Received: 28 August 2020 | Revised: 17 November 2020 | Accepted: 28 November 2020 | Published on: 28 December 2020

DOI: 10.1002/spy2.143

RESEARCH ARTICLE

WILEY

Blockchain-based video forensics and integrity verification framework for wireless Internet-of-Things devices

Suat Mercan[1] | **Mumin Cebe**[2] | **Ramazan S. Aygun**[3] | **Kemal Akkaya**[1] | **Elijah Toussaint**[1] | **Dominik Danko**[4]

[1]Department of Electrical and Computer Engineering, Florida International University, Miami, Florida,

[2]Department of Computer Science, Marquette University, Milwaukee, Wisconsin,

[3]Department of Computer Science, Kennesaw State University, Marietta, Georgia,

[4]Department of Math and Computer Science, Clark University, Worcester, Massachusetts,

Correspondence
Suat Mercan, Department of Electrical and Computer Engineering, Florida International University, Miami, FL.
Email: smercan@fiu.edu

Abstract

A camera footage which is essential for forensic investigations can easily be modified with advanced video tampering techniques. This makes it necessary to employ novel methods to retain and prove the integrity of captured scene in criminal investigations. In this vein, blockchain technology has received a substantial interest in the last decade as it provides trust among users without a trusted third party, which enabled a myriad of applications. To this end, we propose a framework that utilizes blockchain technology to verify integrity of a camera footage recorded by a resource-constrained wireless Internet of Things (IoT) device. The proposed approach computes the hash of the video data before it leaves the IoT device to ensure the integrity. The hash is then stored on a permissioned blockchain platform that enables detection of tampering in the video. The continuous stream is segmented efficiently to have periodic hash value to minimize the risk of video loss in case of device failure. The system has been implemented on a Raspberry Pi and Hyperledger to validate its efficiency. We are able to process high resolution videos on a resource-constrained with reasonable amount of delay. The integrity of recorded video is successfully verified by using the digest kept in permissioned blockchain.

KEYWORDS

blockchain, digital forensics, Hyperledger, IoT device, video integrity

1 | INTRODUCTION

With the widespread use of Internet of Things (IoT) devices, capturing information from our environments turned into a common practice. In particular, visual information can be easily delivered through numerous available cameras with wireless communication capabilities. Indeed, a large number of wireless cameras are increasingly being utilized on buildings and streets for surveillance and in many smart city applications.[1] In addition, wireless cameras and drones are also being used by law enforcement while responding to a call or pursuing crimes.[2,3]

In these applications, video scene provides significant visual information to analyze and interrogate a crime and resolve any dispute about an incident because it has a spatio-temporal context that reveals detailed information about

A very preliminary version of this paper appeared in the Proceedings of 16th International Conference on Mobile Ad Hoc and Sensor Systems Workshops (MASSW), San Diego, 2019.

the case.[4] Since description of visual context by viewers could be biased, miss important details, and sometimes the order of information provided may not be correct, availability of visual information would clarify many ambiguities or vague information provided by witnesses. As video data are to be used as a type of evidence in some scenarios, the authenticity of video information is critical to make decisions and reach conclusions. If the video is captured by a personal camera or a smart device, the verification of visual information is achieved by the testimony of the person who recorded the video.[5] However, for videos that are automatically and continuously recorded by IoT devices, testimonials would not work because there is no actual person actively recording the scene. In fact, for such cases, the problem becomes very challenging since video forgery techniques are now very sophisticated that even a typical user can change a video scene by using off-the-shelf open source video editing tools that are supported by machine learning. As an example, available video editing tools can easily destroy the trust in visual information obtained by video capturing since some scenes may be deleted, new scenes can be inserted and furthermore, even insertion and deletion of objects are possible. The manipulations are so sophisticated that it may almost be impossible to determine whether the video has been edited or not[6] without seeing the original video. In addition to manipulating stored video data, the wireless channels could be another source for data tampering[7] since the videos may be altered while being transmitted through the wireless communication channels.

For a video to be admissible in a trial, it must satisfy several requirements. In addition to (a) relevancy and (b) legal acquisition, (c) chain of custody is highly important to ensure the authenticity which means that the evidence must be maintained in accordance with the conditions it was acquired.[8] Thus, three phases; acquisition, transfer, and storage should be handled in a secure manner to establish authenticity which implies integrity and non-repudiation. In order to prove the authenticity in such cases, various methods have been proposed. For instance, as digital piracy has been an issue for a long time, some of the techniques that are used to fight against piracy such as watermarking[9] are used by embedding a hidden signature into the video. This signature can be analyzed whether the video has been tampered or not. But, IoT-based security cameras are not equipped with this capability, and watermarks are prone to various transformational attacks such as scaling, shearing[10] as well as removal, cryptographic, geometric, and protocol attacks.[11]

Another method for integrity verification is hashing where a hash is computed and delivered with the video. This is a simple and cost-efficient technique as long as the used hash functions are collision-resistant. However, there is also a need to maintain the integrity of the hash from the time it leaves the camera to its storage in a server. To this end, we advocate the use of blockchain in this paper. Blockchain has emerged as an alternative method to transfer money between non-trusting participants without having a trusted third party such as a bank.[12] The idea of using a distributed ledger has surpassed its original goal and led to many applications as being considered as a game changer by some people. In this regard, blockchain can also be utilized to ensure the authenticity of information by storing data in a distributed and retroactively unmodifiable manner which will ensure the integrity of data.

In this paper, we present a blockchain-based forensic framework to verify the integrity and authenticity of videos captured by wireless-based IoT devices where this video data might be used as an evidence in court. It is important to note that we adopt an existing blockchain platform, Hyperledger, to build the system as it is without changing the internal mechanism of the blockchain. We verify the integrity of the data outside by using the hash stored in the Hyperledger. Specifically, we consider drone application cases where they are capable of streaming in many surveillance applications that can record a crime or accident scene to be used as future forensic evidence. The framework includes authenticating the source, maintaining the integrity of video data by getting the hash, and storing in blockchain and being able to receive the original video that came out of the drone in a lossless way while providing real-time streaming.

More specifically, first, our framework performs the hash computation on the video data on the device where it is captured without sending to an edge device or servers. This will mitigate the risk of the hash being tampered during or after transmission. Furthermore, we segment the video stream periodically to reduce the risk of losing the whole footage due to any potential problem on the drone or a glitch in another part of the video. Since processing the stream in real-time is computationally demanding, it is challenging on a resource-constrained device. To handle this issue, we perform cost efficient segmentation on H.264 base profile encoding by splitting the stream at specific points as opposed to using computation intensive video processing tools. Once processing the recent segment is done, the drone communicates with blockchain and sends the video metadata (hash, id, etc) gathered to blockchain immediately. For this communication, we utilize TCP to ensure its reliability.

In this respect, the actual video data should also utilize TCP to ensure that whatever processed at the drone is received at the remote server without any losses. This is very critical to guarantee that the hash sent to blockchain will match the hash of the received video in the server. However, TCP is not suitable for video streaming since it will introduce jitter to video frames when there is any loss during transmissions. Therefore, in order to handle real-time streaming from IoT device to remote server, we propose a UDP-based reliable data transmission that enables low latency streaming in addition

to an external loss recovery mechanism over UDP to achieve data integrity, and adaptive resolution against changing bandwidth. Finally, once a video segment is transferred to the remote server, we compute the hash of each segment there instantly, and compare it with the previously saved hash on blockchain by querying the blockchain via a smart contract. This enables instant integrity check to ensure secure transmission, and detect any compromise in the system that might otherwise be detected in the future. For future forensic investigations, since the hash is retained in blockchain permanently in a tamper-proof way, it could be easily retrieved to perform verification on specific part of the video footage.

We implemented our approach on a Raspberry Pi equipped with a camera that has a wireless connection to a remote server. We used Hyperledger, a permissioned blockchain platform, to establish test environment that will act as a distributed storage for hash values. The decision for a permissioned blockchain environment stems from the fact that the system will be restricted to only registered users, and public blockchains will be cost-inefficient. The results show that we can effectively use wireless-IoT device to transmit video without a quality degradation while providing the integrity and authenticity utilizing blockchain.

This paper is organized as follows. The related work in the literature is summarized in the following section. Section 3 introduces some background on the blockchain concepts while Section 4 provides the system model, and our approach is explained in Section 5. Section 6 presents a security analysis while Section 7 evaluates the performance of the proposed mechanism. The last section concludes the paper.

2 | RELATED WORK

Various ideas have been proposed related to blockchain utilization for data integrity verification. Security and trust related challenges, and potential blockchain-based solutions are identified and discussed in References 13 and 14 for IoT generated data. Block-DEF[15] is proposed as a secure digital evidence framework. The idea is to store evidence and evidence information separately. In order to avoid data bloat, a lightweight blockchain design is utilized. Cebe et al[16] suggest a framework for car accident scenarios to save data in blockchain when an accident happens. They use a simplified public key infrastructure tailored for vehicular networks to preserve the privacy. The data saved in blockchain are used to solve any dispute among the insurer, owner, and manufacturer.

The video data can be tampered during transmission as well as after being stored because of because of outside attackers, malicious employee, transmission failure, and storage loss. It is important to verify that video has originated from the actual source and then the video that has been delivered and stored is the same as the video originated from the source. Current techniques to provide and verify integrity in videos and images are using various methods such as watermarking, device fingerprinting.[4,17] However, many cameras do not support adding watermarks. Along with these existing approaches, there is a growing interest in using blockchain as a tamper-proof environment to protect sensitive information.[18-20]

Gallo et al[21] utilized blockchain to protect video content in addition to camera settings such as angle of camera in surveillance systems. The authors try to prevent hackers from changing camera orientation which might either violate the privacy of neighbors or prevent the recording of some criminal scenes. They distinguish the background and foreground images. Background is used to deduce the camera settings by using some features that do not change over time such as corners and edges, while the foreground is used to identify events occurring in the scene. This requires video analysis in frame level, thus not feasible to implement on an IoT device.

Gipp et al[22] implemented a similar approach on smartphones which is used as dashboard camera in cars using public blockchain Bitcoin. Once smartphone detects an accident via accelerometer sensor, it starts recording the scene and calculates the hash at the end to be written to the public blockchain. In order to keep the cost to minimum, it stores the aggregation of the hashes. In order to prove that the video stored on the phone has not been changed, the user provides the original video with the hash. Using public blockchain will not be feasible in terms of cost and speed when we need to store high volume of data.

Karthik et al[23] work on a case that if the video has to be redacted by the owner after the video has been recorded in order to conceal private information such as faces. It still needs to be proved that significant content has not been modified. They propose to utilize blockchain as trusted platform by both a creator and its recipients to store the initial sanitizable signature and transparently update the signature upon content modification. Ghimire et al[24] adopt segmented video hashing where each segment is connected to the previous and next block with hash chain, which resembles original blockchain structure. They assume that surveillance cameras produce segments every few minutes, thus they do not deal with the segmentation and inter-dependency among frames (eg, dependency of P frames). They compute the hash of

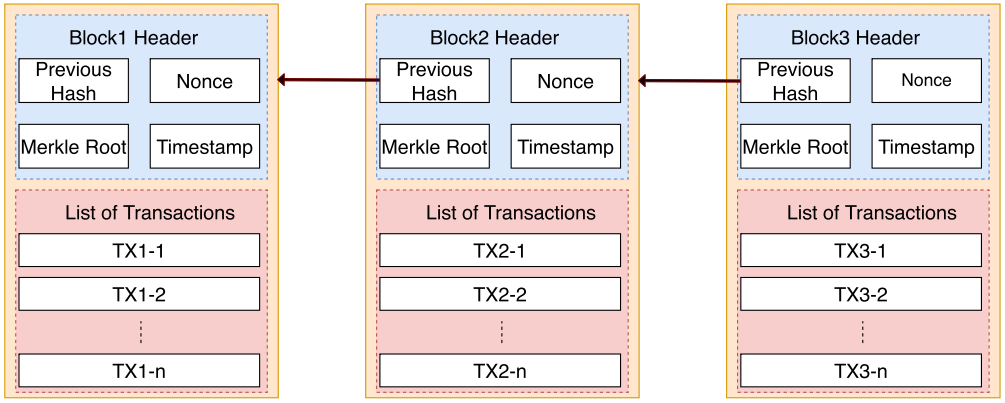


FIGURE 1 Blockchain representation

encoded video from camera and does not deal with encode + decode operation since it is computationally expensive. We handle segmentation problem by using a built-in decoder which enables us to segment the video at any point. They also employed hmac algorithm instead of a standalone hash function. A system has been presented in Reference 25 for CCTV cameras for a smart city scenario. They first register the devices and users to the system through membership service provider (MSP) to be able to interact securely. The devices only send specific frames to the blockchain to be used for validation. They choose Hyperledger as a blockchain platform to store hash values.

Michelin et al[26] proposes a blockchain solution for surveillance cameras in case of airport. Different entities that have CCTV cameras store the video in interplanetary file system network which is blockchain-based file storage system. In this model, video is first transferred to a gateway, a part of the blockchain network, which processes the video to calculate the metadata. The video is not processed at the source which exposes the data to various threats during transmission.

Our work focuses on the video transmission from resource constrained IoT devices as we address those limitations. We segment the video and perform hash calculation for each segment on the device to mitigate potential risks at the source. We also address real-time streaming while providing integrity and adaptive streaming.

3 | BACKGROUND

3.1 | Blockchain

Blockchain is a list of records called blocks linked together by containing the hash of the previous block. The list of blocks continues to grow with the addition of new ones as it is not possible to delete existing blocks. For Bitcoin, a block is simply comprised of transactions (data), timestamp, nonce, the hash of the block, and the hash of the previous block[12] as shown in Figure 1.

The hash of transaction is inserted into a Merkle tree which enables users to easily verify whether a transaction is in the block or not. A critical feature of blockchain is that all of these blocks, and the data they contain, are distributed among many different nodes. These nodes have to agree on the state of the blockchain by using a consensus protocol for the approval of a block, making it nearly impossible to modify any data that has been written to a blockchain. This working scheme of blockchain carries unique properties such as elimination of central authority trust, immutability of record. In blockchain, it is possible to create smart contracts which enables participant to define rules which will be enforced by the network. The joining parties will interact under the defined rules. It provides mechanisms to embed governance rules in verifiable way that can be audited by the consensus algorithm. Blockchain is classified as public and private based on user participation policy. While public blockchains are open anyone, private (permissioned) blockchains are established by a group of stakeholders who are only allowed to make transactions.

3.2 | Consensus mechanism

The process of adding a new block to the chain is carried out via a protocol, which establishes consensus among participants to confirm the new block. In other words, it validates the transactions within the block and provides an agreement

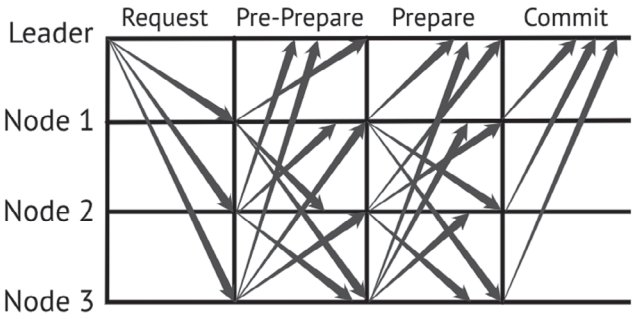


FIGURE 2 An illustration of how BA protocol works with replicated nodes

on the last state of blockchain.[27] There are two types of blockchain structure, public and permissioned, according to the used consensus mechanism.[28] The most widely known blockchains, such as Bitcoin and Ethereum fall into public blockchain category where consensus is established via a mechanism called proof-of-work (PoW). The PoW consensus is typically a form of hash puzzle which requires finding a predefined hash value. This consensus protocol brings a significant level of security on the chain (withstand up to 50% of nodes are being malicious), but at the cost of computational power and time. For instance, Bitcoin's maximum throughput is seven transactions per second and the consensus finality can take an hour. On the other hand, permissioned blockchains utilize some kind of Byzantine fault tolerant voting based algorithm as consensus mechanism, such as practical Byzantine fault tolerance[29] or Stellar consensus protocol,[30] which do not require computationally expensive hash puzzles. As a result, reaching a consensus is faster which means higher transaction throughput. However, permissioned blockchains generally require more than two-thirds of nodes to be trustworthy rather than 51%.

3.3 | Voting based consensus protocol

Voting based consensus protocol first emerged in distributed computing[29] to provide reliability of data or computation, even if arbitrary nodes conduct malicious actions or fail. Permissioned blockchain mechanisms adapt the same idea to establish consensus where some of some nodes may act maliciously.

In this setting, where there are n nodes, a consensus can be achieved if at least $(2n-1)/3$ number of nodes act honestly. Honesty means providing correct information to the other participants. In a permissioned blockchain, there are two different types of nodes called *Leader* and *Validator*. First, a randomly selected Leader builds a block from transactions. This block is then distributed by the leader to Validator nodes for verification. Validators check the transactions within the block, sign it, and distribute it again to the other Validators, as shown in Figure 2. Each node, again, distributes the block captured from the other node. This continues until each Validator node collects individually signed versions of the block from the other ones. After $n-1$ version of blocks are gathered, the Validators check differences between blocks. If $(2n-1)/3$ of these blocks are valid, the Validator nodes inform the Leader about confirmation and add the block to its local chain.

3.4 | Hyperledger

Hyperledger is an open source umbrella community and platform founded by the Linux Foundation for the development of blockchain projects, frameworks, libraries, and tools (Figure 3). Throughout its inception, Hyperledger has received various contributions from Intel and IBM to support a collaborative development of blockchain-based distributed ledgers. It is a permissioned blockchain platform where access is restricted to stakeholders unlike the public blockchain where anyone can access the produced blocks. It is using voting based consensus algorithm defined in the previous section. There are numerous applications of Hyperledger blockchain-based distributed ledgers. The applications include distributed ledgers dealing with finance, banking, healthcare, manufacturing, and supply chains. Unlike most blockchain technologies, Hyperledger does not support cryptocurrencies like Bitcoin, Ethereum, Litecoin, and so on. The executives governing Hyperledger ultimately decided not to have a blockchain infrastructure focused on cryptocurrencies or tokens to promote non-monetary, high scaling industrial applications of blockchain technologies. Presently, Hyperledger's most notable

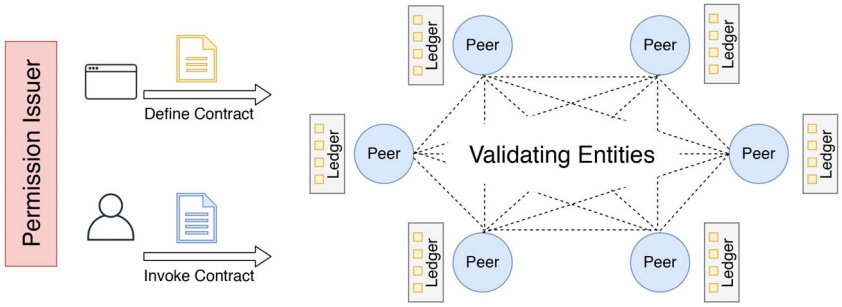


FIGURE 3 Hyperledger platform

frameworks are Hyperledger Fabric, Sawtooth, Iroha, and Besu. It uses JavaScript, Go, and Java Software Development Kits (SDK) for integration projects that needs to use Distributed Ledger Technology (DLT).

3.5 | Camera fingerprinting

Images and videos captured by a camera can be identified using some hardware parameters of the device which is called camera calibration. It involves intrinsic parameters of the camera while some applications require determining extrinsic parameters as well. Intrinsic parameters generally include focal length, scale factor, principle point, and the lens radial distortion.[31] Camera calibration is generally achieved by taking pictures of a template pattern, whose structure (ie, distances between points) is known such as a checkerbox, from many angles and distances. While distortion can be determined from images, information about the environment would be required for verifying other parameters of the camera. If a gyroscope sensor is included, it provides angular velocity and can be used to verify the center point of rotation in the image and can be matched with the principle point of the camera. Available methods to identify the camera that has been used for recording an environment categorized into two whether they aim to determine a model for the camera or use machine learning methods by extracting features from images.[32] The first type of methods uses photo-response non-uniformity (PRNU),[33] lens radial distortion,[34] sensor pattern noise[35] to devise a model. Second method employs machine learning approaches such convolutional neural networks to identify the camera based on an input image.[32]

4 | SYSTEM AND THREAT MODEL

This section defines the problem along with requirements and lists the potential threats.

4.1 | Problem definition

The overall problem can be stated as designing a framework to ensure the integrity of a video (evidence) recorded by a resource constrained wireless device. This main problem can be defined as the combination of subproblems to further clarify the challenges and solutions.

4.1.1 | Integrity verification

Integrity verification refers specifically to determining if the video currently possesses the conditions at the time it was recorded. If the initial video is V_i and the final version is denoted as V_f, when the evidence is investigated in a trial, it must be proved that $V_i = V_f$. In order to minimize the risk of tampering, the hash is computed on the device where the video is being recorded and it is transmitted immediately. This increases the complexity because of additional limitations of IoT device in terms of computation and connectivity as new subproblems emerge to be addressed. In case of any failure such as hardware malfunction, battery exhaustion, Wi-Fi disconnection, we want to save hash periodically not to lose previous portions. The problem turns into verifying each subset of frames individually. Then, $V_i = \{V_{i1}, V_{i2}, \ldots, V_{in}\}$ and $V_f = \{V_{f1}, V_{f2}, \ldots, V_{fn}\}$, thus each corresponding portion must be verified that they are equal such as $V_{ik} = V_{fk}$. Other challenges stemming from resource limitation are further elaborated as system requirements. Those are also subproblems to be addressed within the framework.

4.1.2 | System requirements

- *High resolution processing*. Video processing is a computationally expensive process especially for resource-constrained IoT devices. However, it is important to record a high quality video for clear interpretation of scenes. Therefore, the system should enable processing high resolution videos on IoT devices.
- *Real-time streaming*. Transmitted video could be used to control the device in real-time by the operator. This necessitates the low-delay video transmission.
- *Adaptive streaming*. Since the drone is navigating, the connection bandwidth between the IoT device and the control room might change during the operation. The video quality should be adjusted based on the available bandwidth.
- *Efficient segmentation*. To save the recorded part of the video, the hash must be calculated periodically thus requiring the segmentation of video into small chunks efficiently using limited computational resources.

4.1.3 | Secure transmission and storage

After the video and hash are acquired on the device, they must be securely communicated and stored for later use. Threats that might compromise the integrity might take place during the transmission or storage phases. Various threats to the system are listed in the next section and investigated later in Section 6.

4.2 | Threat model and security goals

As our threat model, we consider both physical and network threats against our framework. We assume that the adversary has both read and write access to the communication interfaces of the device (eg, LTE, wireless). This is a practical presumption as IoT video devices are installed on drones/body-cams which are physically accessible to attackers. We also assume that the adversaries may intercept communication of the device by altering/dropping/replaying the messages sent from the device to video database and blockchains.

Threat 1: In this scenario, the attacker disguises itself as a legitimate device by stealing private key of the device to push false surveillance data to our forensic system.

Security Goal 1: Validate the participation of unauthorized devices to our system by prohibiting them using stolen keys as if they are authenticated.

Threat 2: In this attack scenario, the attacker attacks the device communication layer and drops packets to break the integrity of forensic system. The malicious actor can also fulfill this attack selectively, for example, by dropping packets for particular types or dropping randomly selected packets. Considering the lossy network environment of our system, this attack is hard to detect and may cause differences between the blockchain hash and the video in the database.

Security Goal 2: Track the missing packets and provide holistic integrity of the forensic system along with the integrity of each packet.

Threat 3: In this attack scenario, the attacker first alters the video stored in the database. Then, s/he updates the hashes of corresponding frames in blockchain. Thus, the attacker succeeds in altering the original stored forensic data without breaking its feature of being an evidence in the court.

Security Goal 3: Prevent an attacker altering both the database and blockchain simultaneously to produce false evidence.

5 | BLOCKCHAIN-BASED VIDEO FORENSICS

5.1 | Motivation

Using videos as evidence is a fairly complicated task and poses many questions. Sophisticated video tampering methods enable modifying the original footage such as deleting scenes or frames,[36] inserting new frames, manipulating frames, or removing moving objects in the scene, or cropping regions.[37] Some image processing techniques such as blurring can be applied to blur the important objects or people in the scenes. Moreover, deep learning introduces new challenges for

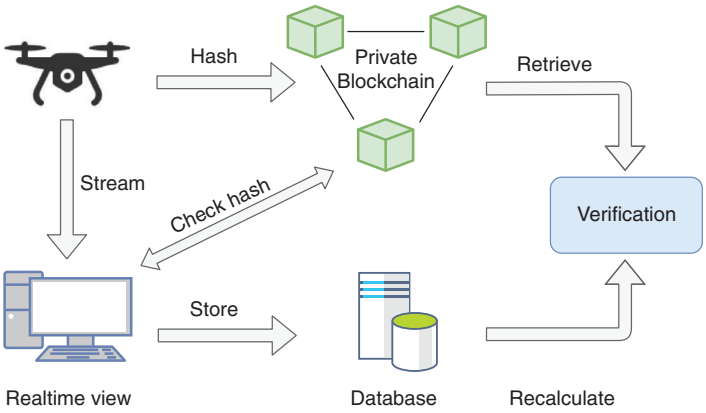


FIGURE 4 Framework for video integrity verification using blockchain

authenticating the originality of a video. Unless the original video is available, it could be impossible for ordinary users to determine whether the video is original or fake. For instance, *DeepFake* is one major video tampering technique where the face of a person in a video can be replaced with the face of another person[38] and it is shown to be very realistic. Furthermore, generative adversarial networks (GANs)[39] are used to generate new content such as adding gestures by analyzing content from multiple sources. For example, GestureGAN can generate new images of a person with new hand gestures by using the original scene and a sample skeleton image of a hand gesture.[40]

Availability of such advanced techniques makes it necessary to come up with novel ideas to retain and prove the integrity and authenticity of critical footages. Thus, as a more reliable alternative, we propose a video forensic framework for integrity verification by leveraging blockchain concept. The proposed system enables capturing high-resolution videos and allows the investigators to detect tampered video captured by drones. The remaining of this section explains the details of the framework.

5.2 | Framework architecture

Camera-equipped drone continually records video as shown in Figure 4 and transmits the captured scene to the remote server. To be able to prove or verify the integrity of the recorded video, the hash of it which is a unique signature is utilized. However, the hash value is also vulnerable to attacks as well as the video itself.

Thus, the secure calculation and storage of hash must be handled. To this end, we propose handling the hash computation in the device before it is transmitted, and store it in blockchain to prevent any further compromise. So, as the video is streamed, the drone computes the hash of the streamed data which is sent directly from the device to the permissioned blockchain consisting of stakeholders' nodes. Once the transmission of a segment is complete, the remote server retrieves the hash by querying blockchain via smart contract to make sure the stored hash matches with the received video. This check ensures the integrity during transmission. Whenever a video is needed for any investigation, its integrity can be verified in the same way. This second check is to protect the information during storage phase against a malicious employee or attacker.

Before any device is used for this type of recording, it must be registered in the system. This process includes assigning keys to device so that it can communicate with system. The admin can revoke the permission at any time. In addition to key generation, our model assumes that the fingerprint of the device such as PRNU, and sample videos from the device is taken and saved in the blockchain to compare to future footage. This information is used to prove the source of a video.

5.3 | Segmentation

While the video is being streamed, it is possible to wait until the recording is complete to transfer the hash of the whole video to the blockchain. However, this method poses several risks that threatens the security of the recording. The drone may stop running because of the battery or physical crash. The wireless connection may go down, any malfunctioning or attack may cause termination of recording or loss of recorded data. In order to eliminate these types of problems that would make the whole video useless, we adopt a segmentation approach. Instead of waiting to transfer the hash of the

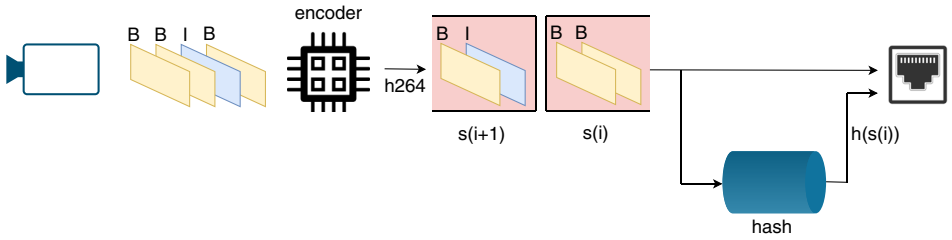


FIGURE 5 Segmentation is illustrated in low level

complete video, we periodically calculate the hash of small units. However, this requires dividing the stream into small chunks which might add delay and computation cost. Since, these operations are to be performed on resource-constrained device, there is a need for an efficient method. Video segmentation can be achieved using video processing tools such as *opencv*, *mpeg-dash*, and *ffmpeg streamer*, which requires "decode & re-encode" process. This is a resource demanding operation, and not feasible for an IoT device. This method allows only processing low resolution videos.

Most security cameras use the base profile of H.264 encoding which only contains **I** and **P** frames which are known as the frames within a typical video (there is no **B** frame). So, it is possible to capture the stream in small units by dividing at appropriate points (ie, **I** frames) without "decode & re-encode" process, thus to handle higher quality videos. For instance, the Raspberry Pi, widely used IoT device, has a hardware encoding unit that can generate H.264 video stream and allows to generate segments with appropriate size.

Figure 5 illustrates the segmentation process. Encoder compresses the raw video coming from camera by generating **I** and **B** frames. Whenever it receives a split command, it waits until next **I** frame to terminate the current segment and start a new one. A segment may contain multiple **I** frames and many **B** frames, but always start with an **I** frame. The length of the segment is customizable but we choose 3 seconds in our implementation.

5.4 | Sending video hash to blockchain

The video integrity verification can be accomplished by storing the video file on a blockchain, which would make it immutable, thus it can be used as evidence for forensic purposes. However, the size would be too big to put on a blockchain, so we compute hash of video stream and put it on a blockchain. The hash is unique and can later be used to verify the video. As the data generated from the camera, before dumped into network, it is fed into hash function as shown in Figure 5. It updates the hash value with new arriving data.

$$h\,(a+b) = h.update\,(a) + h.update\,(b)\,.$$

Thus, hash function does not have to wait for the video to be complete. With this approach, the video is instantly sent from the IoT device which enables real-time streaming. Various hash functions exist such as MD5, SHA2, and SHA3. Even though MD5 is faster in terms of computation time, it is considered weak in terms of collision. We choose SHA3 that can generate 512 bit long output.

Figure 6 explains the interaction between end-user and blockchain. In our case, nodes represents regional police stations, each operates its own node independently with a Certificate Authority which handles issuing and validating certificates for device authentication In order to allow a device to access the network, the admin of a peer creates a user, which means creating cryptographic credentials for the device when it is registered to the system. The interaction between the device and blockchain must always go through peers using APIs. The device invokes smart contract to submit an update or query the ledger. When the user submits a transaction, the signature is verified by the MSP. If the transaction has a valid signature, the transaction is processed. The transaction created by the device consists of video ID (V_{id}), sequence number of segment (S_i), timestamp (TS), and hash of the video segment ($H(S_i)$).

$$T_i = \left\{V_{id}, S_i, TS, H\left(S_j\right)\right\}.$$

The transaction is stored on the Hyperledger with a transaction id which is also sent to remote server to be saved in the database.

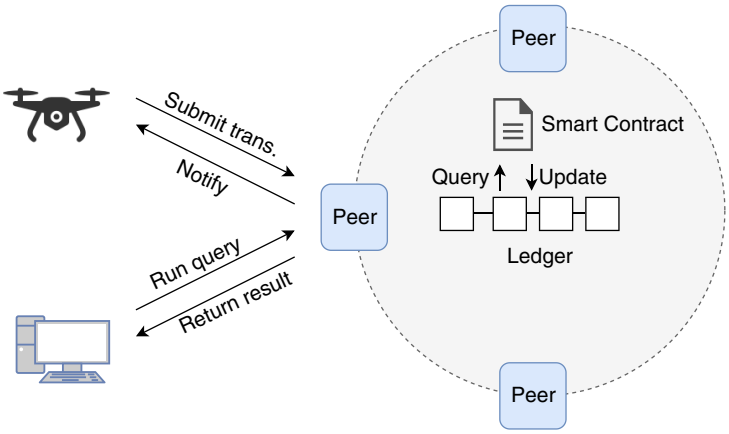


FIGURE 6 End-device interact with Hyperledger through smart contract

5.5 | Real-time streaming with loss recovery

The proposed approach can be used in various scenarios that have different requirements in terms of latency. If the video is only to be stored in database not for real-time viewing, then video transmission is performed using a TCP connection which automatically handles packet loss recovery by requesting re-transmission of dropped packets from the sender. However, this induces a delay on the receiver application especially in a lossy environment. This type of communication can be adopted for some applications where drone is controlled visually but transmission is only needed to store the video for later investigations. Nevertheless, TCP communication is not suitable for real-time control which cannot tolerate high delay. UDP is the only option to utilize for such scenarios as it forwards incoming packets directly to application. Packet loss can be tolerated to some extent although it may cause quality degradation. On the other hand, UDP does not provide reliable transfer as we need it for integrity. As a result, we need a solution that provides both real-time transmission and reliable delivery as detailed next. We utilize datagram transport layer security (DTLS) for data transmission to enable secure communication.

5.5.1 | Loss recovery

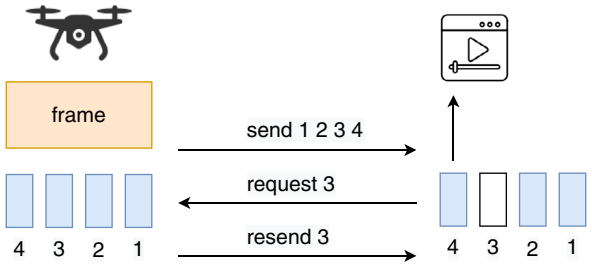
We propose a loss recovery mechanism to UDP communication, which is integrated to the receiver side by augmenting UDP with an explicit re-transmission capability at the application layer. Essentially, this is adding another layer to the protocol stack by introducing new headers to keep track of the lost packets after they are passed to the application layer.

To perform this, we start by identifying the right packet size. As is known, the size of a video frame is mostly much bigger than maximum transmission unit which means that the frames are fragmented by the IP layer. The loss of one fragment requires re-transmission of the whole frame. In order to avoid fragmentation, we divide the frames into a reasonable size so that we can track and request each individual packet exclusively. Basically, each packet is indexed with a unique number that identifies video, segment, and sequence of packet.

As shown in Figure 7, the drone produces small UDP packets and embeds sequence number to each as the frames arrive from the encoder. The packets are retained in the IoT device in order to respond to any re-transmission request. The order of packets may change or any of them may be dropped because of congestion/wireless channel during transmission. On the receiver side, the incoming packets are fed directly into player regardless of loss or out of order. The player itself might have a short buffer before rendering the video. In order to retain the integrity of the video, the missing packets are requested from sender side explicitly using the sequence number in packets after waiting a certain amount of time. This process is repeated until the missing chunks are completed. Whenever all the chunks of a frame are complete, they are written to the corresponding segment file and when all the frames of a segment arrived, the segment file is complete.

Note that we are also utilizing a separate TCP connection between drone and remote server to send control commands, in a similar manner to RTP and real-time control protocol (RTCP) where RTCP is used to communicate with the server such as managing the quality of the video. In our case, the remote server sends re-transmission request messages over TCP, and the drone informs the receiver about end of video using the same channel. Additionally, resolution change command used in the next section is also transmitted over this channel.

FIGURE 7 Receiver plays and store videos



5.5.2 | Adaptive streaming

While the drone is navigating in the air, the bandwidth of wireless connection might change because of physical distance to access point, congestion in the network traffic, blocking objects in the transmission medium. This might cause very long delay and freezes in the stream which makes controlling the drone remotely impossible. Moreover, the integrity of the stream is put in danger because re-transmission requests may not be fulfilled. This necessitates an adaptive streaming scheme. MPEG-DASH is the prominent protocol to handle streaming under variable network conditions. It generates short segments of a source video in different resolutions simultaneously so that the user can switch to any of them depending on the connection quality. Such a solution is not feasible to implement on an IoT device as it requires a high computation power. Changing the capture resolution of the camera can be considered as a possible solution, but this requires restarting the capture process which causes a few seconds pause in the stream. This is also not an acceptable alternative as we need a continuous stream.

Our solution utilizes the resize capability of encoder together with two additional parameters; *quality* and *bandwidth* which also impacts the output size. These parameters can only be adjusted at the beginning of a new segment. Since we are using a segmented scheme, we easily deployed adaptive streaming in our solution. The process is initiated by the receiver which observes the connection quality by considering bandwidth estimation parameters such as number of lost packets, and so on. Then, it notifies the sender either to increase or decrease the resolution, and the sender switches to an appropriate resolution based on the feedback from sender.

5.6 | Video storage

The video sent from the drone is received by the remote server. It is viewed by an operator and also stored for future use. Since the video is split at specific points on the sender side, the receiver must be communicated to store the pieces properly. This is achieved by a "split" message that contains the sequence number at which the new segment starts as shown in Figure 8. Whenever the receiver gets this message, it will start dumping data into another file starting from split point. Once the transmission of a segment is complete, it will be compared to blockchain using its associated transaction id to check the integrity. Then, the frames will be stored in two versions. The first one will be complete file that contains all frames without a segmentation whose purpose is to make view and search easy for the end-user. The second one will be stored as segments in order to make integrity verification for each segment separately.

5.7 | Integrity verification

Validation process is requested by an investigator if a video is needed to be used as evidence. This process includes the following steps:

- *Hash re-calculation*: The investigator needs to access the original video that s/he needs to verify and identify the segment s/he is interested in. Then, the hashing process is performed for the stored video segment to get the current signature.

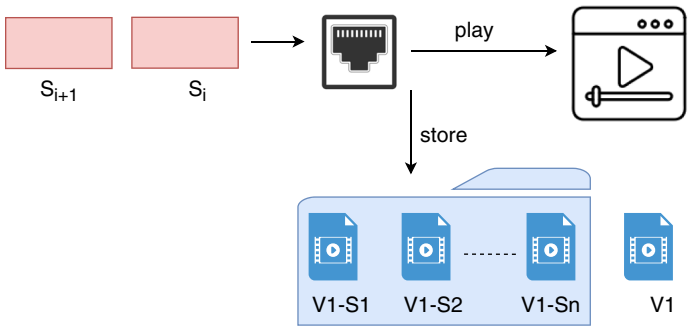


FIGURE 8 Receiver plays and store videos

- *Blockchain query*: The blockchain is queried using transaction ID associated with segment to access previously saved hash along with other metadata such as video ID, segment ID, time stamp, and so on. Querying a transaction is simpler and faster than update which has less steps to finalize the procedure.
- *Validation*: The query will return the original hash stored in Hyperledger. If the retrieved hash and computed hash match, the segment is authentic; if they do not match, it can be inferred that the video is a fake or altered video. The stored hash is secure because it is distributed on all stakeholders and they agree on its correctness. If any stakeholder is compromised, the other nodes will still provide the correct information.

6 | SECURITY ANALYSIS

In this section, we provide a security analysis of our proposed approach with respect to our threat model described in Section 4.2.

Threat 1: Stolen Keys Attack: In our proposed scheme, only registered devices are able to produce forensically sound data to our system. This is achieved through MSP component which is available in Hyperledger. In addition to this, the scheme will refuse any generated video signed with valid stolen keys (ie, cryptographically authentic) due to the fact that the genesis block produced by that device store the fingerprint (PRNU parameters) of the device in blockchain as well. Thus, generated videos from another device will be rejected when the fingerprint of that device and the saved one in genesis block does not match during investigation.

Threat 2: Selective Packet Dropping Attack: Our system excludes this attack since it requires breaking the reliable transmission methods used both streaming the video and transferring the corresponding hashes. We utilize DTLS which prevents eavesdropping, tampering and replay attacks. The applied loss recovery technique in Section 5.5.1 ensures the reliability of video frames while DTLS does not guarantee reliable delivery.

Threat 3: Generation of Forensically Sound Data Attack: Our scheme acts as an unbreakable seal to provide a long-term integrity ensuring mechanism for forensic investigations. This is due to the fact that, changing an old transaction, in other words, rollback will require to counterfeit 2/3 of the peers in the system. Considering the well-protected and distributed nature of the members, this is hard to achieve even for insider attackers since it requires a coordination with other insiders to infiltrate at least 2/3 of the members.

7 | PERFORMANCE EVALUATION

7.1 | Experiment setup

In order to evaluate the performance of the proposed approach, we set up a testbed and performed various tests. We used a Raspberry Pi3 to simulate a drone, or similar IoT device which communicates with remote server and Hyperledger network through Wi-Fi. It ran the code that would be installed on a drone equipped with a camera. This setup is shown in Figure 9. Raspberry Pi3 B+ specs shown in Table 1 are used to run the experiments. Therefore, the values in the results are specific to this configuration.

For the blockchain implementation, we used Hyperledger Fabric, a permissioned blockchain, to create a test network. First, the prerequisites are installed such as Python, Node.js, Docker, Docker Compose, Git, CURL, and Go. Hyperledger

FIGURE 9 System implementation

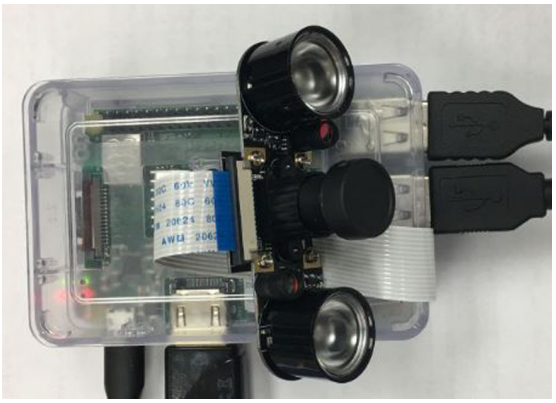


TABLE 1 Raspberry Pi3 specifications

CPU(SoC)	BCM2837B0 quad-core A53 1.4 GHz
RAM	1 GB LPDDR2 SDRAM
Networking	2.4 GHz and 5 GHz 802.11b/g/n/ac Wi-Fi

TABLE 2 Video resolutions

Version	Resolution
v1	320×240
v2	640×480
v3	960×720
v4	1280×960
v5	1600×1200

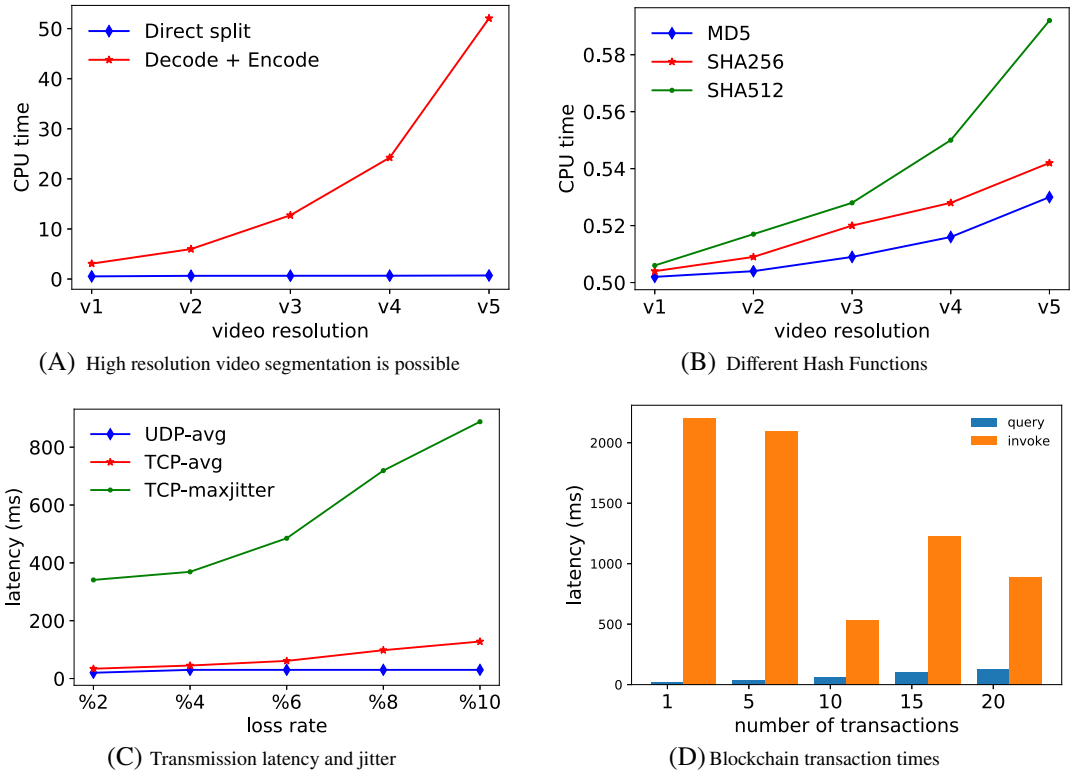
Fabric uses JavaScript, Go, and Java SDK for integration projects that needs to use DLT. We created a test network consists of peers, channels, admin, user to run a test application. The most fundamental component of the Hyperledger Fabric network is peers which have the role of storing the blockchain ledger and validating transactions before they are committed to the ledger. Peers run smart contracts on the blockchain ledger. Every peer in the network belongs to an organization. Channels are a private communication layer between certain network users. Users a part of the network utilizes applications to invoke smart contracts to create transactions on the ledger and execute query to read data from the ledger. In Hyperledger Fabric, smart contracts are deployed on the network in the form of packages known as chaincode. The chaincode is installed on the peers of an organization.

We performed test using various resolutions which are shown in Table 2 with 24 fps.

7.2 | Metrics and baselines

We use the following metrics to assess the performance of the proposed approach:

- *CPU utilization*: Processing time is a crucial metric since we are operating on an IoT device. We measure the CPU utilization required for segmentation, hashing, and so on.
- *Transmission latency*: It shows the latency of video packets from IoT device to remote server. It affects the real-time view and control of the drone.
- *Jitter*: It is defined as the inter-arrival time between the packets. It is important for quality of service.
- *Blockchain transaction latency*: We use this metric to measure the time needed to write and read data from Hyperledger.



(A) High resolution video segmentation is possible

(B) Different Hash Functions

(C) Transmission latency and jitter

(D) Blockchain transaction times

FIGURE 10 Experiment results for balance-aware routing using single connection

7.3 | Performance results

7.3.1 | Video processing efficiency

Figure 10A compares the CPU utilization on Raspberry during video capturing while calculating the hash. We use the OpenCV method as baseline to compare our proposed approach. OpenCV first decodes the stream in order to split, then must encode again which is depicted as "decode + encode." Since these processes are performed on CPU, it is possible to handle only low resolution video. However, using our splitting technique, *Direct split*, we complete the whole process for much higher resolutions under 1 second CPU utilization. This leaves time to perform other computations such as encryption. We perform hash computation explicitly as opposed to segmentation for which we used a built-in decoder instead of doing it on the CPU. The complexity is crucial for the overall performance especially as the resolution increases. Block ciphers work on fixed-size blocks of bits and the work done per block is constant. Thus, the complexity is $O(n)$ where n is the number of blocks in the video segment.

Figure 10B shows computation time of our approach with precise values. We tested three different hashing algorithms. Even though SHA512 needs more time, it is still very close to others. It is also worth to mention that the time needed for a hash algorithm depends on the input size. So, hashing raw frames generated by OpenCV would take a lot more time.

7.3.2 | UDP performance

In Figure 10C, we compare TCP and UDP communication by considering real-time streaming which requires low latency for an efficient control of a drone. We varied the loss rate on Raspberry wireless interface from 2% to 10% and run the test within a WLAN which gives very low transmission latency (0-5 ms) in a lossless environment. That would increase for long distance communication, and it also depends on current network conditions such as congestion etc. However, we want to highlight that the gap between TCP and UDP is significantly increasing with higher loss rate especially after 5%. We achieve very low latency packet transmission over Wi-Fi using UDP-based streaming at the cost of some lost packets

which is tolerable up to some extent. However, reliable transmission provided by TCP incurs delay in order to recover lost packet which increases the average packet delay up to 200 ms which seems not very high because non-delayed packets arrive almost instantly. Thus, the jitter time of lost packets is more important to understand the quality for this case since a lost packet will hold off all the packets waiting behind from being transmitted to application layer. This means frequent pauses will appear in the stream that makes operating an interactive device impossible. We observed that jitter can be up to 1000 ms which means TCP is infeasible for such applications. Our UDP-based solution provides low latency transmission and reliable transfer with explicit re-transmission capability. Although we consider this application for the use of security forces in which case the source of the video may not want to hide himself. However, if the system is adopted for public use and a third party wants to keep himself anonymous, video and hash may be transferred through one of privacy preserving tools[41] such as Tor, although the performance will be dependent on the tool's features used as communication platform.

7.3.3 | Hyperledger performance

In Figure 10D, we present the transaction latency for write and read operations on Hyperledger platform. A bunch of transactions, from 1 to 20, is submitted concurrently in order to test the performance. Query latency increases from 20 to 100 ms with increasing number of transactions, which still can be considered fast. Average time of invoke operation, on the other hand, depends on the number of transactions submitted together and the blocksize. We are using the default block size which is 10 transactions per block. For a single transaction, it takes around 2 seconds because it waits for block timeout to submit the block for validation. When we look at 10 transactions, the average latency is much lower because the block is submitted immediately. This pattern will continue up to a saturation point which is generally measured as 140 tps, then it will start increasing. So, Hyperledger is capable of processing the transactions from a drone fast enough, and the system can scale to handling many drones simultaneously. In order to optimize the blockchain part, other parameters such as block size can also be tuned depending on the transaction arrival expectation.

8 | CONCLUSION

Video recordings are precious for security and judicial system to investigate criminal cases and resolve disputes as they reveal meaningful and trustable information. However, since they are vulnerable to manipulation with the existence of advanced editing tools, it is mandatory to show its authenticity to make it a valid evidence in trials. This calls for efficient integrity validation mechanisms. In this work, we presented a forensic framework specifically designed for video integrity in case of transmission from wireless IoT devices to capture videos for forensic investigations. In order to avoid data loss due to the failure of devices, we adopted segmentation approach for precise integrity check. High computation power requirement for splitting is overcome by utilizing a built-in decoder. The results show that we are able to process high resolution videos on a resource-constrained IoT device. Moreover, we developed a UDP-based reliable transmission by adding external re-transmission feature. It enabled to have real time streaming with lossless file transfer with low delay. We also showed that video quality can be adapted in real time based on available bandwidth with varying distance to control center. The proposed system is tested with a Hyperledger permissioned blockchain platform and the results indicate its feasibility in terms of transaction throughput for such a framework. We also provided a security analysis of the proposed framework that discusses how the desired security goals are achieved.

CONFLICT OF INTEREST

The authors declare no conflicts of interest.

ACKNOWLEDGMENT

Open access funding enabled and organized by Projekt DEAL.

ORCID

Suat Mercan https://orcid.org/0000-0003-1571-9256

REFERENCES

1. Ashby MP. The value of CCTV surveillance cameras as an investigative tool: an empirical analysis. *Eur J Crim Policy Res*. 2017;23(3):441-459.
2. Vattapparamban E, Güvenç İ, Yurekli Aİ, Akkaya K, Uluağaç S. Drones for smart cities: Issues in cybersecurity, privacy, and public safety. In Paper presented at: 2016 International Wireless Communications and Mobile Computing Conference (IWCMC); IEEE; 2016: 216–221.
3. Menouar H, Guvenc I, Akkaya K, Uluagac AS, Kadri A, Tuncer A. UAV-enabled intelligent transportation systems for the smart city: applications and challenges. *IEEE Commun Mag*. 2017;55(3):22-28.
4. Poisel R, Tjoa S. Forensics investigations of multimedia data: a review of the state-of-the-art. In; Paper presented at: 2011 Sixth International Conference on IT Security Incident Management and IT Forensics; IEEE; 2011: 48–61.
5. Using Video for Documentation and Evidence; 2014. https://www.newtactics.org/conversation/using-video-documentation-andevidence. Accessed 20 July, 2020.
6. Ismael O, GhazaliSulong AS. Detection of video forgery: a review of literature. *J Theor Appl Inf Technol*. 2015;74(2):207-220.
7. Liang X, Shetty S, Tosh D, Kamhoua C, Kwiat K, Njilla L. Provchain: a blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability. In Paper presented at: 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID); IEEE; 2017: 468–477.
8. Prayudi Y, Sn A. Digital chain of custody: state of the art. *Int J Comput Appl*. 2015;114(5):975-8887.
9. Echizen I, Singh S, Yamada T, Tanimoto K, Tezuka S, Huet B. Integrity verification system for video content by using digital watermarking. In Paper presented at: 2006 International Conference on Service Systems and Service Management; IEEE; 2006: 1619–1624.
10. Aparna J, Ayyappan S. Comparison of digital watermarking techniques. IEEE; Paper presented at: 2014 International conference on computation of power, energy, information and communication (ICCPEIC), 2014: 87–92.
11. Song C, Sudirman S, Merabti M, Llewellyn-Jones D. Analysis of digital image watermark attacks. In Paper presented at: 7th IEEE Consumer Communications and Networking Conference; IEEE; 2010: 1–5.
12. Nakamoto S, Bitcoin A. A peer-to-peer electronic cash system. *Bitcoin*; 2008. https://bitcoin.org/bitcoin.pdf. Accessed 15 June, 2020.
13. Olufowobi H, Engel R, Baracaldo N, Bathen LAD, Tata S, Ludwig H. Data provenance model for internet of things (IoT) systems. In Paper presented at: International Conference on Service-Oriented Computing; Springer; 2016: 85–91.
14. Polyzos GC, Fotiou N. Blockchain-assisted information distribution for the Internet of Things. In Paper presented at: 2017 IEEE International Conference on Information Reuse and Integration (IRI); IEEE; 2017: 75–78.
15. Tian Z, Li M, Qiu M, Sun Y, Su S. Block-DEF: a secure digital evidence framework using blockchain. *Inform Sci*. 2019;491:151-165.
16. Cebe M, Erdin E, Akkaya K, Aksu H, Uluagac S. Block4forensic: an integrated lightweight blockchain framework for forensics applications of connected vehicles. *IEEE Commun Mag*. 2018;56(10):50-57.
17. Sencar HT, Memon N. Digital image forensics. *Counter-forensics: attacking image forensics*. New York, NY: Springer; 2013:327-366.
18. Liang X, Zhao J, Shetty S, Li D. Towards data assurance and resilience in IoT using blockchain. In Paper presented at: 2017 IEEE Military Communications Conference (MILCOM); IEEE; 2017: 261–266.
19. Khan MA, Salah K. IoT security: review, blockchain solutions, and open challenges. *Future Gener Comput Syst*. 2018;82:395-411.
20. Danko D, Mercan S, Cebe M, Akkaya K. Assuring the integrity of videos from wireless-based IoT devices using blockchain; In Paper presented at: 16th International Conference on Mobile Ad Hoc and Sensor Systems Workshops (MASSW); IEEE; 2019: 48–52.
21. Gallo P, Pongnumkul S, Nguyen UQ. BlockSee: blockchain for IoT video surveillance in smart cities. In Paper presented at: 2018 International Conference on Environment and Electrical Engineering; IEEE; 2018: 1–6.
22. Gipp B, Kosti J, Breitinger C. Securing video integrity using decentralized trusted timestamping on the Bitcoin blockchain. In Paper presented at: Mediterranean Conference on Information Systems (MCIS); IEEE; 2016: 51.
23. Nandakumar K, Ratha N, Pankanti S. Proving multimedia integrity using sanitizable signatures recorded on blockchain. In Paper presented at: Workshop on Information Hiding and Multimedia Security; IEEE; 2019: 151–160.
24. Ghimire S, Choi JY, Lee B. Using blockchain for improved video integrity verification. *IEEE Trans Multimedia*. 2019;22(1):108-121.
25. Khan PW, Byun YC, Park N. A data verification system for CCTV surveillance cameras using blockchain technology in smart cities. *Electronics*. 2020;9(3):484.
26. Michelin RA, Ahmed N, Kanhere SS, Seneviratne A, Jha S. Leveraging lightweight blockchain to establish data integrity for surveillance cameras. *arXiv preprint arXiv:1912.11044*; 2019.
27. Baliga A. Understanding blockchain consensus models. *Persistent*. 2017;2017(4):1-14.
28. Okada H, Yamasaki S, Bracamonte V. Proposed classification of blockchains based on authority and incentive dimensions. In Paper presented at: 19th international conference on advanced communication technology (icact); IEEE; 2017: 593–597.
29. Castro M, Liskov B. Practical Byzantine fault tolerance and proactive recovery. *ACM Trans Comput Syst*. 2002;20(4):398-461.
30. Mazieres D. The stellar consensus protocol: a federated model for internet-level consensus. *Stellar Dev Found*. 2015;32. https://tools.ietf.org/id/draft-mazieres-dinrg-scp-05.html. Accessed 5 August, 2020.
31. Heikkila J, Silven O. A four-step camera calibration procedure with implicit image correction. In Paper presented at: Proceedings of IEEE computer society conference on computer vision and pattern recognition; IEEE; 1997: 1106–1112.
32. Tuama A, Comby F, Chaumont M. Camera model identification with the use of deep convolutional neural networks. In Paper presented at: 2016 IEEE International workshop on information forensics and security (WIFS); IEEE; 2016: 1–6.
33. Rosenfeld K, Sencar HT. A study of the robustness of PRNU-based camera identification. International Society for Optics and Photonics. In Paper presented at: Proceedings of SPIE volume 7254, Media Forensics and Security; 2009: 72540M.

34. Choi KS, Lam EY, Wong KK. Source camera identification using footprints from lens aberration. In Paper presented at: Proceedings Volume 6069, Digital Photography II; IEEE; 2006: 172–179.
35. Lukas J, Fridrich J, Goljan M. Digital camera identification from sensor pattern noise. *IEEE Trans Inform Forensics Secur*. 2006;1(2):205-214.
36. Stamm MC, Lin WS, Liu KJR. Temporal forensics and anti-forensics for motion compensated video. *IEEE Trans Inform Forensics Secur*. 2012;7(4):1315-1329.
37. Wahab AWA, Bagiwa MA, Idris MYI, Khan S, Razak Z, Ariffin MRK. Passive video forgery detection techniques: a survey. In Paper presented at: 10th International Conference on Information Assurance and Security; IEEE; 2014: 29–34.
38. Güera D, Delp EJ. Deepfake video detection using recurrent neural networks. In Paper presented at: 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS); IEEE; 2018: 1–6.
39. Radford A, Metz L, Chintala S. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*; 2015.
40. Tang H, Wang W, Xu D, Yan Y, Sebe N. GestureGAN for Hand Gesture-to-Gesture Translation in the Wild. Paper presented at: MM '18. IEEE. Association for Computing Machinery, New York, NY; 2018: 774–782.
41. Nia MA, Ruiz-Martínez A. Systematic literature review on the state of the art and future research work in anonymous communications systems. *Comput Elect Eng*. 2018;69:497-520.

How to cite this article: Mercan S, Cebe M, Aygun RS, Akkaya K, Toussaint E, Danko D. Blockchain-based video forensics and integrity verification framework for wireless Internet-of-Things devices. *Security and Privacy*. 2021;4:e143. https://doi.org/10.1002/spy2.143