorithm to the video segment,  $H_v = \text{Hash}(V_{seg})$

**HMAC** ( $dK, H_v$ )  $\rightarrow$  ( $H_m$ ). Compute HMAC to the hash value  $H_v$  as:

$$H_m = \text{HMAC}(dK, H_v)$$

**ECC Key pair**  $\rightarrow$  ( $Kb, Kv$ ):

Given an elliptic curve  $E$  over  $Z_n$  with a number of points that is divisible by the large prime  $n$ , choose a base point  $G$  on the curve and a random integer  $d \in [1, n-1]$ , where  $d$  is stored secretly and is known as the private key, i.e.,  $Kv = d$ .

The public key  $Kb = KvG$  is a point on the curve that is published publicly with other domain parameters  $E, G$ , and  $n$ .

**Encryption secret key** ( $Kb, k_e$ )  $\rightarrow SK_E$ : Select a random integer value  $k_e \in Z_n$  as an ephemeral private key, and compute  $k_{ue} = k_e G$  as an ephemeral public key.

Encryption secret key is  $SK_E = k_e Kb$

**Encryption key** ( $SK_E$ )  $\rightarrow E_k$ . Apply a key derivation function to the secret key to generate encryption key,  $E_k = \text{KDF}(SK_E)$ .

**Encrypt** ( $E_k, H_m || dK$ )  $\rightarrow$  VIC. The VIC value of the video segment can be generated as:

$$\text{VIC} = E(E_k, (H_m || dK))$$

**Generate Block Key** ( $k_{ue}, dK$ )  $\rightarrow bK$ .

The HMAC key for block is,  $bK = \text{Hash}_r(k_{ue}, dK)$ , where  $\text{Hash}_r$  is a randomized hash function.

**Block HMAC** ( $bK, (\text{VIC}_{i-1} || \text{timestamp}_{i-1} || \text{Index}_{i-1} || k_{ue,(i-1)} || V_{seg_{i-1}})) \rightarrow \text{Block}_{hmac}$

The previous block of the blockchain is hashed with HMAC using the block key  $bK$ . Here  $i-1$  indicates the index of the previous block.

$\text{Block}_{hmac} = \text{HMAC}(bK, (\text{VIC}_{i-1} || \text{timestamp}_{i-1} || \text{Index}_{i-1} || k_{ue,(i-1)} || V_{seg_{i-1}}))$

**Store on Blockchain** ( $\text{Block}_{hmac}, \text{VIC}, \text{timestamp}, \text{Index}, k_{ue}, V_{seg}$ )  $\rightarrow B_v \rightarrow$  blockchain, where note that

VIC, *timestamp*, *Index*,  $k_{ue}$ , and  $V_{seg}$  are the values of the current block.

### C. Video Integrity Check

The video integrity check starts with decrypting the key ( $dK$ ) that the HMAC algorithm used to hash the video segment submitted for integrity verification. Then, the integrity is checked by comparing the new HMAC value with the one stored in the blockchain. If the comparison produces a negative output, then it specifies the video segment as tampered. Otherwise, we again examine whether the modified hash value has been replaced or not. For this, the block with the current video segment is hashed by using the HMAC algorithm, and the generated value is compared with the previous HMAC value of the next block in the chain. A positive result indicates the video as untampered, while a negative result indicates a forgery, with replacement of the hash value with a modified one. The overall process of video integrity check is described in the following steps:

**Extract the corresponding block of the video segment submitted for integrity verification**

$$B_v \rightarrow (\text{Block}_{hmac}, \text{VIC}, \text{timestamp}, \text{Index}, k_{ue}, V_{seg})$$

**Decryption secret key** ( $Kv, k_{eu}$ )  $\rightarrow SK_D$ : The secretly stored private key  $Kv$  is extracted, and is used to generate the decryption secret key as:

$$SK_D = k_{eu} Kv$$

**Decryption key**  $SK_D \rightarrow D_k$ . Apply Key derivation function to the secret key,  $D_k = \text{KDF}(SK_D)$

**Decrypt** ( $D_k, \text{VIC}$ )  $\rightarrow (H_m || dK)$ . Decrypt VIC value of the video segment,

$$(H_m || dK) = D(D_k, \text{VIC})$$

$H_m$  and  $dK$  are split from the concatenated output.

**Extract video segment** ( $V_{seg}$ )  $\rightarrow V_{seg}$ .

The video segment is extracted from the path  $V_{seg}$  of the storage.

**Hash** ( $V_{seg}$ )  $\rightarrow (H'_v)$ , compute hash algorithm to the video segment,  $H'_v = \text{Hash}(V_{seg})$

**HMAC** ( $dK, H'_v$ )  $\rightarrow (H'_m)$ , compute HMAC to the hash value  $H'_v$ ,  $H'_m = \text{HMAC}(dK, H'_v)$

**Verify** ( $H_m, H'_m$ )  $\rightarrow ev$ .

If ( $H_m = H'_m$ )  $\rightarrow ev = 1$  else  $ev = 0$

Case 1: If ( $ev = 1$ )  $\rightarrow$  **Verify** ( $\text{Block}_{hmac,j}, \text{Block}'_{hmac,j}$ ).

Case 2: Else, the video segment is tampered.

Here  $j$  represents the index of next block in blockchain, i.e.,  $j = \text{Index} + 1$ , where  $\text{Index}$  is the index value of the current block.

**Verify** ( $\text{Block}_{hmac,j}, \text{Block}'_{hmac,j}$ )  $\rightarrow eb_j$ ,

- Decryption secret key** ( $Kv, k_{eu,j}$ )  $\rightarrow SK_{D,j}$ : The secretly stored private key  $Kv$  is extracted, and is used to generate the decryption secret key of the  $j$ -th block as:  $SK_{D,j} = k_{eu,j} Kv$ , where  $k_{eu,j}$  is an ephemeral public key of the  $j$ -th block.

- Decryption key** ( $SK_{D,j}$ )  $\rightarrow D_{k,j}$ . Decryption key of the  $j$ -th block,  $D_{k,j} = \text{KDF}(SK_{D,j})$



TABLE II  
VIDEO FILES USED IN THE DETECTION TEST

Video	Original frames	Tampering type	Tampered frames	Time length	Forgery detection
1	450	Copy-paste	450	15 s	Yes
2	900	Copy-paste	900	30 s	Yes
3	1797	Insert	1800	60 s	Yes
4	3593	Insert	3600	120 s	Yes
5	4867	Delete	4800	160 s	Yes

- **Decrypt** ( $D_{k,j}$ ,  $VIC_j$ )  $\rightarrow$  ( $H_{m,j}||dK_j$ ). Decrypt VIC of the  $j$ -th block, ( $H_{m,j}||dK_j$ ) = D( $D_{k,j}$ ,  $VIC_j$ ).  $H_{m,j}$  and  $dK_j$  are split from the concatenated output.
- **Block Key** ( $k_{ue,j}$ ,  $dK_j$ )  $\rightarrow$   $bK_j$   
The HMAC key for hashing previous block from  $j$ -th block  $bK_j = \text{Hash}_r(k_{ue,j}, dK_j)$
- **Block HMAC** ( $bK_j$ , ( $VIC||timestamp||index||k_{ue}||V\_seg$ ))  $\rightarrow$   $Block'_{hmac,j}$   
The current block of the blockchain is hashed with HMAC using the block key  $bK_j$ .  
 $Block'_{hmac,j} = \text{HMAC}(bK_j, (VIC||timestamp||index||k_{ue}||V\_seg))$
- **Verify**  
If ( $Block_{hmac,j} = Block'_{hmac,j}$ )  $\rightarrow$   $eb_j = 1$  else  $eb_j = 0$

Case 1: If  $eb_j = 0 \rightarrow$  Forgery with replacement of the HMAC value with modified one.

Case 2: Else, the video segment is untampered.

#### IV. EXPERIMENTAL RESULTS

We used five different publicly available video segments from [47]–[51] with  $1280 \times 720$  pixels to evaluate the proposed method. The frame rate is set to 30 frames per second (fps). To add forgery to the test videos, a commercial video editing tool, the AVS video editor [52], is used. We created tampered video sequences by randomly deleting, inserting, copying, and pasting frames in the video segments. The five test videos and their tampered sequences (with the type of tampering and number of frames) used for the experiments are listed in Table II. In addition, the secure hash algorithm-512 (SHA-512) is used for hashing, and SHA-256 is used in the calculation of the HMAC value. ECIES is used for the key encryption.

The experiments are performed on five different test videos, as tampered by different forgery types, as listed in Table II. The results show that the proposed method successfully detected the forgeries on all test videos, and also indicates that our method is robust regardless of any forgery type.

In Table III, we compare the proposed method with conventional IVMs [1], [17]–[20] based on verification capabilities, where we compare integrity verification capabilities among file system integrity-based approaches, video editing tools, file structure-based methods, cryptography-based schemes, and the proposed method. The methods in Kwon *et al.* [20] and Kim *et al.* [19] are independent of video file types and video editing tools for tampering, whereas the methods in Lee *et al.* [1]

TABLE III  
COMPARISON OF VARIOUS METHODS BASED ON INTEGRITY VERIFICATION CAPABILITIES

Method	File type dependent	Tool dependent	Image editing detection	Segment wise detection
Lee [1][17]	Yes	No	No	No
Song [18]	No	Yes	Yes	No
Kim [19]	No	No	Yes	No
Kwon [20]	No	No	Yes	No
Proposed	No	No	Yes	Yes

[17] and Song *et al.* [18] are dependent. Furthermore, Lee *et al.* [1], [17] cannot detect an image edit forgery. In contrast, the methods of Song *et al.* [18], Kim *et al.* [19], and Kwon *et al.* [20] and the proposed method are able to detect any kind of image editing forgery, such as copy-paste and copy-move. The proposed method is based on a video segment-wise IVM, which is more efficient in terms of memory consumption than other methods, where every individual frame must be processed to get the integrity value.

Three kinds of experimental analyses have been performed for verification of the proposed method: (a) ECC and RSA are compared in terms of running times and security levels; (b) performance comparisons with and without the blockchain are performed; and (c) security issues and attacks on the proposed method are analyzed in the security analysis section.

##### A. Analysis of ECC Over RSA

In Fig. 3, ECC can be replaceable with RSA in our proposed method. As described in Section II, asymmetric encryption algorithms that are based on integer factorization or discrete log problems including RSA or ElGamal are less secure than elliptic curve discrete log problems such as ECC. To ensure the effectiveness of the algorithms, we analyzed the performance of ECC and RSA in terms of execution time and security bit level, with varying sizes of the key. Fig. 7 shows the comparison of the total execution time (encryption and decryption time) for RSA and ECC with different test videos with different video sizes. The experiments are repeated ten times, and the average values are computed. It can be seen in Fig. 7 that the time consumptions by RSA are slightly higher than those by ECC for all video sizes.

Fig. 8 shows the security bit levels for cryptography key length between RSA and ECC. As shown in Fig. 8, RSA requires a larger key size as compared to ECC to achieve a similar security level. The difference of key sizes between ECC and RSA is significantly large over the security bit levels. Thus, for a given security level, the smaller key size of ECC makes the computation faster, with lower power and memory consumption [53]. The results in Fig. 7 and 8 indicate that ECC is, overall, more efficient and secure than RSA. Hence, we adopted ECC for encryption of the key in the proposed method.

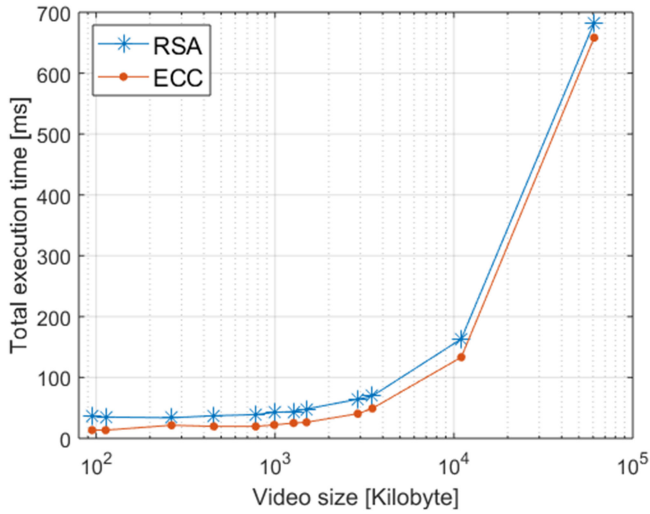


Fig. 7. Comparison of total execution time (encryption and decryption time) for Rivest–Shamir–Adleman (RSA) and ECC with different test videos of varying frame length.

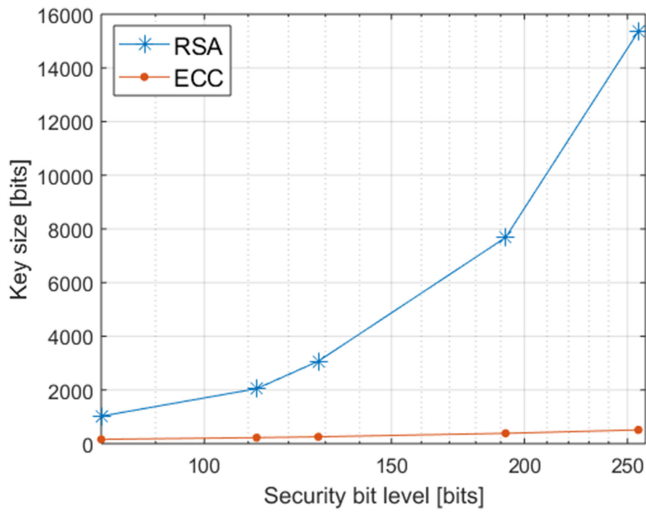


Fig. 8. Comparable security bit level for cryptography key length.

### B. Performance Evaluation of the Proposed Method in Terms of Running Time With Blockchain and Without Blockchain

First, we encoded the test videos using the H.264/AVC video codec by FFmpeg [54], and then the compressed stream is hashed and added in a blockchain. During verification, the same process of hashing is performed on the stored bitstream, and is compared with the value in a blockchain after decryption. The experiments are performed on a PC with an Intel Core i7-770K CPU @ 4.20 GHz, and 8 GB RAM. The time comparisons for hash value generation and verifications of videos having various lengths with and without blockchain are shown in Figs. 9, 10, and 11, respectively.

As shown in the figures, the average time for hashing and integrity verification by blockchain is 8 ms more than that without blockchain, which is very nominal. The time required for hashing a block and the key is the reason for this insignificant increase

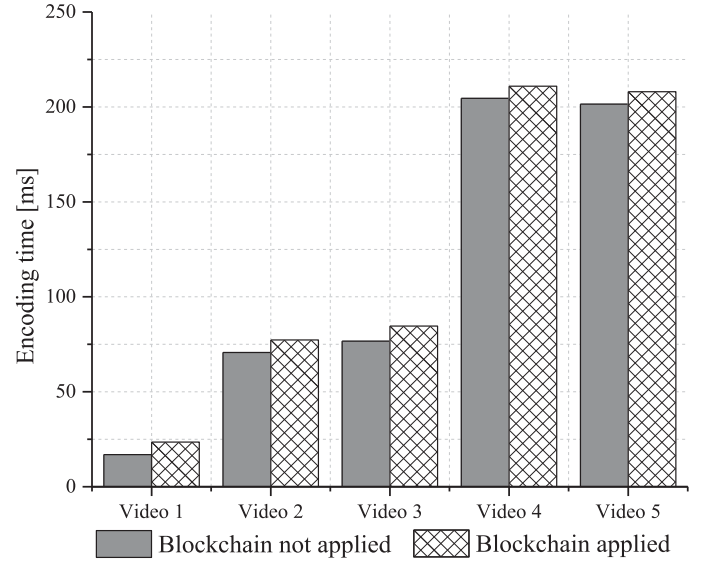


Fig. 9. Comparison of average encoding time for different test videos of varying frame lengths with and without blockchain.

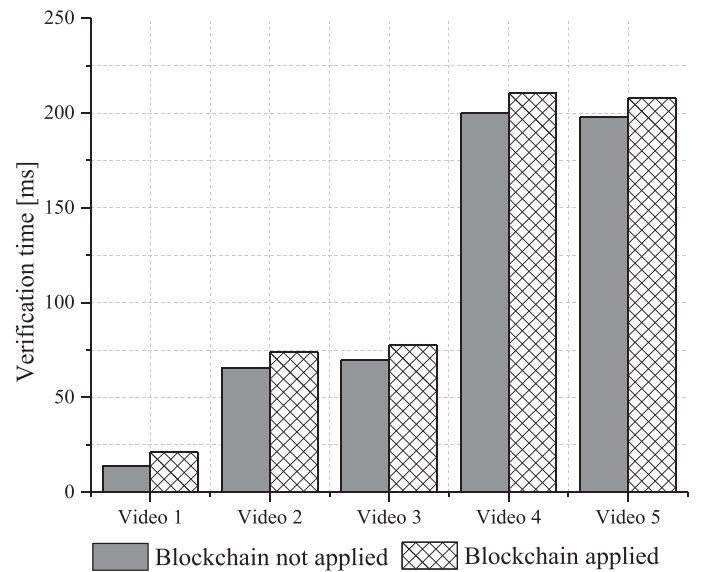


Fig. 10. Comparison of average verification time for different test videos of varying frame length with and without blockchain.

in execution time for both the processes. Similarly, to check the feasibility in real-world scenarios, we analyzed the proposed method based on execution time on an Ambarella board [55] of a 792-MHz ARM Cortex-A9 CPU, DDR3 / DDR3L with up to 600 MHz memory, which is one of the most widely used embedded board prototypes for the ADR made by Ambarella Inc. The ADR system records videos through the front windshield, the rear window, or inside the vehicle. The execution time for the encryption and verification processes for various length videos with and without blockchain as implemented on the Ambarella board is illustrated in Fig. 12.

The results in Fig. 12 show that the execution time for encryption and validation increases with the size of the video for

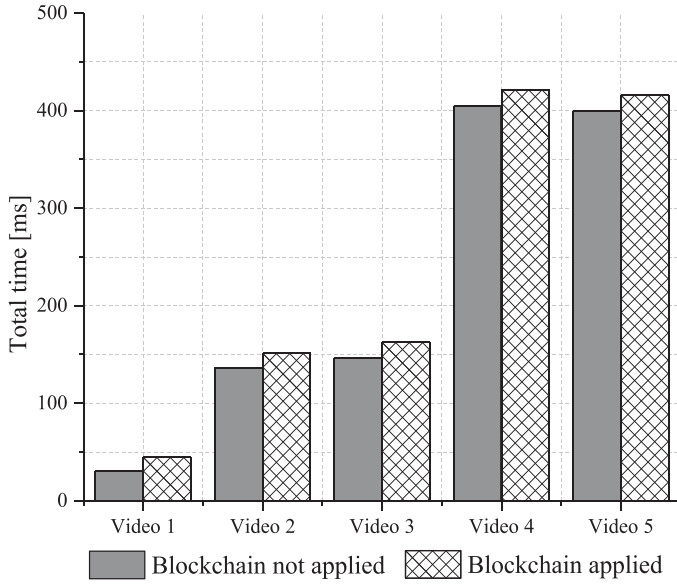


Fig. 11. Comparison of average time for different test videos of varying frame length with and without blockchain.

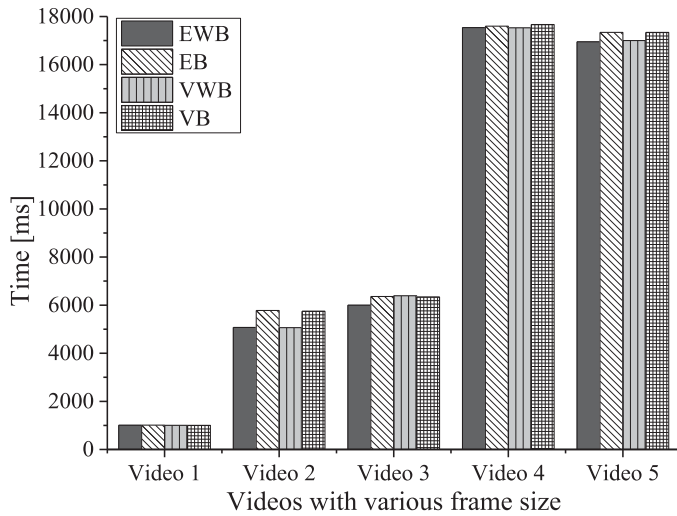


Fig. 12. Comparison of execution time for different test videos of varying frame length with and without blockchain on the Ambarella board. EWB: Encryption without blockchain, EB: Encryption with blockchain, VWB: Validation without blockchain, VB: Validation with blockchain.

both methods, with and without blockchain. A slightly longer time (encryption and decryption) is required for the method with blockchain as compared to the method without blockchain. The execution times with blockchain for all test videos slightly increase, by 0.5s on average. It can be concluded that the use of blockchain in the proposed method does not produce extra computation time. The execution times in Fig. 12 range from 1s to 20s, depending on the video sizes. Note that video 3, with 6s execution time, can be feasible in a real-world scenario by considering both execution time and video size.

The comparison of the physical memory size consumed by the method with and without blockchain is shown in Table IV. The comparison shows that the memory needed for our method

TABLE IV  
COMPARISON OF PHYSICAL MEMORY CONSUMED BY HASH OUTPUT OF ONE VIDEO SEGMENT WITH AND WITHOUT BLOCKCHAIN [UNIT: BYTES]

Output	With blockchain	Without
ID	4	0
Previous HMAC	32	0
VIC value	32	32
Ephemeral public key	32	32
Timestamp	4	0
Filename	16	0
Total	128	64

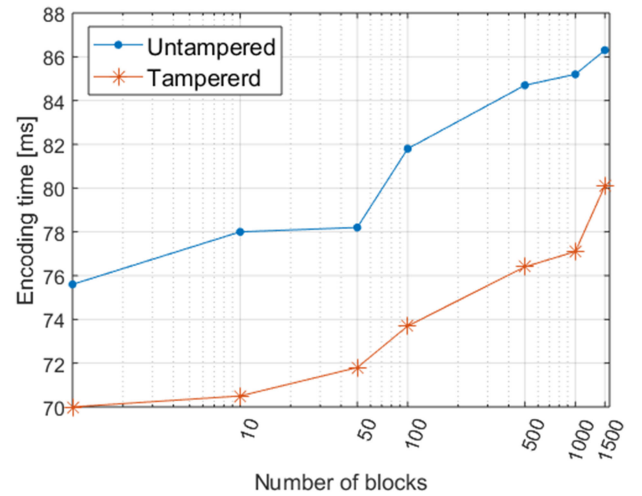


Fig. 13. Comparison of average encoding time along with change in the size of blockchain with tampered and untampered video segments.

with blockchain is twice that of the method without blockchain. Even though the size of the memory in the proposed method with blockchain is double, the method is more adequate because of the security level, as will be discussed in the security analysis of Section IV-C. The memory size can be a compromise with the level of security provided.

The time consumption for the verification process in the integrity verification scheme is an important factor to consider. To verify the significance of the proposed method with an increase in the size of the blockchain, we evaluated the time required for the scheme to perform with differently-sized blockchains for both tampered and untampered videos, as illustrated in Fig. 13. The result shows that the proposed method requires a very small increment of time with the increase in the number of blocks in the blockchain. In addition, the time deviation between the untampered and tampered video is also very marginal (at most 8ms), with the untampered video draining slightly more time. However, the case is identical if the video is modified and the hash value is replaced with a new modified hash; this time is almost equal to the time required to encode untampered video. For instance, the verification terminates if the first step of matching yields a negative result, and conversely, the second step is carried out if the result is positive. Thus, the multiple-step matching

procedure tends to increase the time consumption of the overall verification process.

### C. Security Analysis

As described in Section III-B, with the security acquired by the use of blockchain, the chained hash can trace an alteration in the video content if anybody is able to discover the key. Similarly, the violation of the video content and running of a hash function in a loop to remove the traces of modification are mitigated by applying the HMAC algorithm to every block. Furthermore, as the proposed method utilizes the HMAC algorithm for both the block and data, it is less affected by a collision attack than the underlying hashing algorithms alone [56], [57]. However, the system is mostly dependent on the key used in the HMAC algorithm. If it is revealed, the probability of an integrity violation is increased. Hence, the likelihood of an attack on the ECC key is the security threat, as the key is encrypted using the ECC algorithm. Consequently, we evaluate the legitimacy and probability of such an attack.

If we assume that the private key of the ECC is kept secret and not exposed, the use of a unique ephemeral private key (for every video segment) and the exponential time requirement (for solving the elliptic curve discrete logarithm problem) [58] makes the determination of the private key  $k$  from the public key  $kP$  very difficult. Moreover, the problem requires an extremely large amount of time to solve. The computational complexity to compute encryption and decryption can be used to measure the strength of any algorithm [41].

In RSA, with a fixed public exponent  $e$ , in computing encryption,  $M^e \bmod N$  has a time complexity of  $O(\log(N)^3)$ , where  $M$  is the input message and  $N$  is the product of two large prime numbers. In addition, to decrypt the cipher text  $C$  with the private exponent  $d$ ,  $C^d \bmod N$ , the complexity is  $O(\log(N)^3)$ . Moreover,  $d$  has a size in bits in proportion to that of  $N$ . However, given a point  $P$  on an elliptic curve  $C$  over a finite field  $F_q$  and integer  $x$ , the coordinates of the point  $xP$  can be computed in  $O((\log x) \times (\log q)^3)$  bits operations. Here,  $F_q$  is a finite field of  $q$  elements, where  $q$  is a very large prime power  $q = p^r$  for a prime  $p$  and large positive integer  $r$ . In addition, the point multiplication  $xP$  is the basic principle for encryption and decryption in ECC algorithm, thus the same complexity of  $O((\log x) \cdot (\log q)^3)$ -bit operations are required to compute ECC encryption and decryption. Conversely, a cryptographic algorithm is computationally secure if the cost to break the algorithm by a cryptanalyst needs ample time and power, i.e., the time complexity in terms of difficulty. The difficulty in solving an integer factorization problem like RSA is given by  $O(\sqrt{d})$ , where  $d$  is a private exponent [41]. This computational difficulty is identical for integer factorization problems and discrete logarithm problems, such as the ElGamal algorithm and digital signature algorithm (DSA). However, the difficulty of solving an elliptic curve discrete log problem is substantially larger than the difficulty of solving an integer factorization discrete log problem, owing to the large prime power  $q$ , given by the complexity of  $O(\sqrt{q})$  [41]. Thus, it would take more computational time to break the ECC key, which provides a high level of security.

Moreover, the proposed method incorporates ECC in blocks of the blockchain to strengthen the security. Considering  $n$  blocks of blockchain, the time complexity of the proposed method is  $O(n\sqrt{q})$ . However, an IVM without blockchain has  $O(\sqrt{q})$  complexity, which is equivalent to ECC. Hence, it shows that the introduction of blockchain in an IVM increases the computational complexity, thus resulting in higher security.

Furthermore, as the HMAC value of the block is stored in the blockchain without encryption, there is a likelihood of attack on the HMAC. The security of HMAC functions are dependent on the security strength of the embedded hash function [30], e.g., SHA-256 [59]. The probability of a successful attack on the HMAC value is equivalent to the probability of attacks on the embedded hash function.

1) *Brute-Force Attack*: For this attack, the attacker replaces an initialization vector (IV) of the compression function of the hash function with a secret random value. Moreover, the attack requires a brute force attack on the key. However, a level of effort on the order of  $2^n$  is required to find a possible key for one block. That means an effort of  $n$  times  $2^n$  is required to brute force all of the keys of the blockchain to alter the single block. Thus, this attack is infeasible.

2) *Collision Attack*: In this attack, the attacker tries to determine two messages  $M$  and  $M'$  that produce the same hash:  $H(M) = H(M')$ . To attack a standard function like SHA-256, the attacker requires a level of computations equal to  $2^{n/2}$ , where  $n$  is the size of the output in bits, which is  $2^{128}$  for SHA-256 of  $n = 256$ . As the attacker knows the value of IV and the algorithm, he/she can generate the hash code for the number of messages to find a collision. However, for HMAC, the attacker should know the key to generate a (message, code) pair. For this, the attacker must learn the sequence of the (message, code) pair generated under the same key. For a code length of 128 bits,  $2^{64}$  blocks generated from same key must be observed [59]. This may be possible with a dedicated computing facility working on all of the message/code pairs. However, the proposed method uses different keys for different video segments, making it difficult to learn the sequence of a message/code pair and consequently obtain the key.

Thus, it is infeasible to attack the HMAC value of the block in the blockchain.

Furthermore, as per the properties of the hashing algorithm, i.e., the one-way function [60], it is difficult to find the key  $dK$  if the key  $bK$  used in the block hashing is discovered. However, the key can be threatened by a hash collision attack. Despite this, the randomization of the key with the seed before hashing in the proposed method may help to mitigate such an attack.

## V. CONCLUSION AND FUTURE WORKS

In this paper, we proposed a novel method for video IVM based on blockchain in centralized video data. In the proposed method, the blockchain includes the HMAC and ECC algorithms for hashing and encrypting the video segments and keys, respectively. Moreover, the VIC generated from the encryption output is stored in a chronologically-chained manner. The integrity of the video segment is then validated by comparing the stored



integrity code with the newly-generated code of the video at the time of validation. The experimental results show that the proposed method is much more robust against tampering detection than other conventional methods equipped in a PC environment, as well as in an embedded system. In addition, the analysis on the computational complexity confirms that the proposed method yields a negligible increase in execution time, and at the same time guarantees a higher level of security as compared to the conventional methods, even if blockchain is additionally used as in our method. We also analyzed the performance of the system based on the size of the blockchain, i.e., the number of blocks contained in the blockchain. The results demonstrate that the proposed method produces very low overhead for an increase in the number of blocks, which justifies the applicability in real-world scenarios. Finally, the analysis of security shows that the proposed IVM is more robust against several attacks, and has time complexity of  $O(n\sqrt{q})$  that confirms a higher level of security.

For future work, it is desirable to investigate the robustness of the proposed blockchain-based integrity check especially when quite high-performance computing power such as the quantum computer is used as the attacker. In addition, it is also worthwhile to investigate whether the proposed method can be applicable to any type of multimedia data such as audios, images, and documents etc.

## REFERENCES

- [1] C. Lee, J. Lee, Y. Pyo, and H. Lee, "Broken integrity detection of video files in video event data recorders," *KSH Trans. Internet Inf. Syst.*, vol. 10, no. 8, pp. 3943–3957, 2016.
- [2] X. Nie, Y. Yin, J. Sun, J. Liu, and C. Cui, "Comprehensive feature-based robust video fingerprinting using tensor model," *IEEE Trans. Multimedia*, vol. 19, no. 4, pp. 785–796, Apr. 2017.
- [3] C. Chou, H. Chen, and S. Lee, "Pattern-based near-duplicate video retrieval and localization on web-scale videos," *IEEE Trans. Multimedia*, vol. 17, no. 3, pp. 382–395, Mar. 2015.
- [4] M. Asikuzzaman, M. J. Alam, A. J. Lambert, and M. R. Pickering, "Robust DT CWT-based DIBR 3D video watermarking using chrominance embedding," *IEEE Trans. Multimedia*, vol. 18, no. 9, pp. 1733–1748, Sep. 2016.
- [5] L. Yu *et al.*, "Exposing frame deletion by detecting abrupt changes in video streams," *Neurocomputing*, vol. 205, pp. 84–91, Sep. 2016.
- [6] O. Al-Sanjary, G. Sulong, and O. Ismael Al-sanjary, "Detection of video forgery: A review of literature," *J. Theor. Appl. Inf. Technol.*, vol. 74, no. 2, pp. 207–220, 2015.
- [7] H. Ravi, A. V. Subramanyam, G. Gupta, and B. A. Kumar, "Compression noise based video forgery detection," in *Proc. IEEE Int. Conf. Image Process.*, 2014, pp. 5352–5356.
- [8] Y. Su, W. Nie, and C. Zhang, "A frame tampering detection algorithm for MPEG videos," in *Proc. 6th IEEE Joint Int. Inf. Technol. Artif. Intell. Conf.*, 2011, vol. 2, pp. 461–464.
- [9] X. Jiang, W. Wang, T. Sun, Y. Q. Shi, and S. Wang, "Detection of double compression in MPEG-4 videos based on Markov statistics," *IEEE Signal Process. Lett.*, vol. 20, no. 5, pp. 447–450, May 2013.
- [10] C. C. Hsu, T. Y. Hung, C. W. Lin, and C. T. Hsu, "Video forgery detection using correlation of noise residue," in *Proc. IEEE 10th Workshop Multimedia Signal Process.*, 2008, pp. 170–174.
- [11] W. Wang and H. Farid, "Exposing digital forgeries in video by detecting double MPEG compression," in *Proc. 8th Workshop Multimedia Secur.*, New York, NY, USA, 2006, pp. 37–47.
- [12] A. V. Subramanyam and S. Emmanuel, "Pixel estimation based video forgery detection," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2013, pp. 3038–3042.
- [13] P. He, X. Jiang, T. Sun, and S. Wang, "Double compression detection based on local motion vector field analysis in static-background videos," *J. Vis. Commun. Image Representation*, vol. 35, pp. 55–66, 2016.
- [14] D. Vázquez-Padrón *et al.*, "Detection of video double encoding with GOP size estimation," in *Proc. IEEE Int. Workshop Inf. Forensics Secur.*, 2012, pp. 151–156.
- [15] J. Song, Y. Yang, Z. Huang, H. T. Shen, and J. Luo, "Effective multiple feature hashing for large-scale near-duplicate video retrieval," *IEEE Trans. Multimedia*, vol. 15, no. 8, pp. 1997–2008, Dec. 2013.
- [16] V. E. Liong, J. Lu, Y. Tan, and J. Zhou, "Deep video hashing," *IEEE Trans. Multimedia*, vol. 19, no. 6, pp. 1209–1219, Jun. 2017.
- [17] S. Lee, J. E. Song, W. Y. Lee, Y. W. Ko, and H. Lee, "Integrity verification scheme of video contents in surveillance cameras for digital forensic investigations," *IEICE Trans. Inf. Syst.*, vol. E98D, no. 1, pp. 95–97, 2015.
- [18] J. Song, K. Lee, W. Y. Lee, and H. Lee, "Integrity verification of the ordered data structures in manipulated video content," *Digit. Investigation*, vol. 18, pp. 1–7, 2016.
- [19] M. Kim and K. Kim, "Data forgery detection for vehicle black box," in *Proc. Int. Conf. Inf. Commun. Technol. Convergence*, 2014, pp. 636–637.
- [20] H. Kwon, S. Kim, and H. Lee, *Sigmata: Storage Integrity Guaranteeing Mechanism Against Tampering Attempts for Video Event Data Recorders*. Winter Garden, FL, USA: Int. Inst. Inform. Systemics, 2016.
- [21] Z. Yong-Xia and Z. Ge, "MD5 research," in *Proc. Int. Conf. MultiMedia Inf. Technol.*, 2010, vol. 2, pp. 271–273.
- [22] B. Preneel, H. Dobbertin, and A. Bosselaers, "The cryptographic hash function RIPEMD-160," *CryptoBytes*, vol. 3, no. 2, pp. 9–14, 1997.
- [23] P. E. Jones and D. Eastlake, "US Secure Hash Algorithm 1 (SHA-1)," 2001. [Online]. Available: <https://tools.ietf.org/html/rfc3174>
- [24] R. A. Dobre, R. O. Preda, C. C. Oprea, and I. Pirnog, "Authentication of JPEG images on the blockchain," in *Proc. Int. Conf. Control, Artif. Intell., Robot. Optim.*, 2018, pp. 211–215.
- [25] S. M. Soliman, B. Magdy, and M. A. A. E. Ghany, "Efficient implementation of the AES algorithm for security applications," in *Proc. 29th IEEE Int. Syst.-Chip Conf.*, 2016, pp. 206–210.
- [26] S. Ghimire and B. Lee, "An architecture of integrity check for accident video data recording system," in *Proc. KIIT Conf.*, Jun. 2018, pp. 419–422.
- [27] S. Ghimire and B. Lee, "Data integrity verification algorithms and performance evaluation for vehicle accident data recording system," in *Proc. KSAE Annu. Autumn Conf. Exhib.*, Nov. 2018, pp. 788–791.
- [28] F. X. Olleror and M. Zhegu, *Research Handbook on Digital Transformations*. Camberley, U.K.: Edward Elgar, 2016.
- [29] N. D. Doulamis, A. D. Doulamis, G. E. Konstantoulakis, and G. I. Stassinopoulos, "Efficient modeling of VBR MPEG-1 coded video sources," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, no. 1, pp. 93–112, Feb. 2000.
- [30] M. U. Arshad, A. Kundu, E. Bertino, A. Ghafoor, and C. Kundu, "Efficient and scalable integrity verification of data and query results for graph databases," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 5, pp. 866–879, May 2018.
- [31] M. Singh and S. Kim, "Introduce reward-based intelligent vehicles communication using blockchain," in *Proc. Int. SoC Des. Conf.*, 2017, pp. 15–16.
- [32] N. Z. Aitzhan and D. Svetinovic, "Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams," *IEEE Trans. Dependable Secure Comput.*, vol. 15, no. 5, pp. 840–852, Sep./Oct. 2018.
- [33] S. Singh and N. Singh, "Blockchain: Future of financial and cyber security," in *Proc. 2nd Int. Conf. Contemporary Comput. Inform.*, 2016, pp. 463–467.
- [34] A. S. Bruyn, *Blockchain: An Introduction*. Amsterdam, The Netherlands: Univ. Amsterdam, 2017.
- [35] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman, "MedRec: Using blockchain for medical data access and permission management," in *Proc. 2nd Int. Conf. Open Big Data*, 2016, pp. 25–30.
- [36] L. D. Singh and K. M. Singh, "Image encryption using elliptic curve cryptography," in *Proc. Comput. Sci.*, 2015, vol. 54, pp. 472–481.
- [37] D. Hankerson, A. Menezes, and S. V. Springer, *Guide to Elliptic Curve Cryptography*. New York, NY, USA: Springer, 2012.
- [38] L. E. Dickson, *Linear Groups: With an Exposition of the Galois Field Theory*. Chelmsford, MA, USA: Courier Corporation, 2003.
- [39] V. G. Martínez, L. H. Encinas, and C. S. Ávila, "A survey of the elliptic curve integrated encryption scheme," *J. Comput. Sci. Eng.*, vol. 2, no. 2, pp. 7–13, 2010.

- [40] L. D. Singh and K. M. Singh, "Implementation of text encryption using elliptic curve cryptography," *Procedia Comput. Sci.*, vol. 54, no. 1, pp. 73–82, 2015.
- [41] M. Calabresi, "An introduction to elliptic curve cryptography," Ohio State Univ., Columbus, OH, USA, p. 10, 2016.
- [42] "Taiwan digital time-lapse interval video recorder CCTV security camera-1," Forever Plus Corp., Taipei, Taiwan. [Online]. Available: <https://foreverplus.en.taiwantrade.com/product/digital-time-lapse-interval-video-recorder-cctv-security-camera-1-582444.html#>, Accessed on: Feb. 11, 2019.
- [43] M. Bellare, R. Canetti, and H. Krawczyk, "Keying hash functions for message authentication," in *Proc. Adv. Cryptol.*, 1996, pp. 1–15.
- [44] "Descriptions of SHA-256, SHA-384, and SHA-512" [Online]. Available: <https://web.archive.org/web/20130526224224/http://csrc.nist.gov/groups/STM/cavp/documents/shs/sha256-384-512.pdf>, Accessed on: Feb. 12, 2019.
- [45] B. Gipp, K. Jagrut, and C. Bretinger, "Securing video integrity using decentralized trusted timestamping on the blockchain," in *Proc. 10th Mediterranean Conf. Inf. Syst.*, 2016, vol. 26, pp. 3–17.
- [46] S. Halevi and H. Krawczyk, "Strengthening digital signatures via randomized hashing," in *Proc. Adv. Cryptol.*, 2006, pp. 41–59.
- [47] "Real yellow car—YouTube." [Online]. Available: <https://www.youtube.com/watch?v=o2YNaYcwdbA>, Accessed on: Feb. 12, 2019.
- [48] "Nature in 30 seconds —YouTube." [Online]. Available: <https://www.youtube.com/watch?v=MHna8CzxPLk>, Accessed on: Feb. 12, 2019.
- [49] "1 min of nature footage—4K (Ultra HD)—YouTube." [Online]. Available: <https://www.youtube.com/watch?v=WLKJnHu0GC4>, Accessed on: Feb. 12, 2019.
- [50] "4K video Ultra HD—Epic footage!—YouTube." [Online]. Available: <https://www.youtube.com/watch?v=od5nla42Jvc>, Accessed on: Feb. 12, 2019.
- [51] "029-Realistic beautiful flower painting timelapse by artistbrownlion—Satisfying video—2.5 Min—YouTube." [Online]. Available: <https://www.youtube.com/watch?v=2g8bS-nNYE&t=7s>, Accessed on: Feb. 12, 2019.
- [52] "AVS video editor—Easy video editing software for Windows." [Online]. Available: <https://www.avsyou.com/avs-video-editor.aspx>, Accessed on: Jan. 31, 2019.
- [53] Gemalto, "Benefits of elliptic curve cryptography," Gemalto, Amsterdam, The Netherlands, Mar. 2012.
- [54] FFmpeg. [Online]. Available: <http://www.ffmpeg.org/>, Accessed on: Jul. 30, 2018.
- [55] "Ambarella | embedded computer vision SoCs." [Online]. Available: <https://www.ambarella.com/>, Accessed on: Mar. 11, 2019.
- [56] H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: Keyed-hashing for message authentication," RFC Ed. RFC2104, Feb. 1997.
- [57] "SHA-1 broken—Schneier on security." [Online]. Available: [https://www.schneier.com/blog/archives/2005/02/sha1\\_broken.html](https://www.schneier.com/blog/archives/2005/02/sha1_broken.html), Accessed on: Jul. 27, 2018.
- [58] J. Bappaditya and P. Jayanta, "A performance analysis on elliptic curve cryptography in network security," in *Proc. Int. Conf. Comput., Elect. Commun. Eng.*, 2016, pp. 1–7.
- [59] W. Stallings, *Cryptography and Network Security Principles and Practices*, 4th ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 2005.
- [60] *NIST Special Publication 800-107 Revision 1, Recommendation for Applications Using Approved Hash Algorithms*, Nat. Inst. Standards Technol., Gaithersburg, MD, USA, 2012.



Sarala Ghimire received the B.E. degree in electronics and communication engineering and the M.E. degree in computer system and knowledge engineering from Tribhuvan University, Kirtipur, Nepal, in 2011 and 2017, respectively. She is currently working toward the M.S. degree in information and communication engineering with Chosun University, Gwangju, South Korea. From 2012 to 2017, she was a Senior Software Developer with XclusiveMinds, Pvt., Ltd., Kathmandu, Nepal, where she was involved in analysis and development of various information and financial management software. Her research interests include data analysis, image/video processing, and video security.



Jae Young Choi (M'12) received the B.S. degree from Kwangwoon University, Seoul, South Korea, in 2004, and the M.S. and Ph.D. degrees from the Korea Advanced Institute of Science and Technology, Daejeon, South Korea, in 2008 and 2011, respectively. In 2008, he was a Visiting Scholar with the University of Toronto, and from 2011 to 2012, he was a Postdoctoral Researcher with the University of Toronto. He was a Postdoctoral Fellow with the University of Pennsylvania from 2012 to 2013. He was a Senior Engineer with Samsung Electronics from 2013 to 2014. He is currently a tenure-track Associated Professor with the Hankuk University of Foreign Studies, Seoul, South Korea. He is the author or coauthor of more than 80 refereed research publications in the mentioned research areas. His research interests include pattern recognition, machine learning, deep learning, and computer vision. Especially, he has developed several pioneering algorithms for automatic face recognition using facial color information. He was a regular reviewer in many journals including the IEEE TRANSACTIONS ON IMAGE PROCESSING, the IEEE TRANSACTIONS ON MULTIMEDIA, and the IEEE SIGNAL PROCESSING LETTER. He was the recipient of the Samsung HumanTech Thesis Prize in 2010.



Bumshik Lee (M'07) received the B.S. degree in electrical engineering from Korea University, Seoul, South Korea, in 2000, and the M.S. and Ph.D. degrees in information and communications engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2006 and 2012, respectively. He was a Postdoctoral Scholar with the University of California, San Diego (UCSD), San Diego, CA, USA, from 2012 to 2013, and a Research Professor with KAIST in 2014. He was a Principal Engineer with Advanced Standard R&D Lab., LG Electronics, Seoul, from 2015 to 2016. He is currently an Assistant Professor with the Department of Information and Communications Engineering, Chosun University, Gwangju, South Korea. His research interests include video compression, video processing, video security, medical image processing, etc.