Mesopotamian journal of Cybersecurity
Vol.5,No.1, **pp**. 121–146
DOI: https://doi.org/10.58496/MJCS/2025/009; ISSN: 2958-6542
https://mesopotamian.press/journals/index.php/cybersecurity

Research Article

Enhancing Crime Detection in Video Surveillance via a Lightweight Blockchain and Homomorphic Encryption-based Computer Vision System

Tanya Abdulsattar Jaber[1,*,]

[1] *Institute of Applied Arts, Middle Technical University, Baghdad, Iraq*

ARTICLE INFO

Article history
Received 04 Nov 2024
Accepted 18 Jan 2025
Published 12 Feb 2025

Keywords
Lightweight blockchain homomorphic encryption
VGG16 and the blockchain crime prevention system
Okamoto–Uchiyama cryptosystem

ABSTRACT

Blockchain technology consists of distributed ledgers or database systems, regarded as immutable, secure, and innovative, characterized by unsupervised internal maintenance with a special security protocol used to prevent inference from malicious or third parties. The widespread use of this technology has led to deep research into the problems posed by this technology, which can be summarized in terms of computational cost and latency time. The crime detection process in video surveillance has made great progress with the use of technologies such as the Internet of Things and blockchain technologies. However, to reach high levels of security in the physical crime detection process in which data are sent to servers via a computer network, there must be a high degree of security for Internet of Things systems related to the crime detection process. There has been a significant increase in the number of problems associated with crime detection in video surveillance systems, including the modification of surveillance data during transfer to and from servers. For this reason, establishing a reliable and secure system for transferring video surveillance data to servers has become a high priority. This paper presents a lightweight security system to protect data generated in the crime detection process, both from video surveillance cameras and the servers that store these data. The challenges related to IoT-based video surveillance cameras and monitoring and control centers have been considered, turning the system primarily into a decentralized system. In this paper, a lightweight blockchain system based on a proof of secret share consensus algorithm technology is proposed, along with the encryption of surveillance data via modified Okamoto–Uchiyama homomorphic encryption technology. The proposed system is evaluated via standard blockchain and security evaluation metrics, demonstrating efficient utilization of computational costs and realization of security, with a high scalability rate. The VGG16 deep learning model is employed in the proposed system to detect and classify criminal activities in surveillance videos. Owing to its ability to identify patterns and anomalies, the model achieved an accuracy of 94%, demonstrating a high level of performance in crime detection and prevention. Overall, the use of VGG16 provides an efficient and reliable approach for improving the security of public spaces and reducing the risk of criminal activity.

1. INTRODUCTION

In smart cities, crime prevention through the use of computer vision technology has become an increasingly important topic. One application of computer vision involves the use of surveillance cameras equipped with machine learning algorithms to detect violent behavior and identify potential perpetrators.

By analysing patterns of movement and behavior, a computer can identify when a person is acting aggressively and potentially about to commit a violent act. This information can then be relayed to law enforcement or security personnel, who can intervene before a crime occurs.

While the use of computer vision technology for crime prevention can be highly effective, it also raises concerns about privacy and the potential for abuse. It is important to have appropriate safeguards and regulations in place to ensure that this technology is used ethically and with the utmost respect for individuals' rights [1].

Blockchains can be described as autonomous distributed storage data systems, which can be divided into public, private, and hybrid unsupervised systems and provide the characteristic of immutability [19]. The term autonomous here refers to the operation of maintenance, which is performed without the interference of a third party, meaning that all participants are somehow responsible for the maintenance of the blockchain and that the protocols used ensure the elimination of malicious action or data corruption by means of many techniques, including time stamps, digital signatures, and hashing, to achieve consistency and security of the data.

*Corresponding author. Email: tanya.galxy@mtu.edu.iq

The blockchain should be distributed such that each participant possesses a copy of the blockchain database, guaranteeing that the data will not be lost or misused.
Immutability refers to the fact that the data within the blockchain cannot be changed for any reason. One variant of blockchain technology is lightweight blockchain technology, which aims to reduce the time and computational resources required for traditional standard blockchain systems and is designed to work within the context of the Internet of Things and mobile devices. Blockchain technology can be effective in preventing crime by providing immutable and transparent record-keeping, enabling smart contracts, decentralizing control, and facilitating digital identities [24][2].
Homomorphic encryption is a cryptographic technique that makes it possible to perform operations on encrypted text, such as addition, multiplication or both, thus increasing the security and privacy of encrypted data, which can be processed in a secure form.
Employing homomorphic encryption within crime prevention systems can enhance the security and privacy of the electoral process. This technology can be divided into three main types: partial homomorphic, somewhat homomorphic, and fully homomorphic [30][3].
Okamoto–Uchiyama homomorphic encryption is a partially homomorphic encryption algorithm in which addition is applied to encrypted text without decrypting it.
The contribution of this paper is the proposal of a new system that combines blockchain technology with a lightweight modified version of the Okamoto–Uchiyama homomorphic algorithm to secure the transmission and collection processes of crime prevention decision data from a system comprising multiple distributed surveillance cameras [31][4].
This paper presents a new consensus algorithm called proof of secret sharing (PoSS) that employs Shamir's secret sharing algorithm, which reduces the time required for the verification and validation of transactions and blocks.
These contributions can potentially improve the efficiency and security of crime prevention detection when operating and performing data sharing with monitoring and control centers, where crime prevention within smart cities is used as a case study.
The remainder of this paper presents basic information about blockchain technology and homomorphic encryption and a review of the relevant literature, taking into consideration different aspects of the integration of blockchain and homomorphic encryption, featuring sections in which such a system is proposed and evaluated.

1.1 dataset

The real-life violence situation (RLVS) dataset is a collection of over 10,000 video clips depicting real-world violent events, each labelled as containing violence or not, with the information provided on the type of violence, location, and number of people involved, spanning a total duration of approximately 140 hours and released in 2019 by a team of researchers from the University of Central Florida for use in developing and evaluating machine learning algorithms for violence detection; this dataset has been used in a number of research studies and competitions, including the IEEE International Conference. Table (1) shows the main description of the dataset employed within this work [5].

TABLE I. MAIN DESCRIPTION OF THE RLVS DATASET

Dataset	Real-Life Violence Situations (RLVS)
Description	A collection of video clips depicting real-world violent events
Total Duration	Approximately 140 hours
Number of Clips	Over 10,000
Violence Labels	Each clip is labelled as either containing violence or not
Type of Violence	Information about the type of violence is included for each clip
Location	The location of the event is provided for each clip
Number of People	The number of people involved in the event is provided for each clip
Use Cases	Developing and evaluating machine learning algorithms for violence detection
Release Date	2019
Research Studies	Used in a number of research studies and competitions, including the IEEE International Conference on

2. THEORETICAL BACKGROUND OF BLOCKCHAIN

Blockchain consists of a chain of blocks that contain transaction details and some security information that are used to secure cryptocurrency or data exchange. The content of the block is split into two main parts: the header, which contains the hash value of the previous connected block, and the body, which mainly contains the transaction and time stamp, along with other related information. The first block of the chain is usually called the genesis block, and its initial data are provided by the owner of the chain, whereas other blocks are connected as a linked list to this chain, with each hash being calculated via a cryptographic hash and saved to the block, and each block saves the hash of the previous block, thus securing the block against modification. A hacker changing the content of one of the previous blocks will result in the block being invalid because the domino effect of the previous hashes makes the tampered-with hash invalid [6].

If a hacker aims to change the content of the current block, they need to change the consecutive blocks from the beginning to achieve an acceptable consensus.
The blockchain reaches high levels of data security via the following:

1. Decentralization: In a traditional system, the fact that all data are saved within a single location makes it vulnerable to cyberattacks, whereas in a blockchain peer-to-peer network, the data are saved in decentralized nodes, making the system harder for an attacker to compromise.
2. Encryption: The use of an encryption algorithm to encrypt the block's data makes the system more secure and the data more resistant to modification.
3. Immutability: the data saved within the blockchain cannot be modified or deleted, making tampering within the blockchain more difficult and ensuring the integrity of the data.
4. Consensus protocol: blockchain relies on the consensus mechanism for the validation and verification of the transaction, ensuring that only valid transactions are merged with the blockchain and that attackers are unable to add fraudulent data.
5. Smart contract: an agreement signed between two sides (usually referred to as a buyer and a seller) included within the blockchain code, where it cannot be modified once it has been agreed upon [7].

2.1. Consensus Algorithm

The Byzantine generalization problem inspired the derivation of the consensus algorithm. As explained previously, only honest generals participate in deciding whether to attack or not, and to determine whether to attack or retreat, each generally sends their order to all generals and receives orders from all generals. A traitor general may send an incorrect order to others. This technique helps to achieve consensus in the process.
In a blockchain, the consensus algorithm is mainly employed to ensure data consistency in the case of failures within the nodes connected via the distributed network; failure nodes can be either Byzantine fault nodes or crash nodes. Byzantine fault nodes behave arbitrarily, whereby either the message sent by the node could itself be wrong or different messages could have been sent to different nodes to undermine the consensus among the nodes. Moreover, crash fault nodes result from system halting, which means that the node does not function properly, but there is no suspicious or malicious behavior, thus limiting the impact to message delay or message loss [7].
If the system is faced with a crash fault node, the process of reaching a consensus among the nodes is generally considered easy, and many algorithms have been proposed to address crash fault nodes.
A more difficult problem is represented by Byzantine fault nodes, which need to be handled to achieve consensus; therefore, a consensus algorithm should be designed following a specific standard and should meet the following requirements:

1. Consistency: A valid transaction from any honest node is valid for every other honest node; this allows the blockchain to counteract any double-spending attack and eliminate it if the majority of nodes are honest.
2. Liveness: the validity of a single transaction that is sent by an honest node within the blockchain should eventually be confirmed within the system, ensuring the validity of the system.

The consensus algorithm of the blockchain is considered correct only if it meets the consistency and liveness requirements. In addition, other aspects could certainly be added to each algorithm to meet the needs of the application for which the algorithm is designed, such as increased scalability and throughput and a reduction in costs related to resources. The consensus algorithm is an old concept in the context of distributed systems (i.e., in traditional models), and there are many known consensus algorithms, including proof of work (POW), proof of stake (PoS), and practical Byzantine fault tolerance (PBFT) [8].

3. HOMOMORPHIC ENCRYPTION

Homomorphic encryption is a technique that enables secure computation of special encrypted data. This means that the encrypted data being processed can be processed in a secure manner without the need to decrypt it first. Homomorphic encryption has the potential to enable new applications and use cases not previously possible, such as secure data sharing and cloud computing. It is a rapidly developing field of research in cryptography and has the potential to greatly improve the security and privacy of data in a variety of applications [9].
One of the key advantages of homomorphic encryption is that it enables data to be shared and processed securely without revealing the raw data to any of the parties involved. This approach can be particularly useful in scenarios where multiple parties need to collaborate on a project or share data but where there are security concerns that prevent the raw data from being shared.
Another advantage of homomorphic encryption is that it allows for a high degree of privacy and security. Because the data remain encrypted throughout the entire process, even the parties performing the mathematical operations on the encrypted

data cannot see the raw data. This makes it much harder for any unauthorized party to access the data, even if they are able to intercept the encrypted data as they are being transmitted [10].

Overall, homomorphic encryption is an exciting area of research in cryptography that has the potential to revolutionize the way that data are shared and processed. While there are still many challenges and open questions in this field, the potential applications of homomorphic encryption are numerous and could have a significant impact on the way that data are handled in the future [11].

Homomorphic encryption and blockchain technology are two separate fields, but they have the potential to be integrated in various ways. For example, homomorphic encryption could be used to enable the secure sharing of data on a blockchain platform. This would allow data to be encrypted and then stored on the blockchain, where they could be processed and manipulated without the need to decrypt them first. This could be useful for protecting sensitive data on the blockchain and enabling secure collaboration between multiple parties on blockchain-based projects.

Another potential use for the integration of homomorphic encryption and blockchain technology is in the area of secure transactions. Homomorphic encryption can be used to encrypt financial data such as credit card numbers and then perform mathematical operations on the encrypted data to verify the authenticity of the transactions without revealing the raw data. This could be integrated with blockchain technology to enable secure, transparent, and auditable financial transactions on a decentralized platform.

Overall, the integration of homomorphic encryption and blockchain technology has the potential to enable a wide range of new applications and use cases. As these two fields continue to evolve and develop, we are likely to see an increasing number of examples of how they can be integrated to provide enhanced security and functionality [12].

Many modern applications can employ homomorphic encryption and blockchain, such as the integration of blockchain with the IoT [13], the preservation of privacy when artificial intelligence techniques are used, especially in computer networks [14], data retrieval via smart contracts [15], key generation via smart boxes and PLL algorithms [16], saving keys generated via magic cubes [17], and network packet security in SDN [18].

3.1. The Okamoto–Uchiyama Cryptosystem [20]

The Okamoto–Uchiyama cryptosystem is a public key encryption algorithm that possesses homomorphic characteristics that works with (Z/nZ)*.

The Okamoto–Uchiyama algorithm accepts integer inputs since all the characters or groups of characters are represented by integer values used within the secret communications between parties.

The key generation algorithm steps are explained in Algorithm 1, **w**here two values are selected randomly—P and Q—and then *N* values are calculated by multiplying the square of P with Q; then, the g value is generated according to the condition where $g \in \{2 \ldots. N^{-1}\}$ is chosen such that $gp^{-1} \not\equiv 1 \bmod p^2$, and finally, the value of H is calculated.

Algorithm 1 Okamoto–Uchiyama key generation

Input: randomly selected values
Output: public and private pairs
Step1: generate two large primes P and Q
Step2: compute n as follows:
$$N = P^2 * Q$$
Step3: choose $g \in \{2 \ldots. N^{-1}\}$ *such that* $gp^{-1} \not\equiv 1 \bmod p^2$
Step4: compute H as follows:
$$H = g^N \bmod N$$
Step5: public keys (n, g, h), private keys (p, q)

The encryption process of Okamoto–Uchiyama is described in Algorithm 2.

Algorithm 2 Okamoto–Uchiyama encryption

Input: plaintext, keys
Output: cipher text
Step1: randomly select a value of r that is between 1 and n-1
Step2: compute c as follows:
$$C = g^m h^r \bmod N$$

The decryption process of Okamoto–Uchiyama is carried out by calculating the values of A and B and the inverse of B with reference to module P and the value of the original message, which is found by multiplying these values Modulo p, as shown in Algorithm 3.

Algorithm3 Okamoto–Uchiyama decryption

Input: cipher text, keys
Output: plaintext
Step1: compute the value of A as follows:

$$A = \frac{(c^{p-1} mod\ p^2)-1}{p}$$

Step2: compute the value of B as follows:

$$B = \frac{(g^{p-1} mod\ p^2)-1}{p}$$

Step3: compute the inverse of B Modulo P as follows:

$$B' = B^{-1}\ mod\ P$$

Step4: compute m as follows:

$$M = A\ B'\ mod\ P$$

4. VGG16

VGG16 for crime prevention could be used to analyse surveillance camera footage and detect suspicious behavior or individuals. The network could be trained on a large dataset of labelled images, such as people who loitered, carry suspicious objects, or engage in other criminal activities. The trained network could then be used to process real-time video feeds from surveillance cameras and flag any instances of suspicious behavior for further investigation by law enforcement personnel.
Table (2) displays the description of the main layers in VGG16 and the characteristics of each layer.

TABLE II. VGG16 MAIN LAYER DESCRIPTIONS.

Characteristics	Description
Architecture	Convolutional neural network
Layers	16 layers, including 13 convolutional layers and 3 fully connected layers
Input size	Fixed input size of 224 x 224 pixels
Preprocessing	Mean subtraction of RGB pixel values
Activation function	Rectified Linear Unit (ReLU)
Pooling layers	Five max pooling layers
Output layer	Softmax layer with 1000 nodes corresponding to 1000 ImageNet classes
Training	Trained on ImageNet dataset with 1.2 million images
Accuracy	Achieved top-5 error rate of 7.32% on ImageNet test set
Applications	Image classification, feature extraction for transfer learning, object detection, semantic segmentation, and more.

The specifications of each layer are shown in Table (3).

TABLE III. VGG16 LAYER SPECIFICATIONS

Layer Type	Layer Name	Output Shape	Number of Parameters
Input	Input	(224, 224, 3)	0
Convolutional	Block 1 Convolutional Layers	(224, 224, 64)	179,136
Max Pooling	Block 1 Max Pooling	(112, 112, 64)	0
Convolutional	Block 2 Convolutional Layers	(112, 112, 128)	802,816
Max Pooling	Block 2 Max Pooling	(56, 56, 128)	0
Convolutional	Block 3 Convolutional Layers	(56, 56, 256)	1,605,632
Convolutional	Block 4 Convolutional Layers	(56, 56, 256)	1,605,632
Max Pooling	Block 4 Max Pooling	(28, 28, 256)	0
Convolutional	Block 5 Convolutional Layers	(28, 28, 512)	6,418,176
Convolutional	Block 6 Convolutional Layers	(28, 28, 512)	2,415,872
Max Pooling	Block 6 Max Pooling	(14, 14, 512)	0
Convolutional	Block 7 Convolutional Layers	(14, 14, 512)	2,415,872
Convolutional	Block 8 Convolutional Layers	(14, 14, 512)	2,415,872
Max Pooling	Block 8 Max Pooling	(7, 7, 512)	0
Dense	Flatten	(25,088)	0
Dense	FC1	(4,096)	102,764,544
Dense	FC2	(4,096)	16,781,312
Output	Output	(1000)	4,097,000

5. RELATED WORK

In [21], the authors proposed a system called “BeeKeeper”, which is an IoT system that employs blockchain to securely store data while applying homomorphic computations; interactions between blockchain nodes and IoT devices are managed

via smart contracts. Additionally, this system uses homomorphic computations to enable mathematical calculations to be performed on encrypted data, securing the storage data. This system was evaluated in terms of execution time and transaction throughput. The results showed that BeeKeeper is able to handle many transactions while maintaining integrity and confidentiality.

In [22], the authors proposed a data aggregation scheme that constituted a data preservation system for IoT data called "PrivDA", which employed blockchain and homomorphic encryption algorithms to ensure the privacy and security of aggregated IoT data, wherein each data consumer is able to generate a smart contract and data producers are put together who are able to answer consumer requests, selecting the aggregated ones; then, a group of results is requested, computed via homomorphic computations. This system was evaluated via standard blockchain evaluation methods, and the private Ethereum blockchain was used as the backbone.

In [23], a combination of blockchain and homomorphic encryption was proposed for privacy preservation purposes in the context of the aggregation and savings of data generated by smart home systems. This work aimed to verify the working nodes and the transactions via a node verification process and to suggest a new block data structure based on homomorphic encryption for smart home system devices, with each device information transaction being recorded. The Paillier partial homomorphic encryption algorithm was used to encrypt the sensitive data obtained from all of the peers in the smart home system and upload them to the blockchain. The authors analysed the encrypted data and verified its security, whereas attack experiments were conducted to show the effect of insecure nodes within the system. The results showed that this system demonstrated better results with respect to user privacy than did standard smart home systems.

The authors of [25] proposed a new protocol for electronic voting secured via sign encryption homomorphic encryption, which was used for both authentication and encryption. This protocol solves the challenges faced by traditional e-voting systems, such as the inability to verify ballots, low efficiency, and susceptibility to fraud, by employing homomorphic encryption and sign encryption algorithms for ballot signing and encryption, while homomorphic encryption was used to aggregate the data; this may lead to the elimination of the need for a third party to supervise the voting operation, and the votes are secured via blockchain. The authors claimed that this protocol was efficient, flexible, and secure.

Moreover, in [26], the authors proposed an electronic voting system that merged technologies related to homomorphic encryption such that operations could be performed on encrypted data, whereas blockchain is tampering proof due to the use of a distributed ledger. Thus, this system ensures the security and accuracy of the voting process, preventing duplicate votes and maintaining voter privacy.

In [27], the authors proposed an e-voting system in which a combination of blockchain and homomorphic encryption was used to realize privacy, encryption of voter data, system verifiability, and credibility. This system, instead of focusing on cast votes, focused on the encryption of voter information, meaning that, with the ability to securely analyse votes without preserving voters' identities, this system could also be used with different types of data rather than voting data.

In [28], a model was proposed that targeted personal healthcare data applications, where blockchain was used to securely save data and fully homomorphic encryption was used to secure the data when it was shared; this model combined the distributed hash table and the blockchain network to save and transmit metadata and data separately, with end-to-end encryption being promoted with the aim of reducing the unencrypted data present within the system. This system was employed to save the data records of COVID-19 patients, with the authors claiming that their system reduced the potential risks to user privacy and improved the interoperability of different health institutions.

In [29] and [30], a survey of human activity recognition that can be employed within crime prediction and smart cities was conducted, with the main description of each dataset.

In[31] authors proposes an advanced cybersecurity approach using flow-based anomaly detection with Min-Max Game Theory optimized Artificial Neural Networks (MMGT-ANN). By leveraging the KDD dataset and data-driven modeling, the system achieves enhanced precision and accuracy in detecting malicious activities while reducing time complexity compared to previous methods.

6. PROPOSED SYSTEM

This section provides a detailed description of the lightweight security system based on the blockchain (LSSBB) via the proof of secret sharing (PoSS) consensus algorithm, and the effects of the lightweight blockchain integrated with a homomorphic algorithm on security and the consumption of resources are explained.

Figure 1 presents a system diagram of the main lightweight security system based on the blockchain (LSSBB), where the system consists of three main parts, with each part representing a main process employed within the system and each part consisting of internal phases or processes that will be explained in detail later.

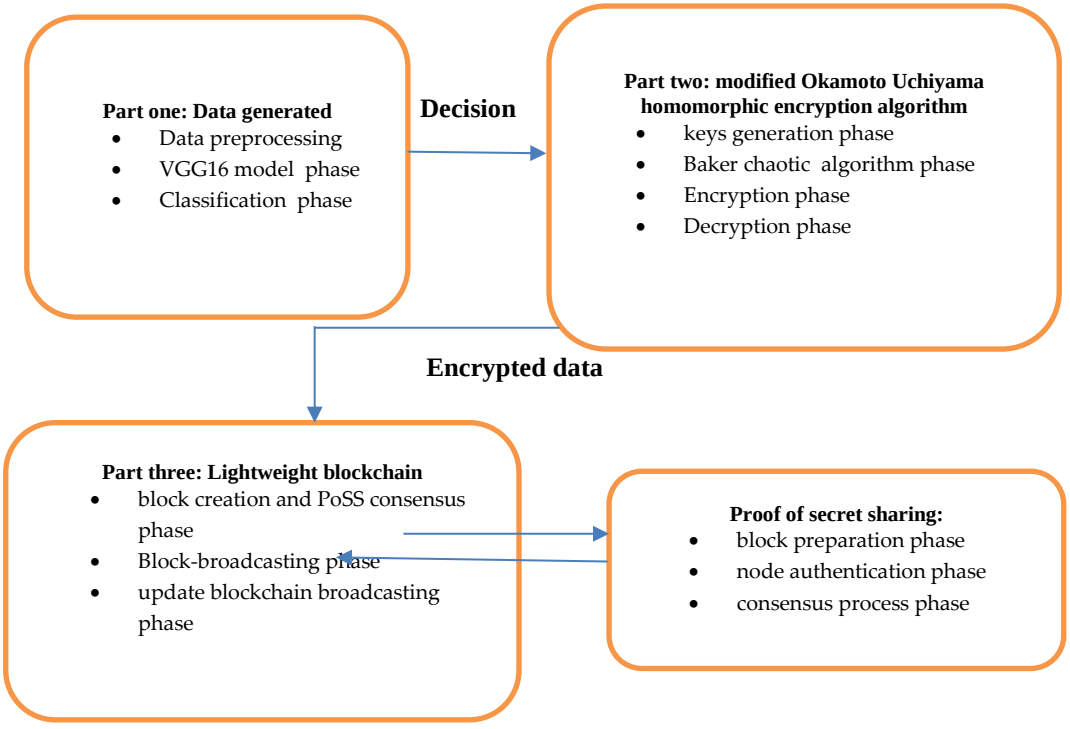

Fig. 1. System diagram of the proposed lightweight security system based on the blockchain (LSSBB).

1. Part one—data generation—involves data preprocessing, the VGG16 model phase, and classification via VGG16. The data preprocessing step includes key frame extraction, data cleaning, and frame resizing. The VGG16 model is used to extract features from the preprocessed frames, and the extracted features are used for classification of frames into different categories or classes. Data generation consists of three main steps:
   - Data preprocessing: This involves several tasks, including keyframe extraction, data cleaning, and frame resizing. Keyframe extraction involves selecting the most representative frames from a video sequence, whereas data cleaning involves removing any errors, inconsistencies, or duplicates in the data. Frame resizing is the process of adjusting the size of the frames to a standardized resolution for further processing.
   - VGG16 model phase: VGG16 is a deep learning model that is commonly used for image recognition tasks. In this phase, the VGG16 model is used to extract features from the preprocessed frames.
   - Classification VGG16 phase: In this phase, the extracted features are used to classify the frames into different categories or classes. This is typically done via VGG16
2. Part two—the modified Okamoto–Uchiyama homomorphic encryption algorithm— consists of four main phases:
   - Key generation phase: the public and private keys within the system are generated.
   - Baker chaotic algorithm phase: the K value is generated (the chaotic map algorithm is employed to generate the chaotic value employed within the algorithm).
   - Encryption phase: This step includes the encryption.
   - Decryption phase: This step includes decryption.
3. Part three—lightweight blockchain: In this part, a proof of secret sharing and a proof of homomorphic secret sharing are employed, which can be broken down into three main phases:
   - Block preparation phase: the transaction is collected into a block that is ready to add to the chain.
   - Node authentication phase: This phase includes the authentication of each node participating in the consensus process.
   - The consensus phase includes consensus and block validation, which are used for authentication and verification, as well as the consensus process, and the block is merged into the blockchain, which consists of three main phases, including block creation and PoSS consensus.
   - Block-broadcasting phase: the block is broadcast to all nodes within the peer-to-peer network, and blockchain broadcasting is updated (i.e., updating the ledger).

6.1. Part One: Data Generation

Data generation is a crucial step in machine learning and computer vision, as it involves preparing the data for further analysis and processing. The process of data generation is divided into three main steps, namely, data preprocessing, the VGG16 model phase, and classification via VGG16.

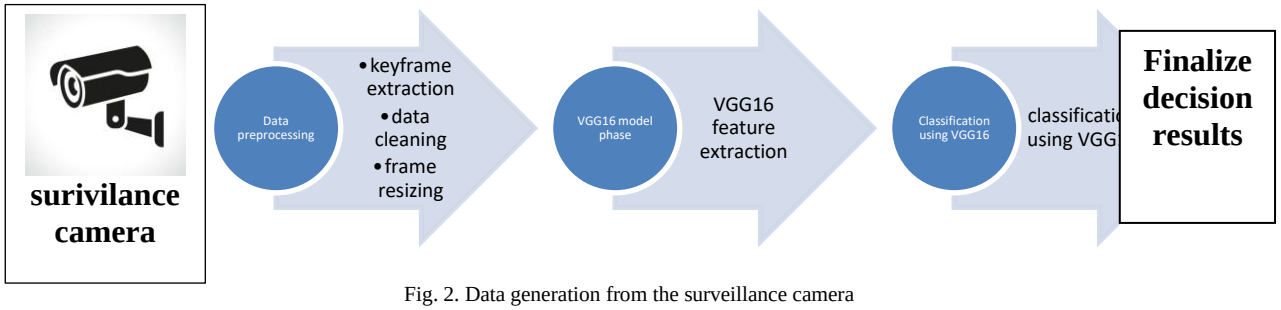

Fig. 2. Data generation from the surveillance camera

The monitoring process consists of several steps, as follows:

1. Data preprocessing: This involves several tasks, including keyframe extraction, data cleaning, and frame resizing. Keyframe extraction is the process of selecting the most representative frames from a video sequence. Data cleaning is the process of removing any errors, inconsistencies, or duplicates in the data. Frame resizing is the process of adjusting the size of the frames to a standardized resolution for further processing.
2. VGG16 model phase: VGG16 is a deep learning model that is commonly used for image recognition tasks. In this phase, the VGG16 model is used to extract features from the preprocessed frames. The VGG16 model has been pretrained on a large dataset and can recognize various features in an image.
3. Classification via VGG16: In this phase, the extracted features are used to classify the frames into different categories or classes. This is typically done via VGG16, as it has been pretrained on a large dataset and can recognize various features in an image. The classification process involves comparing the extracted features with predefined categories or classes and assigning a label to each frame.

The proposed system uses VGG16, a deep learning model that has shown remarkable performance in image classification tasks. By using VGG16, the system is able to detect and classify criminal activities accurately in surveillance videos. The model has been trained on a large dataset of crime-related images and videos, allowing it to identify patterns and anomalies in video footage that may indicate criminal activity.

To evaluate the performance of the proposed system, a set of standard metrics, including accuracy, precision, recall, and F1 score, was used. The system was tested on a large dataset of surveillance videos, and the results demonstrated an accuracy of 96%, indicating a high level of accuracy in crime detection and prevention.

Overall, the use of VGG16 in the proposed system provides a reliable and efficient approach to crime prevention and detection in video surveillance systems, with the potential to significantly improve the security of public spaces and reduce the risk of criminal activity.

Using VGG16 with the real-life violence situation (RLVS) dataset to detect violence and prevent crimes is a promising application of machine learning. By analysing video feeds from cameras in public places, VGG16 can help identify violent events as they occur and send notifications to a control center for prompt intervention. Using homomorphic encryption and a lightweight blockchain can help ensure the security and privacy of the notifications while also enabling secure and efficient communication between the cameras and the control center.

The overall process of using VGG16 with RLVS dataset for violence detection and prevention and sending notifications via homomorphic encryption and blockchain can be summarized as follows:

- Video feeds from cameras in public places are collected and preprocessed.
- The VGG16 and RLVS datasets are used to detect violent events in the video feeds.
- Send notifications to the control center for prompt intervention when violent events are detected.
- Encrypt the notifications via homomorphic encryption to ensure the privacy of the data.
- A lightweight blockchain is used to secure and efficiently transmit encrypted notifications between the cameras and the control center.

Algorithm 4 shows the VGG16 classification model for crime prevention:

Algorithm 4 crime prevention using VGG16 classification model

Input: Video feeds from cameras in public places, Real Life Violence Situations (RLVS) dataset
Output: Encrypted notifications of violent events
Step 1: Load the RLVS dataset.
Step 2: Initialize the Okamoto-Uchiyama homomorphic encryption scheme.

Step 3: Initialize the lightweight blockchain for secure communication between the cameras and the control center.
step4: For each frame in the video feeds, perform the following steps:
- *a. Resize the frame to the input size required by the violence detection algorithm.*
- *b. Use a violence detection algorithm based on the RLVS dataset to classify the frame as containing violence or not.*
- *c. If the frame contains violence, create a notification containing the time, location, and type of violence.*
- *d. Encrypt the notification using the Okamoto-Uchiyama homomorphic encryption scheme. e. Add the encrypted notification to the lightweight blockchain for secure transmission to the control center.*

Step 5: Periodically transmit the encrypted notifications from the cameras to the control center via the lightweight blockchain.
Step 6: Decrypt the notifications using the Okamoto-Uchiyama homomorphic encryption scheme at the control center.
Step 7: Process the notifications and take appropriate action to intervene in violent situations.

6.2. Okamoto–Uchiyama Homomorphic Algorithm

To integrate the Okamoto–Uchiyama homomorphic algorithm with IoT and blockchain systems, it needs to be modified to reduce the time required for algorithm encryption/decryption to be performed and to add some chaotic behavior to its original execution.
This necessitates the proposal of a modified version of the homomorphic algorithm in which the amount of time required is reduced and the chaotic attitude of the algorithm is improved. The main modifications to the Okamoto–Uchiyama homomorphic algorithm are as follows:
The encryption equation is partitioned, and precomputation is used for

$$H = g^{\wedge nkr} \quad (1)$$

These values should be the same until the system key and the distribution process are regenerated; therefore, instead of calculating the values of encrypted data each time, they can be calculated before the encryption process itself.
A chaotic value generated via Baker map chaotic generation is added to generate a single value, which is randomly distributed to each node, to add chaotic behavior to the algorithm.
The Baker algorithm is an example of a discrete-time dynamical system, and it is used to model the growth of populations. The variable x represents the population density at time n, and the constant r represents the growth rate of the population. The Baker map algorithm is able to generate a single chaotic value when given an initial value, a constant value, and several iterations. The algorithm iterates the Baker map function n times, using the previous value of x to generate a new value each time. The final value of x after n iterations is the chaotic value generated by the algorithm.

Algorithm 5 and Figure 3 show the modifications implemented in the standard Okamoto–Uchiyama algorithm.

Algorithm 5 modified Okamoto–Uchiyama algorithm

Input: randomly selected values, plaintext
Output: public and private pairs, the cipher text
Keys generation:
Step 1: generate two large primes P and Q.
Step 2: compute n as follows:
*N = P2 * Q*
Step 3: choose g ∈{2 …. N^{-1}} such that gp-1 ≠1 mod p2.
Step 4: compute H as follows:
H = gN mod N
Step 5: randomly select a value of r that is between 1 and n-1, which can be generated using Steps 6 to 10.
Baker chaotic algorithm to generate K value:
Step 6: set the initial values x and y.
Step 7: use a loop to iterate n times through the Baker map.
Step 8: within the loop, check if x is less than 0.5, if it is:
a. update x by multiplying it by 2
b. update y by dividing it by 2

Step 9: else
a. update x by multiplying it by 2 and subtracting 1
b. update y by adding 1 and dividing it by 2
Step 10: use the round function to round x to the nearest integer and it is the K value.
Precomputations:
Step 11: compute HH as follows:
$HH = gk*n*r \bmod N$
The final keys generated:
Step 12: set the keys, public keys (n, g, h), private keys (p, q, HH,k)
Encryption:
Step 13: compute cc as follows:
$CC = gK*m \bmod N$
Step 14: compute final C as follows:
$C = CC*HH \bmod N$
Decryption:
Step 15:compute the value of A as follows:
$A = \frac{(c^{p-1} \bmod p^2)-1}{p}$
Step 16: compute the value of B as follows:
$B = \frac{(g^{p-1} \bmod p^2)-1}{p}$
Step 17: compute the inverse of B module P.
$B' = B\text{-}1 \bmod P$
Step 18: compute m as follows:
$M = (A\ B' \bmod P)/k$

The modified Okamoto–Uchiyama algorithm uses the Baker chaotic algorithm, whereas the Okamoto–Uchiyama algorithm, which is a public-key homomorphic algorithm that uses a pair of keys—a public key and a private key—is used to encrypt and decrypt messages. The public key is used to encrypt messages, and the private key is used to decrypt them. The Baker chaotic algorithm can be used to enhance the security of the Okamoto–Uchiyama encryption algorithm in several ways:

1. Key generation: R is a random value used within the Okamoto–Uchiyama algorithm to add random behavior to the algorithm. The R value can be generated via the Baker chaotic algorithm. Additionally, the Baker chaotic algorithm can be used to generate pseudorandom numbers, which can be used in various encryption operations, such as key generation, where the key is generated via the K value.
2. Key updating: Chaotic algorithms can be used to update keys regularly. By using chaotic algorithms to update the keys, the keys will be more difficult for an attacker to predict, making it harder for them to break the encryption. This step is related to the key distribution and key updating.

Precomputation of modular exponentiation: Okamoto–Uchiyama homomorphic encryption algorithms use modular exponentiation to encrypt and decrypt data. This operation can be time consuming, so precomputing the modular exponentiation of specific values that would not change rapidly if the keys were used a certain number of times can accelerate the encryption process, benefiting the proposed lightweight security system based on the blockchain (LSSBB), where lightness and time are important factors.
Two additional methods were used to improve the speed of the Okamoto–Uchiyama algorithm:

1. Square and multiply:
In this method, a binary representation of the exponent is created to find the modular exponent within a logarithmic number of steps, whereas in the traditional method, repeated multiplication is performed, which consumes time.

The main implementation steps are as follows:

- The results variable is assigned a value of 1.
- When the exponent is greater than 0, then
    1. If the LSB of the exponent is equal to 1, multiply the results by the base and take the modulo of the resulting modulus.
    2. The exponent results are shifted by one bit.
    3. Square the base and take the modulo of the resulting modulus.
- Return the results.

This approach can be much more efficient in the case of large exponents, and it can save considerable computation time and memory.
Therefore, in general, the modular exponentiation algorithm is considered to be a better method than the standard method of repeated multiplication for the computation of large exponentiations with a modulo of a given integer.

2. Montgomery exponentiation algorithm:

This method is a variant of the square multiplication algorithm, in which modular multiplication is used to speed up the computation process of modular exponentiation, especially in cases with large exponents.
The main steps of the algorithm are as follows:

1. Choose R, which is a large power of 2, and the value of R should be larger than the modulus.
2. Find the n value where n * modulus ≡ -1 mod R (an inverse modular reference to R).
3. Precomputation of the x value where x = base * R mod modulus.
4. Initialize the results variable to the R mod modulus.
5. When the exponent is greater than 0, then

a. If the least significant bit of the exponent is 1, multiply the result by x and take the modulo of the resulting modulus.
b. Square the value x and take the modulo of the resulting modulus.
c. The exponent is shifted by 1 bit (equivalent to dividing by 2).

6. The final result is computed by multiplying the results by a modulus of 1 mod.

The square and multiplication algorithms and the Montgomery exponentiation algorithm are merged with the Okamoto–Uchiyama homomorphic encryption scheme for the implementation of exponential operations within the encrypted values without affecting the homomorphic characteristics of the homomorphic methods.
Both algorithms, along with precomputed modification, can be used to achieve this goal, and choosing between them requires the specific requirements and constraints of the system to be specified; in general, the Montgomery exponentiation algorithm should be more efficient for the calculation of large exponentiations, whereas the square and multiplication algorithms should be more efficient for the calculation of small exponentiations.
As shown in Figure 3 and Algorithm 4, the proposed modified Okamoto–Uchiyama encryption algorithm can be split into four main phases as follows:

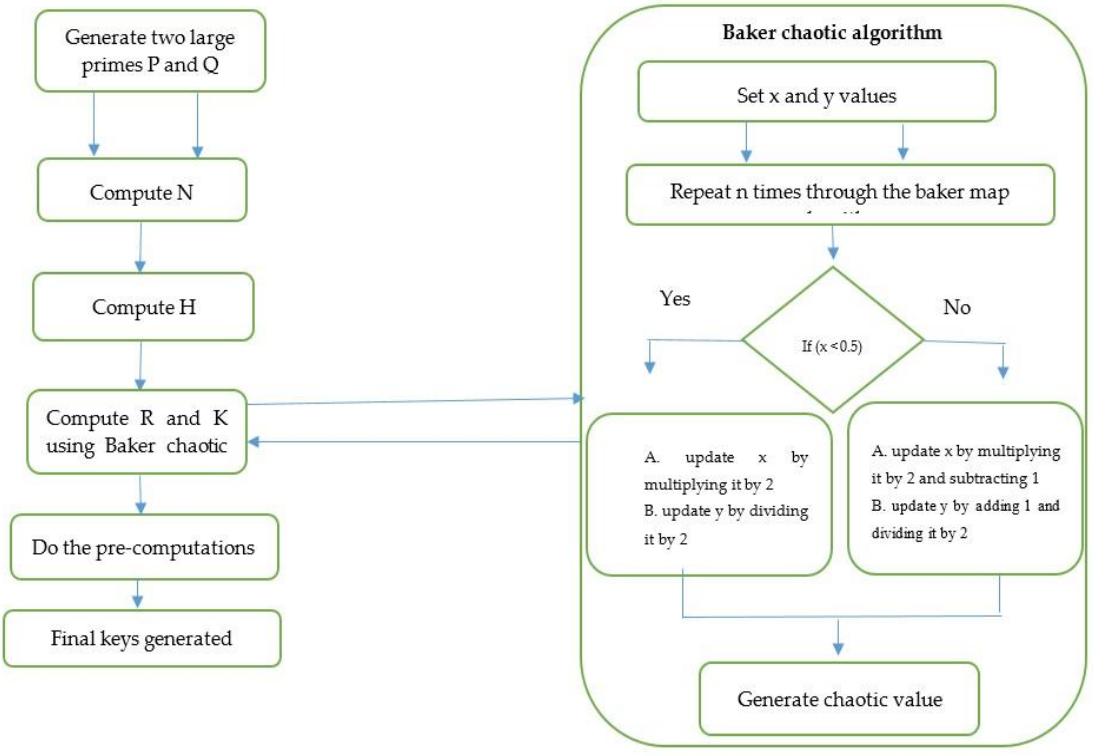

Fig. 3. Modified Okamoto–Uchiyama encryption process.

3. Phase One—Key Generation:

All of the keys related to the encryption and decryption process are generated within this phase, where the N values can be calculated via both P and Q, which are large numbers, and the value of g is chosen on the basis of a special condition that belongs to the range between 2 and N−1 and $gp^{-1} \not\equiv 1 \bmod p2$ and considers a public key.

The value of the public key H is calculated via the factorization of g and N modular N, and the r value should also be chosen between 1 and N^{-1} to ensure that this value, even if it is deterministic with a special range, can be chosen randomly. The Baker chaotic map algorithm can be used to find it, or it can be chosen randomly.

The final list keys include public keys (n, g, h) and private keys (p, q,k).

4. Phase Two—Baker Chaotic Algorithm to Generate the K Value:

To add a factor to the factorization values to present an additional level of complexity to attackers in the case of an attack, in this method, a K value should be randomly chosen and should participate in precomputation phase three to achieve the highest security levels.

This phase starts with two initial values X and Y being chosen, and a loop of n iterations of the Baker map is used to calculate a final value of K on the basis of the value of X to determine the next step, where if the value of X within the loop is less than 0.5, then the value of X is updated by simple multiplication by 2 and the value of Y is updated by simple division by 2, and if it is not less than 0.5, then the value of X is updated by simple multiplication by 2 and subtraction of 1 and the value of Y is updated by the addition of 1 to Y and its division by 2. The final value is rounded to the nearest integer.

5. Phase Three—Encryption:

To speed up the process of factorization and reduce the time needed for this calculation, the value of HH is added to the secret keys.

The factorization of the g-based value is exponential to the values of the k,n,r multiplicities modular N, and the final value is distributed as part of the secret keys group. This phase constitutes the encryption process of the plaintext, where g is the base of power K and the message modular N, and the final result is the multiplication of this value and the value generated from the precomputations.

6. Phase Four—Decryption:

For decryption, the modified Okamoto–Uchiyama system calculates the value of A by first raising *c* to the power of (p−1) and then taking the modulus with p^2, subtracting 1, and dividing by p.

$$A = \frac{(c^{p-1} mod\ p^2)-1}{p} \tag{2}$$

The value of B is subsequently calculated by raising g to the power of (p−1) and then taking the modulus with p^2, subtracting 1, and dividing by p.

$$B = \frac{(g^{p-1} mod\ p^2)-1}{p} \tag{3}$$

Then, the inverse of B modulo P is calculated, and the value of M is found by first multiplying A and B' together, then taking the modulus with P, and then dividing by k.

$$M = (A\ B' \bmod P)/k \tag{4}$$

6.3. Lightweight Blockchain

A lightweight blockchain refers to a blockchain system that is designed to have low resource requirements, such as with respect to processing power and memory, as well as low latency. This enables transactions to take place more rapidly, with lower energy consumption, and at a more scalable solution.

Consensus algorithms play a crucial role in meeting the requirements of a lightweight blockchain, as they determine how transactions are validated and added to the blockchain. By choosing an efficient consensus algorithm, it is possible to reduce the amount of resources required to maintain the blockchain network and increase its scalability.

This section explores the use of the proof of secret sharing as a consensus mechanism within a blockchain to create a lightweight blockchain. Secret sharing is a technique that enables a secret to be divided into multiple parts, with each part held by a different participant. Proof of secret sharing, on the other hand, is a method for verifying that a participant holds a secret share without revealing the secret itself. By combining these techniques, it is possible to achieve a consensus mechanism that is both secure and lightweight, making it ideal for use in a blockchain environment. This section delves into the details of how these consensus algorithms can be implemented to meet the demands of a lightweight blockchain.

New consensus algorithms will be proposed for utilization in the context of blockchain technology, namely, proof of secret sharing (PoSS). This algorithm is designed to be more efficient, require less time and resources, and have lower complexity than traditional algorithms such as PoW. The details of this algorithm will be further explored later in the paper, along with the details of the lightweight blockchain using the proposed consensus algorithm.

6.3.1. Proof of Secret Sharing (PoSS) Consensus Algorithm

Proof of secret sharing (PoSS) is a consensus algorithm in which cryptographic security difficulty puzzles are replaced with proof of zero knowledge via the Shamir secret sharing algorithm; it has been applied in real-life problems such as preserving the security of data generated from automated crime prevention systems. A schematic representation of the PoSS is presented in Figure 4.

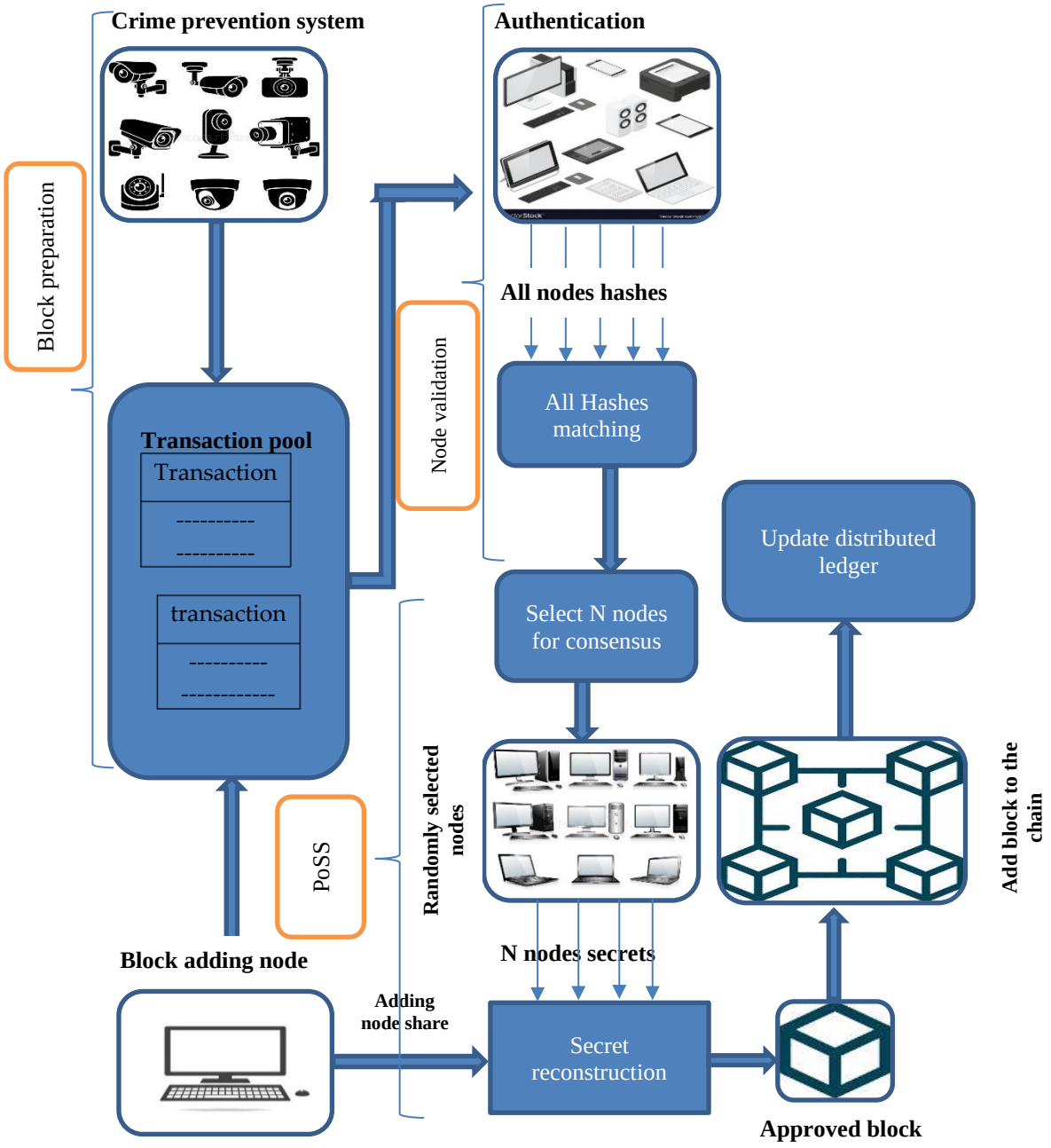

Fig. 4. Proof of secret sharing (PoSS) consensus algorithm.

The participants within the peer-to-peer network are crime prevention system nodes, from which the decision data are sent; control and monitoring center nodes, where issues are solved and block addition is managed; and block-adding nodes, where the blockchain data are monitored. These are the main impact roles that need to be incorporated into the blockchain network to secure the decision made via the software and hardware required to implement the blockchain.

Since the size of the chain could be large and storing the data could be expensive, as well as with the aim of reducing the impact of the blockchain on the environment, PoSS is employed to reduce the time and the resources required when adding

a new block to the chain. A specified secret, generated by control and monitoring center nodes with reference to the total number of authorized crime prevention system nodes that can be connected to the chain, is generated and distributed to each node within the network via one of the key distribution techniques. To verify each node, its secret share needs to be verified and authorized by sharing it with the management node to order the addition of a block to the blockchain. A predefined number, called threshold N random N, of nodes will provide their shares as proof of their secret share, where if an attacker was able to compromise N nodes, then they would be able to control the decision of the consensus.
Each crime prevention system generates decision data that are added to the blockchain from surveillance cameras, and the hash values are calculated for these data via the standard SHA-256 algorithm. Since all camera center nodes need to save their data within the blockchain, there is no need to find a mining method or solve complex mathematical puzzles; rather, this can be replaced by proof of secret sharing for the authentication of nodes, thus validating the blocks added to the chain. The timestamp is included for each block, and the main conditions for verification are as follows:

- In this case study, the time stamp needs to be within a specified range.
- The current node timestamp should be larger than the time stamp of its parent and within the acceptable range of the monitoring period.

Figure 4 and Algorithm 6 provide a general description of the proof of secret sharing (PoSS).
Proof of secret sharing (PoSS) can be split into three main phases as follows:

Algorithm 6 proof of secret sharing (PoSS)

Input: data, secret shares, distributed ledgers
Output: updated distributed ledgers
Step 1: data collection: collect the data generated from the crime prevention systems.
Step 2: formulate the block: all the data collected are saved in one single block.
Step 3: authenticate the peer-to-peer network node: authenticate the participating node by matching the distributed ledger of each node and sharing the secret share.
Step 4: if the ledger was not matched and the node does not have a secret share, the operation is aborted, and further investigation is performed to detect the error node.
Step 4: selection of consensus nodes: N-1 nodes are selected from all the validated nodes to participate in the consensus process in addition to the block-adding node.
Step 5: secret sharing authentication: each one of the selected nodes shares its secret share to participate in secret reconstruction.
Step 6: secret reconstruction: all shares are used with the appropriate secret-sharing algorithm to reconstruct the secret of the system.
Step 7: distributed ledger update: the block is added to the chain, and the distributed ledger is updated for all nodes in the peer-to-peer network.
Step 8: end.

All the secrets are distributed to all nodes before the initiation of the blockchain.
Phase one—block preparation: This phase includes the process in which the data are collected and turned into a block that is ready for validation and addition to the node.
This phase consists of many internal steps to accommodate the preparation of blocks within the system:

1. Data generation: The crime prevention nodes generate decision data via surveillance cameras to initiate a block to be added to the blockchain. Each piece of data related to a node among the crime prevention nodes is considered a transaction.
2. Transaction pool: If there is more than one adding node, the data generated (i.e., the transactions) are collected in a special pool, and the block is formulated.
3. Block-adding node action: one of the crime prevention nodes takes the role of a block-adding node and requests permission to add a new node to the blockchain distributed ledger.

Phase two—node authentication: the authentication of adding nodes as well as the nodes participating in the consensus process is carried out in this phase.
This phase consists of many internal steps to accommodate the validation of the nodes within the system:

1. Authentication of network nodes: Each node, including the block-adding node, needs to be authenticated before the consensus process begins. If one of the nodes fails the authentication process, then the whole process is aborted, and further investigation is required to identify this node and solve the issue.
2. The authentication is carried out by comparing the hashes of all nodes and the distributed ledger of all nodes within the private peer-to-peer network.
3. Selection of consensus nodes: From among all authenticated nodes within the peer-to-peer network, N-1 nodes are randomly selected to participate in the consensus decision by reconstructing the secret during phase three, where N represents the threshold of the Shamir secret sharing algorithm.

4. Authenticated node secret sharing: Each of the rebuilt selected nodes and the block-adding node need to share their shares of the secret as part of the consensus process.

Phase three—the consensus process: the secret is reconstructed via the secret shares distributed among the nodes to validate the process of connecting the block to the chain and updating the distributed ledger.
This phase consists of many internal steps to accommodate the consensus process and add the block to the chain:

1. Reconstruction of the secret: This step includes collecting all the shares and reconstructing the secret via the secret sharing algorithm used within the system (in this system, Shamir's secret sharing was used). If the secret is reconstructed correctly, then all the participating nodes are validated, including the block-adding node and all the consensus nodes, and the block is added to the chain. If it is not rebuilt correctly, then the operation is aborted, and further investigation is needed to detect the node that has shared the incorrect information to correct or isolate it, thus avoiding malicious behavior.
2. Block addition and ledger update: After consensus has been achieved and authenticated in the previous step for the secret on the basis of all randomly selected nodes in phase 2, the block is added to the blockchain, and the ledger is updated; therefore, all nodes within the peer-to-peer network ledgers need to be updated.

6.3.2. Proposed lightweight blockchain using the new consensus algorithm

A lightweight blockchain, referred to as a light chain, represents a type of blockchain in which specific characteristics are modified so that fewer resources are needed, thus achieving a smaller footprint than traditional blockchains do. This can be achieved as follows:

1. New consensus algorithms such as proof of secret sharing (PoSS) and proof of homomorphic secret sharing (PoHSS) have been proposed.
2. The amount of blockchain that needs to be saved is reduced such that not all the data are saved, only sensitive data are saved, and a file comparison is used instead of the Merkle tree to reduce the amount of data within the header.
3. Limit the number of nodes participating in the network using a predefined number of nodes within the network via a private network. Each node has its own share of the secret used within the PoSS, which is updated periodically.

This makes the blockchain more accessible and applicable in a wider range of applications.

Figure 5 and Algorithm 7 illustrate the main steps of the proposed lightweight blockchain algorithm.

Algorithm 7 lightweight blockchain

Input: keys, crime prevention decision output, distributed ledger
Output: distributed ledger update.
Start
Step 1: crime prevention decisionoutput by creating a new transaction block, which includes the following details:

- *The sender's details*
- *The crime prevention decision output*
- *A timestamp*
- *A reference to the previous block (also known as the "hash")*

Step 2: the network nodes, also known as "miners", send their secret shares to the added node via the network.
Step 3: N nodes are randomly selected to be used in the proof of secret sharing (PoSS) consensus algorithm to validate the transaction. This typically involves solving secret sharing problems, also known as mining.
Step 4: once a miner successfully solves the problem and generates a valid PoSS secret, the new block is added to the blockchain.
Step5: the added node miner creates a new block header, which includes the following details:

- *A reference to the previous block's hash*
- *A timestamp*

Step 5: the added miner node then broadcasts the new block header to the network.
Step 6: the other nodes in the network receive the new block header and check it against the existing blockchain. If the new block header is valid, the nodes add it to their copy of the blockchain.
Step 7: the blockchain is then updated and broadcast to all other nodes in the network to ensure that everyone has the same copy of the blockchain.
Step 8: the transaction is then considered to have been confirmed and added to the blockchain.
End

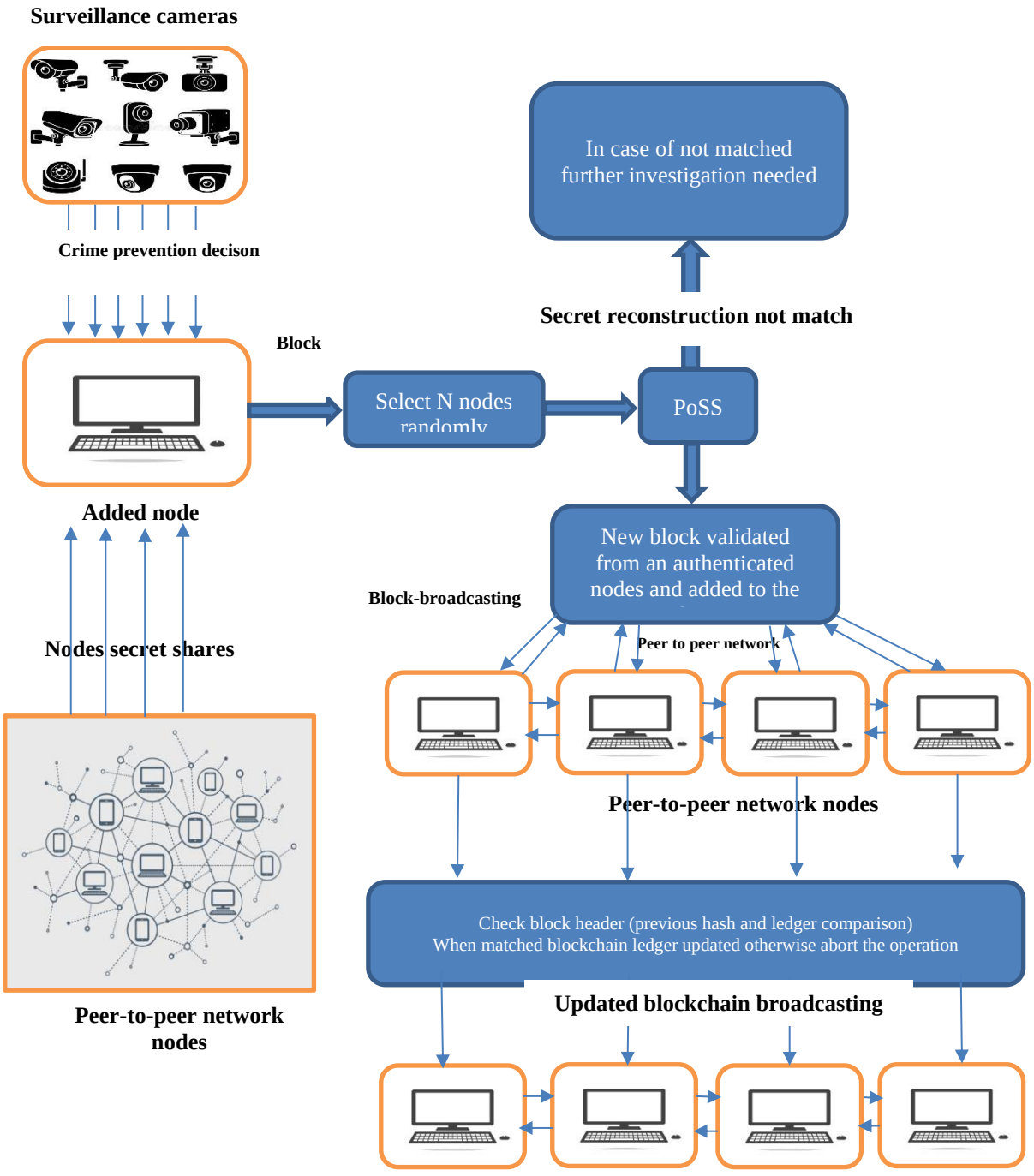

Fig. 5. Lightweight blockchain main phases.

The lightweight blockchain consists of the following phases:

Phase one: Block creation and PoSS: The output of the crime prevention node is accepted as a decision, and a block is formulated on the basis of the transactions. These blocks need to be verified and validated via an authenticated node in which N are selected and authenticated via Shamir secret sharing and participate in the validation of the block and its addition to the chain, as explained previously.

The block can consist of different content, depending on the purpose of the blockchain; in this work, the block consists of the following content:

1. The sender's details verify the node that created the block and add it to the chain. This is useful when retrieving data or when auditing blockchain data.

2. The output of the crime prevention node is the decision data that the node would like to add to the blockchain. In the context of this system, the nodes need to send the decision output of the crime prevention node to the control and monitoring system, where it will be saved in decentralized locations as nodes and media, whereas monitoring agency nodes will be able to retrieve the chain data without having the ability to add to it.
   The nature of the blockchain means that the output of these devices is immutable and cannot be modified or deleted.
3. A timestamp is an important factor within the chain, as the time of block addition cannot fall outside the day time, as well as including the exact date and time of the addition of the block to the chain.
4. A reference to the previous block (also known as the "hash"); the blockchain is saved as a linked list in which each node has a reference to the next node in the blockchain. Each block has a hash calculated via one of the hash family methods, whereby the hash of the previous node is saved within the current node. If an attacker were to attempt to attack the blockchain, then they would need to modify all the blocks in the chain right back to the beginning, which is practically impossible.
5. Current block hash: This is a critical security factor of blockchain in which the data belonging to the current block are hashed via a predefined hash algorithm—SHA-256 is used in this work, as it is useful for guaranteeing the consistency and integrity of the data—and connected to the next node when generated.

If the block is validated and the node is authenticated, then it can be connected to the chain. In cases where the PoSS secrets do not match, further investigation is needed to verify the node/nodes that do not provide a valid share of the secret, either to isolate it or to correct it, and that node will not be able to retrieve or add blocks to the chain until it has been correctly validated via PoSS. The use of PoSS within the chain reduces the time needed for verification and authentication and reduces the computational resources needed, where instead of solving complex mathematical problems such as proof of work and its variations or proving the share amount, a random number of nodes are selected, and it is only necessary for them to provide their secret share. If all of the selected nodes are verified and authenticated correctly, including the block-adding node, then the data will be fed to the chain as a block.

Phase two: Block broadcasting

This verified blockchain block needs to be broadcast to all nodes within the chain so that they can update their ledger, or the ledger needs to be updated and sent as binary data to the nodes within the peer-to-peer network. The behavior of the chain determines the method of broadcasting the block to each node of the network, each of which saves an exact copy of the distributed ledger.

Phase three: Updated blockchain broadcasting

This includes the operation of checking the header of a broadcasted block and using it to perform a comparison with the previous block added to the chain, where the hash within the last block of the chain in the distributed ledger must match the hash of the broadcasted block. If the match is valid, the blockchain ledger is updated; otherwise, the operation is aborted. The type of distributed ledger differs with respect to the type of blockchain used, where it could be text or a binary file, and the header content represents the reason. Text or binary files can sometimes hold files of greater size.

Instead of using the Merkle tree to compare the hashes of the file, a simple file comparison of the file content is performed to decide on the basis of matching that this file has not been modified or corrected.

This blockchain has the following characteristics:

1. The improved total security is more resistant to a 51% attack of the blockchain since the attacker, even if they were to successfully compromise 51% of the peer-to-peer network, would be unable to take control of the network, since no one could guarantee that this percentage contained all of the selected nodes.
2. Increased scalability: the main evaluation metrics are related to throughput and the number of transactions per second, which means that more transactions and blocks per unit of time are accepted, thus increasing the chain's scalability.
3. Reduced energy consumption: the standard blockchain consumes a high amount of power, and the lightweight blockchain with the new consensus algorithm is more efficient with respect to power consumption.
4. Faster confirmation time: the standard blockchain typically uses proof of work or other consensus algorithms, which require more time for confirmation.

7. RESULTS

In this section, the performance of the proposed system is measured on the basis of many measures to ensure that the system is efficient, scalable, secure, and cost-effective and to show the manner in which the proposed system improves these general factors.

Part two: evaluation of the modified Okamoto–Uchiyama algorithm:

The most important factor for the proposed system is the reduction in the required time and the efficient use of resources. Three methods are proposed to reduce the time required for the encryption process within the Okamoto–Uchiyama homomorphic encryption algorithm:

1. Precomputations: This method aims to precalculate some values to reduce the time required for the calculation.
2. Square and multiplication algorithm: This algorithm is employed to determine the modulus using the binary representation of the exponent.

3. Montgomery exponentiation algorithm: This algorithm uses modular multiplication to find the results.

The following tables and figures present comparisons of the standard and proposed methods using different key sizes. The size of Okamoto–Uchiyama homomorphic encryption represents the private and public key bits employed for the main encryption and decryption operations.

The size of these keys impacts the computational efficiency, execution time, and security, with larger keys being more secure, countering brute force attacks, but resulting in slower encryption and decryption processes. Therefore, the differences in the execution time required for each key size and the different levels of time complexity, which are calculated with reference to and compared with those of the original algorithm, are presented.

In the proposed system, larger keys can be used for encryption when secret keys are shared, whereas smaller keys can be used for normal data.

The time complexity presented in the following table and figure represents the time required to perform the encryption process when the standard algorithm is used compared with the modified Okamoto–Uchiyama algorithm, providing a general metric for the computational resources needed to perform the encryption operation, whereby low time complexity indicates that fewer computational resources are required to perform the operation, making that version of the algorithm more efficient.

Table 4 and Figure 6 provide plaintext encrypted via different variants of the Okamoto–Uchiyama algorithm (i.e., the original and modified variants) on the basis of the same ground data, and the key used and the time complexities demonstrate the time needed to encrypt each plaintext via each method.

TABLE IV. KEY SIZE 32 STANDARD OKAMOTO-UCHIYAMA ALGORITHM AND PROPOSED MODIFICATIONS,

Plaintext	Standard Algorithm	Precomputation	Square and Multiply Method	Montgomery Exponentiation Method	Square and Multiply Method and Precomputed Method	Montgomery Exponentiation Method and Precomputed Method
2,571,999	0.0001603000000010013	1.449999999891815 6e-05	0.00014459999999871798	0.00010819999999966967	0.00013990000000063674	0.00016530000000081202
2,571,986	9.21000000007055e-05	8.399999999575414e-06	8.009999999991635e-05	8.060000000043033e-05	7.940000000061787e-05	8.85999999997722e-05
2,431,991	9.529999999990935e--05	8.499999999855845e-06	8.439999999865222e-05	8.85999999997722e-05	8.270000000010214e-05	9.049999999888314e-05
1,112,017	9.190000000014464e--05	7.599999999996498 4e-06	8.080000000010301e-05	8.000000000008001e-05	7.970000000012689e-05	8.749999999979607e-05
2,682,020	0.0001987999999997 2142	8.999999999481645e-06	8.430000000014815e-05	8.21000000001959e-05	8.63000000004277e-05	9.670000000028267e-05

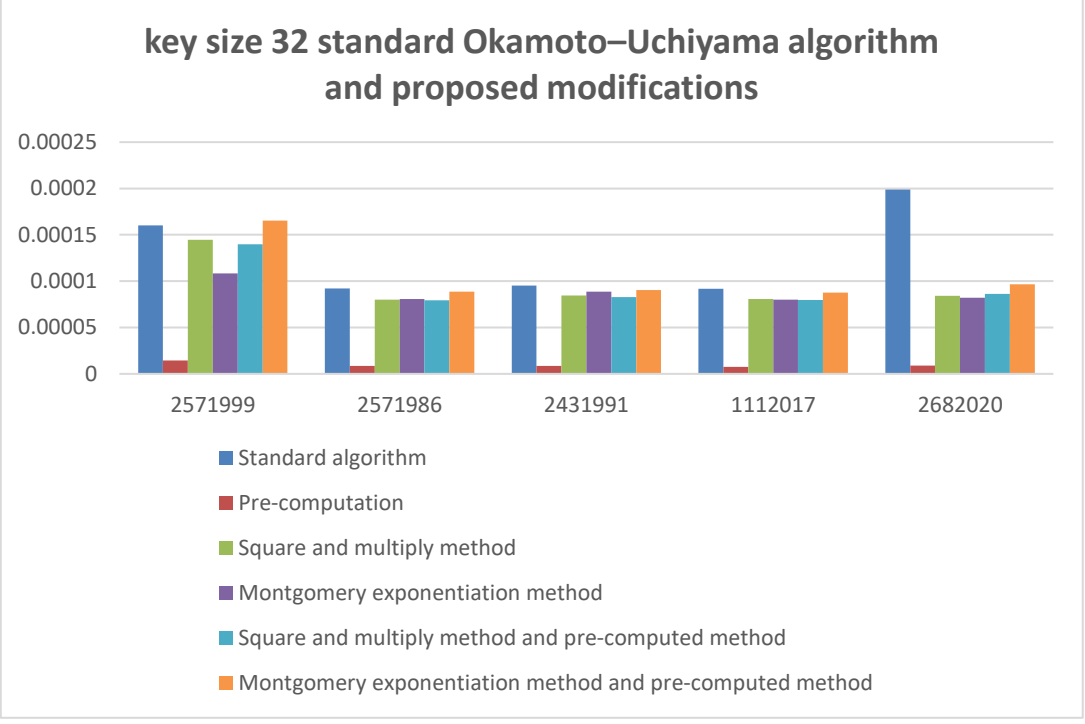

Fig. 6. Key size 32 standard Okamoto–Uchiyama algorithm and proposed modifications.

The standard Okamoto–Uchiyama algorithm took the longest time among the various algorithm variants proposed, with all other variants successfully reducing the time required for encryption; the precomputation method requires the shortest amount of time for encryption, and all of the algorithms used values generated via the Baker chaotic algorithm.
As in the case of a 32-bit key size, with a 64-bit key size, the precomputed modification required the least time among the different methods, followed by the square and multiplication methods and the Montgomery exponentiation method. However, when combined with precomputation, the square and multiplication methods were the slowest, whereas the Montgomery exponentiation method combined with precomputation still performed relatively well.
The results obtained for the suggested methods with respect to time complexity when a 32-bit key length is used compared with a 64-bit key length can be summarized as follows:

- Precomputational method: This method provides significant speedup and demonstrates the effectiveness of precomputation in reducing the encryption time.
- Square and multiplication methods: The speedup is not as significant as that achieved by the precomputation method, but it still represents a notable improvement.
- Montgomery exponentiation method: the Montgomery exponentiation method is similar to that achieved by the square and multiplication methods
- Square and multiply methods and precomputed methods: This suggests that the combination of these two techniques may not always result in faster encryption times and that careful consideration is required when selecting encryption methods.
- Montgomery exponentiation method and precomputed method: While the speed-up achieved by this method is not as significant as that achieved by the precomputation method alone, it still represents a notable improvement over the standard algorithm.

Part three: Blockchain evaluation metrics:
Twelve evaluation metrics were used to evaluate the performance of the proposed blockchain in terms of speed, efficiency, and resource utilization. Each of these metrics provides insight into blockchain performance and identifies areas that require improvement.

- Transactions per second (TPS): an important metric that reflects the capacity of the blockchain to handle many transactions within a given period; this metric is crucial for applications of the blockchain that require high throughput, such as supply chains, payment methods, and data generated by the IoT.
- Block generation time and verification time: These two metrics indicate how fast the blockchain can validate and accept new transactions and the time required to add them and update the ledger, which guarantees that the blockchain is up-to-date for each participating node.
- CPU usage and RAM usage: These metrics indicate the amount of resources required by the blockchain to operate effectively. This may ensure that the blockchain works efficiently, consuming resources at acceptable levels at acceptable costs and levels of scalability. As shown in Table 5, the results obtained for the standard blockchain using a proof of work consensus algorithm where the size of the blocks was predefined and the lightweight blockchain using the proof of secret sharing algorithm where the size of the blocks was not fixed but the number of transactions was previously determined since the input to the system was already in an expected format show that the lightweight blockchain using the PoSS consensus algorithm performed better in terms of transactions per second, block and transaction size, and block generation and verification times, as well as with respect to final time.

TABLE V. EVALUATION METRICS FOR STANDARD AND LIGHTWEIGHT BLOCKCHAINS

Metric	Standard Blockchain	Lightweight Blockchain
Transactions per second (TPS)	7–37	262
Transaction size	0.02 KB	0.02 KB
Block size	0.3310546875 kilo-byte	0.3310546875 kilo-byte
Block generation time	1.7 s	0.012 s
Block verification time	6.302859499999997 s	0.9951314999999994 s
Final time (Block generation + verification time)	8.002859499999997 s	1.0071314999999994 s
Average CPU usage	14.2	3.8
Average CPU user time	53,868.93749999999	53,582.03125
Average CPU system time	38,787.609375	38,640.4375
Average idle time	1,897,998.1249999998	1,890,245.7031249998
Average interrupt time	2382.5	2373.40625
Average RAM %	46.8/100	50.8/100

1. Transactions per second (TPS):two blockchains were evaluated via this metric with the same transaction input: the standard blockchain (based on the proof-of-work consensus algorithm)and the proposed lightweight

blockchain (based on the proof-of-secret-sharing consensus algorithm).The results were 7 to 37 transactions per second and 262 transactions per second, respectively, which indicates that the proposed lightweight blockchain is faster than the standard blockchain in terms of the time required for processing transactions, as shown in Figure 7.

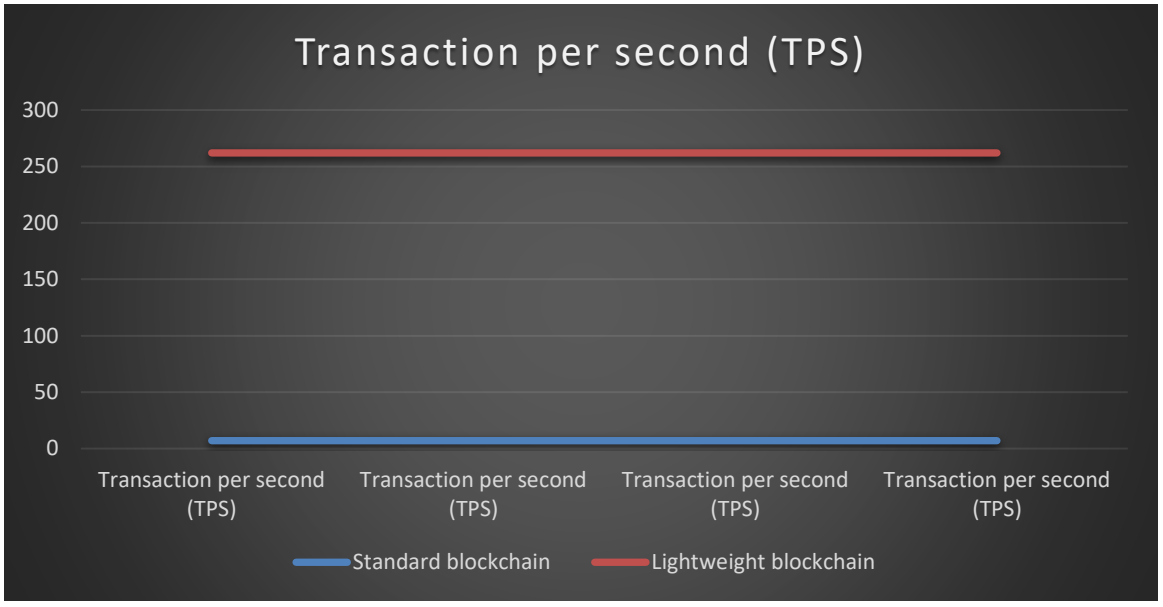

Fig. 7. Transactions per second for standard and lightweight blockchains.

2. Transaction size: For evaluation purposes, both algorithms were evaluated using the same transaction size, whereby the size of a single transaction measured in kilobytes was 0.02 KB.
3. Block size: This represents the size of each block within the blockchain. For evaluation purposes, both algorithms were evaluated using the same block size of 0.3310546875 kilobytes.
4. Block generation time: the time required for the blockchain to generate a new block and connect it to the chain. This metric was used to evaluate the two blockchains—the standard blockchain (based on the proof-of-work consensus algorithm) and the proposed lightweight blockchain (based on the proof-of-secret sharing consensus algorithm). The standard blockchain required 1.7 s to generate a new block, whereas the proposed lightweight blockchain required 0.012 s to generate a new block, indicating that the proposed lightweight blockchain is faster than the standard blockchain with respect to block generation, as shown in Figure 8.

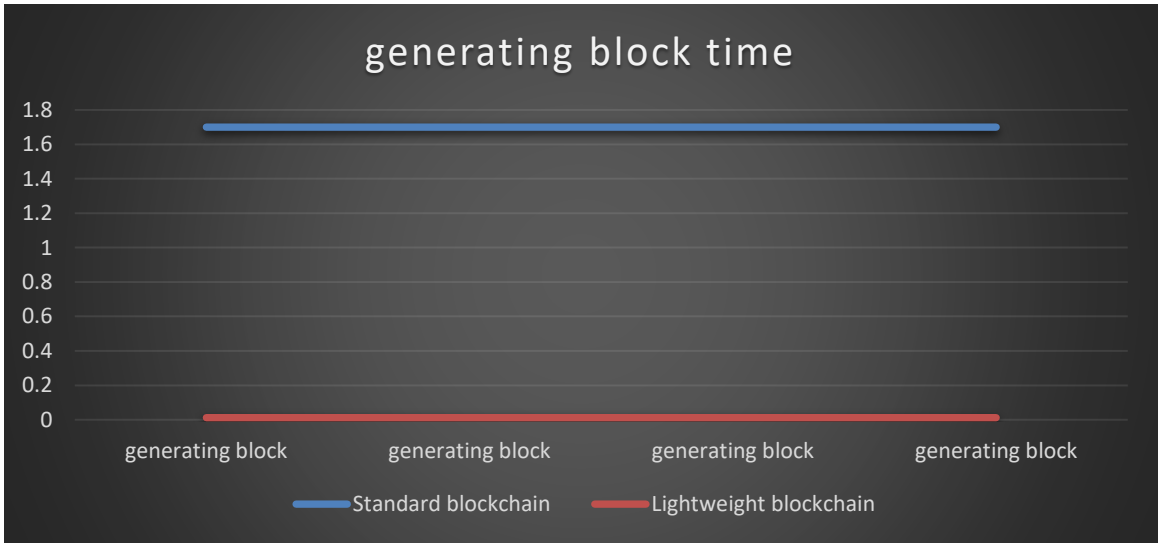

Fig. 8. Block generation time for standard and lightweight blockchains.

5. Block verification time: the time required for a blockchain to verify a block. This metric was used to evaluate the two blockchains—the standard blockchain (based on the proof-of-work consensus algorithm) and the proposed lightweight blockchain (based on the proof-of-secret sharing consensus algorithm). The standard blockchain requires 6.302859499999997 s to verify a new block, whereas the proposed lightweight blockchain requires 0.9951314999999994 s to verify the new block, indicating that the proposed lightweight blockchain is faster than the standard blockchain with respect to block verification, as shown in Figure 9.

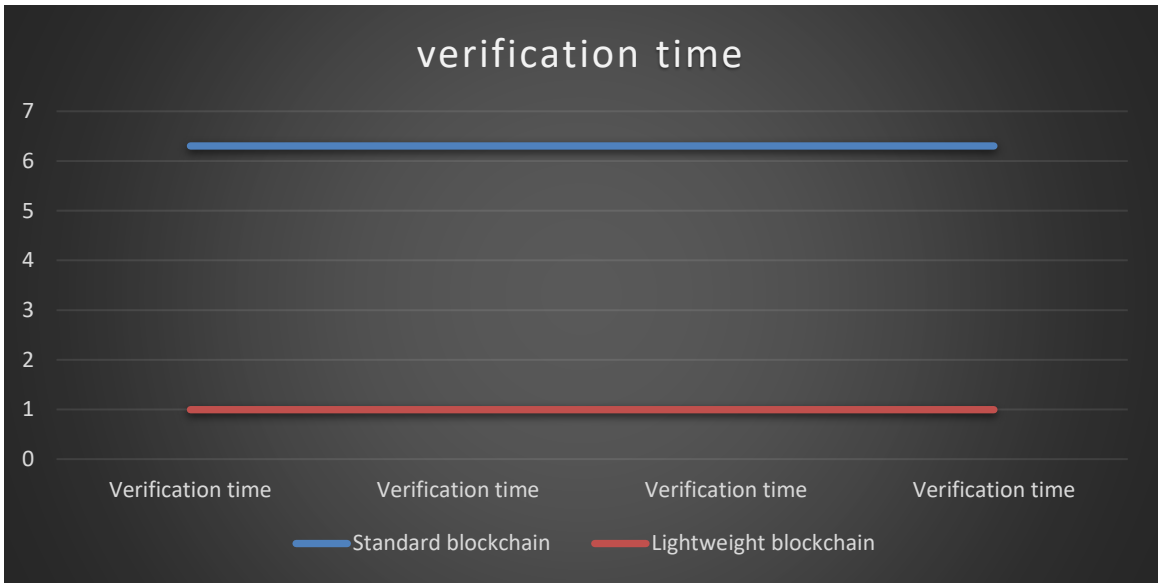

Fig. 9. Verification time for standard and lightweight blockchains.

6. Final time (block generation + verification time): the time required for the blockchain to generate and verify a block. This metric was used to evaluate the two blockchains—the standard blockchain (based on the proof-of-work consensus algorithm) and the proposed lightweight blockchain (based on the proof-of-secret sharing consensus algorithm). The standard blockchain requires 6.302859499999997 s to generate and verify a new block, whereas the proposed lightweight blockchain requires 0.9951314999999994 s to generate and verify a new block, indicating that the proposed lightweight blockchain is faster than the standard blockchain with respect to block generation and verification, as shown in Figure 10.

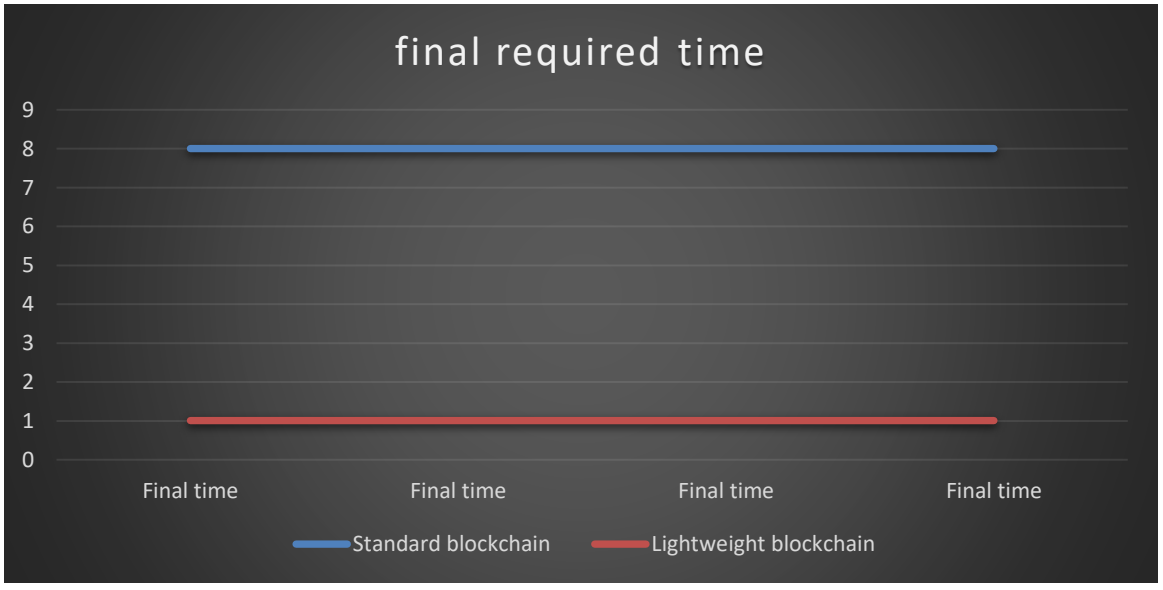

Fig. 10. Final required time for standard and lightweight blockchains.

7. Average CPU usage: the percentage of node consumption refers to the CPU usage required by the blockchain. This metric was used to evaluate the two blockchains—the standard blockchain (based on the proof-of-work consensus algorithm) and the proposed lightweight blockchain (based on the proof-of-secret sharing consensus algorithm). The standard blockchain required an average CPU usage of 14.2% to verify a new block, whereas the proposed lightweight blockchain required an average CPU usage of 3.8% to verify a new block.
8. Average CPU user usage: the percentage of user consumption refers to the CPU usage by the blockchain. This metric was used to evaluate the two blockchains—the standard blockchain (based on the proof-of-work consensus algorithm) and the proposed lightweight blockchain (based on the proof-of-secret sharing consensus algorithm). The standard blockchain had an average user CPU usage of 53,868.93749999999 to verify a new block, whereas the proposed lightweight blockchain had an average user CPU usage of 53,582.03125 to verify a new block. This shows that both blockchains have almost the same CPU user time.
9. Average CPU system usage: the percentage of system consumption refers to the CPU usage required by the blockchain. This metric was used to evaluate the two blockchains—the standard blockchain (based on the proof-of-work consensus algorithm) and the proposed lightweight blockchain (based on the proof-of-secret sharing consensus algorithm). The standard blockchain requires an average system CPU usage of 38,787.609375 to verify a new block, whereas the proposed lightweight blockchain requires an average system CPU usage of 38,640.4375 to verify a new block. This shows that both blockchains have almost the same CPU system time requirements.
10. Average idle time: the percentage of system consumption refers to the required CPU usage when the blockchain is idle. This metric was used to evaluate the two blockchains—the standard blockchain (based on the proof-of-work consensus algorithm) and the proposed lightweight blockchain (based on the proof-of-secret sharing consensus algorithm). The standard blockchain has an average system CPU idle requirement of 1,897,998.1249999998, whereas the proposed lightweight blockchain has an average system CPU idle requirement of 1,890,245.7031249998. This shows that both blockchains have almost the same CPU system time requirements.
11. Average interrupt time: the percentage of system consumption refers to the percentage of CPU usage consisting of interrupts. This metric was used to evaluate the two blockchains—the standard blockchain (based on the proof-of-work consensus algorithm) and the proposed lightweight blockchain (based on the proof-of-secret sharing consensus algorithm). The standard blockchain had an average system CPU consisting of 2382.5 interrupts, whereas the proposed lightweight blockchain had an average system CPU consisting of 2373.40625 interrupts. This shows that both blockchains have almost the same CPU system time requirements.
12. Average RAM percentage: This percentage indicates the percentage of RAM used by the blockchain. The standard blockchain used 46.8/100 of the available RAM, whereas the lightweight blockchain used 50.8, indicating that the standard blockchain required less RAM than the lightweight blockchain did.

Figure 11 presents a comparison of the standard blockchain and the lightweight blockchain with respect to resource consumption.

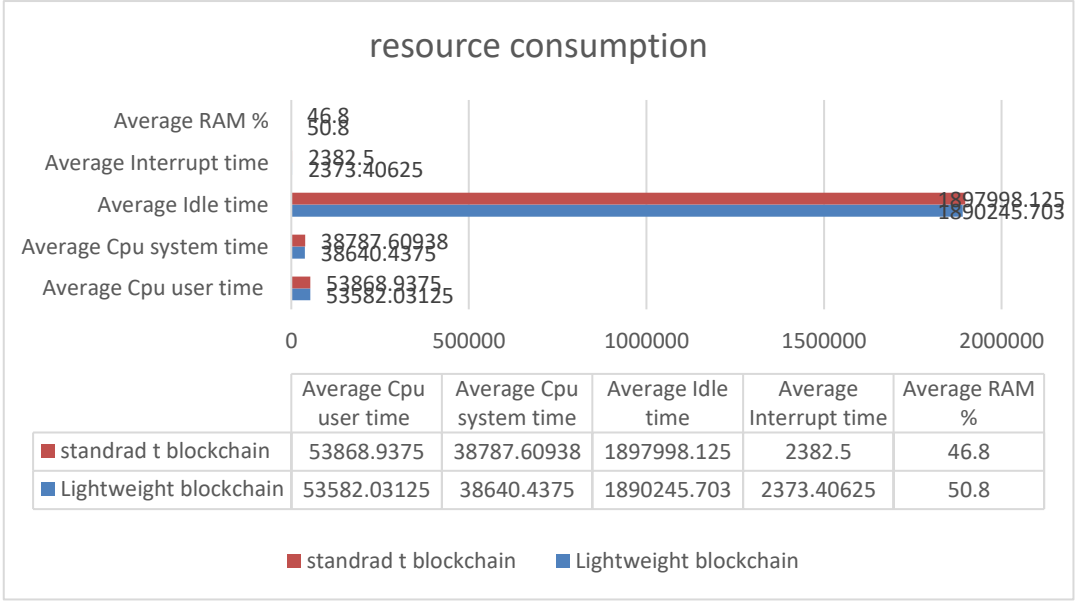

Fig. 11. Resource consumption for standard and lightweight blockchains.

Figure 12 shows the accuracy of VGG16 for detecting crime within smart cities, and Figure 13 shows the confusion matrix of this model.

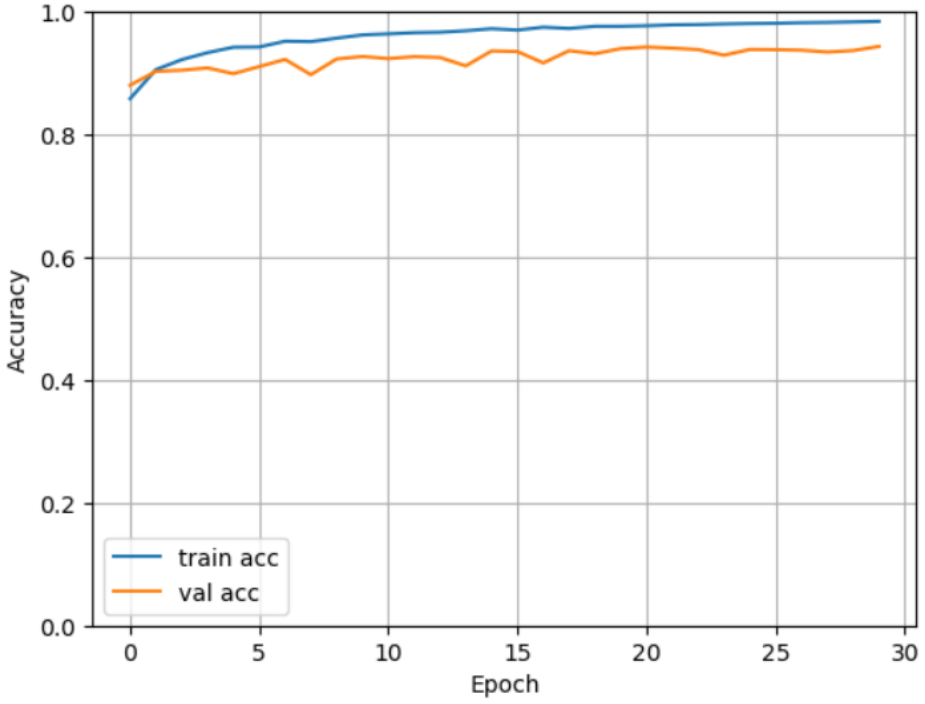

Fig. 12. VGG16 for crime prevention accuracy for training and testing

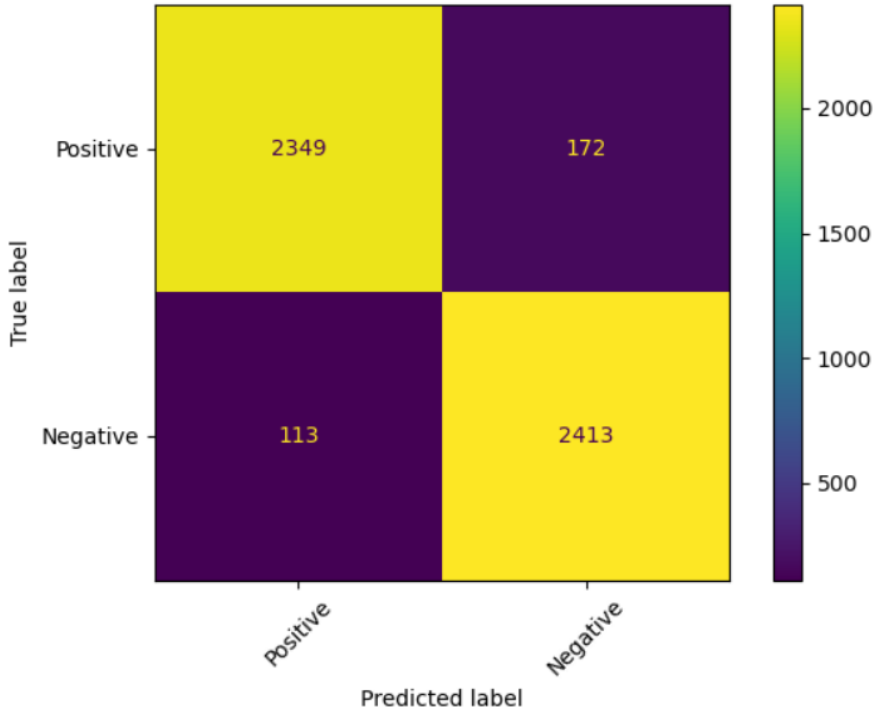

Fig. 13. VGG16 for crime prevention confusion matrix

The VGG16 model was used to detect violence and prevent crimes in smart cities, achieving an accuracy of 94%. The confusion matrix obtained was [[2349, 172], [113, 2413]], with a 0.3 dropout rate and a loss value for both the training and testing phases. Figure 14 shows the loss function for training and testing.

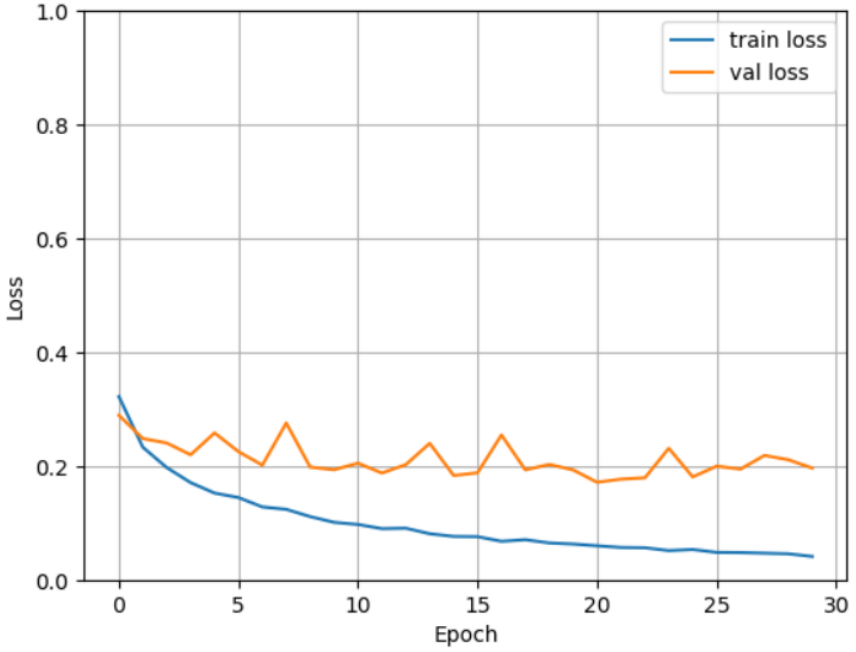

Fig. 14. Loss function for training and testing within the VGG16 model

8. CONCLUSIONS

The security and innovation of blockchain technology and its distributed ledger, whereby each node is responsible for its own security and maintenance, has led to the widespread use of this technology in many fields, including financial and supply chain applications, and in crime prevention systems, decision saving and sharing. Such applications may face issues arising from the technology itself, such as with respect to latency time and computational costs. To address these problems, a lightweight blockchain system is proposed on the basis of the integration of the proof of secret sharing consensus algorithm with modified Okamoto–Uchiyama homomorphic encryption technology.

The privacy of the data comprising each block is considered, and all sensitive data are encrypted via modified Okamoto–Uchiyama homomorphic encryption with three suggested modifications to improve the time latency and incorporate the Baker chaotic algorithm to increase the complexity of the algorithm without affecting the encryption/decryption speed.

The proposed system was demonstrated to possess efficient utilization of computational resources, reliable security metrics, and good scalability compared with the standard blockchain system via proof of work.

The proposed system achieves better results with respect to the evaluation metrics used, including transactions per second, transaction size, block size, block generation time, block verification time, final time, average CPU usage, average CPU user time, average CPU system time, average idle time, and average interrupt time.

The results of the evaluation indicate that the proposed lightweight blockchain system, integrated with a modified Okamoto–Uchiyama algorithm, offers a good solution to the known challenges facing crime prevention systems on the basis of the use of surveillance camera technology and control and monitoring servers. This system is able to enhance the security, privacy, and reliability of the decision collection process and mitigate problems related to data system hacking and the modification of real results during the transfer of classification results between crime prevention and the control and monitoring servers of smart cities.

In our future work, the implementation of lightweight blockchain in different applications in the context of the IoT and e-commerce will be explored.

Funding

This research received no external funding.

Data availability statement:

The authors confirm that the data supporting the findings of this study are available within the article and are available at this link at https://www.kaggle.com/mohamedmustafa/real-life-violence-situations-dataset.

Conflicts of interest:

The authors declare that they have no conflicts of interest.

References

[1] W. Al Saidi and A. M. Zeki, "The use of data mining techniques in crime prevention and prediction," in 2nd Smart Cities Symposium (SCS 2019), pp. 1-4, IET, 2019.

[2] R. F. Ghani, A. A. Salman, A. B. Khudhair, and L. Aljobouri, "Blockchain-based student certificate management and system sharing using hyperledger fabric platform," Periodicals of Engineering and Natural Sciences, vol. 10, no. 2, pp. 207-218, 2022.DOI: 10.21533/pen.v10i2.1953

[3] M. Z. Mohammed, H. B. A. Wahab, and A. M. J. A. Hossen, "Proposed Image Encryption Algorithm based on Paillier and Elliptic Curve Algebra," in 2021 14th International Conference on Developments in eSystems Engineering (DeSE), pp. 302-306, 2021.

[4] T. Okamoto and S. Uchiyama, "A new public-key cryptosystem as secure as factoring," in Advances in Cryptology—EUROCRYPT'98: International Conference on the Theory and Application of Cryptographic Techniques Espoo, Finland, May 31–June 4, 1998 Proceedings, vol. 17, pp. 308-318, Springer Berlin Heidelberg, 1998, doi: 10.1007/BFb0055716.

[5] M. Soliman, M. Kamal, M. Nashed, Y. Mostafa, B. Chawky, and D. Khattab, "Violence Recognition from Videos using Deep Learning Techniques," in Proc. 9th International Conference on Intelligent Computing and Information Systems (ICICIS'19), Cairo, Egypt, 2019, pp. 79-84.

[6] M. A. Mohammed and H. B. Abdul Wahab, "Proposed New Blockchain Consensus Algorithm," JIM, vol. 16, no. 20, pp. 162-167, Mar. 2022.

[7] T. A. Jaber, "Security Risks of the Metaverse World," Int. J. Interact. Mobile Technol., vol. 16, no. 13, pp. 1-7, Feb. 2022.

[8] A. Singh, G. Kumar, R. Saha, M. Conti, M. Alazab, and R. Thomas, "A survey and taxonomy of consensus protocols for blockchains," J. Syst. Archit., vol. 125, pp. 102503, Feb. 2022.

[9] A. Wood, K. Najarian, and D. Kahrobaei, "Homomorphic encryption for machine learning in medicine and bioinformatics," ACM Comput. Surv., vol. 53, no. 4, pp. 1-35, Nov. 2020.

[10] B. Alaya, L. Laouamer, and N. Msilini, "Homomorphic encryption systems statement: Trends and challenges," Comput. Sci. Rev., vol. 36, pp. 100235, Dec. 2020.

[11] M. R. Suma and P. Madhumathy, "Brakerski-Gentry-Vaikuntanathan fully homomorphic encryption cryptography for privacy preserved data access in cloud assisted Internet of Things services using glow-worm swarm optimization," Trans. Emerg. Telecommun. Technol., vol. 33, no. 12, pp. e4641, Dec. 2022.

[12] G. Xu, J. Zhang, U. G. O. Cliff, and C. Ma, "An efficient blockchain-based privacy-preserving scheme with attribute and homomorphic encryption," Int. J. Intell. Syst., pp. 1-18, Feb. 2022.

[13] T. A. Jaber and M. A. Hussein, "Study on known models of NB-IoT Applications in Iraqi environments," in IOP Conf. Ser. Mater. Sci. Eng., vol. 518, no. 5, p. 052013, 2019.

[14] T. A. Jaber, "Artificial intelligence in computer networks," Period. Eng. Nat. Sci., vol. 10, no. 1, pp. 309-322, 2022.

[15] Z. A. Kamal and R. Fareed, "Data retrieval based on the smart contract within the blockchain," Periodicals of Engineering and Natural Sciences, vol. 9, no. 4, pp. 491-507, Oct. 2021.

[16] R. M. Zaki and H. B. A. Wahab, "A novel of substitution-box design using PLL algorithms in magic cube," Period. Eng. Nat. Sci., vol. 9, no. 4, pp. 674-684, 2021.

[17] Zaki, Rana Mohammed. "SNOW3G Modified by using PLL Algorithms in Magic Cube." Iraqi Journal of Computers, Communications, Control and Systems Engineering 22, no. 3 (2022): 1-8.

[18] Hussein, M. A. (2023). A Proposed Multi-Layer Firewall to Improve the Security of Software Defined Networks. International Journal of Interactive Mobile Technologies, 17(2).

[19] W. G. Yass and M. Faris , Trans., “Integrating computer vision, web systems and embedded systems to develop an intelligent monitoring system for violating vehicles”, BJIoT, vol. 2023, pp. 69–73, Sep. 2023, doi: 10.58496/BJIoT/2023/009.

[20] C. K. Koç, F. Özdemir, and Z. Ödemiş Özger, "Okamoto-Uchiyama Algorithm," in Partially Homomorphic Encryption, Cham, Switzerland: Springer, 2021, pp. 83-93.

[21] L. Zhou, L. Wang, Y. Sun and P. Lv, "Beekeeper: A blockchain-based iot system with secure storage and homomorphic computation," in IEEE Access, vol. 6, pp. 43472-43488, 2018.

[22] F. Loukil, C. Ghedira-Guegan, K. Boukadi, and A.-N. Benharkat, "Privacy-preserving IoT data aggregation based on blockchain and homomorphic encryption," Sensors, vol. 21, no. 7, pp. 2452, Apr. 2021.

[23] She, W.E.I., Gu, Z.H., Lyu, X.K., Liu, Q.I., Tian, Z. and Liu, W., 2019. Homomorphic consortium blockchain for smart home system sensitive data privacy preserving. IEEE Access, 7, pp.62058-62070.
[24] L. Hussain, “Fortifying AI Against Cyber Threats Advancing Resilient Systems to Combat Adversarial Attacks”, EDRAAK, vol. 2024, pp. 26–31, Mar. 2024, doi: 10.70470/EDRAAK/2024/004.
[25] W. Qu, L. Wu, W. Wang, Z. Liu and H. Wang, "A electronic voting protocol based on blockchain and homomorphic signcryption," Concurrency and Computation: Practice and Experience, vol. 34, no. 16, e5817, 2022.
[26] C. Sravani and G. Murali, "Secure electronic voting using blockchain and homomorphic encryption," Int. J. Recent Technol. Eng, vol. 8, no. 2, pp. 1002-1007, 2019.
[27] H. Kim, K. E. Kim, S. Park, and J. Sohn, "E-voting System Using Homomorphic Encryption and Blockchain Technology to Encrypt Voter Data," arXiv preprint arXiv:2111.05096, Nov. 2021.
[28] F.N.S. Vanin, L.M. Policarpo, R.D.R. Righi, S.M. Heck, V.F. da Silva, J. Goldim and C.A. da Costa, "A Blockchain-Based End-to-End Data Protection Model for Personal Health Records Sharing: A Fully Homomorphic Encryption Approach," Sensors, vol. 23, no. 1, pp. 14, Jan. 2022.
[29] T. A. Jaber, "Sensor based human action recognition and known public datasets a comprehensive survey," in AIP Conference Proceedings, vol. 2591, no. 1, 2023, p. 020031.
[30] Ezzeldin, Mustafa, Amr Ghoneim, Laila Abdelhamid, and Ayman Atia. "Survey on Multimodal Complex Human Activity Recognition." FCI-H Informatics Bulletin 7, no. 1 (2025): 26-44.
[31] Hephzipah, J. Jasmine, Ranadheer Reddy Vallem, M. Sahaya Sheela, and G. Dhanalakshmi. "An efficient cyber security system based on flow-based anomaly detection using Artificial neural network." Mesopotamian Journal of Cybersecurity 2023 (2023): 48-56.
[32] Y. L. Khaleel, M. A. Habeeb, and H. Alnabulsi , Trans., “Adversarial Attacks in Machine Learning: Key Insights and Defense Approaches ”, Applied Data Science and Analysis, vol. 2024, pp. 121–147, Aug. 2024, doi: 10.58496/ADSA/2024/011.
[33] S. Q. Y. Al-Hashemi, M. S. Naghmash, and A. Ghandour, “Development of Smart Video Surveillance Systems: Technical and Security Challenges in Urban Environments”, SHIFRA, vol. 2024, pp. 24–38, Mar. 2024, doi: 10.70470/SHIFRA/2024/004.