Secure Delivery Method for Preserving Data Integrity of a Video Frame with Sensitive Objects

You-Rak Choi , Do-Yeob Yeo , Yunhee Kang *

Posted Date: 27 February 2025

doi: 10.20944/preprints202502.2137.v1

Keywords: Video data frame; Data integrity; Authenticity; Logger; XOR encoding; Sensitive object

Article

Secure Delivery Method for Preserving Data Integrity of a Video Frame with Sensitive Objects

You-Rak Choi [1,†,‡], Do-Yeob Yeo [1,†] and Yunhee Kang [2,*]

[1] Div. of Nuclear System Integrity Sensing and Diagnosis, Korea Atomic Energy Research Institute (KAERI)
[2] Div. of Computer Engineering, Baekseok University
* Correspondence: yhkang@bu.ac.kr; Tel.: +82-41-550-0504
† 989-111 Daedeok-daero, Yuseong, Daejeon 34057, Republic of Korea: Nuclear System Integrity Sensing and Diagnosis Division, Korea Atomic Energy Research Institute (KAERI).
‡ These authors contributed equally to this work.

Abstract: Data integrity and authenticity verification are essential requirements to provide reliability for applications running in a video delivery system. We design a prototype system for handling data integrity as well as encoding sensitive objects in a video frame. This system is composed of two components, a logger and a verifier; The logger operates a **cryptography hash function** of a feature vector of a video data frame and an **XOR encoding scheme** used for hiding privacy information of sensitive objects in the video data frame. The XOR encoding scheme is operated at **the encoder**, as part of the logger. The hash value outputted from the hash function is signed by a private key owned by the logger. It is listed into one of attributes in a metadata of the video frame. In the key distribution phase for the authenticity verification process, the logger maintains the private key and the public key is stored on the verifier by handshaking. In order to reduce the processing load for data integrity verification, a hash tree configured for each stream of sensor data is used to minimize verification time. This paper is mainly focused on the design of a logger for *data integrity verification* and *encoding of sensitive objects*; To evaluate performance of the proposed method, we build a prototype in a client-server manner. The designed method operated on the logger uses to mitigate the risk of video data frame tampered and to help significantly in preserving sensitive objects securely.

Keywords: video data frame; data integrity; authenticity; logger; XOR encoding; sensitive object

1. Introduction

1.1. Background

Recently, digital image integrity has become an important research area within digital content management. The integrity of image data refers to the completeness and soundness (error detection) of the content record [1].

Digital content, including text, video, and sound, is created in various fields. It is then transmitted across various network environments to repositories, which may be located on servers, in clusters, or in the cloud, for storage and use [2]. Digital content has unique and essential characteristics that make it susceptible to alteration, deletion, and corruption. Ensuring the integrity of digital content is a fundamental task in information technology to ensure safe data management [3].

The General Data Protection Regulation (GDPR) is a comprehensive European Union(EU) law protecting the privacy and personal data of individuals within the EU. It strictly regulates organizations' collection, storage, and processing of such data [4,5]. Under the GDPR, protecting personal data against unauthorized access, alteration, or disclosure is crucial. In terms of integrity and confidentiality, data must be processed in a manner that ensures appropriate security, including protection against unauthorized or unlawful processing and accidental loss, destruction, or damage.

 doi:10.20944/preprints202502.2137.v1

1.2. Motivation

Verification of data integrity is crucial for the correct use of digital content, and ensuring the efficiency of network bandwidth used for verification data is essential [6]. However, many commercial data acquisition and storage systems lack an integrity verification function to detect content modifications such as forgery and deletion. Figure 1 shows the original image and its altered version.

(**a**) (**b**)

Figure 1. (**a**) is original image (**b**) modified image.

The purpose of this paper is to preserve and protect sensitive data from unauthorized access throughout its lifecycle, while maintaining the ability to access and utilize the original image when necessary. By encoding sensitive objects, the system mitigates risks associated with unauthorized access or misuse while ensuring data integrity for authorized purposes. Annotating sensitive data in images is crucial for privacy protection and compliance with data regulations. In this process, to protect a sensitive object from exposure, such as privacy infringement, a protection technique involving secure transmission of the sensitive object is employed [7].

1.3. Contributions

In this work, we define a digital fingerprint that is an output of the cryptographic hash function $SHA-256$, applied to a record of a data frame captured by a web camera. It captures the essential idea of unique identification of digital content using a cryptographic hash function. The digital fingerprint represents the unique value derived from a data frame and can originate from any device between edge devices and a server in a cloud. Its details are described in section 2.2.

The determination of the authenticity of digital records must maintain the continuity of evidence storage or the linkage of digital records [8]. During investigations of incidents caused by intentional contamination or accidentally exposed data, these fingerprints can be used to identify the incidents. Additionally, the logger, plays a role as a data creator, generates situational information by extracting object information included in the captured video. In order to maintain data reliability, authenticity, verification, a function that detects whether data has been modified by an unauthorized user, is required [9].

Designing an encoding and decoding algorithm can help preserve sensitive data that could be misused in deep-fakes. It can also help verify the authenticity of the original data and detect any unauthorized modifications. This paper presents a privacy protection method based on XOR operation, which offers several advantages, including simple and efficient computation, reversibility, enhanced security through the use of keys, diverse applicability, and ease of integration with additional security techniques.

The contributions of this paper are as follows.

- The designing an XOR encoding scheme with a filter to securely and efficiently handle both encoding and decoding operations within an edge device.

 doi:10.20944/preprints202502.2137.v1

- This scheme is designed to securely and efficiently handle both encoding and decoding operations within an edge device.
- We apply the fingerprint of an image as the value of a leaf node in the hash tree.
- The evaluatation of the performance of the data delivery method using a prototype based on an edge device.

2. Related Works

To verify the integrity of video data, there are methods such as a hash function, hash tree, digital signature, and blockchain [10–13]. In this section, we discuss the methods applied in this paper.

2.1. Hash Function

A hash function is used for any size of data and outputs a fixed size hash value corresponding to the data. It is the one way function where the return value is called a hash value. A hash function is a function that takes an arbitrary string of length as an argument and calculates a fixed length value n. The resulting value is called a hash value of the given string [14]. A cryptographic hash function is a type of hash function and has the following properties, making it difficult to find the relationship between the hash value and the original input value [14,15].

An important technique for confirming data integrity is to construct hash values for the data by a cryptographic hash function such as $SHA-256$ that maps of an arbitrary binary string to a binary string with a fixed size. The cryptographic hash functions is a a hash function that tries to make it computationally infeasible to invert them.

For large amounts of data, such as sets of images, ordered sequences of data blocks with variable sizes can be processed by generating hash values. Traditionally, a single hash value for a list or sequence of data blocks is computed by concatenating them in order and regarding the resulting concatenation as a long single input to the hash function. Clearly, the blocks need to be available in the right sequence in order to compute the overall hash value correctly.

2.2. Hash Tree

A hash tree is a hierarchical data structure where each leaf node represents a data block and is labeled with its corresponding cryptographic hash [16]. In a blockchain, a hash tree enables efficient data integrity verification without requiring the download of the entire blockchain. Each block contains a root hash, a cryptographic hash of all its transactions. The hash tree provides a tamper-proof way to summarize and verify the transactions within a block [17]. For instance, a Merkle tree is a tree data structure where each leaf node is labeled with the cryptographic hash of a data block, and each non-leaf node is labeled with the cryptographic hash of its children's labels. The hash tree used for integrity verification is created and used as a Merkle tree with a height of $O(log_2 n)$ and $2n-1$ nodes, for the given n verification target data. Merkle tree is also used in blockchain as a technique to ensure that block data has not been changed or damaged since it was recorded [18]. This allows efficient and secure verification of the contents of large data structures. Figure 2 describes the structure of a hash tree with leaves of five hash values, k_0 to k_4. $hash_0$ is the result of hashing the concatenation of $hash_{0-0}$ and $hash_{0-1}$, where $||$ denotes concatenation. $hash_{root}$ is the root of a hash tree given its leaves.

$$hash_0 = hash(hash_{0-0}||hash_{0-1})$$

$$hash_{root} = hash(hash_0||hash_1)$$

 doi:10.20944/preprints202502.2137.v1

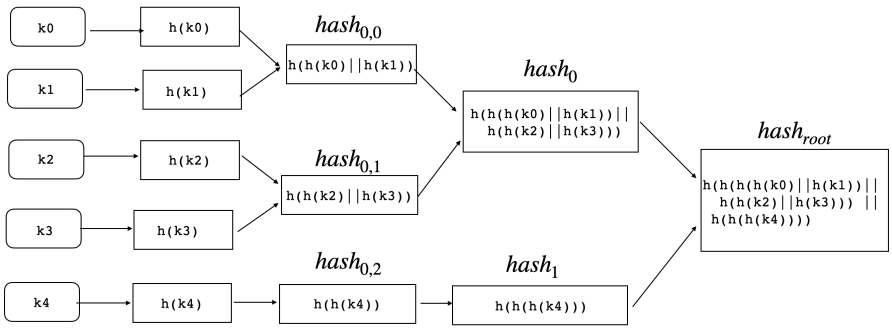


Figure 2. Structure of hash tree.

To verify the data integrity in the proposed method, we apply the hash tree to find whether the data frame is tempered or not. In the case of receiving a tempered image, the part of the image with tempered is detected by the verification method using the hash tree.

2.3. Digital Certificate

Public Key Infrastructure(PKI) is a public key infrastructure based on the X.509 standard that provides a chain of trust. PKI uses public-key cryptography to authenticate users and systems [19]. It relies on digital certificates issued by trusted certificate authorities (CAs). PKI is a combination of policies, procedures, and technologies for secure exchange of digital information online, using algorithms to protect messages and files and ensure delivery to the intended recipient. PKI authentication involves verifying the identity of users and systems by a trusted third-party certification authority. This centralized authority manages key generation, renewal, and revocation based on digital certificates. PKI employs two types of cryptographic keys: public and private keys. Together, these keys ensure the confidentiality, integrity, and authenticity of communications over the internet. Figure 3 illustrates the process of signing and verifying a message between Alice and Bob. Alice generates a public-private key pair and sends the public key to Bob, who then uses it to verify the received message.

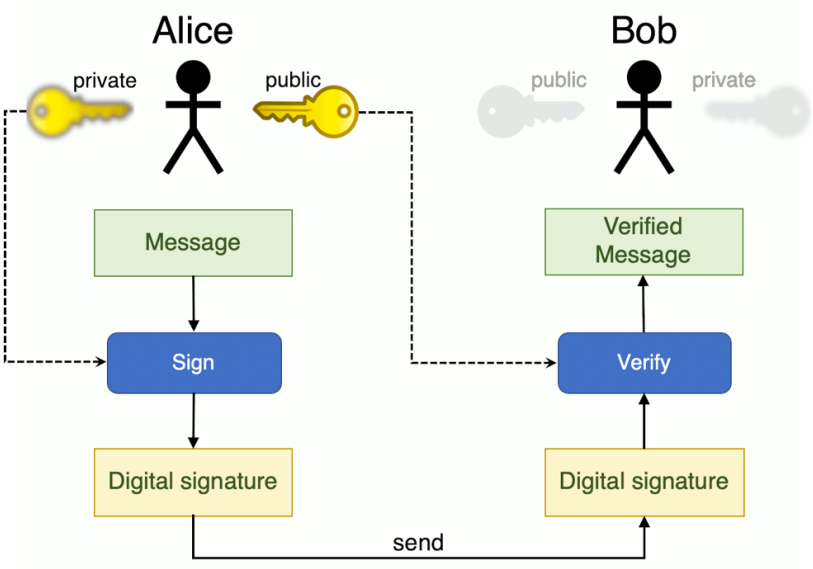


Figure 3. Overall process of sign and verify using PKI keys (*https://en.wikipedia.org/wiki/Digital_signature*).

3. Methods

3.1. System Structure Design

Figure 4 shows the software architecture associated with the primary functions of the logger and the verifier in the designed system. The logger generates metadata during thread-based concurrent

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 27 February 2025 doi:10.20944/preprints202502.2137.v1

video frame handling. The logger generates a hash value for each video frame, signs it, and stores it on the verifier. As shown in Figure 4, CID is a given video identifier. The hash value of the video frame and its signature are delivered to the verifier from the loggger by the thread.The verifier performs hash verification, constructs a hash tree to verify future video changes, and performs hash tree verification. The verifier stores the integrity verification results on the repository.

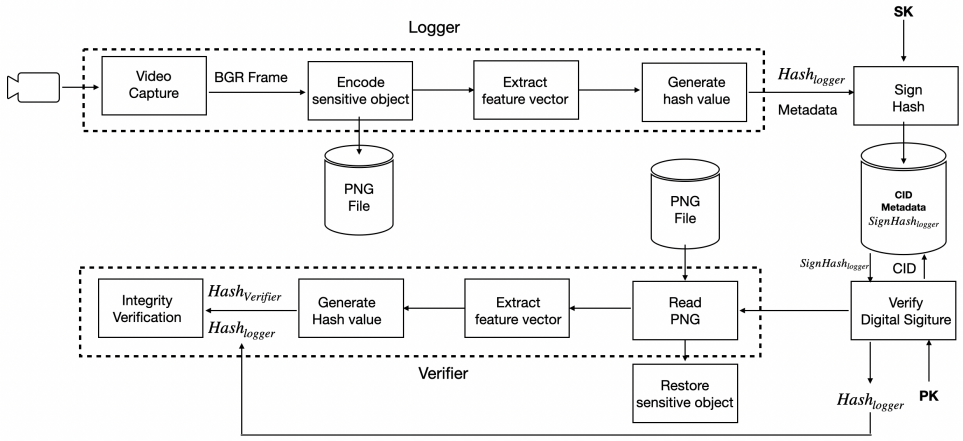


Figure 4. Overall system structure with logger and verifier.

For data life-cycle management, a logger plays a role as a software component that records events while a digital content producer captures an image. The producer then checks for the presence of a sensitive object within the image. If found, the sensitive object is encoded. The encoded object, along with related metadata, is securely delivered to a verifier. Finally, the original image is recovered through decoding. Further details are provided in the subsequent section 3.2.

The designed logger is used in an edge-based environment to extract features from an image for integrity verification. By extracting features at the edge, which is close to the data source, it ensures low latency in acquiring the information required for integrity checks and delivers it to the verifier, enabling real-time processing and facilitating the design of a distributed architecture. The system for verifying the integrity and authenticity of digital video data consists of a logger and a verifier. Lists below are the requirements for the logger and verifier.

- The verifier receives video data from the logger, which is the module responsible for acquiring video data, stores it, and performs verification of the video data.
- Communication between the logger and the verifier is based on Representational State Transfer(REST), using a request-response model, and data transmission and processing are carried out by defining message specifications in advance.
- The logger and verifier operate as entities responsible for signing and verification.

3.2. Delivery Scheme

The XOR operation is a simple and fast bitwise operation used to hide information. Due to its reversibility, the XOR operation exhibits the characteristic of restoring the original data when performed twice with the same key. In the process of transmitting metadata, XOR operation is utilized for encoding to conceal information, and XOR operation is also used for encoding sensitive objects. For the filter for encoding sensitive objects, encoding is performed with a separately configured seed. A seed is computed using the CID of the original video data, along with a public key sharing with a logger and a verifier.

$$filter = seed \oplus data$$

$$seed = CID|public\_key$$

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 27 February 2025 doi:10.20944/preprints202502.2137.v1

After detecting objects within the video frame, if a sensitive object is identified, it is encoded to prevent exposure by the seed, and the proposed transmission scheme is applied. It works by combining the secret data with a given seed. This pattern is then embedded into the video frame, To restore the encoded data, the same key is used to XOR the modified values, revealing the original information. Figure 5 shows an encode data.

Figure 5. Encoded data using filter.

Secure delivery of metadata enhances data privacy by configuring the PKI key pair used for signing as a filter and performing an XOR (exclusive OR) operation to verify authenticity [19,20]. A secure transmission scheme for transmitting a metadata after encoding it with a filter is expected to infringe privacy within video data. Figure 6 shows the logical structure of secure delivery scheme.

The verifier interprets the video and metadata received from the logger and stores them in a database. The stored video and metadata are then checked for integrity through the verifier. The integrity check is divided into two parts: authenticity verification through the validation of the signed hash value, and data integrity verification by comparing the hash value generated from the video with the received hash value.

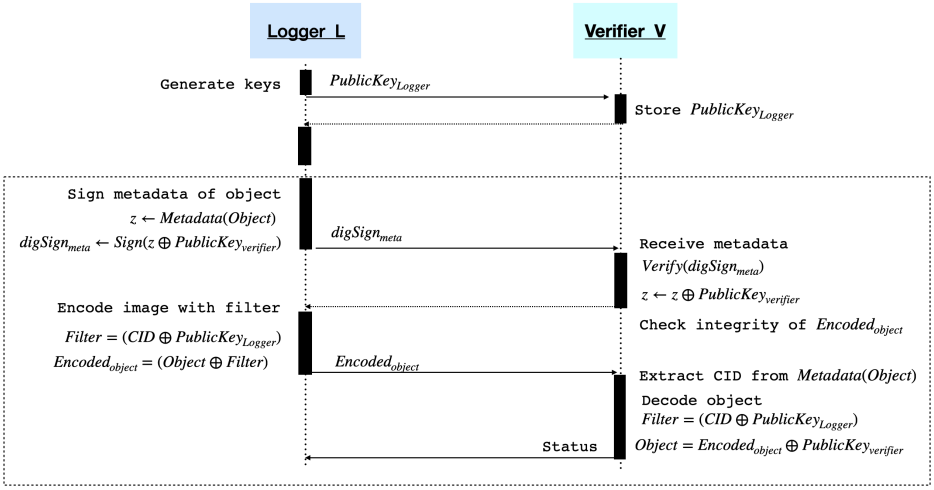


Figure 6. Secure delivery of metadata and sensitive objects.

3.3. Construction of Digital Fingerprint

Table 1 presents the fingerprint generation process for ensuring data integrity. Key generation and exchange are performed by the logger. The logger generates a secrete key (SK) and a public key (PK), storing the private key in a secure location and transmitting the public key to the verifier. The verifier uses the public key for subsequent operations. The logger generates a hash value for each video frame, signs it, and constructs a hash tree with 30 video frames, transmitting the root hash value. The verifier then conducts hash verification using the root hash, detecting any video frames with integrity violations.

 doi:10.20944/preprints202502.2137.v1

Table 1. Fingerprint Generation Procedure.

Phase	Description	Remarks
1	Generate a key pair, *pk* and *sk*	
2	Transfer *pk* to verifier	
3	Generate a feature of given image	Canny edge detection
4	Generate a hash value	SHA-256
5	Sign the hash value	
6	Build a hash tree	Merkle tree

3.4. Multi-Thread Logger

To ensure data integrity and authenticity, the logger creates digital fingerprints and signatures using hash functions. These are transmitted to a Video Management System(VMS) that is composed of a repository and a verifier. By collaborating with an object detection server, the logger extracts features from video data to construct a unique digital fingerprint. This fingerprint, along with the original data, is used for integrity verification in the VMS.

The functions of the logger list as follows:

- **Captureing an image**:Video data is captured through a camera. The camera is connected via USB, and after reading the shape file, video capture is performed according to the shape information. The captured video data in this paper is in BGR format. BGR (Blue, Green, Red) is a color format where the color values are stored in the order of Blue, Green, and Red.
- **Encoding sensitive objects**: BGR image is passed to the encoder with object detection. Encoded videos are stored in a separate folder. The file name for video storage is assigned a sequential number using the MAC address of the logger as a prefix for logger identification. Since the currently operating edge device of the logger does not have a real-time clock(RTC), sequential numbers representing logical time are used for information to determine the chronological order of file creation.
- **Extracting a feature vector** : The BGR data of the captured video is converted to grayscale. Canny Edge detection is used for constructing the feature vector of the video. A feature vector is a numerical representation of a video frame. The grayscale conversion is performed by calling the OpenCV library.
- **Making a hash value**: Hashes are generated for the feature vectors. $SHA-256$ is a cryptographic hash function that generates a fixed-size 256-bit (64-character) unique hexadecimal. The hash value generated by $SHA-256$ is represented as a digital fingerprint, ensuring the integrity of the data.
- **Signing the hash value**:A digital signature, generated by signing the hash with a private key, confirms its origin from a specific logger.
- **Sending image and its metadata**: Video data and metadata are transmitted to the VMS. The HTTP-based REST((Representational State Transfer) protocol is used for data transmission.

3.5. Scheme for Encoding and Decoding Sensitive Objects

In this work, we encrypt sensitive data in transit and at rest. We use a designed encoding and decoding algorithm to protect data transferred and stored on servers in a cloud. When transferring data, we do not apply any cryptographic protocol like TLS/SSL designed to secure communication over a network. Instead of using cryptographic protocol, a logger initiates a connection to the VMS server. The designed encoding and decoding algorithm consists two phases, handshake and encoding

 doi:10.20944/preprints202502.2137.v1

1. (*Handshake*) The logger sends its digital certificate with a public key, part of a key pair in PKI, to the server. The server verifies the certificate to ensure that it's valid and issued by a trusted Certificate Authority (CA). If the certificate is valid, the logger sends the logger a filter, which plays a role as a shared secret key.
2. (*Encoding*) All subsequent communication between the client and server is encrypted using the shared secret key. This encryption ensures that any data transmitted is unreadable to anyone who might intercept it.

In the proposed sensitive data safe transmission scheme, the object detection model detects an object in an image, and when a sensitive object is identified, the sensitive object is encoded to prevent exposure and the proposed transmission scheme is applied. To ensure the safe delivery of sensitive objects, the PKI public key used for signatures is configured as a filter and XOR is performed to strengthen the data hiding function.

Data encoding is performed on sensitive objects within videos, such as human figures, using the YOLOv8 [21,22] object detection model. The encoding and decoding processes are as follows. This paper does not cover the eight integrity check processes.

1. The logger generates a public-private key pair based on PKI and transmits the public key to the verifier.
2. The logger obtains object information from video data using an object detection model.
3. Based on the object information of identified sensitive objects, the logger applies a filter to perform encoding.

4. Results

4.1. Experimental Setup

Embedded systems such as Raspberry Pi have smaller capacities and lower processing speeds than general PCs in terms of memory and CPU. They are used as a video frame in an edge device of a data creator. Therefore, memory savings and processing efficiency in embedded systems are major essential considerations in application development, and this should be reflected in the implementation of the integrity verification function. Jetson Nano Orin is one of embedded computing platforms developed by NVIDIA. The constructed hash tree is used to efficiently detect integrity violations in a data verifier. The Jetson Nano serves as an AI accelerator for object detection, employing the OpenSSL library for key generation, signing, and decryption. It utilizes the curl library to transmit video data via REST API. Figure 7 illustrates the physical hardware configuration.

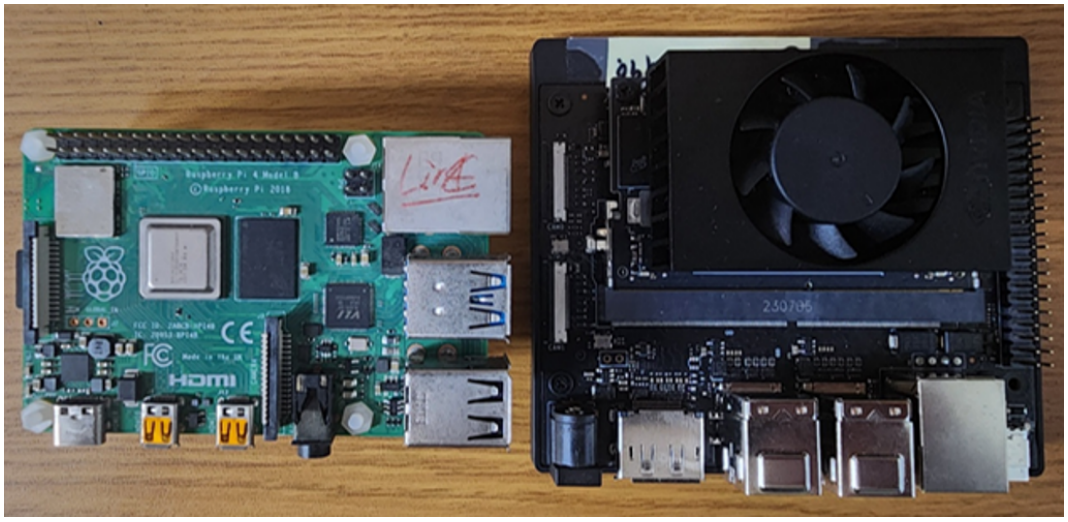


Figure 7. HW device: Raspberry Pi(left) and Jetson Nano Orin(Right).

 doi:10.20944/preprints202502.2137.v1

The experimental setup is shown in Table 2. Jetson Nano Orin is a compact, system-on-module (SOM) from NVIDIA, designed for edge computing and AI applications. It is based on NVIDIA Orin architecture with an ARM Cortex-A78AE CPU and Ampere GPU. It is running on Jetpack 5.1.4-b17. YOLOv8 model is used to detect objects in a video frame. To run the object detector using YOLOv8, Jetson nano orin is applied to run it, which is CUDA-supported. The Raspberry Pi is a series of small, affordable, single-board computers (SBCs) developed by the Raspberry Pi Foundation, Raspberry PI 4 is used for running a logger, which uses a webcam connected to USB.

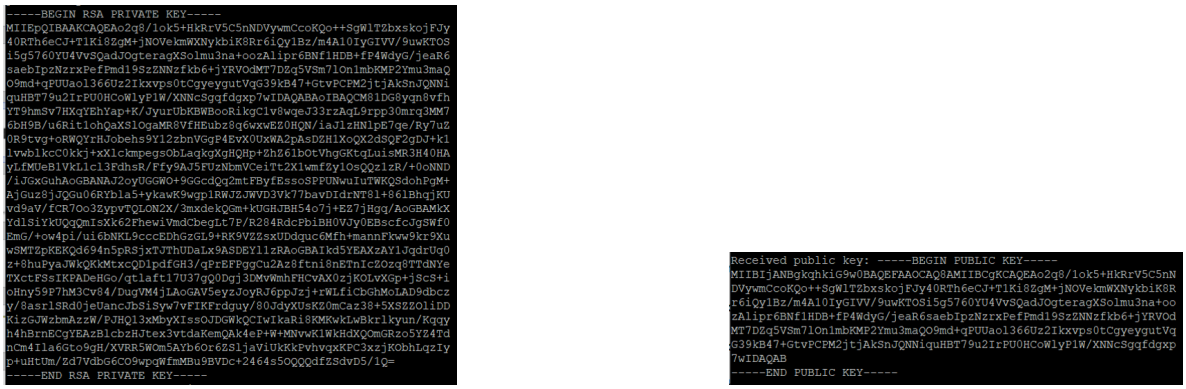
Table 2. Experimental setup for the designed system.

Type		Name	Specification
Object detector	HW	Jetson Nano Orin	8G byte (Memory)
	SW	OS	Ubuntu 20.04.6LTS
		Language	Python 3.8.10
		Libraries	PyTorch 2.1.0
			Torchvision 0.16.1
			YOLOv8
Logger	HW	Raspberry PI 4	8G byte (Memory)
		WebCam	Logitech Webcam c270
	SW	OS	Raspbian
		Language	gcc 12
		Libraries	OpenCV 4.10.0
			Curl 7.1
			OpenSSL 1.1.0

4.2. Result of a Generated Key Pair

In this experiment, a key pair with a public key and a private key is generated in a logger using the OpenSSL library, and the private key is maintained in the wallet of the logger. The public key associated with the generated private key is delivered to the verifier by the key exchange scheme. The key pair is in PEM format that is used to store and transmit cryptographic keys. PEM format is text-safe format, which encoded using Base64. We apply the secure PEM format for transmitting the public key from the logger to the verifier over the network. Figure 8 shows the key pair.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 27 February 2025 doi:10.20944/preprints202502.2137.v1



(**a**) (**b**)

Figure 8. (a) Private key, (b) Public key.

4.3. Result of Preprocessed Image

To extract features from threat-related images associated with Gemini, the images are first converted from the BGR format to grayscale. Then, Canny edge detection is applied to the converted grayscale images. This process is performed using the OpenCV library. Figure 9(a) shows the image converted to gray scale, and Figure 9(b) shows the Canny edge detection image of the gray scale image.

(**a**) (**b**)

Figure 9. (**a**) gray scale image (**b**) transformed image by Canny edge detection.

4.4. Result of Delivering Image and Its Metadata

The logger transmits metadata about images and videos to the Video Management System (VMS). In this process, the logger uses the HTTP-supporting curl library. Images are transferred using the POST method, and image-related metadata is formatted in JSON and also transmitted via POST. Figure 10 shows the metadata transmitted to the VMS. The object detection information includes object labels and bounding box information.

Figure 10. Generated Metadata.

 doi:10.20944/preprints202502.2137.v1

The transmitted images are stored as files, and the metadata is stored in the database. Metadata is data that describes other data. The metadata is described in Table 3.

Table 3. Metadata.

Name	Description
CID	unique ID of captured image
Hash	hash value of a feature vector in character format
Sign hash	digital signature of hash value
Object detection result	information of object with bounding box
Media type	media format like BGR

Figure 11 shows the server's received message from the logger, which includes CID(Content ID), *sign_hash* signed hash value formatted in BASE64, *hash* a hash value, object detection results, and media data format. The signed*sign_hash* value is used to verify the fingerprint obtained after feature extraction on subsequently received video. CID is a unique identifier for the content.

Figure 11. Received message.

4.5. Result of Hash Tree and Verification

According to the designed data verification process flow, it can detect whether data has been changed by using the hash value of the hash tree's root node (root hash) delivered from the logger, who is the digital content creator. Figure 12 shows a hash tree composed of four leaf nodes. Figure 13 shows the processing results after verification using the hash tree's root hash value, followed by integrity and authenticity verification."

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 27 February 2025 doi:10.20944/preprints202502.2137.v1

Figure 12. Constructed hash tree.

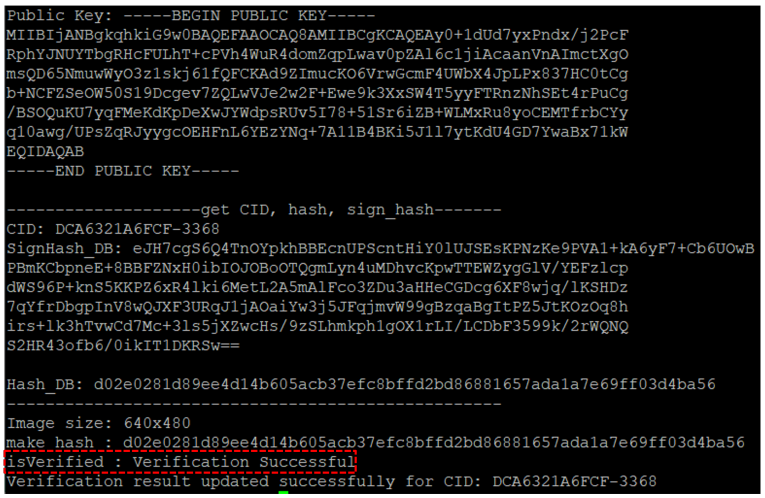


Figure 13. Result of verification.

4.6. Result of Encoding Sensitive Object

Figure 14(a) shows an image detected by the object detector interacted with logger, containing sensitive information such as human objects. Therefore, object detection must be followed by information obfuscation of these objects. Figure 14(b) presents the encoding results of the sensitive object by the logger, and the verifier recovers the sensitive object through XOR operation with the public key. The encoded image is used for integrity checking. Both the encoded image and the object information, including the contours of the sensitive object, are transmitted to the REST server using the POST method.

Performance evaluation indicates that the Jetson device is capable of real-time object detection, with an average processing time of 65 milliseconds per image.

(**a**) (**b**)

Figure 14. (a) Image with objects detected by Yolo8, (b) Encoded the image contained hidden human objects.

 doi:10.20944/preprints202502.2137.v1

5. Performance Evaluation

5.1. Evaluation of logger threads

Table 4 presents the performance evaluation results of threads constituting the logger. Processing performance is compared by calculating the average execution time and standard deviation of the execution time for each individual thread. Among the total of eight threads, the object detection thread has the longest execution time at 76 ms. This thread is configured to operate using a GPU and includes the encoding time of sensitive objects. Figure 16 presents the elapsed time and standard deviation for each thread within the logger.

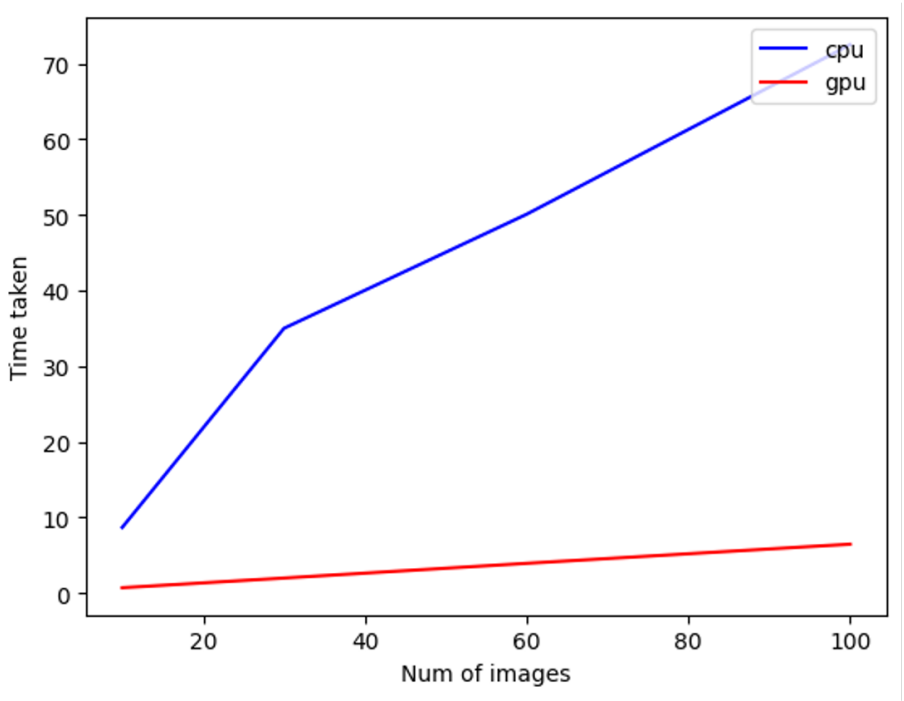


Figure 15. Elapsed time and std. deviation per logger threads.

Table 4. Elapsed time of detecting object

No	Thread	Performance Index	
		Mean	Std Dev
1	Capture thread	31.64	6.73
2	Image save thread	19.12	13.86
3	Convert frame thread	5.56	3.50
4	Object detection thread	76.00	2.62
5	Feature extraction thread	4.97	2.62
6	Make hash thread	26.10	3.99
7	Sign hash thread	9.89	1.76
8	Image send thread	11.04	2.51

 doi:10.20944/preprints202502.2137.v1

5.2. Evaluation of Object Detection and Encoding

The majority of the encoding time for sensitive objects is determined by the object detection time, which relies on the inference performance of the YOLOv8 model. Table 5 illustrates the comparative performance of CPU and GPU for object detection, with the GPU demonstrating a considerable reduction in processing latency. In our experiments, we observed that object detection using a GPU was approximately 12.57 times faster than using a CPU for a set of 10 images. When the number of images was increased to 100, the GPU-based object detection was still significantly faster, at about 11.24 times the speed of the CPU. We also observed that the standard deviation of the object detection time for images was smaller when using a GPU compared to a CPU. This indicates that the object detection time remains consistent, leading to more stable and efficient processing. Our evaluation demonstrates that object detection utilizing GPU-accelerated Jetson equipment achieves an average processing time of 65 milliseconds per image, indicating the feasibility of real-time operation.

Table 5. Elapsed time of detecting object.

No of Images	CPU		GPU	
	Mean	Std Dev	Mean	Std Dev
10	8.67	1.09	0.69	0.04
30	34.99	0.10	1.97	0.05
60	50.07	0.12	3.91	0.06
100	72.49	5.24	6.45	0.05

Figure 16 shows the performance of encoding images using CPU and GPU.

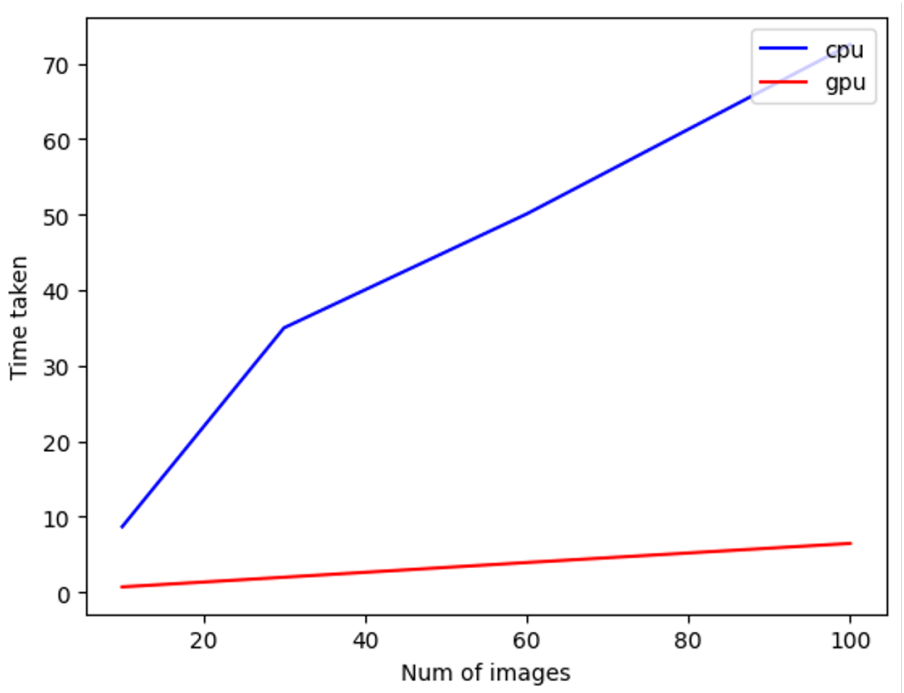


Figure 16. Elapsed time of encoding images.

6. Discussion

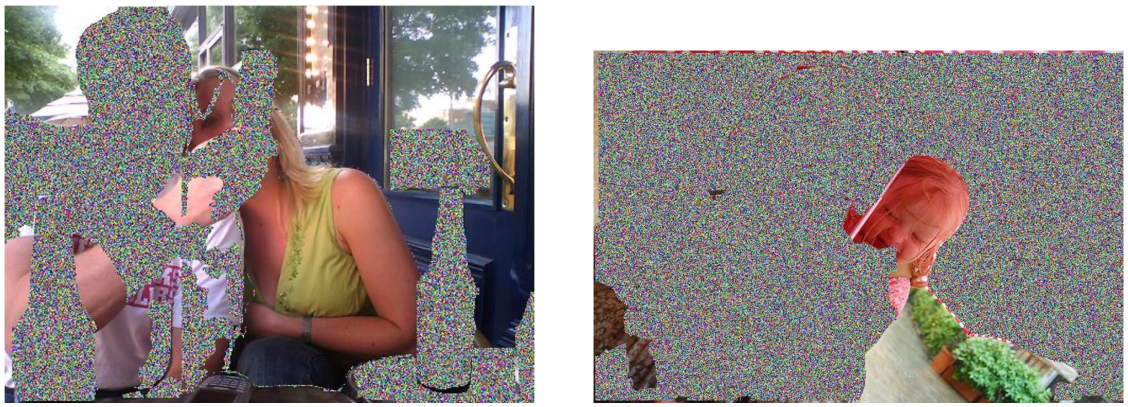
Video data frame generated from data sources verifies the cause of an accident when it occurs. To handle evidence of forensic, data stemmed from the evidence should be preserved the integrity. AI

 doi:10.20944/preprints202502.2137.v1

solutions like machine learning are used routinely across many industries. To ensure the governance and trustworthiness of AI models, it is necessary to introduce AI TRiSM, which stands for AI Trust, Risk, and Security Management. For this reason, ensuring rapid and media-independent integrity of training data is becoming increasingly important.

The results of this experiment confirmed that performing the XOR operation requires minimal time for encoding sensitive data during the image detection process. Furthermore, through parallel processing of multiple threads, feature extraction, metadata construction, and data transmission could be performed without delay for 30 images per second.

The YOLOv8 used in the experiment is an object detection model designed for general-purpose object recognition. During the experiment, as shown in Figure 17, it has limitations such as false positives and false negatives, which can lead to the exposure of sensitive objects. To address this issue, it is necessary to reconstruct the learning model by retraining and fine-tuning it on the dataset acquired from the given environment, thereby improving the accuracy of detected objects."



(**a**) (**b**)

Figure 17. Examples of encoding with false positives and false negatives

7. Conclusions

With the emergence of the Internet of Things environment where smartphones, CCTV, and sensors are connected, the amount of digital content generated from devices connected to the internet is rapidly increasing, and the demand for applications utilizing this content is growing.

Digital materials have the characteristic of being easy to change, and it is important to ensure integrity and authenticity in order to utilize digital materials. Integrity and authenticity verification is used for the purpose of verifying that digital data has been provided by the producer without any changes to its contents. The proposed integrity scheme is used for the purpose of verification of authenticity and integrity to confirm that the video frame was created and provided by a specific logger.

We designed a system for preserving data integrity based on a digital fingerprint generated from XOR encoding scheme. The system operates on a logger within an edge device. This paper focuses on the privacy concerns associated with transmitting video data. By leveraging an object detection model using a Jetson device, we have developed a novel automated encoding technique to obfuscate sensitive objects within video data. The proposed method employs a public key filter to ensure the security of sensitive information and can be applied to a wide range of applications. Through quantitative performance analysis of GPU-accelerated object detection, we observed results with minimal processing time variation, confirming the stability of the overall system performance.

Author Contributions: Conceptualization, Y.C. and Y.K.; methodology, Y.K.; software, Y.K.; validation, Y.C., D.Y. and Y.K.; formal analysis, Y.K.; investigation, D.Y.; resources, D.Y.; writing—original draft preparation, Y.K.; writing—review and editing, Y.K.; visualization, Y.K.; supervision, Y.K.; project administration, Y.C.; funding acquisition, Y.C. All authors have read and agreed to the published version of the manuscript.'

 doi:10.20944/preprints202502.2137.v1

Funding: This work was supported by the National Research Foundation of Korea through the Government (Ministry of Science and Information and Communications Technology) in 2022 under Grant RS-2022-00144000.

Data Availability Statement: The datasets used and/or analysed during the current study available from the corresponding author on reasonable request.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Zimmermann, A.; Schmidt, R.; Jain, L.C., Eds. *Architecting the Digital Transformation - Digital Business, Technology, Decision Support, Management*; Vol. 188, *Intelligent Systems Reference Library*, Springer, 2021. https://doi.org/10.1007/978-3-030-49640-1.
2. Jaspin, K.; Selvan, S.; Sahana, S.; Thanmai, G. Efficient and Secure File Transfer in Cloud Through Double Encryption Using AES and RSA Algorithm. In Proceedings of the 2021 International Conference on Emerging Smart Computing and Informatics (ESCI), 2021, pp. 791–796. https://doi.org/10.1109/ESCI50559.2021.9397005.
3. Sivathanu, G.; Wright, C.P.; Zadok, E. Ensuring data integrity in storage: techniques and applications. In Proceedings of the StorageSS; Atluri, V.; Samarati, P.; Yurcik, W.; Brumbaugh, L.; Zhou, Y., Eds. ACM, 2005, pp. 26–36.
4. Voigt, P.; Bussche, A.v.d. *The EU General Data Protection Regulation (GDPR): A Practical Guide*, 1st ed.; Springer Publishing Company, Incorporated, 2017.
5. Hjerppe, K.; Ruohonen, J.; Leppanen, V. The General Data Protection Regulation: Requirements, Architectures, and Constraints. In Proceedings of the 2019 IEEE 27th International Requirements Engineering Conference (RE). IEEE, 2019. https://doi.org/10.1109/re.2019.00036.
6. Fotiou, N.; Xylomenos, G.; Thomas, Y. Data integrity protection for data spaces. In Proceedings of the Proceedings of the 17th European Workshop on Systems Security, New York, NY, USA, 2024; EuroSec '24, p. 44–50. https://doi.org/10.1145/3642974.3652284.
7. Wu, C. Data privacy: From transparency to fairness. *Technology in Society* **2024**, *76*, 102457. https://doi.org/https://doi.org/10.1016/j.techsoc.2024.102457.
8. Duranti, L.; Rogers, C. Trust in digital records: An increasingly cloudy legal area. *Computer Law & Security Review* **2012**, *28*, 522–531. https://doi.org/https://doi.org/10.1016/j.clsr.2012.07.009.
9. Geus, J.; Ottmann, J.; Freiling, F. Systematic Evaluation of Forensic Data Acquisition using Smartphone Local Backup, 2024, [arXiv:cs.CR/2404.12808].
10. Hansen, T.; 3rd, D.E.E. US Secure Hash Algorithms (SHA and HMAC-SHA). RFC 4634, 2006. https://doi.org/10.17487/RFC4634.
11. Goldwasser, S.; Micali, S.; Rivest, R.L. A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIAM Journal on Computing* **1988**, *17*, 281–308.
12. Swan, M. *Blockchain : blueprint for a new economy*; O'Reilly Media: Sebastopol, Calif., 2015.
13. Gao, T.; Yao, Z.; Si, X. Data Integrity Verification Scheme for Fair Arbitration Based on Blockchain. In Proceedings of the 2024 4th International Conference on Blockchain Technology and Information Security (ICBCTIS), Aug 2024, pp. 191–197. https://doi.org/10.1109/ICBCTIS64495.2024.00038.
14. Cherckesova, L.V.; Safaryan, O.A.; Lyashenko, N.G.; Korochentsev, D.A. Developing a New Collision-Resistant Hashing Algorithm. *Mathematics* **2022**, *10*. https://doi.org/10.3390/math10152769.
15. Rogaway, P.; Shrimpton, T. Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance. In Proceedings of the Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004, Revised Papers; Roy, B.K.; Meier, W., Eds. Springer, 2004, Vol. 3017, *Lecture Notes in Computer Science*, pp. 371–388. https://doi.org/10.1007/978-3-540-25937-4_24.
16. Liu, H.; Luo, X.; Liu, H.; Xia, X. Merkle Tree: A Fundamental Component of Blockchains. In Proceedings of the 2021 International Conference on Electronic Information Engineering and Computer Science (EIECS), 2021, pp. 556–561. https://doi.org/10.1109/EIECS53707.2021.9588047.
17. Belchior, R.; Vasconcelos, A.; Guerreiro, S.; Correia, M. A Survey on Blockchain Interoperability: Past, Present, and Future Trends. *ACM Comput. Surv.* **2021**, *54*. https://doi.org/10.1145/3471140.
18. Merkle, R.C. A Digital Signature Based on a Conventional Encryption Function. In Proceedings of the A Conference on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology, Berlin, Heidelberg, 1987; CRYPTO '87, p. 369–378.

 doi:10.20944/preprints202502.2137.v1

19. Bashiri, K.; Fluhrer, S.; Gazdag, S.L.; Geest, D.V.; Kousidis, S. Internet X.509 Public Key Infrastructure: Algorithm Identifiers for SLH-DSA. Internet-Draft draft-ietf-lamps-x509-slhdsa-03, Internet Engineering Task Force, 2024. Work in Progress.
20. Rivest, R.L.; Shamir, A.; Adleman, L. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* **1978**, *21*, 120–126. https://doi.org/10.1145/359340.359342.
21. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 779–788. https://doi.org/10.1109/CVPR.2016.91.
22. Varghese, R.; M., S. YOLOv8: A Novel Object Detection Algorithm with Enhanced Performance and Robustness. In Proceedings of the 2024 International Conference on Advances in Data Engineering and Intelligent Computing Systems (ADICS), 2024, pp. 1–6. https://doi.org/10.1109/ADICS58448.2024.10533619.