sustainability

MDPI

Article

Blockchain-Based Community Safety Security System with IoT Secure Devices

Chin-Ling Chen [1,2,3], Zi-Yi Lim [3,*] and Hsien-Chou Liao [3,*]

[1] School of Information Engineering, Changchun Sci-Tech University, Changchun 130600, China; clc@mail.cyut.edu.tw
[2] School of Computer and Information Engineering, Xiamen University of Technology, Xiamen 361024, China
[3] Department of Computer Science and Information Engineering, Chaoyang University of Technology, Taichung 41349, Taiwan
* Correspondence: zylim@cyut.edu.tw (Z.-Y.L.); hcliao@cyut.edu.tw (H.-C.L.)

Abstract: Humans frequently need to construct a huge number of buildings for occupants in large cities to work or live in a highly developed civilization; people who live in the same building or same area are defined as a community. A thief stealing items, a burglary, fire hazards, flood hazards, earthquakes, emergency aid, abnormal gas leakage, strange behavior, falling in a building, fainting in a building, and other incidents all threaten the community's safety. Therefore, we proposed a blockchain-based community safety security system that is combined with IoT devices. In the proposed scheme, we designed multiple phases to process the alarm triggered by IoT devices. IoT devices can be set up in two types areas: private and public areas. Both types of IoT devices' alarms have different process flow for the response and records checking phase. All records are saved in the Blockchain Center to assure the data can be verified and cannot be forged. During the communication between sender and receiver, we implemented some security methods to prevent message repudiation, prevent transmission intercept, prevent replay attacks, and ensure data integrity. We also implemented a clarifying mechanism to ensure that all system participants can have confidence in the system's processing methods. The proposed scheme can be used in communities to improve community safety and prevent unnecessary conflicts.

Keywords: community safety; security system; blockchain; Internet of Things; security analysis

Citation: Chen, C.-L.; Lim, Z.-Y.; Liao, H.-C. Blockchain-Based Community Safety Security System with IoT Secure Devices. *Sustainability* **2021**, *13*, 13994. https://doi.org/10.3390/su132413994

Academic Editor: Fadi Al-Turjman

Received: 20 November 2021
Accepted: 15 December 2021
Published: 18 December 2021

1. Introduction

1.1. Background

In a highly developed society, people often need to build a lot of buildings to allow employees or occupants in large cities to work or live; those people in the same building become a community in the city. Generally, working or living in a community is often inseparable from the problem of safety. Recently, there have been serious fire incidents in Taiwan [1,2], causing many injuries and deaths. After the investigation, the building did not have fully functional security sensors or systems, and when a fire broke out, it did not notify the community's occupants, which resulted in the tragedy. Many incidents can harm the safety of the community, including thieves stealing things, burglary, fire hazards, flood hazards, earthquake, emergency help, abnormal gas leakage, abnormal behavior, falling in building, fainting in building, etc.

The above-mentioned unsafe concerns are common in community buildings. The community should construct a community safety system to prevent such problems. Furthermore, more people started to deploy private security guards services in their communities, and these have become hot topics to discuss [3–6].

Some of the communities deploy smart building or smart home technologies to solve the community safety problem; most of these technologies are related to various types of

Internet of Things (IoT) devices (e.g., sensors or surveillance cameras) to achieve the function, such as fire detection, gas leaked detection, motion detection, abnormal detection, human intruder detection, access control, fall detection, earthquake detection, etc. According to a smart buildings market report [7], the research company shows the market is projected to grow from $66.3 billion in 2020 to $108.9 billion in 2025. In addition, due to improvements in medical science and technology, life expectancy has grown rapidly over the world in recent years. Humans are living longer lives, resulting in a fast increase in the number of old people in the population. Around 727 million persons over the age of 65 were recorded globally in 2020, with the number predicted to be 1.5 billion in 2050 [8].

As a result, a safety device or system is one of the community's required components to guarantee that the community's occupants have a safer living or working environment.

1.2. Related Works

Several related works are surveyed as follows. Dutta et al. proposed a system called "enhanced security system for smart building using IoT (ES3B) [9]. The authors proposed an IoT device with Arduino, RFID, and Bluetooth for access control. An exception to avoid non-resident going into the building, the guest information also can be built by a resident with an RFID owner. Except for the access control, another author Prasetyo et al. proposed an IoT device with multiple sensors (e.g., metal sensor, fire sensor, vibrator sensor, PIR Sensor, etc.) and a Raspberry Pi camera to detect the threats in the office [10]. The device can detect threats as follows: dangerous objects made from metals, fires, earthquakes, intruders, or theft.

Moreover, Saad et al. designed a fire detection system to prevent fire hazards [11]. The system proposed by the authors is implemented with ZigBee technology as a protocol to receive the data from multiple sensors, such as smoke sensor, gas sensor, heat sensor, and UI/IV sensor. Then, the system processes the data with Raspberry Pi to identify fire. Furthermore, Taryudi et al. proposed a home security and monitoring system with various types of sensors and integrated with a microcontroller [12]. The system can be monitored and controlled remotely with the smartphone application. To allow building security guards to monitor all the building status more effectively, Al-Hudhud et al. proposed a security guard system with an infrared biosensor and augmented reality device to monitor the IoT status [13].

The above-mentioned authors proposed safety systems with IoT, but some security issues will cause the IoT system to disable. Ray et al. pointed out the security issues and challenges in the smart home system [14]: what is important is that the network between routers or gateway to IoT will be the vulnerabilities of the system. Except for the physical network equipment that should be improved, more researchers are implementing the IoT system with blockchain technology in recent years. Khan et al. proposed a data verification system for surveillance cameras with blockchain [15]. Rahman et al. proposed a distributed IoT Software-Defined Network(IoT-SDN) model to ensure the security and privacy of condominium networks [16]. Furthermore, Khalid et al. also designed an authentication mechanism for IoT systems with blockchain [17]. Unfortunately, the authors did not analyze their system security to prove that their proposed scheme is feasible.

As evident in the above-mentioned research, not all research can provide complete system architecture and system security analysis. The research gap has been addressed as follows: (1) Although many researchers proposed IoT applications in smart cities or smart buildings, fewer established a decentralized or blockchain-based system. (2) Less research exists that shows the completed architecture for a community safety application in a blockchain system. (3) There seems to be no scheme to protect the device or system against threats (such as message repudiation, data integrity, cyberattacks, and so on), and there is no security analysis for the proposed scheme. (4) If a dispute occurs, there is a lack of discussion in reading the historical record in encrypted data for clarification process.

In recent years, more and more scholars or industries have begun to use blockchain as the basis of system architecture to execute or store records because of the advantages

of blockchain, such as decentralization, unforgeable data, traceability, and clarifiable illegal records. The type of blockchain can be separated into public, private, hybrid blockchain [18,19]. The public blockchain is a high decentralized blockchain, permissionless, with high power consumption, and low throughput for executing smart contracts. Conversely, the private blockchain is a low decentralized blockchain, access controlled, low power consumption, and high throughput. The representative private blockchain today is Hyperledger Fabric [20]. There are more and more applications implemented based on the Hyperledger Fabric, such as hospital information system [21], access control system [22], supply chain management [23], etc. In addition, Chen et al. have also proposed blockchain-based schemes in brand clothing industries [24] and insurance industrial [25].

As a consequence, we proposed a more secure IoT safety security system that applies IoT and blockchain technologies. We proposed a system with the HyperledgerFabric-based [20] blockchain. HyperledgerFabric-based blockchain's characteristics make it suitable to be used in the community safety system, and it is utilized to improve the system's security.

The outline of the remaining sections is as follows. Section 2 introduces the technologies that are used in our research. Section 3 proposes our architecture and research method. Then, the security issues analysis and some performance discussion are given in Sections 4 and 5. Lastly, we conclude this paper in Section 6.

2. Preliminary

2.1. Internet of Things (IoT) Devices

Internet of Things (IoT) is an electronic "thing" that can connect to the internet. Scholars have also made a clear definition of IoT: "*the pervasive presence around us of a variety of "things" or "objects", such as RFID, sensors, actuators, mobile phones, which, through unique addressing schemes, are able to interact with each other and cooperate with their neighboring "smart" components to reach common goals*" [26]. In this generation, IoT is continuously implemented and applied around us; our lives are gradually inseparable from IoT. The IoTs are widely used in agriculture, retailers, smart manufacturing, smart home, smart building, smart transportation, smart city, and so on.

In this research, we apply the IoT on the application in building safety. According to the research we found, it can be found that various types of IoT devices are proposed to apply to building safety applications, such as:

- Fire detection: device with detection sensor, smoke detection sensor to prevent a fire hazard [27,28].
- Poison gas alert: device with a variety of gas sensors to prevent gas leaks, such as methane, carbon monoxide, smoke, nitrogen dioxide, and propane [29–31].
- Motion detection: device with infrared sensor or camera to detect motion in an area [32–34].
- Abnormal detection: smart surveillance system implemented by Artificial Intelligence (AI) that can detect some abnormal situation in the images [35–38].
- Human intruder detection: detect humans by analyzing ground vibrations [39].
- Fall detection: detect with cameras or 3-axis sensors to prevent human falls in the building [40–44].
- Smart door: authentication with face recognition, fingerprint, smartphone, or pin code to unlock the door without keys [45–47].
- Earthquake detection: microelectromechanical systems accelerometer detects vibration of an area [48–50].

From the above IoT device application, we can know that IoT technology is already a very mature technology. Therefore, these functional IoT devices or algorithms can be applied or integrated into our proposed safety security system.

2.2. Blockchain-Based Smart Contract

The smart contract is an execution program that is full of transaction logic, which comes from the transformation of a traditional contract in the real world. In today's era, we can meet smart contracts all around the world, such as vending machines and online shopping platforms. Many smart contracts require cash, credit card, or digital currency to process the transactions, but with the rapid development of blockchain and cryptocurrency, many smart contract applications have begun to be deployed on the blockchain network.

Szabo has developed a concept of decentralized digital assets called "Bit Gold". It is considered a pioneer before the advent of Bitcoin [51,52]. The technology of Bitcoin was carried forward by Nakamoto [53], who proposed and announced Bitcoin in a mysterious fashion. Bitcoin is a blockchain-based cryptocurrency technology. All transaction records of Bitcoin must be verified by most hosts before they can be synchronized to all participant hosts to avoid problems such as tampering and forgery.

In addition to Bitcoin, the most popular cryptocurrency on the market is Ethereum [54]. Ethereum provides faster and more stable transaction capabilities than Bitcoin. It also provides the function of deploying smart contracts to the public blockchain. With the characteristics of Ethereum, more commercial frameworks of blockchain systems have been developed, such as Hyperledger Fabric [20], Corda [55], and Azure Blockchain [56].

In particular, Hyperledger Fabric is an open-source licensed blockchain framework. It is a blockchain framework that was proposed by IBM in 2018 [57]. It adopts a modular universal framework, unique identity management, access control functions, and channels to transfer data. Those features are suitable for various industrial applications, such as supply chain tracking [58,59], financial management [60], insurance financial [61], healthcare records [62,63], etc. The advantages of the blockchain's characteristics are listed as follows:

1. Decentralization: The operating mode of the blockchain is a technology composed of multiple decentralized peers. All ledger data will be synchronized to all participating peers.
2. Authentication: All participants need to be registered on the blockchain's Certificate Authority (CA); the participant receives the authentication certificates from the CA. Then, the participants' transactions can be updated to the ledger after authentication.
3. Privacy and anonymity: Unlike the other public blockchains like Ethereum, Fabric-based blockchains have a feature that allows peer-to-peer transactions privately in the channel. Except for the transaction, any log of transactions saved in the ledger is kept secret and anonymous.
4. Unforgeable data: Every peer is storing the same content of the ledger in the blockchain. All the transactions invoked by participants need to be saved as logs in the ledger, the data are chaining between block to block. Every block also records the previous block's hash value as a link between blocks, so it is hard to forge data.
5. Traceability: Because of the unforgeable data characteristics of the blockchain mentioned above, this also makes the modification record of the data traceable.
6. Clarifying illegal records: All change records will be synced in the blockchain peer's ledger. There are signatures or timestamp data to provide evidence to protect the rights of the community's occupants if any conflicts arise in the future.

Therefore, the characteristics of HyperledgerFabric-based blockchain with the smart contract are much more suitable to implement in our safety security system.

2.3. Threat Model

Furthermore, we have sorted out the threat that must be addressed, such as system security vulnerabilities and attacks from third parties. The related threats are described as follows:

1. Message repudiation issues: In the general safety security system, some issues should be solved. Message repudiation is one of the issues, and we must ensure that the message sent to the receiver was indeed sent by the sender [64].

2. Data integrity issues: Sometimes a network or attacker will damage the data during transfer, which makes the data come to non-integrity. Moreover, data integrity must be maintained in the database so that there are no inconsistencies when security records are traced later [64,65].
3. Transmission intercept: The message that is transmitted in the network is easily intercepted by the attackers, for example, man-in-the-middle attacks. The action makes the message sent between sender and receiver become disclosed [66,67].
4. Replay attacks: When attackers intercept an original message sent in the network, the attacker can resend the same message to pretend the attacker is the original sender [68,69].

3. Proposed Architecture and Methods

We proposed a community safety security system with IoT and blockchain technologies. The proposed system architecture is presented in Figure 1. The proposed system is constituted of the following parties: blockchain center, occupants, security guards, log server, and IoT devices (including cameras and sensors).

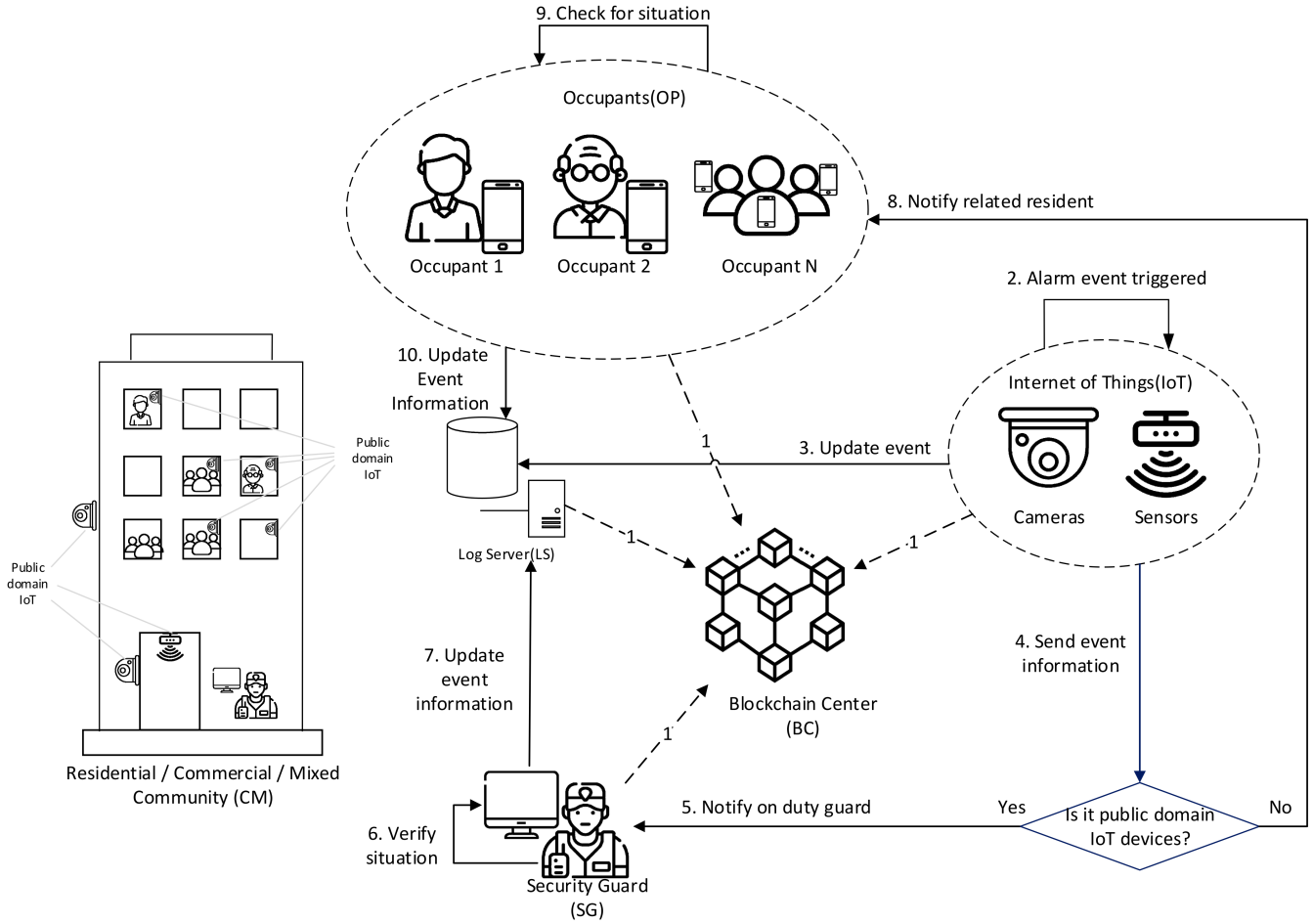

Figure 1. Proposed system architecture.

3.1. System Architecture

Firstly, all the involved parties in this system are introduced as follows in detail:

1. Blockchain Center (BC): A blockchain center composed of multiple device nodes that are storing all the records from IoT. All the involved parties must register in the blockchain center. The monitoring records of the community are saved in the blockchain center. The records in the blockchain center are unforgeable and verifiable.

When the parties request to view the specified record in detail, the blockchain center will send votes to other occupants. If more than half of the votes are agreed, the parties can view with the security guard.

2. Community (CM): We separate the community type into three types: residential, commercial, and mixed. It is a physical structure in which occupants live or work. Inside the community is installed with several public or private domain IoT devices, variance age or type of occupants, and at least one security guard.
3. Occupants (OP): The occupants who live or work in the residential/commercial/mixed community. All occupants involved in this system are given the option to choose whether to install the IoT devices in their private spaces to ensure the safety of their property. Every occupant must have a mobile phone with a decentralized application (dAPP). The dAPP in the mobile phone is an application that connects to the blockchain center. The application needs to registers and login with the blockchain center. The occupants have the right to give the security guard permission to view the monitoring records from his/her private domain IoT.
4. Security Guard (SG): The security guard hired by the community's management administrator. The SG must monitor the condition of the community at all times. If a dangerous incident occurs, the SG must deal with it immediately to ensure the safety of the community. To view any public or private domain IoT record, SG must request the blockchain center to read the records.
5. Supervisor (SP): The security guards' supervisor is in charge of the responsibility of managing security guards. The supervisor is hired by the community's management administrator. The security guards who are on duty need the supervisor's permission to check for the public domain history record.
6. Internet of Things devices (IoT): The IoT included the security sensor, for example, cameras and sensors. The cameras take images from the corners of the community and detect suspicious events, e.g., burglary and abnormal behaviors.
    a. The cameras in the community can be categorized into two types: public domain cameras and private domain cameras. Public domain cameras are installed at the public corner of the community, the installation of the private domain cameras can be chosen by occupants, and they can choose how many cameras and which position they need to install.
    b. The sensors in the community can be smoke detectors, motion detectors, emergency buttons, or more devices that can help to notice dangerous moments.
7. Log Server (LS): Every video record and event from the camera and sensors in the community are saved to the server. The server can be a physical device in a community or a cloud service on the internet. SG needs access to the log server to read the records.

Figure 1 shows the overall system architecture, and the detailed process flow with numbered is description is as follows:

Step 1. Every participant must register and get a private key and public key from BC.
Step 2. An alarm event triggered by an IoT device.
Step 3. The IoT device updates the event information and status to LS and updates the information via chaincode to BC.
Step 4. If the triggered IoT device is the public domain device, it will send the event information to SG in the next step (step 5), otherwise, the information will be sent to the relevant occupant in step 8.
Step 5. The event information sends to the security guard in the community, the security guard received the event in a surveillance system on the LS.
Step 6. SG receive the event information and check for the video record immediately to verify the situation.
Step 7. When the situation is checked, SG starts to resolve the alarm event and update the resolved information as a remark to LS and update the information to BC via chaincode.

Step 8. If the IoT device is a private domain, the alarm event information will be sent to the related occupant's mobile phone application.
Step 9. The occupant checks the situation with his/her application on the mobile phone.
Step 10. The occupant responds and updates the event information to BC. If the occupant needs SG to resolve the situation of the alarm event, then go to step 4. SG will be notified, and SG will help to deal with it. Every action chosen from OP will update to BC via invoking chaincode.

3.2. Notations

The notations used in the following sections are described as shown in Table 1.

Table 1. The description of the notations.

Notations	Description
ID_X	X is the identity of the participant (such as IoT devices and occupants), issued by the blockchain center.
ID_E	The event ID that generated by IoT devices, included the ID of IoT and User. The format is [UserID + IoTID + timestamp]
q	A k-bit of prime number
$GF(q)$	Finite group of q
E	The elliptic curve defined on finite group
G	A generating point based on the elliptic curve E
k_i	The i^{th} random value on the elliptic curve
(r_{X_i}, s_{X_i})	Elliptic curve signature value of X
(x_{X_i}, y_{X_i})	The ECDSA signature value of X
d_X	The ECDSA's private key of participant X
Q_X	The ECDSA's public key of participant X
Puk_X	The public key of party X, issued by the BC's CA
Prk_X	The private key of party X, issued by the BC's CA
C_{X_i}	The ith ciphertext of X
$H(M)$	One way hash function
h_{X_i}	The ith hash value of X
T_i	The ith timestamp
τ	The threshold for checking the validity of a timestamp
M_i	The ith message from a sender
$E_{Puk_X}(M)/D_{Prkx_X}(M)$	Encrypt or decrypt message M with a public key or private key of participant X

3.3. Initialization and Registration Phase

All participants that want to be a part of the system need to register with BC's certificate authority (CA). CA generates the public key and private key pair and sends it to the participant. Figures 2 and 3 show the chaincode's data structures and types of information of a user, IoT, and event information. Note that the "chaincode" is a "smart contract" that is defined in Hyperledger Fabric. It is almost the same function as the "smart contract" we mentioned before, which can execute the code in the blockchain.

Figure 4 shows the process of the registration phase between the new participant (X) and the blockchain center's certificate authority (CA). The steps are described as follows:

Step 1. Participant X sends the information of registration to CA.
Step 2. CA generates a set of a private key d_X and a public key Q_X of the Elliptic Curve Digital Signature Algorithm (ECDSA) [70]. The Q_X is generated by the follows equation:

$$Q_X = d_X G \quad (1)$$

```
type User_Information struct {
        User_ID string
        Name string
        Detail string
        Address string
        Telephone string
        var Role RoleType
        var IoTs []IoT_Information
}

type IoT_Information struct {
        IoT_ID string
        User_ID string
        Name string
        Detail string
        var IoT_Type IoTType
        var Privacy PrivacyType
}
```

```
type RoleType string
const(
        Occupant
        SecurityGuard
        Supervisor
        IoT
)
type IoTType string
const(
        Camera
        Sensor
)
type PrivacyType string
const(
        Public
        Private
)
```

Figure 2. Structure of user information, IoT information, and the enumeration of three types (Role, IoT, and Privacy).

```
type Event_Information struct {
        Event_ID string
        IoT_ID string
        User_ID string
        var IoT_Type IoTType
        Detail string
        Remark string
        ResponseMessage string
        Triggered_Datetime time.Time
        LS_Received_Datetime time.Time
        Notify_Datetime time.Time
        User_Received_Datetime time.Time
        User_Response_Datetime time.Time
        SG_Received_Datetime time.Time
        SG_Response_Datetime time.Time
        LS_Response_Datetime time.Time
        IoT_Triggered_Signature string
        LS_Received_Signature string
        User_Received_Signature string
        User_Response_Signature string
        SG_Received_Signature string
        SG_Response_Signature string
        LS_Response_Signature string
}
```

Figure 3. Structure of triggered event information.

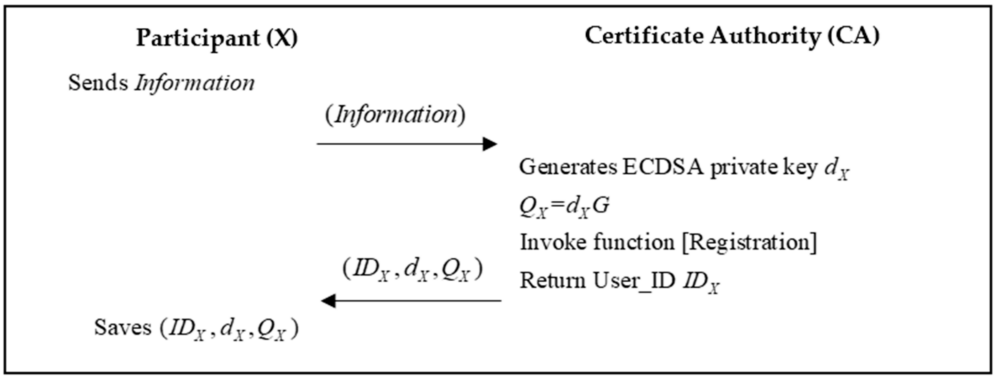

Figure 4. The flowchart of the registration phase for a participant.

Next, CA invokes the chaincode function "Registration" in Algorithm 1 to generate participant X's user ID_X. The chaincode also updates the user's information to the BC ledger.

Step 3. Participant X gets and saves the unique parameters ID_X, Q_X, and d_X for future transactions.

Algorithm 1. Chaincode function of registration and add IoT.

```
var UserList []User_Information
func Registration (var user User_Information) (User_ID string) {
User_ID = GenerateUniqueID()
user.User_ID = User_ID
UserList = append (UserList, user)
return User_ID
}
func Add_IoT(User_ID string, IoT IoT_Information) (IoT_ID string, d string, Q string) {
index: = SearchUID(User_ID)
IoT.IoT_ID = GenerateUniqueIoTID()
UserList[index].IoTs = append(UserList[index].IoTs, IoT)
d = GenerateECDSAPrivateKey()
Q = d * G
return (IoT.IoT_ID, d, Q)
}
```

Furthermore, every IoT device in the CM must also register with CA. The registration flow of adding a new IoT is shown in Figure 5. The details are as follows:

Step 1. Firstly, user X generates a random number k_1, then generates the message with a sender ID, receiver ID, IoT's information, and timestamp.

$$M_1 = (ID_X||ID_{CA}||< IoT\_Information >||T_1) \quad (2)$$

A hash value h_{X_1} is calculated by a hash function.

$$h_{X_1} = H(M_1) \quad (3)$$

Then, X executes the "Sign" function with parameters in Algorithm 2 to get a set of ECDSA signatures (r_{X_1}, s_{X_1}). The detailed calculation is described in the authentication phase.

$$(r_{X_1}, s_{X_1}) = Sign(h_{X_1}, k_1, d_X) \quad (4)$$

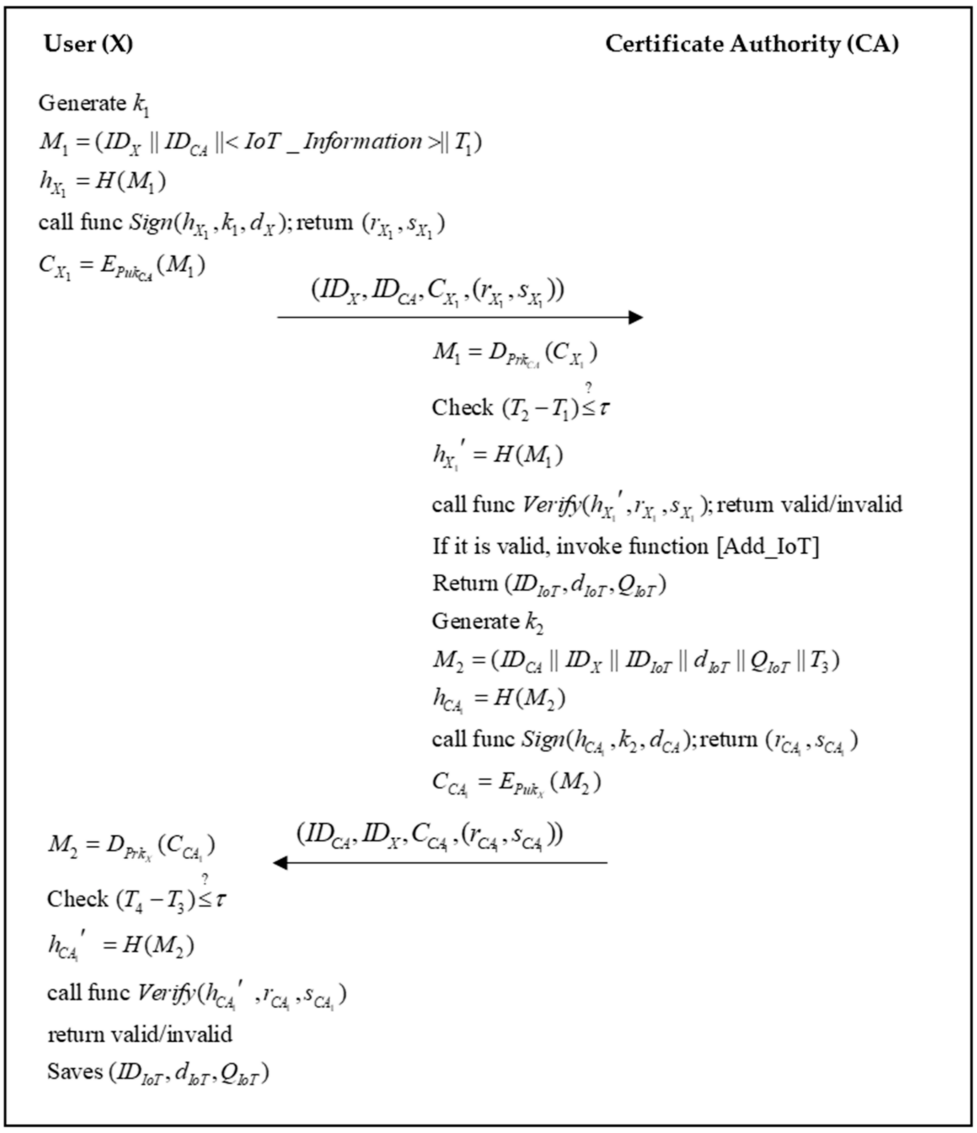

Figure 5. The flowchart of the registration phase for IoT devices.

Next, user X encrypts the message M_1 into a cipher message C_{X_1}. Then, user X sends cipher messages with signatures $(ID_X, ID_{CA}, C_{X_1}, (r_{X_1}, s_{X_1}))$ to CA.

$$C_{X_1} = E_{Puk_{CA}}(M_1) \tag{5}$$

Step 2. After CA receives the message from X, CA decrypts the cipher message with its private key and gets the decrypted messages.

$$M_1 = D_{Prk_{CA}}(C_{X_1}) \tag{6}$$

CA also check the validity of the message's timestamp.

$$\text{Check } (T_2 - T_1) \stackrel{?}{\leq} \tau \tag{7}$$

Then, CA calculates the hash value $h_{X_2}{}'$ from the message M_1 and invokes the "Verify" function in Algorithm 2 to check the validity of signatures. The "Verify" function is described in the next subsection "authentication phase".

$$h_{X_2}{}' = H(M_3) \tag{8}$$

$$(\text{valid/invalid}) = Verify(h_{X_1}{}', r_{X_1}, s_{X_1}) \tag{9}$$

If the signatures are valid, CA invokes a chaincode function "Add_IoT" in Algorithm 1 to update the IoT device's information into BC. Then, CA sends a response message to user X. Firstly, CA generates a random number k_2 and a response message with the IoT's ID ID_{IoT} and ECDSA parameters d_{IoT} and Q_{IoT}.

$$M_2 = (ID_{CA}||ID_X||ID_{IoT}||d_{IoT}||Q_{IoT}||T_3) \tag{10}$$

Then, CA calculates signatures with the hash value h_{CA_1}. The signatures are calculated by calling the "Sign" function in Algorithm 2, then returning the signatures (r_{CA_1}, s_{CA_1}).

$$h_{CA_1} = H(M_2) \tag{11}$$

$$(r_{CA_1}, s_{CA_1}) = Sign(h_{CA_1}, k_2, d_{CA}) \tag{12}$$

Next, CA encrypts the message to cipher message by using user X's public key.

$$C_{CA_1} = E_{Puk_X}(M_2) \tag{13}$$

Step 3. User X receives the response message from CA. Then, user X decrypts the cipher message with his/her private key and gets the decrypted messages.

$$M_2 = D_{Prk_X}(C_{CA_1}) \tag{14}$$

User X checks the message's timestamp is valid or not:

$$\text{Check } (T_4 - T_3) \stackrel{?}{\leq} \tau \tag{15}$$

Then, user Y calculates the hash value $h_{CA_1}{}'$ from the message and invokes the "Verify" function in Algorithm 2 to check the validity of signatures by the following equations.

$$h_{CA_1}{}' = H(M_2) \tag{16}$$

$$(\text{valid/invalid}) = Verify(h_{CA_1}{}', r_{CA_1}, s_{CA_1}) \tag{17}$$

The function returns "valid" if the signatures sent from user Y are valid. Then the IoT device saves parameters $(ID_{IoT}, d_{IoT}, Q_{IoT})$ to its configurations for future communication in the system.

3.4. Authentication Phase

In the authentication phase, we assigned user X as a sender and user Y as a receiver. As shown in Figure 6, every sender and receiver need to sign and verify their message by invoking the "Sign" and "Verify" function in Algorithm 2. The following are the steps in detail.

Step 1. User X generates a random number k_3 and generates a message with a timestamp, sender ID, and receiver ID.

$$M_3 = (ID_X||ID_Y||T_5) \tag{18}$$

Then, a hash value h_{X_2} is calculated by a hash function

$$h_{X_2} = H(M_3) \tag{19}$$

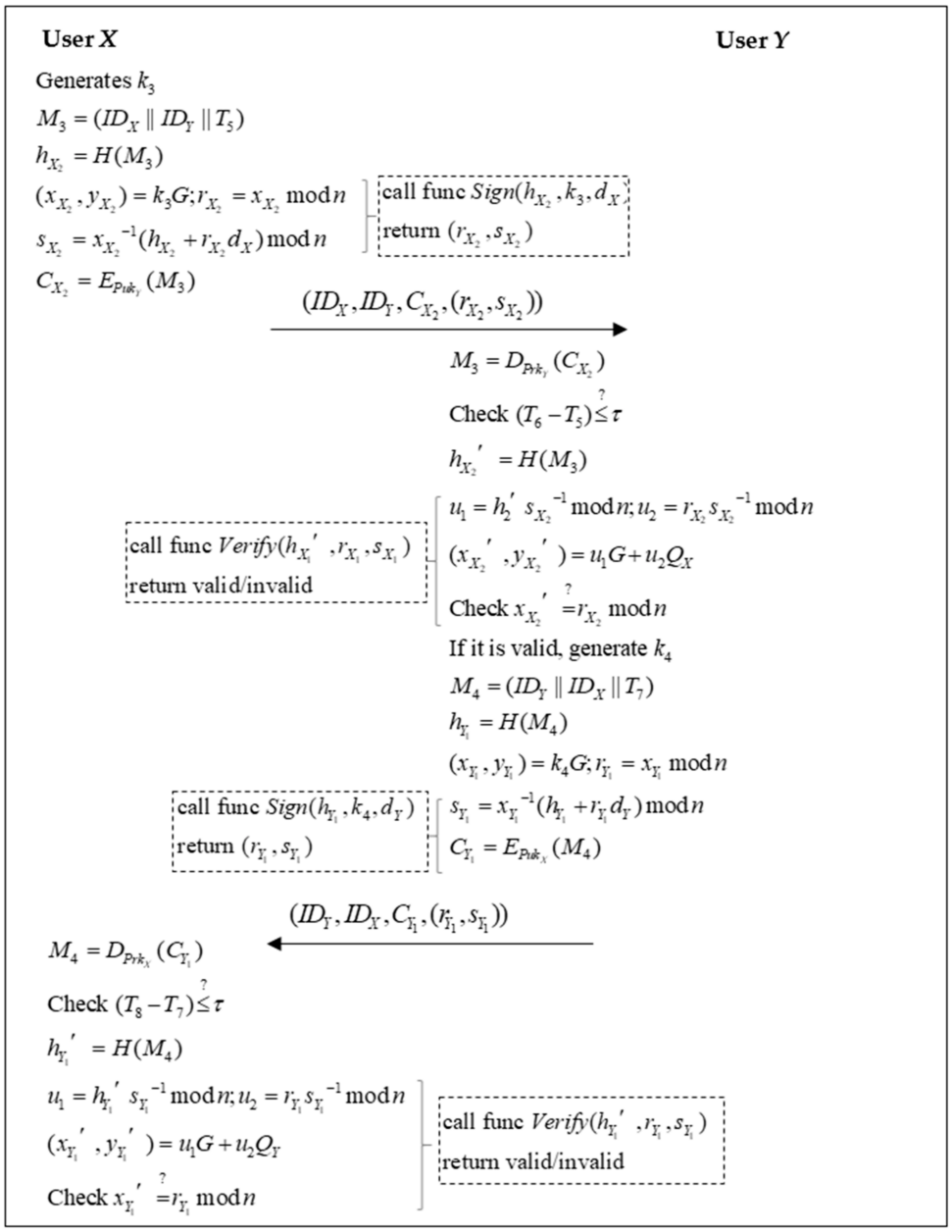

Figure 6. The flowchart of the phase of authentication.

Then, X executes the "Sign" function with parameters in Algorithm 2 to return a set of ECDSA signatures (r_{X_2}, s_{X_2}). The details are described in the following equations.

$$(x_{X_2}, y_{X_2}) = k_3 G \tag{20}$$

$$r_{X_2} = x_{X_2} \bmod n \tag{21}$$

$$s_{X_2} = {x_{X_2}}^{-1}(h_{X_2} + r_{X_2} d_X) \bmod n \tag{22}$$

Next, user X encrypts the message M_3 into a cipher message C_{X_2}. Then, user X sends cipher messages with signatures $(ID_X, ID_Y, C_{X_2}, (r_{X_2}, s_{X_2}))$ to user Y:

$$C_{X_2} = E_{Puk_Y}(M_3) \tag{23}$$

Step 2. User Y receives the message from user X. Then, user Y decrypts the cipher message with his/her private key and gets the decrypted messages.

$$M_3 = D_{Prk_Y}(C_{X_2}) \tag{24}$$

Then, user Y checks whether the message's timestamp is valid or not.

$$T_6 - T_5 \stackrel{?}{\leq} \tau \tag{25}$$

Then, user Y calculates the hash value ${h_{X_2}}'$ from the message M_3 and invokes the "Verify" function in Algorithm 2 to check the validity of signatures.

$${h_{X_2}}' = H(M_3) \tag{26}$$

$$u_1 = {h_2}'{s_{X_2}}^{-1} \mathrm{mod} n \tag{27}$$

$$u_2 = r_{X_2}{s_{X_2}}^{-1} \mathrm{mod} n \tag{28}$$

$$({x_{X_2}}', {y_{X_2}}') = u_1 G + u_2 Q_X \tag{29}$$

$$\text{Check } {x_{X_2}}' \stackrel{?}{=} r_{X_2} \mathrm{mod} n \tag{30}$$

If the signatures are valid, then user Y sends a response message to user X. Firstly, user Y generates a random number k_4 and a response message with a timestamp.

$$M_4 = (ID_Y || ID_X || T_7) \tag{31}$$

Then, user Y calculates signatures with the hash value h_{Y_1}. The signatures are calculated by calling the "Sign" function in Algorithm 2, and the function returns the signatures (r_{Y_1}, s_{Y_1}) after the following calculations.

$$h_{Y_1} = H(M_4) \tag{32}$$

$$(x_{Y_1}, y_{Y_1}) = k_4 G \tag{33}$$

$$r_{Y_1} = x_{Y_1} \mathrm{mod} n \tag{34}$$

$${s_{Y_1}} = {x_{Y_1}}^{-1}(h_{Y_1} + r_{Y_1} d_Y) \mathrm{mod} n \tag{35}$$

Then, user Y encrypts the message to cipher message with user X's public key.

$$C_{Y_1} = E_{Puk_X}(M_4) \tag{36}$$

Step 3. User X receives the response message from Y. Then, user X decrypts the cipher message with his/her private key and gets the decrypted messages.

$$M_4 = D_{Prk_X}(C_{Y_1}) \tag{37}$$

Then, user X check whether the message's timestamp is valid or not:

$$\text{Check } (T_8 - T_7) \stackrel{?}{\leq} \tau \tag{38}$$

Then, user Y calculates the hash value ${h_{Y_1}}'$ from the message and invokes the "Verify" function in Algorithm 2 to check the validity of signatures by the following equations.

$${h_{Y_1}}' = H(M_4) \tag{39}$$

$$u_1 = {h_{Y_1}}'{s_{Y_1}}^{-1} \mathrm{mod} n \tag{40}$$

$$u_2 = r_{Y_1}{s_{Y_1}}^{-1} \mathrm{mod} n \tag{41}$$

$$({x_{Y_1}}', {y_{Y_1}}') = u_1 G + u_2 Q_Y \tag{42}$$

$$\text{Check } {x_{Y_1}}' \stackrel{?}{=} r_{Y_1} \text{mod} n \tag{43}$$

The function return is valid if the signatures sent from user Y are legal.

Algorithm 2. The function of authentication with the sign and verify.

```
func Sign (h string, k string, d string) (r string, s string) {
(x, y) = k * G;
r = x % n
s = (h + r * d)/x % n
return r, s
}
func Verify (h string, r string, s string) (result string) {
u1 = h/s % n
u2 = r/s % n
(x, y) = u1 * G + u2 * Q
if x = r {
return "valid"
}else{
return "invalid"
}
}
```

3.5. Alarm Triggered Phase

When an IoT device detects an abnormal occurrence, it must trigger and send information to the LS immediately. The flow of the alarm triggered phase is provided in Figure 7, and the descriptions are as follows.

Step 1. IoT generates a random number k_5, then invokes the chaincode function "Event_Trigger" as shown in Algorithm 3. This will send a transaction and update the information to the BC's ledger. The function also returns an event's ID, and the IoT generates a message and adds with the event's ID and a timestamp.

$$M_5 = (ID_{IoT}||ID_{LS}||ID_{OP}||ID_E||T_9) \tag{44}$$

A hash value h_{IoT_1} is calculated by a hash function.

$$h_{IoT_1} = H(M_5) \tag{45}$$

Then, IoT executes the "Sign" function with parameters in Algorithm 2 to generate a set of ECDSA signatures (r_{IoT_1}, s_{IoT_1}).

$$(r_{IoT_1}, s_{IoT_1}) = Sign(h_{IoT_1}, k_5, d_{IoT}) \tag{46}$$

Next, IoT encrypts the message M_5 into a cipher message C_{IoT_1} with LS's public key. Then, IoT sends cipher messages with signatures $(ID_{IoT}, ID_{LS}, C_{IoT_1}, (r_{IoT_1}, s_{IoT_1}))$ to LS.

$$C_{IoT_1} = E_{Puk_{LS}}(M_5) \tag{47}$$

Step 2. After LS receives the message from IoT, LS decrypts the cipher message with its private key and gets the decrypted messages.

$$M_5 = D_{Prk_{LS}}(C_{IoT_1}) \tag{48}$$

Then, LS checks the interval of the message's timestamp.

$$\text{Check } (T_{10} - T_9) \stackrel{?}{\leq} \tau \tag{49}$$

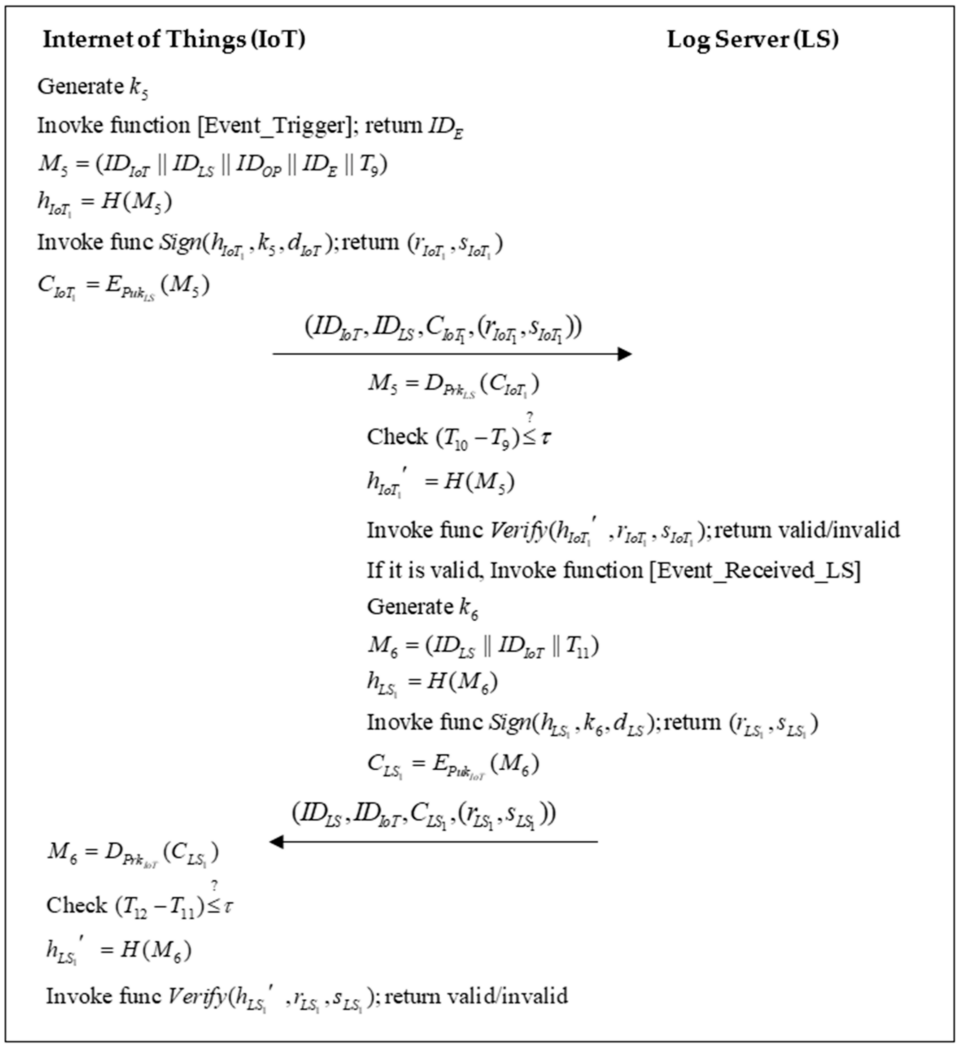

Figure 7. The flowchart of the alarm triggered phase.

Next, LS calculates the hash value $h_{IoT_1}{}'$ from the message M_5 and invokes the "Verify" function in Algorithm 2 to check the validity of signatures.

$$h_{IoT_1}{}' = H(M_5) \tag{50}$$

$$(\text{valid/invalid}) = Verify(h_{IoT_1}{}', r_{IoT_1}, s_{IoT_1}) \tag{51}$$

If the signatures are valid, the LS invokes the chaincode function "Event_Received_LS" as shown in Algorithm 3 which updates the LS's received timestamp and signature of the event to BC. After that, LS generates a random number k_6 and a response message M_6, then transmits the message to IoT later.

$$M_6 = (ID_{LS} || ID_{IoT} || T_{11}) \tag{52}$$

Then, CA calculates signatures with the hash value h_{LS_1}. The signatures are calculated by calling the "Sign" function in Algorithm 2, then the function returns the signatures (r_{LS_1}, s_{LS_1}).

$$h_{LS_1} = H(M_6) \tag{53}$$

$$(r_{LS_1}, s_{LS_1}) = Sign(h_{LS_1}, k_6, d_{LS}) \tag{54}$$

LS encrypts the message to cipher message with IoT's public key and sends the message to IoT.

$$C_{LS_1} = E_{Puk_{IoT}}(M_6) \tag{55}$$

Step 3. IoT receives the response message from LS. Then, IoT decrypts the cipher message with its private key and gets the decrypted messages.

$$M_6 = D_{Prk_{IoT}}(C_{LS_1}) \tag{56}$$

Then, IoT checks whether the message's timestamp is valid or not:

$$\text{Check } (T_{12} - T_{11}) \stackrel{?}{\leq} \tau \tag{57}$$

Next, IoT calculates the hash value ${h_{LS_1}}'$ from the message and invokes the "Verify" function in Algorithm 2 to check the validity of signatures by the following equations.

$${h_{LS_1}}' = H(M_6) \tag{58}$$

$$(\text{valid/invalid}) = Verify({h_{LS_1}}', r_{LS_1}, s_{LS_1}) \tag{59}$$

Algorithm 3. Chaincode function of alarm triggered phase.

```
var EventLog []Event_Information
func Event_Trigger (OPID string, IoTID string, type IoTType, signature string) (EventID string) {
EventID = GenerateUniqueEventID()
EventLog = append (EventLog, new Event_Information{
Event_ID:EventID,
IoT_ID: IoTID,
User_ID: OPID,
IoT_Type: type,
Triggered_Datetime: time.Now(),
IoT_Triggered_Signature: signature
})
return EventID
}
func Event_Received_LS (Event_ID string, signature string) {
index: = SearchEventID(Event_ID)
EventLog[index].LS_Received_Datetime = time.Now()
EventLog[index].LS_Received_Signature = signature
}
```

3.6. Notification Phase

When an IoT device detects an abnormal occurrence, the IoT must alert relevant personnel at the same time. If the IoT is set to the public domain, the notice will be delivered to SG; otherwise, if the IoT is set to the private domain, the notification will be sent to OP. The flow of the notification phase is depicted in Figure 8, the descriptions are as follows.

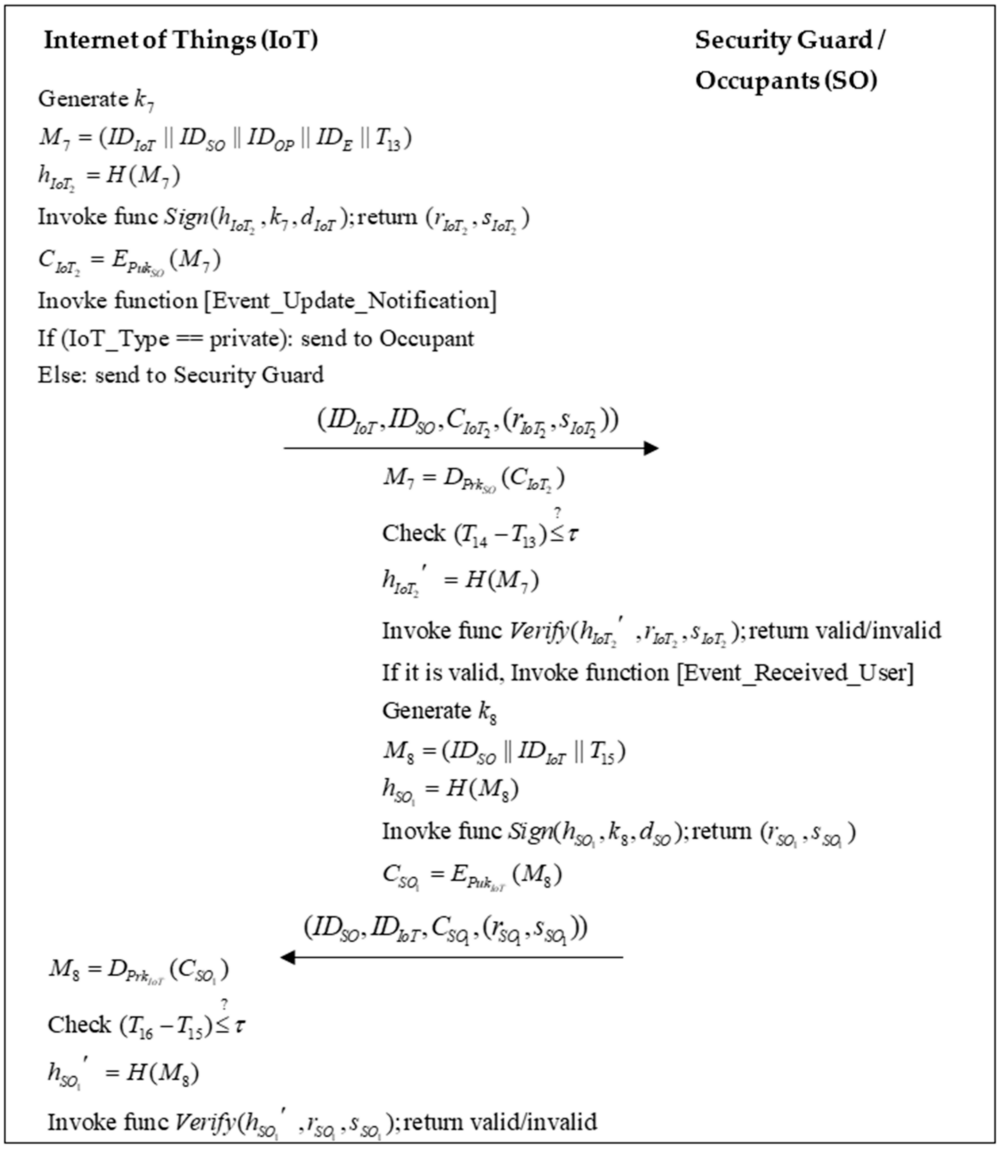

Figure 8. The flowchart of the notification phase.

Step 1. Firstly, IoT generates a random number k_7 and a message with IoT's ID, security guard/occupant's ID, IoT occupant's ID, event's ID, and send timestamp.

$$M_7 = (ID_{IoT}||ID_{SO}||ID_{OP}||ID_E||T_{13}) \tag{60}$$

A hash value h_{IoT_2} is calculated by a hash function.

$$h_{IoT_2} = H(M_7) \tag{61}$$

Then, IoT executes the "Sign" function with parameters in Algorithm 2 to generate a set of signatures (r_{IoT_2}, s_{IoT_2}).

$$(r_{IoT_2}, s_{IoT_2}) = Sign(h_{IoT_2}, k_7, d_{IoT}) \tag{62}$$

Next, IoT encrypts the message M_7 into a cipher message C_{IoT_2}.

$$C_{IoT_2} = E_{Puk_{SO}}(M_7) \tag{63}$$

A chaincode function "Event_Update_Notification" as shown in Algorithm 4 is invoked by IoT to update the IoT notification timestamp and signature of the event to BC. Then, IoT sends cipher messages with signatures $(ID_{IoT}, ID_{SO}, C_{IoT_2}, (r_{IoT_2}, s_{IoT_2}))$ to the security guard or occupant depending on the IoT type. We abbreviate the security guard or occupant to SO.

Step 2. After SO receives the message from IoT, SO decrypts the cipher message with his/her private key and gets the decrypted messages.

$$M_7 = D_{Prk_{SO}}(C_{IoT_2}) \tag{64}$$

Then, SO checks the interval of the message's timestamp.

$$\text{Check } (T_{14} - T_{13}) \stackrel{?}{\leq} \tau \tag{65}$$

Next, LS calculates the hash value ${h_{IoT_2}}'$ from the message M_7 and invokes the "Verify" function in Algorithm 2 to check the validity of signatures.

$${h_{IoT_2}}' = H(M_7) \tag{66}$$

$$(\text{valid/invalid}) = Verify({h_{IoT_2}}', r_{IoT_2}, s_{IoT_2}) \tag{67}$$

If the signatures are valid, the SO invokes the chaincode function "Event_Received_User" as shown in Algorithm 4 which updates the SO's received timestamp and signature to BC. After that, SO generates a random number k_8 and a response message M_8.

$$M_8 = (ID_{SO}||ID_{IoT}||T_{15}) \tag{68}$$

Then, SO calculates signatures with the hash value h_{SO_1}. The signatures are calculated by executing the "Sign" function in Algorithm 2, then the function returns the signatures (r_{SO_1}, s_{SO_1}).

$$h_{SO_1} = H(M_8) \tag{69}$$

$$(r_{SO_1}, s_{SO_1}) = Sign(h_{SO_1}, k_8, d_{SO}) \tag{70}$$

Then, SO encrypts the message to cipher message with IoT's public key, then SO sends the message to IoT.

$$C_{SO_1} = E_{Puk_{IoT}}(M_8) \tag{71}$$

Step 3. IoT receives the response message from SO. Then, IoT decrypts the cipher message with its private key and gets the decrypted messages.

$$M_8 = D_{Prk_{IoT}}(C_{SO_1}) \tag{72}$$

Then, IoT checks whether the message's timestamp is valid or not:

$$\text{Check } (T_{16} - T_{15}) \stackrel{?}{\leq} \tau \tag{73}$$

Next, IoT calculates the hash value ${h_{SO_1}}'$ from the message and invokes the "Verify" function in Algorithm 2 to check the validity of signatures by the following equations.

$${h_{SO_1}}' = H(M_8) \tag{74}$$

$$(\text{valid/invalid}) = Verify({h_{SO_1}}', r_{SO_1}, s_{SO_1}) \tag{75}$$

The real situation must be verified or checked by SO. The response message should be sent to LS in the next phase after being checked or solved.

Algorithm 4. Chaincode function of notification phase.

```
func Event_Update_Notification (Event_ID string) {
index := SearchEventID(Event_ID)
EventLog[index].Notify_Datetime = time.Now()
}

func Event_Received_User (Event_ID string, signature string) {
index := SearchEventID(Event_ID)
EventLog[index].User_Received_Datetime = time.Now()
EventLog[index].User_Received_Signature = signature
}
```

3.7. Response Phase (Security Guard with Public Domain IoT)

The security guard must report the results back to LS once the public IoT alarm has been investigated or solved by the security guard. Figure 9 shows the flow of the response phase with public domain IoT, and the details of the steps are as follows.

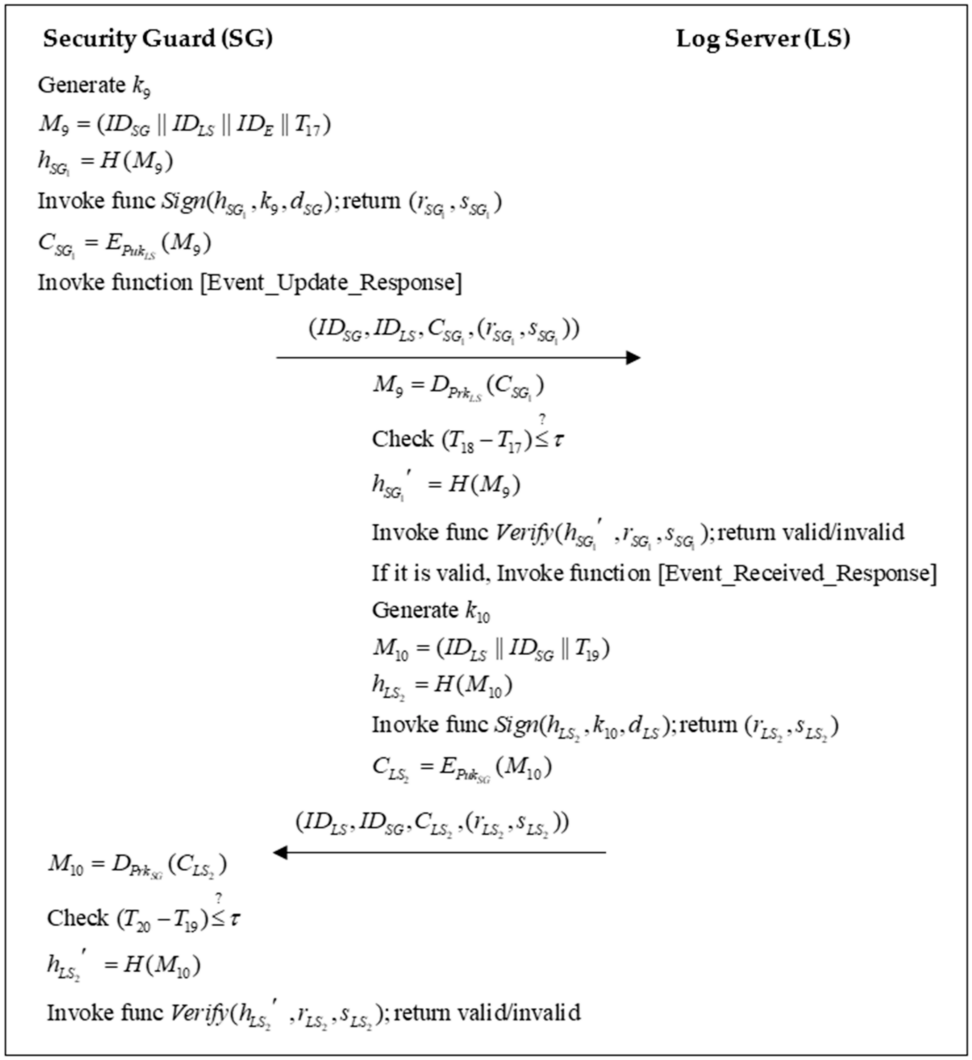

Figure 9. The flowchart of the response phase from the security guard.

Step 1. Initially, SG generates a random number k_9 and a message with the event's ID and a timestamp.

$$M_9 = (ID_{SG}||ID_{LS}||ID_E||T_{17}) \tag{76}$$

A hash value h_{SG_1} is calculated by a hash function.

$$h_{SG_1} = H(M_9) \tag{77}$$

Next, SG executes the "Sign" function with parameters in Algorithm 2 to generate a set of signatures (r_{SG_1}, s_{SG_1}).

$$(r_{SG_1}, s_{SG_1}) = Sign(h_{SG_1}, k_9, d_{SG}) \tag{78}$$

Then, SG encrypts the message M_9 into a cipher message C_{SG_1} with LS's public key.

$$C_{SG_1} = E_{Puk_{LS}}(M_9) \tag{79}$$

SG invokes a chaincode function "Event_Update_Response" in Algorithm 5 to update the SG's response message, timestamp, and signature of the event to BC. Then, SG sends cipher messages with signatures $(ID_{SG}, ID_{LS}, C_{SG_1}, (r_{SG_1}, s_{SG_1}))$ to the LS.

Step 2. LS receives the message from SG and decrypts the cipher message with its private key and gets the decrypted messages.

$$M_9 = D_{Prk_{LS}}(C_{SG_1}) \tag{80}$$

Then, LS checks the interval of the message's timestamp.

$$\text{Check } (T_{18} - T_{17}) \stackrel{?}{\leq} \tau \tag{81}$$

Next, LS calculates the hash value ${h_{SG_1}}'$ from the message M_9 and invokes the "Verify" function in Algorithm 2 to check the validity of signatures.

$${h_{SG_1}}' = H(M_9) \tag{82}$$

$$(\text{valid/invalid}) = Verify({h_{SG_1}}', r_{SG_1}, s_{SG_1}) \tag{83}$$

If the signatures are valid, the LS invokes the chaincode function "Event_Received_Response" in Algorithm 5 to update the LS's received timestamp and signature to BC. Afterward, LS generates a random number k_{10} and a response message M_{10}.

$$M_{10} = (ID_{LS}||ID_{SG}||T_{19}) \tag{84}$$

LS calculates signatures with the hash value h_{LS_2}. The signatures (r_{LS_2}, s_{LS_2}) are calculated by executing the "Sign" function in Algorithm 2.

$$h_{LS_2} = H(M_{10}) \tag{85}$$

$$(r_{LS_2}, s_{LS_2}) = Sign(h_{LS_2}, k_{10}, d_{LS}) \tag{86}$$

Then, LS encrypts the message to cipher message with SG's public key, then sends the message to SG.

$$C_{LS_2} = E_{Puk_{SG}}(M_{10}) \tag{87}$$

Step 3. SG receives the message from LS. Then, SG decrypts the cipher message with his/her private key and gets the decrypted messages.

$$M_{10} = D_{Prk_{SG}}(C_{LS_2}) \tag{88}$$

After that, SG checks whether the message's timestamp is valid or not.

$$\text{Check } (T_{20} - T_{19}) \stackrel{?}{\leq} \tau \tag{89}$$

Next, SG calculates the hash value $h_{LS_2}{}'$ from the message and invokes the "Verify" function in Algorithm 2 to check the validity of signatures.

$$h_{LS_2}{}' = H(M_{10}) \tag{90}$$

$$(\text{valid/invalid}) = Verify(h_{LS_2}{}', r_{LS_2}, s_{LS_2}) \tag{91}$$

Algorithm 5. Chaincode function of response phase from security guard.

```
func Event_Update_Response (Event_ID string, signature string, message string) {
index: = SearchEventID(Event_ID)
EventLog[index].User_Response_Datetime = time.Now()
EventLog[index].User_Response_Signature = signature
EventLog[index].ResponseMessage = message
}

func Event_Received_Response (Event_ID string, signature string) {
index: = SearchEventID(Event_ID)
EventLog[index].LS_Received_Datetime = time.Now()
EventLog[index].LS_Received_Signature = signature
}
```

3.8. Response Phase (Occupant with Private Domain IoT)

If the IoT is in the private domain, the occupant has the alternative of resolving it himself/herself or authorizing the security guard to solve the alarm situation. Figure 10 shows the flow of the response phase with private domain IoT. The details of the steps are as follows.

Step 1. Firstly, OP generates a random number k_{11} and a message with the event's ID, a timestamp, and the option of self or permitting guard to solve $< Self/Guard >$.

$$M_{11} = (ID_{OP}||ID_{LS}||ID_E||T_{21}||< Self/Guard >) \tag{92}$$

A hash value h_{OP_1} is calculated by a hash function.

$$h_{OP_1} = H(M_{11}) \tag{93}$$

Next, OP executes the "Sign" function with parameters in Algorithm 2 to generate a set of signatures (r_{OP_1}, s_{OP_1}).

$$(r_{OP_1}, s_{OP_1}) = Sign(h_{OP_1}, k_{11}, d_{OP}) \tag{94}$$

Then, OP encrypts the message M_{11} into a cipher message C_{OP_1} with LS's public key.

$$C_{OP_1} = E_{Puk_{LS}}(M_{11}) \tag{95}$$

OP invokes a chaincode function "Event_Update_Response" in Algorithm 6. The function updates the OP's response message, timestamp, and signature to BC. Then, OP sends cipher messages with signatures $(ID_{OP}, ID_{LS}, C_{OP_1}, (r_{OP_1}, s_{OP_1}))$ to the LS.

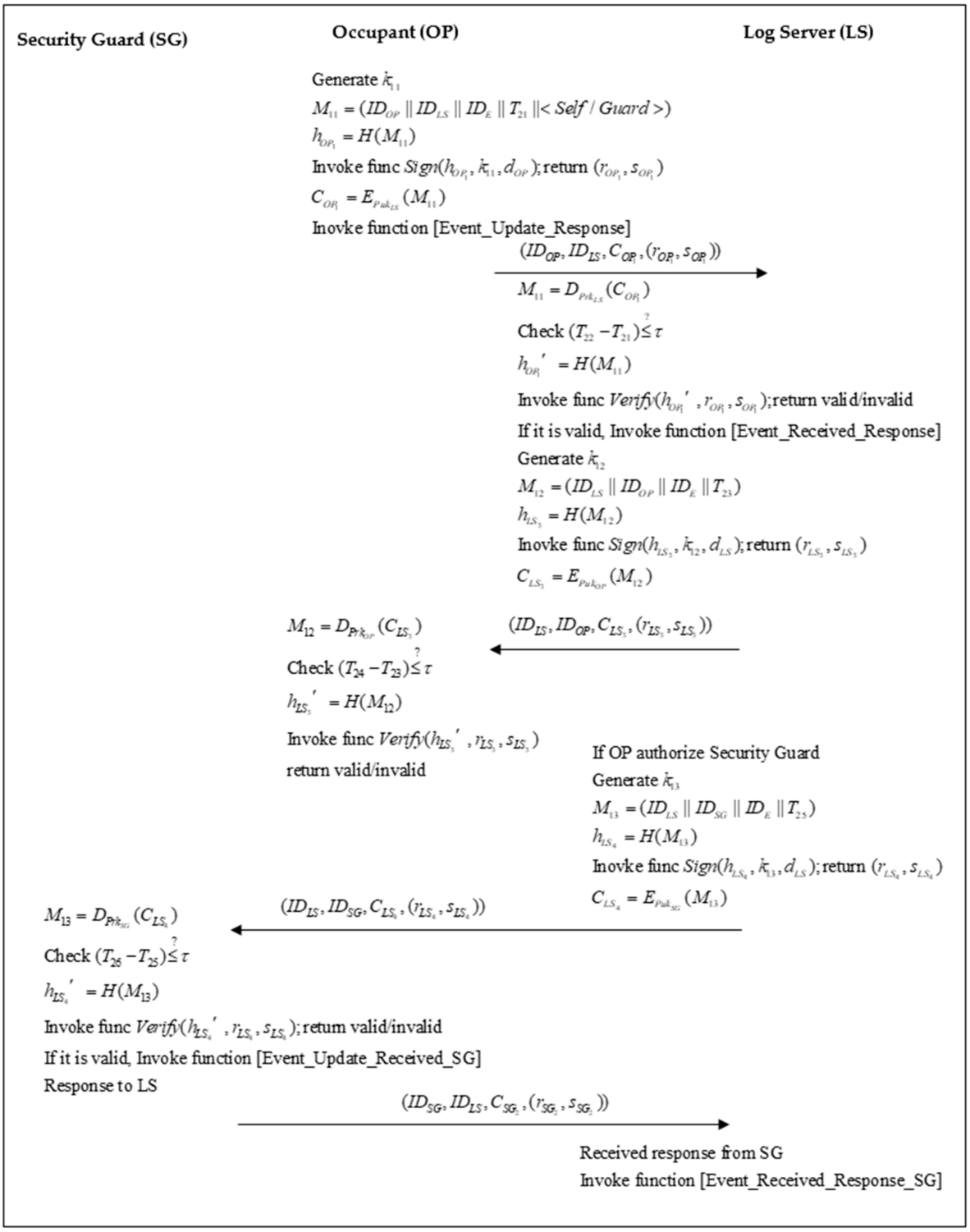

Figure 10. The flowchart of the response phase from an occupant.

Step 2. LS receives the message from OP. LS decrypts the cipher message with its private key and gets the decrypted messages.

$$M_{11} = D_{Prk_{LS}}(C_{OP_1}) \tag{96}$$

Then, LS checks the interval of the message's timestamp.

$$\text{Check } (T_{22} - T_{21}) \stackrel{?}{\leq} \tau \tag{97}$$

Next, LS calculates the hash value ${h_{OP_1}}'$ from the message M_{11} and invokes the "Verify" function in Algorithm 2 to check the validity of signatures.

$${h_{OP_1}}' = H(M_{11}) \tag{98}$$

$$(\text{valid/invalid}) = Verify({h_{OP_1}}', r_{OP_1}, s_{OP_1}) \tag{99}$$

If the signatures are valid, the LS invokes the chaincode function "Event_Received_ Response" in Algorithm 6 to update the LS's received timestamp and signature to BC. Afterward, LS generates a random number k_{12} and a response message M_{12}.

$$M_{12} = (ID_{LS}||ID_{OP}||ID_E||T_{23}) \tag{100}$$

LS calculates signatures with the hash value h_{LS_3}. The signatures (r_{LS_3}, s_{LS_3}) are calculated by executing the "Sign" function in Algorithm 2.

$$h_{LS_3} = H(M_{12}) \tag{101}$$

$$(r_{LS_3}, s_{LS_3}) = Sign(h_{LS_3}, k_{12}, d_{LS}) \tag{102}$$

Then, LS encrypts the message to cipher message with OP's public key, then sends the message $(ID_{LS}, ID_{OP}, C_{LS_3}, (r_{LS_3}, s_{LS_3}))$ to OP.

$$C_{LS_3} = E_{Puk_{OP}}(M_{12}) \tag{103}$$

Step 3. OP receives the message from LS. Then, OP decrypts the cipher message with his/her private key.

$$M_{12} = D_{Prk_{OP}}(C_{LS_3}) \tag{104}$$

After that, OP check whether the message's timestamp is valid or not.

$$\text{Check } (T_{24} - T_{23}) \stackrel{?}{\leq} \tau \tag{105}$$

Next, OP calculates the hash value ${h_{LS_3}}'$ from the message and invokes the "Verify" function in Algorithm 2 to check the validity of signatures.

$${h_{LS_3}}' = H(M_{12}) \tag{106}$$

$$(\text{valid/invalid}) = Verify({h_{LS_3}}', r_{LS_3}, s_{LS_3}) \tag{107}$$

Step 1. If the OP permits the security guard in the community to solve the situation, then LS will send the notification to the security guard as provided in the following step. LS generates a random number k_{13} and a response message M_{13}.

$$M_{13} = (ID_{LS}||ID_{SG}||ID_E||T_{25}) \tag{108}$$

LS calculates signatures with the hash value h_{LS_4}. The signatures (r_{LS_4}, s_{LS_4}) are calculated by executing the "Sign" function in Algorithm 2.

$$h_{LS_4} = H(M_{13}) \tag{109}$$

$$(r_{LS_4}, s_{LS_4}) = Sign(h_{LS_4}, k_{13}, d_{LS}) \tag{110}$$

Then, LS encrypts the message to cipher message with SG's public key, then sends the message $(ID_{LS}, ID_{SG}, C_{LS_4}, (r_{LS_4}, s_{LS_4}))$ to SG.

$$C_{LS_4} = E_{Puk_{SG}}(M_{13}) \tag{111}$$

Step 2. SG received the message from LS. Then, SG decrypts the cipher message with his/her private key.

$$M_{13} = D_{Prk_{SG}}(C_{LS_4}) \tag{112}$$

After that, SG checks the validity of the message's timestamp.

$$\text{Check } (T_{26} - T_{25}) \stackrel{?}{\leq} \tau \tag{113}$$

Next, SG calculates the hash value ${h_{LS_4}}'$ from the message and invokes the "Verify" function in Algorithm 2 to check the validity of signatures.

$${h_{LS_4}}' = H(M_{13}) \tag{114}$$

$$(\text{valid/invalid}) = Verify({h_{LS_4}}', r_{LS_4}, s_{LS_4}) \tag{115}$$

SG invokes a chaincode function "Event_Update_Received_SG" in Algorithm 6. The function updates the SG response message, timestamp, and signature of the event to BC. Afterward, SG generates a random number k_{14} and a response message M_{14}.

$$M_{14} = (ID_{SG}||ID_{LS}||ID_E||T_{27}) \tag{116}$$

SG calculates signatures with the hash value h_{SG_2}. The signatures (r_{SG_2}, s_{SG_2}) are calculated by executing the "Sign" function in Algorithm 2.

$$h_{SG_2} = H(M_{14}) \tag{117}$$

$$(r_{SG_2}, s_{SG_2}) = Sign(h_{SG_2}, k_{14}, d_{SG}) \tag{118}$$

Then, SG encrypts the message to cipher message with LS's public key, then sends the cipher message to SG.

$$C_{SG_2} = E_{Puk_{LS}}(M_{14}) \tag{119}$$

Then, SG sends cipher messages with signatures $(ID_{SG}, ID_{LS}, C_{SG_2}, (r_{SG_2}, s_{SG_2}))$ to LS.

Step 3. LS receives the message from SG, LS decrypts the cipher message with its private key and gets the decrypted messages.

$$M_{14} = D_{Prk_{LS}}(C_{SG_2}) \tag{120}$$

Then, LS checks the interval of the message's timestamp.

$$\text{Check } (T_{28} - T_{27}) \stackrel{?}{\leq} \tau \tag{121}$$

Next, LS calculates the hash value ${h_{SG_2}}'$ from the message M_{14} and invokes the "Verify" function in Algorithm 2 to check the validity of signatures.

$${h_{SG_2}}' = H(M_{14}) \tag{122}$$

$$(\text{valid/invalid}) = Verify({h_{SG_2}}', r_{SG_2}, s_{SG_2}) \tag{123}$$

If the signatures are valid, the LS invokes the chaincode function "Event_Received_Response_SG" in Algorithm 6 to update the LS's received timestamp and signature to BC.

Algorithm 6. Chaincode function of response phase from an occupant.

```
func Event_Update_Response (Event_ID string, signature string, message string) {
index: = SearchEventID(Event_ID)
EventLog[index].User_Response_Datetime = time.Now()
EventLog[index].User_Response_Signature = signature
EventLog[index].ResponseMessage = message
}

func Event_Received_Response (Event_ID string, signature string) {
index: = SearchEventID(Event_ID)
EventLog[index].LS_Received_Datetime = time.Now()
EventLog[index].LS_Received_Signature = signature
}

func Event_Update_Recieved_SG (Event_ID string, signature string) {
index: = SearchEventID(Event_ID)
EventLog[index].SG_Received_Datetime = time.Now()
EventLog[index].SG_Received_Signature = signature
}

func Event_Received_Response_SG (Event_ID string, signature string) {
index: = SearchEventID(Event_ID)
EventLog[index].SG_Response_Datetime = time.Now()
EventLog[index].SG_Response_Signature = signature
}
```

3.9. Check for History Records Phase

In this stage, the system is designed in two ways to obtain history videos or records. It is divided into the private and public domain. Figure 11 shows how OP viewing for the public records through SG. The detailed step is as follows:

Step 1. OP requests for viewing the private domain IoT devices videos or records from SG face to face.

Step 2. The SG sends his/her encryption key and request information (such as start date and time, end date and time, and name or ID of private domain IoT device(s)) to LS. Then, the LS sends that information to BC for verification.

Step 3. BC notifies the related occupant's mobile device and asks for permission to let SG and OP view the related records.

Step 4. The related occupant replies to the permission request back to BC by mobile device.

Step 5. If the related occupant accepts the request, then BC shows the history videos or records in the security system. Otherwise, BC responds to the SG and OP that the request is not permitted.

Furthermore, the SG can also request for viewing the private domain IoT devices videos or records himself/herself without OP; the authorization of the relevant occupant is also required. On the other hand, SG needs SP to authorize to view the public records, as shown in Figure 12. The major steps of obtaining authorization are the same, the difference is only the step of notifying and asking permission from SG's supervisor SP, who is not related to OP. The detailed step is described as follows:

Step 1. SG requests for viewing the public domain IoT devices' videos or records. The SG sends his/her encryption key and request information (such as start date and time, end date and time, and name or ID of private domain IoT device(s)) to LS.

Step 2. Then, the LS sends that information to BC for verification.

Step 3. BC notifies SG's supervisor (SP) and asks for permission to let SG view the related records.

Step 4. The SP replies to the permission request back to BC by mobile device.

Step 5. If the related occupant accepts the request, then BC shows the history videos or records in the security system to SG. Otherwise, BC responds to the SG that request is not permitted.

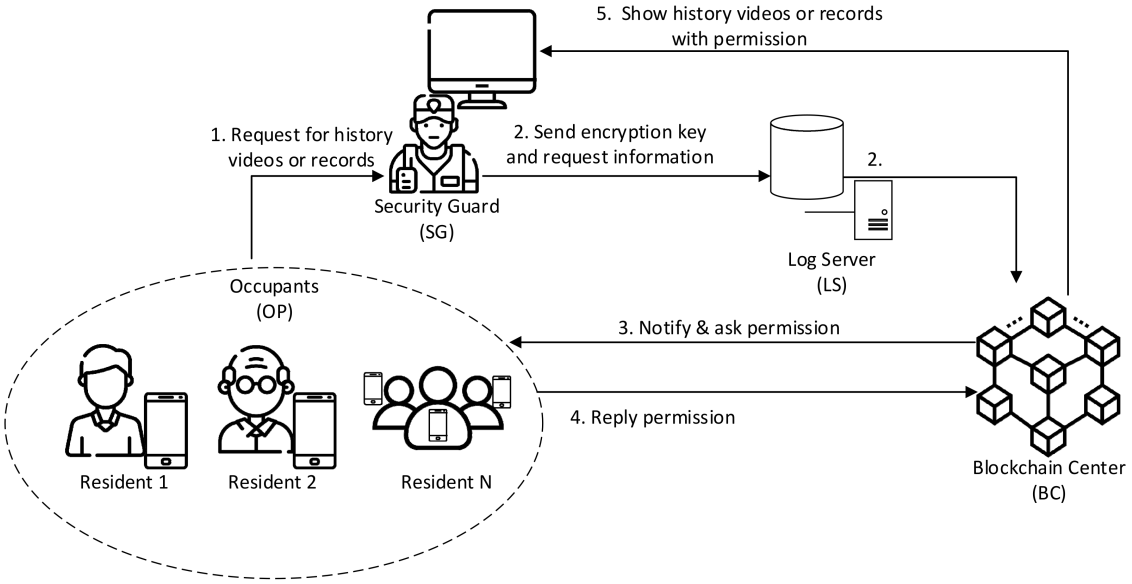

Figure 11. The phase of obtaining authorization when viewing the private history videos or records.

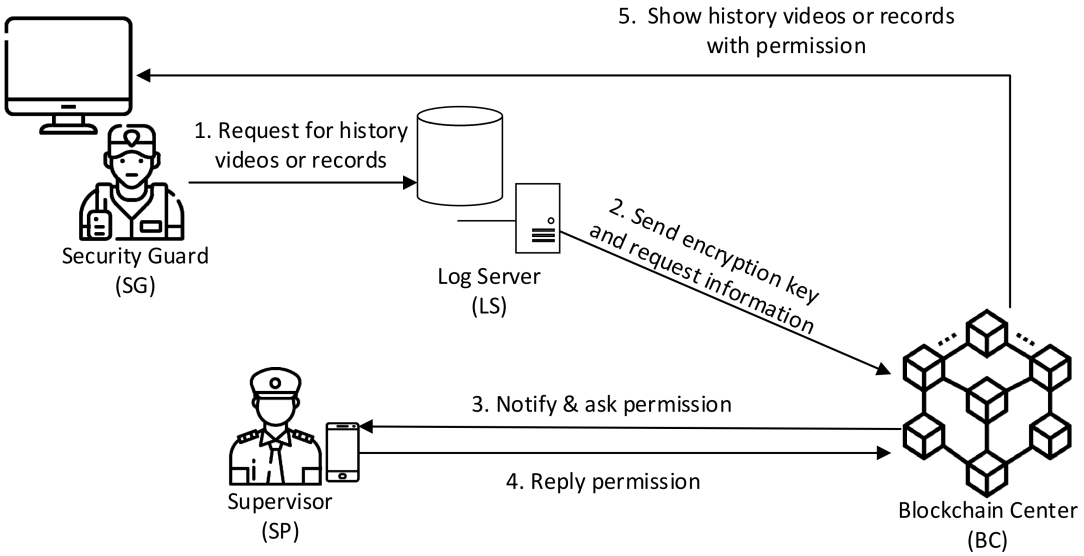

Figure 12. The phase of obtaining authorization when viewing the public history videos or records.

3.10. Clarification Phase

When any party's participating role determines that there is a problem with the secure record, it can request a third-party impartial unit or person to verify it. Figure 13 shows the flow of clarifying the information of safety security system logs. The explanation is as follows:

Step 1. The participant (such as security guard or occupant, SO) sends a clarifying request with the specified event ID and signatures to a third party (TP).
Step 2. TP sends the request message and his/her signature to BC.
Step 3. The signatures are checked by the BC, then the event's information are sent to TP.
Step 4. The TP checks the validity of every signature in the event's information.

a. Check if the event is triggered from a public domain IoT: go to step 4b if it is "no", else go to step 4d.
b. If the SG response signature is not valid, then the information is forged by LS.
c. If the SG received signature is not valid, then the information is forged by SG.
d. If the LS response signature is not valid, then the information is forged by LS.
e. If the SO response signature is not valid, then the information is forged by SO.
f. If the SO received signature is not valid, then the information is forged by SO.
g. If the LS received signature is not valid, then the information is forged by LS.
h. If the IoT triggered signature is not valid, then the information is forged by IoT.
i. The specified event information is valid if all the signatures are legal.

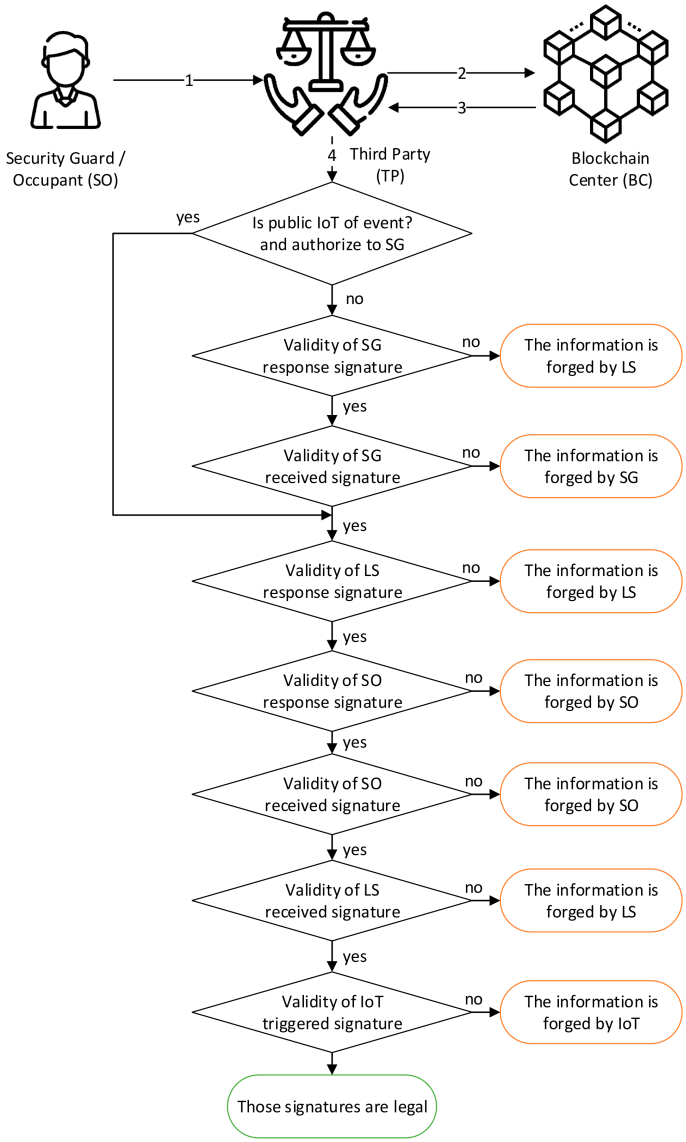

Figure 13. The flow of clarifying information and signature.

4. Security Analysis

In this research, we have done some important security analyses to prevent the system's vulnerabilities or attacks from the proposed scheme. The analyses are explained in the following subsections, such as the data integrity, non-repudiation of the message, unforgeable data, traceability of records, man-in-the-middle attack, and replay attack.

4.1. Data Integrity

Firstly, we analyzed the integrity of the message in this subsection. The hash function with the ECDSA algorithm is used to ensure the integrity of the message. Every sender must calculate a hash value from the message and generate a set of signatures, and the receiver needs to verify the validation of the message in hash value and signatures by the "Verify" function in Algorithm 2.

For example, in the phase of registration of IoT, the sender X sends the message M_1 to the receiver CA. The sender needs to generate h_{X_1} and calculate (r_{X_1}, s_{X_1}) with the "Sign" function in Algorithm 2. Next sender sends M_1 in cipher message with a signature (r_{X_1}, s_{X_1}) to the receiver, then the receiver decrypts and generates hash value ${h_{X_1}}'$ by the message M_1. Lastly, the hash value ${h_{X_1}}'$ with the signature (r_{X_1}, s_{X_1}) is verified in the "Verify" function of Algorithm 2. All detailed messages in every phase are listed in Table 2 respectively.

Table 2. Verification of data integrity of the proposed scheme.

Phase	Party		Message	Hash Value	Verification
	Sender	Receiver			
Registration of IoT	X	CA	$M_1 = (ID_X \| \| ID_{CA} \| \| < IoT\_Information > \| \| T_1)$	$h_{X_1} = H(M_1)$	$Verify({h_{X_1}}', r_{X_1}, s_{X_1})$
	CA	X	$M_2 = (ID_{CA} \| \| ID_X \| \| ID_{IoT} \| \| d_{IoT} \| \| Q_{IoT} \| \| T_3)$	$h_{CA_1} = H(M_2)$	$Verify({h_{CA_1}}', r_{CA_1}, s_{CA_1})$
Authentication	X	Y	$M_3 = (ID_X \| \| ID_Y \| \| T_5)$	$h_{X_2} = H(M_3)$	$Verify({h_{X_1}}', r_{X_1}, s_{X_1})$
	Y	X	$M_4 = (ID_Y \| \| ID_X \| \| T_7)$	$h_{Y_1} = H(M_4)$	$Verify({h_{Y_1}}', r_{Y_1}, s_{Y_1})$
Alarm triggered	IoT	LS	$M_5 = (ID_{IoT} \| \| ID_{LS} \| \| ID_{OP} \| \| ID_E \| \| T_9)$	$h_{IoT_1} = H(M_5)$	$Verify({h_{IoT_1}}', r_{IoT_1}, s_{IoT_1})$
	LS	IoT	$M_6 = (ID_{LS} \| \| ID_{IoT} \| \| T_{11})$	$h_{LS_1} = H(M_6)$	$Verify({h_{LS_1}}', r_{LS_1}, s_{LS_1})$
Notification	IoT	SG/OP	$M_7 = (ID_{IoT} \| \| ID_{SO} \| \| ID_{OP} \| \| ID_E \| \| T_{13})$	$h_{IoT_2} = H(M_7)$	$Verify({h_{IoT_2}}', r_{IoT_2}, s_{IoT_2})$
	SG/OP	IoT	$M_8 = (ID_{SO} \| \| ID_{IoT} \| \| T_{15})$	$h_{SO_1} = H(M_8)$	$Verify({h_{SO_1}}', r_{SO_1}, s_{SO_1})$
Response (Public IoT)	SG	LS	$M_9 = (ID_{SG} \| \| ID_{LS} \| \| ID_E \| \| T_{17})$	$h_{SG_1} = H(M_9)$	$Verify({h_{SG_1}}', r_{SG_1}, s_{SG_1})$
	LS	SG	$M_{10} = (ID_{LS} \| \| ID_{SG} \| \| T_{19})$	$h_{LS_2} = H(M_{10})$	$Verify({h_{LS_2}}', r_{LS_2}, s_{LS_2})$
Response (Private IoT)	OP	LS	$M_{11} = (ID_{OP} \| \| ID_{LS} \| \| ID_E \| \| T_{21})$	$h_{OP_1} = H(M_{11})$	$Verify({h_{OP_1}}', r_{OP_1}, s_{OP_1})$
	LS	OP	$M_{12} = (ID_{LS} \| \| ID_{OP} \| \| ID_E \| \| T_{23})$	$h_{LS_3} = H(M_{12})$	$Verify({h_{LS_3}}', r_{LS_3}, s_{LS_3})$
	LS	SG	$M_{13} = (ID_{LS} \| \| ID_{SG} \| \| ID_E \| \| T_{25})$	$h_{LS_4} = H(M_{13})$	$Verify({h_{LS_4}}', r_{LS_4}, s_{LS_4})$
	SG	LS	$M_{14} = (ID_{SG} \| \| ID_{LS} \| \| ID_E \| \| T_{27})$	$h_{SG_2} = H(M_{14})$	$Verify({h_{SG_2}}', r_{SG_2}, s_{SG_2})$

4.2. Non-Repudiation

Furthermore, we also analyzed the non-repudiation of the message in this subsection. The sign function with the ECDSA algorithm is used to ensure the signature is from the sender. Every sender must generate the signature from the message, and the receiver needs to verify the signatures by the "Verify" function in Algorithm 2.

For example in the phase of registration of IoT, the sender X generates (r_{X_1}, s_{X_1}) with the ECDSA "Sign" function in Algorithm 2 by multiple parameters, such as hash value h_{X_1}, random number k_1, and its ECDSA parameter d_X. Then, the receiver generates a hash value ${h_{X_1}}'$ by the received message. The hash value ${h_{X_1}}'$ and signature (r_{X_1}, s_{X_1}) are verified in the ECDSA "Verify" function in Algorithm 2. The signature is signed by the sender if the verification return is valid. All the signatures and verification in every phase are listed in Table 3.

Table 3. Verify non-repudiation of the proposed scheme.

Phase	Party		Signature	Verification
	Sender	Receiver		
Registration of IoT	X	CA	$(r_{X_1}, s_{X_1}) = Sign(h_{X_1}, k_1, d_X)$	$Verify(h_{X_1}', r_{X_1}, s_{X_1})$
	CA	X	$(r_{CA_1}, s_{CA_1}) = Sign(h_{CA_1}, k_2, d_{CA})$	$Verify(h_{CA_1}', r_{CA_1}, s_{CA_1})$
Authentication	X	Y	$(r_{X_2}, s_{X_2}) = Sign(h_{X_2}, k_3, d_X)$	$Verify(h_{X_1}', r_{X_1}, s_{X_1})$
	Y	X	$(r_{Y_1}, s_{Y_1}) = Sign(h_{Y_1}, k_4, d_Y)$	$Verify(h_{Y_1}', r_{Y_1}, s_{Y_1})$
Alarm triggered	IoT	LS	$(r_{IoT_1}, s_{IoT_1}) = Sign(h_{IoT_1}, k_5, d_{IoT})$	$Verify(h_{IoT_1}', r_{IoT_1}, s_{IoT_1})$
	LS	IoT	$(r_{LS_1}, s_{LS_1}) = Sign(h_{LS_1}, k_6, d_{LS})$	$Verify(h_{LS_1}', r_{LS_1}, s_{LS_1})$
Notification	IoT	SG/OP	$(r_{IoT_2}, s_{IoT_2}) = Sign(h_{IoT_2}, k_7, d_{IoT})$	$Verify(h_{IoT_2}', r_{IoT_2}, s_{IoT_2})$
	SG/OP	IoT	$(r_{SO_1}, s_{SO_1}) = Sign(h_{SO_1}, k_8, d_{SO})$	$Verify(h_{SO_1}', r_{SO_1}, s_{SO_1})$
Response (Public IoT)	SG	LS	$(r_{SG_1}, s_{SG_1}) = Sign(h_{SG_1}, k_9, d_{SG})$	$Verify(h_{SG_1}', r_{SG_1}, s_{SG_1})$
	LS	SG	$(r_{LS_2}, s_{LS_2}) = Sign(h_{LS_2}, k_{10}, d_{LS})$	$Verify(h_{LS_2}', r_{LS_2}, s_{LS_2})$
Response (Private IoT)	OP	LS	$(r_{OP_1}, s_{OP_1}) = Sign(h_{OP_1}, k_{11}, d_{OP})$	$Verify(h_{OP_1}', r_{OP_1}, s_{OP_1})$
	LS	OP	$(r_{LS_3}, s_{LS_3}) = Sign(h_{LS_3}, k_{12}, d_{LS})$	$Verify(h_{LS_3}', r_{LS_3}, s_{LS_3})$
	LS	SG	$(r_{LS_4}, s_{LS_4}) = Sign(h_{LS_4}, k_{13}, d_{LS})$	$Verify(h_{LS_4}', r_{LS_4}, s_{LS_4})$
	SG	LS	$(r_{SG_2}, s_{SG_2}) = Sign(h_{SG_2}, k_{14}, d_{SG})$	$Verify(h_{SG_2}', r_{SG_2}, s_{SG_2})$

All the important signatures and received timestamps will be sent via chaincode in the alarm triggered phase, notification phase, and response phase. When there is a dispute that has to be resolved, Section 3.10 offers a clarification phase to determine whether there are any signature issues.

4.3. Unforgeable Data and Traceability

In the proposed system we applied HyperledgerFabric-based blockchain technology in the proposed scheme. It is hard to forge any data stored in the BC compared to the traditional database storing scheme. In every phase we designed in the system, every participant must invoke the chaincode function and update the related information to BC, especially updating the signature and timestamp when processing the IoT's triggered alarm.

Figure 14 shows how the data are updated in the BC. When a participant invokes a chaincode function, the chaincode mechanism will be proposed to every peer (the peer could be more than 1, we demonstrated 3 peers in the scheme) in the blockchain center. Every peer sign and response after the transaction is proofed. After that, the ledger will be updated to every peer via an ordering peer (OP) after sending those endorsed signs and the transaction.

Because of the chaincode mechanism, it is impossible to forge data in the blockchain center; every transaction and ledger are duplicated in all the peers in BC, and the only way to update the data in the ledger is from pre-designed chaincode.

According to the characteristics of the blockchain, every transaction record will be chained and stored in the ledger of every peer. Therefore, the records can be traced in the ledger, and they are also unforgeable. In addition, we designed a clarification phase in Section 3.10. The section can help participants to clarify the record in the blockchain with the third party. If any signatures cannot be verified in the phase of clarifying, it means some signatures are being forged during executing chaincode in other phases.

4.4. Man-in-the-Middle Attack

We implemented asymmetric encryption and decryption in every communication message to defend from the man-in-the-middle attack. Every message that sends from

the sender must encrypt with the receiver's public key. After the receiver received the encrypted cipher message, the receiver uses its private key to decrypt the message. Table 4 shows all the asymmetric encryption and decryption in every phase.

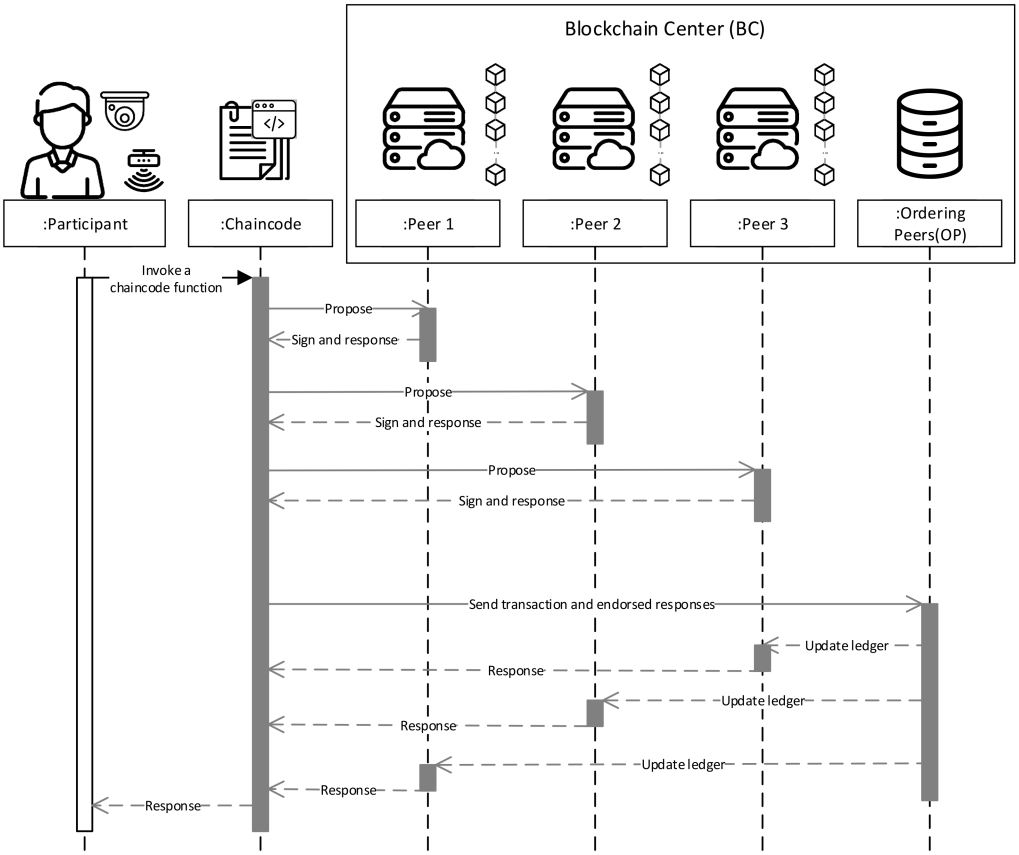

Figure 14. Update data to Blockchain Center.

Table 4. Encryption and decryption to prevent man-in-the-middle attack.

Phase	Party		Encryption	Decryption
	Sender	Receiver		
Registration of IoT	X	CA	$C_{X_1} = E_{Puk_{CA}}(M_1)$	$M_1 = D_{Prk_{CA}}(C_{X_1})$
	CA	X	$C_{CA_1} = E_{Puk_X}(M_2)$	$M_2 = D_{Prk_X}(C_{CA_1})$
Authentication	X	Y	$C_{X_2} = E_{Puk_Y}(M_3)$	$M_3 = D_{Prk_Y}(C_{X_2})$
	Y	X	$C_{Y_1} = E_{Puk_X}(M_4)$	$M_4 = D_{Prk_X}(C_{Y_1})$
Alarm triggered	IoT	LS	$C_{IoT_1} = E_{Puk_{LS}}(M_5)$	$M_5 = D_{Prk_{LS}}(C_{IoT_1})$
	LS	IoT	$C_{LS_1} = E_{Puk_{IoT}}(M_6)$	$M_6 = D_{Prk_{IoT}}(C_{LS_1})$
Notification	IoT	SG/OP	$C_{IoT_2} = E_{Puk_{SO}}(M_7)$	$M_7 = D_{Prk_{SO}}(C_{IoT_2})$
	SG/OP	IoT	$C_{SO_1} = E_{Puk_{IoT}}(M_8)$	$M_8 = D_{Prk_{IoT}}(C_{SO_1})$
Response (Public IoT)	SG	LS	$C_{SG_1} = E_{Puk_{LS}}(M_9)$	$M_9 = D_{Prk_{LS}}(C_{SG_1})$
	LS	SG	$C_{LS_2} = E_{Puk_{SG}}(M_{10})$	$M_{10} = D_{Prk_{SG}}(C_{LS_2})$
Response (Private IoT)	OP	LS	$C_{OP_1} = E_{Puk_{LS}}(M_{11})$	$M_{11} = D_{Prk_{LS}}(C_{OP_1})$
	LS	OP	$C_{LS_3} = E_{Puk_{OP}}(M_{12})$	$M_{12} = D_{Prk_{OP}}(C_{LS_3})$
	LS	SG	$C_{SG_2} = E_{Puk_{LS}}(M_{13})$	$M_{13} = D_{Prk_{LS}}(C_{SG_2})$
	SG	LS	$C_{LS_4} = E_{Puk_{SG}}(M_{14})$	$M_{14} = D_{Prk_{SG}}(C_{LS_4})$

For example, in the phase of registration of IoT, X sends a message M_1 encrypted by CA's public key $E_{Puk_{CA}}(M_1)$, then generates a cipher message C_{X_1}. Next, X sends the cipher message C_{X_1} to CA, and CA decrypts the cipher message C_{X_1} by his/her private key $D_{Prk_{CA}}(C_{X_1}) = M_1$ to get the original message M_1. Consequently, the attacker is not able to decrypt the message without having a private key of the receiver.

4.5. Replay Attack

To prevent the replay attack, we added a timestamp in every message sent from the sender. The receiver needs to calculate the difference of the timestamp when receiving the message. If the interval of two timestamps is over a threshold value, it means the message is being replayed. Table 5 shows all timestamp validation in every phase.

Table 5. Timestamp validation to prevent man-in-the-middle attack.

Phase	Party		Send Time	Receive Time	Validation
	Sender	Receiver			
Registration of IoT	X	CA	T_1	T_2	Check $(T_2 - T_1) \overset{?}{\leq} \tau$
	CA	X	T_3	T_4	Check $(T_4 - T_3) \overset{?}{\leq} \tau$
Authentication	X	Y	T_5	T_6	Check $(T_6 - T_5) \overset{?}{\leq} \tau$
	Y	X	T_7	T_8	Check $(T_8 - T_7) \overset{?}{\leq} \tau$
Alarm triggered	IoT	LS	T_9	T_{10}	Check $(T_{10} - T_9) \overset{?}{\leq} \tau$
	LS	IoT	T_{11}	T_{12}	Check $(T_{12} - T_{11}) \overset{?}{\leq} \tau$
Notification	IoT	SG/OP	T_{13}	T_{14}	Check $(T_{14} - T_{13}) \overset{?}{\leq} \tau$
	SG/OP	IoT	T_{15}	T_{16}	Check $(T_{16} - T_{15}) \overset{?}{\leq} \tau$
Response (Public IoT)	SG	LS	T_{17}	T_{18}	Check $(T_{18} - T_{17}) \overset{?}{\leq} \tau$
	LS	SG	T_{19}	T_{20}	Check $(T_{20} - T_{19}) \overset{?}{\leq} \tau$
Response (Private IoT)	OP	LS	T_{21}	T_{22}	Check $(T_{22} - T_{21}) \overset{?}{\leq} \tau$
	LS	OP	T_{23}	T_{24}	Check $(T_{24} - T_{23}) \overset{?}{\leq} \tau$
	LS	SG	T_{25}	T_{26}	Check $(T_{26} - T_{25}) \overset{?}{\leq} \tau$
	SG	LS	T_{27}	T_{28}	Check $(T_{28} - T_{27}) \overset{?}{\leq} \tau$

5. Discussion

5.1. Computation Cost

In this subsection, we calculated the computation costs as shown in Table 6. We consider all computational costs in all phases of communication protocol. The costs included hash operation, additional operation, subtraction operation, multiplication operation, division operation, and asymmetrical encryption/decryption.

Table 6. Computation costs of the proposed scheme.

Phase	Participant 1	Participant 2
Registration of IoT	User X: $2T_h + 2T_{add} + 1T_{sub} + 4T_{mul} + 3T_{div} + 2T_{asy}$	Certificate Authority: $2T_h + 2T_{add} + 1T_{sub} + 3T_{mul} + 3T_{div} + 2T_{asy}$
Authentication	User X: $2T_h + 2T_{add} + 1T_{sub} + 4T_{mul} + 3T_{div} + 2T_{asy}$	User Y: $2T_h + 2T_{add} + 1T_{sub} + 3T_{mul} + 3T_{div} + 2T_{asy}$
Alarm triggered	Internet of Things: $2T_h + 2T_{add} + 1T_{sub} + 4T_{mul} + 3T_{div} + 2T_{asy}$	Log Server: $2T_h + 2T_{add} + 1T_{sub} + 3T_{mul} + 3T_{div} + 2T_{asy}$
Notification	Internet of Things: $2T_h + 2T_{add} + 1T_{sub} + 4T_{mul} + 3T_{div} + 2T_{asy}$	Security Guard/Occupant: $2T_h + 2T_{add} + 1T_{sub} + 3T_{mul} + 3T_{div} + 2T_{asy}$
Response (Public IoT)	Security Guard: $2T_h + 2T_{add} + 1T_{sub} + 4T_{mul} + 3T_{div} + 2T_{asy}$	Log Server: $2T_h + 2T_{add} + 1T_{sub} + 3T_{mul} + 3T_{div} + 2T_{asy}$
Response (Private IoT)	Occupant: $2T_h + 2T_{add} + 1T_{sub} + 4T_{mul} + 3T_{div} + 2T_{asy}$	Log Server: $2T_h + 2T_{add} + 1T_{sub} + 3T_{mul} + 3T_{div} + 2T_{asy}$
	Log Server: $2T_h + 2T_{add} + 1T_{sub} + 4T_{mul} + 3T_{div} + 2T_{asy}$	Security Guard: $2T_h + 2T_{add} + 1T_{sub} + 3T_{mul} + 3T_{div} + 2T_{asy}$

Notes: T_h: a hash operation; T_{add}: an additional operation; T_{sub}: a subtraction operation; T_{mul}: a multiplication operation; T_{div}: a division operation; T_{asy}: asymmetrical encryption/decryption.

5.2. Communication Cost

In this subsection, we calculated the communication costs as shown in Table 7. We calculated the total message length transmitted in every phase with 4G and 5G networks. The maximum speed is 100 Mbps (Megabit Per Second) in the 4G network and 20 Gbps in the 5G network. We assume that the length of an ID (L_{ID}) is 144 bits, the length of a cipher message (L_m) is 512 bits, and the length of a set of signatures (L_{sig}) is 1024 bits in the transmission message. For example, in the phase of registration of IoT, the calculation of message length is $4 \times L_{ID} + 2 \times L_m + 2 \times L_{sig} = 4 \times 144$ bits $+ 2 \times 512$ bits $+ 2 \times 1024$ bits $= 3648$ bits. Therefore, the communication cost in the 4G network is 3648 bits/100 Mbps = 35 ms, and the communication cost in the 5G network is 3648 bits/20 Gbps = 0.17 ms.

Table 7. Communication costs of the proposed scheme.

Phase	Message Length	4G (100 Mbps)	5G (20 Gbps)
Registration of IoT	3648 bits	35 ms	0.17 ms
Authentication	3648 bits	35 ms	0.17 ms
Alarm triggered	3648 bits	35 ms	0.17 ms
Notification	3648 bits	35 ms	0.17 ms
Response (Public IoT)	3648 bits	35 ms	0.17 ms
Response (Private IoT)	7296 bits	70 ms	0.34 ms

Notes: L_{ID}: an ID (144 bits); L_m: a cipher message (512 bits); L_{sig}: signature parameters (1024 bits).

5.3. Comparison

Table 8 shows the comparison with the previous researches we surveyed. In this research, we proposed a more secure IoT system with blockchain technology. Our research also provides a complete system architecture and scheme, then the security issues of the proposed method are proven in the security analysis section.

Table 8. Comparison with surveyed related works.

Authors	Year	Objective	Technologies	1	2	3	4	5	6
Dutta et al. [9]	2018	IoT security system	IoT, Arduino	Y	N	Y	N	N	N
Prasetyo et al. [10]	2018	Smart office system with threat detection	IoT, Arduino, Raspberry Pi	Y	N	Y	Y	N	N
Saeed et al. [11]	2018	Smart home environment for fire prevention	ZigBee	Y	N	Y	N	Y	N
Taryudi et al. [12]	2018	Home security and monitoring system with various types of sensor, such as PIR, DHT-22, rain, fire, LDR sensors.	Arduino-nano, NodeMCU ESP8266	Y	N	Y	N	N	N
Al-Hudhud et al. [13]	2019	Security guard system with augmented reality to monitor IoT status	Infrared biosensor, google glass	Y	N	Y	Y	Y	N
Ray et al. [14]	2020	The security issues of smart home network	Information security, networking	Y	N	Y	Y	Y	N
Khan et al. [15]	2020	Data verification system for surveillance cameras	Blockchain, IoT	Y	Y	Y	Y	Y	N
Rahman et al. [16]	2020	Distributed IoT-SDN Model for condominium	Blockchain, IoT-SDN	Y	Y	Y	N	Y	N
Khairuddin et al. [71]	2021	Smart building system with face detection and recognition	Image processing, Raspberry Pi	Y	N	Y	Y	Y	N
Our proposed method	2021	A Blockchain-based community safety security system with IoT secure devices	Blockchain, IoT	Y	Y	Y	Y	Y	Y

Notes: 1: Application for community or building safety, 2: Blockchain-based architecture, 3: Internet of Things (IoT), 4: Surveillance Camera, 5: Complete architecture or framework, 6: Security analysis, Y: Yes, N: No.

5.4. Limitations

The legal effect of the privacy issues for installing the IoT to surveillance will depend on the legal provisions in various countries. In the proposed scheme, before using the system, every participant must register with the blockchain center and have their relevant data filled in during the registration process; other relevant information is also required. After successful registration, they need to obtain the public key and private key pair of the elliptic curve signature before they participate in this system.

The most important thing is that all communities must have basic and stable electricity and networks to operate IoT devices and safety security systems. The establishment of a backup power supply in the building or community can also prevent the proposed approach from failing due to power problems. Alternatively, if the community does not have a stable Internet access network, a local network safety security system can be realized with the same architecture of the proposed scheme. Concerning the notification phase or response phase, it would be a better solution to use a more traditional Short Message Service (SMS) to send messages to related occupant or security guard mobile phones instead of network transmission.

Overall, there are some limitations to the proposed system. The proposed scheme in this article is focused on applying blockchain technology to solve security and clarification issues. Therefore, the unpopularity or the slow speed of the Internet environment will be potential issues.

6. Conclusions

Working or living in a community is frequently inextricably linked to the issue of safety. To solve the problem of community safety, we proposed a community security system based on blockchain and IoT security devices. Our research proposes a complete system architecture and provides a communication flow for multiple phases after the alarm is triggered. Security mechanisms are also integrated into the communication flow to prevent the system from being attacked.

First, all participants and IoT devices must register with BC for future communication authentication. When an IoT device triggers an alarm in the community, the message will be sent to the LS to record the alarm message, and the relevant personnel (occupier or security guard) will be notified that there is a possibility of an unsafe situation in the community. Relevant personnel must confirm and respond to LS to record. In the process of all status updates of the phase, all senders and receivers must update their signatures to BC by invoking a chaincode we provide. The chaincode execution updates all the peers' ledgers in the BC.

We have also devised different methods for managing both private and public IoT devices. Any participant that needs to check for the history records must obtain the right from the IoT's owner. The private IoT devices should be permitted by the occupant, and the public IoT should be permitted by the SG's supervisor to ensure the privacy of the occupants. Furthermore, when a participant in this system raises a dispute, the participant can raise questions about the community safety handling process with a third party in the clarification phase. The third-party can obtain the signature and timestamp data through the BC by the IoT trigger alarm ID specified by the participant. Then the third party confirms the legality of the process by verifying the validity of the signature.

With the proposed method, we achieved the following features in the safety security system:

(1) Blockchain decentralization and authentication to ensure the privacy and anonymity of participants.
(2) Unforgeable and traceability of data by the blockchain characteristic.
(3) The privacy protection when grabbing a history records from Log Server.
(4) Designed a clarifying phase for clarifying the safety system process.
(5) Signature mechanism to ensure message repudiation during communication.
(6) Asymmetrical encryption/decryption to ensure data integrity during communication.

(7) Transmission intercept prevention, prevent replay attacks from cyberattacks.
(8) Multiple security analyses have also been presented to prove the system's security.
(9) The features of other works and our proposed scheme are also compared and concluded in Table 8.

Finally, we expect that the system can be applied to various communities to strengthen the safety of the community and avoid unnecessary disputes and tragedies.

Author Contributions: The authors' contributions are summarized below. Z.-Y.L. and C.-L.C. made substantial contributions to the conception and design. Z.-Y.L. and C.-L.C. were involved in drafting the manuscript. Z.-Y.L. acquired data and analyzed and conducted the interpretation of the data. The critically important intellectual content of this manuscript was revised by H.-C.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the Ministry of Science and Technology, Taiwan, R.O.C., under Contract MOST 110-2218-E-305-001–MBK, Contract MOST 110-2410-H-324-004-MY2.

Data Availability Statement: The data used to support the findings of this study are available from the corresponding author upon request.

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

References

1. Yahoo News: Taiwan's 'Ghost Building' Fire: A Death Trap for Dozens of Elderly. Available online: https://news.yahoo.com/taiwans-ghost-building-fire-death-122230630.html (accessed on 1 November 2021).
2. The New York Times: Taiwan Couple Are Suspected of Negligent Homicide in Building Fire. Available online: https://www.nytimes.com/2021/10/18/world/asia/taiwan-building-fire.html (accessed on 1 November 2021).
3. Kammersgaard, T. Private security guards policing public space: Using soft power in place of legal authority. *Polic. Soc.* **2019**, *31*, 117–130. [CrossRef]
4. Nalla, M.K.; Crichlow, V.J. Have the standards for private security guards become more stringent in the post 9/11 era? An assessment of security guard regulations in the US from 1982 to 2010. *Secur. J.* **2017**, *30*, 523–537. [CrossRef]
5. Gurinskaya, A.L.; Nalla, M.K.; Rafailova, D.K. Are Private Security Guards Capable of Protecting Life and Property? Exploring Russian Youth's Perceptions. *Russ. J. Criminol.* **2018**, *12*, 338–348. [CrossRef]
6. Van Steden, R.; Nalla, M.K. Citizen satisfaction with private security guards in the Netherlands: Perceptions of an ambiguous occupation. *Eur. J. Criminol.* **2010**, *7*, 214–234. [CrossRef]
7. Smart Buildings Market by Component (Solution (Safety and Security Management, Energy Management, Building Infrastructure Management, Network Management, IWMS), Services), Building Type (Residential, Commercial, Industrial), Region—Global Forecast to 2025. Available online: https://www.marketsandmarkets.com/Market-Reports/smart-building-market-1169.html#:~{}:text=According%20to%20MarketsandMarkets%2C%20the%20global,14.2%25%20during%20the%20forecast%20period (accessed on 1 November 2021).
8. Medical Alert Systems Market by Type (Personal Emergency Response System (PERS) [Home-based/Landline-based System, Mobile PERS], Nurse Calling System (NCS) [Button-Based Systems, Integrated Communication Systems, Mobile Systems, Intercom Systems] and Smart Belt) by Offering (Hardware, Services and Software), by Connection Type (Wired, Wireless), by End, and by Region, Forecast to 2028. Available online: https://www.reportsanddata.com/report-detail/medical-alert-systems-market (accessed on 1 November 2021).
9. Dutta, J.; Wang, Y.; Maitra, T.; Islam, S.H.; Rawal, B.S.; Giri, D. ES3B: Enhanced Security System for Smart Building Using IoT. In Proceedings of the 2018 IEEE International Conference on Smart Cloud (SmartCloud), New York, NY, USA, 21–23 September 2018; pp. 158–165.
10. Prasetyo, T.F.; Zaliluddin, D.; Iqbal, M. Prototype of smart office system using based security system. *J. Phys. Conf. Ser.* **2018**, *1013*, 012189. [CrossRef]
11. Saeed, F.; Paul, A.; Rehman, A.; Hong, W.H.; Seo, H. IoT-Based Intelligent Modeling of Smart Home Environment for Fire Prevention and Safety. *J. Sens. Actuator Netw.* **2018**, *7*, 11. [CrossRef]
12. Taryudi; Adriano, D.B.; Ciptoning Budi, W.A. Iot-based Integrated Home Security and Monitoring System. *J. Phys. Conf. Ser.* **2018**, *1140*, 012006. [CrossRef]
13. Al-Hudhud, G.; AlSaeed, D.; Al-Baity, H.; Al-Humaimeedy, A.; Al-Turaiki, I. iGuard: Mobile security guard system with infrared biosensor and google glass article information. *Biosci. Biotechnol. Res. Commun.* **2019**, *12*, 333–337. [CrossRef]
14. Ray, A.K.; Bagwari, A. IoT based Smart home: Security Aspects and security architecture. In Proceedings of the 2020 IEEE 9th International Conference on Communication Systems and Network Technologies (CSNT), Gwalior, India, 10–12 April 2020; pp. 218–222.
15. Khan, P.W.; Byun, Y.C.; Park, N. A Data Verification System for CCTV Surveillance Cameras Using Blockchain Technology in Smart Cities. *Electronics* **2020**, *9*, 484. [CrossRef]

16. Rahman, A.; Islam, M.J.; Rahman, Z.; Reza, M.M.; Anwar, A.; Mahmud, M.A.P.; Nasir, M.K.; Noor, R.M. DistB-Condo: Distributed Blockchain-Based IoT-SDN Model for Smart Condominium. *IEEE Access* **2020**, *8*, 209594–209609. [CrossRef]
17. Khalid, U.; Asim, M.; Baker, T.; Hung, P.C.K.; Tariq, M.A.; Rafferty, L. A decentralized lightweight blockchain-based authentication mechanism for IoT systems. *Cluster Comput.* **2020**, *23*, 2067–2087. [CrossRef]
18. Van Wassenaer, L.; Verdouw, C.; Wolfert, S. What Blockchain Are We Talking About? An Analytical Framework for Understanding Blockchain Applications in Agriculture and Food. *Front. Blockchain* **2021**, *4*, 20. [CrossRef]
19. Johar, S.; Ahmad, N.; Asher, W.; Cruickshank, H.; Durrani, A. Research and Applied Perspective to Blockchain Technology: A Comprehensive Survey. *Appl. Sci.* **2021**, *11*, 6252. [CrossRef]
20. Hyperledger Fabric. Available online: https://www.hyperledger.org/use/fabric (accessed on 1 November 2021).
21. Leng, Z.; Tan, Z.; Wang, K. Application of Hyperledger in the Hospital Information Systems: A Survey. *IEEE Access* **2021**, *9*, 128965–128987. [CrossRef]
22. Iftekhar, A.; Cui, X.; Tao, Q.; Zheng, C. Hyperledger Fabric Access Control System for Internet of Things Layer in Blockchain-Based Applications. *Entropy* **2021**, *23*, 1054. [CrossRef] [PubMed]
23. Kamilaris, A.; Fonts, A.; Prenafeta-Boldύ, F.X. The rise of blockchain technology in agriculture and food supply chains. *Trends Food Sci. Technol.* **2019**, *91*, 640–652. [CrossRef]
24. Chen, C.-L.; Shang, X.; Tsaur, W.-J.; Weng, W.; Deng, Y.-Y.; Wu, C.-M.; Cui, J. An Anti-Counterfeit and Traceable Management System for Brand Clothing with Hyperledger Fabric Framework. *Symmetry* **2021**, *13*, 2048. [CrossRef]
25. Chen, C.-L.; Deng, Y.-Y.; Tsaur, W.-J.; Li, C.-T.; Lee, C.-C.; Wu, C.-M. A Traceable Online Insurance Claims System Based on Blockchain and Smart Contract Technology. *Sustainability* **2021**, *13*, 9386. [CrossRef]
26. Atzori, L.; Iera, A.; Morabito, G. The Internet of Things: A survey. *Comput. Netw.* **2010**, *54*, 2787–2805. [CrossRef]
27. Chen, S.J.; Hovde, D.C.; Peterson, K.A.; Marshall, A.W. Fire detection using smoke and gas sensors. *Fire Saf. J.* **2007**, *42*, 507–515. [CrossRef]
28. Maguluri, L.P.; Srinivasarao, T.; Ragupathy, R.; Syamala, M.; Nalini, N.J. Efficient Smart Emergency Response System for Fire Hazards using IoT. *Int. J. Adv. Comput. Sci. Appl.* **2018**, *9*, 314–320.
29. Jacob, T.P.; Pravin, A. Environmental Pollution Alerting System Using IOT. *Res. J. Pharm. Biol. Chem. Sci.* **2018**, *9*, 403–406.
30. Lee, J.; Kim, J.; Im, J.P.; Lim, S.Y.; Kwon, J.Y.; Lee, S.M.; Moon, S.E. MEMS-Based NO_2 Gas Sensor Using ZnO Nano-Rods for Low-Power IoT Application. *J. Korean Phys. Soc.* **2017**, *70*, 924–928. [CrossRef]
31. Suh, J.H.; Cho, I.; Kang, K.; Kweon, S.J.; Lee, M.; Yoo, H.J.; Park, I. Fully integrated and portable semiconductor-type multi-gas sensing module for IoT applications. *Sens. Actuator B Chem.* **2018**, *265*, 660–667. [CrossRef]
32. Gu, Z.M. Home smart motion system assisted by multi-sensor. *Microprocess. Microsyst.* **2021**, *80*. [CrossRef]
33. Choo, K.D.; Xu, L.; Kim, Y.; Seol, J.H.; Wu, X.; Sylvester, D.; Blaauw, D. Energy-Efficient Motion-Triggered IoT CMOS Image Sensor With Capacitor Array-Assisted Charge-Injection SAR ADC. *IEEE J. Solid State Circuit* **2019**, *54*, 2921–2931. [CrossRef]
34. Kim, J.W.; Sul, S.H.; Choi, J.B. Development of real-time Internet of Things motion detection platform applying non-contact sensor based on open source hardware. *Int. J. Distrib. Sens. Netw.* **2020**, *16*, 20. [CrossRef]
35. McCay, K.D.; Ho, E.S.L.; Shum, H.P.H.; Fehringer, G.; Marcroft, C.; Embleton, N.D. Abnormal Infant Movements Classification With Deep Learning on Pose-Based Features. *IEEE Access* **2020**, *8*, 51582–51592. [CrossRef]
36. Shehzed, A.; Jalal, A.; Kim, K. Multi-Person Tracking in Smart Surveillance System for Crowd Counting and Normal/Abnormal Events Detection. In Proceedings of the 2019 International Conference on Applied and Engineering Mathematics (ICAEM), Taxila, Pakistan, 27–29 August 2019; pp. 163–168.
37. Sreenu, G.; Saleem Durai, M.A. Intelligent video surveillance: A review through deep learning techniques for crowd analysis. *J. Big Data* **2019**, *6*, 48. [CrossRef]
38. Zhou, S.; Shen, W.; Zeng, D.; Fang, M.; Wei, Y.; Zhang, Z. Spatial–temporal convolutional neural networks for anomaly detection and localization in crowded scenes. *Signal Process. Image Commun.* **2016**, *47*, 358–368. [CrossRef]
39. Ermis, A.; Yurttadur, A.A.; Karacay, T. Human Intruder Detection by Measuring and Analysing Ground Vibrations. *J. Fac. Eng. Archit. Gazi Univ.* **2015**, *30*, 207–215.
40. Auvinet, E.; Multon, F.; Saint-Arnaud, A.; Rousseau, J.; Meunier, J. Fall Detection With Multiple Cameras: An Occlusion-Resistant Method Based on 3-D Silhouette Vertical Distribution. *IEEE Trans. Inf. Technol. Biomed.* **2011**, *15*, 290–300. [CrossRef]
41. Kong, X.; Meng, Z.; Meng, L.; Tomiyama, H. A Privacy Protected Fall Detection IoT System for Elderly Persons Using Depth Camera. In Proceedings of the 2018 International Conference on Advanced Mechatronic Systems (ICAMechS), Zhengzhou, China, 30 August–2 September 2018; pp. 31–35.
42. Leone, A.; Diraco, G.; Siciliano, P. Detecting falls with 3D range camera in ambient assisted living applications: A preliminary study. *Med. Eng. Phys.* **2011**, *33*, 770–781. [CrossRef] [PubMed]
43. Shojaei-Hashemi, A.; Nasiopoulos, P.; Little, J.J.; Pourazad, M.T. Video-based Human Fall Detection in Smart Homes Using Deep Learning. In Proceedings of the 2018 IEEE International Symposium on Circuits and Systems (ISCAS), Florence, Italy, 27–30 May 2018; pp. 1–5.
44. Gia, T.N.; Sarker, V.K.; Tcarenko, I.; Rahmani, A.M.; Westerlund, T.; Liljeberg, P.; Tenhunen, H. Energy efficient wearable sensor node for IoT-based fall detection systems. *Microprocess. Microsyst.* **2018**, *56*, 34–46. [CrossRef]
45. Kim, T.; Park, H.; Hong, S.H.; Chung, Y. Integrated System of Face Recognition and Sound Localization for a Smart Door Phone. *IEEE Trans. Consum. Electron.* **2013**, *59*, 598–603. [CrossRef]

46. Huang, Z.G.; Zhang, L.; Meng, X.Y.; Choo, K.K.R. Key-Free Authentication Protocol Against Subverted Indoor Smart Devices for Smart Home. *IEEE Internet Things J.* **2020**, *7*, 1039–1047. [CrossRef]
47. Song, S.J.; Nam, H. Visible light Identification System for Smart Door Lock Application with Small Area Outdoor Interface. *Curr. Opt. Photonics* **2017**, *1*, 90–94. [CrossRef]
48. Won, J.; Park, J.; Park, J.W.; Kim, I.H. BLESeis: Low-Cost IoT Sensor for Smart Earthquake Detection and Notification. *Sensors* **2020**, *20*, 13. [CrossRef]
49. Taale, A.; Ventura, C.E.; Marti, J. On the feasibility of IoT-based smart meters for earthquake early warning. *Earthq. Spectra* **2021**, *37*, 2066–2083. [CrossRef]
50. Khan, I.; Choi, S.; Kwon, Y.W. Earthquake Detection in a Static and Dynamic Environment Using Supervised Machine Learning and a Novel Feature Extraction Method. *Sensors* **2020**, *20*, 21. [CrossRef] [PubMed]
51. Popper, N. Decoding the enigma of Satoshi Nakamoto and the birth of Bitcoin. *New York Times*, 15 May 2015.
52. Peck, M.E. The cryptoanarchists' answer to cash. *IEEE Spectrum* **2012**, *49*, 50–56. [CrossRef]
53. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. Available online: https://bitcoin.org/bitcoin.pdf (accessed on 29 December 2020).
54. Buterin, V. A next-generation smart contract and decentralized application platform. *Ethereum White Pap.* **2014**, *3*, 36.
55. Corda. Available online: https://www.corda.net/ (accessed on 1 November 2021).
56. Azure Blockchain. Available online: https://azure.microsoft.com/en-us/solutions/blockchain/ (accessed on 1 November 2021).
57. Nasir, Q.; Qasse, I.A.; Abu Talib, M.; Nassif, A.B. Performance Analysis of Hyperledger Fabric Platforms. *Secur. Commun. Netw.* **2018**, *2018*, 3976093. [CrossRef]
58. Chen, C.-L.; Lim, Z.-Y.; Liao, H.-C.; Deng, Y.-Y.; Chen, P. A Traceable and Verifiable Tobacco Products Logistics System with GPS and RFID Technologies. *Appl. Sci.* **2021**, *11*, 4939. [CrossRef]
59. Iftekhar, A.; Cui, X.H. Blockchain-Based Traceability System That Ensures Food Safety Measures to Protect Consumer Safety and COVID-19 Free Supply Chains. *Foods* **2021**, *10*, 12. [CrossRef] [PubMed]
60. Zhang, Y.; Wang, Z.; Deng, J.; Gong, Z.; Flood, I.; Wang, Y. Framework for a Blockchain-Based Infrastructure Project Financing System. *IEEE Access* **2021**, *9*, 141555–141570. [CrossRef]
61. Xiao, Z.; Li, Z.; Yang, Y.; Chen, P.; Liu, R.W.; Jing, W.; Pyrloh, Y.; Sotthiwat, E.; Goh, R.S.M. Blockchain and IoT for Insurance: A Case Study and Cyberinfrastructure Solution on Fine-Grained Transportation Insurance. *IEEE Trans. Comput. Soc. Syst.* **2020**, *7*, 1409–1422. [CrossRef]
62. Tanwar, S.; Parekh, K.; Evans, R. Blockchain-based electronic healthcare record system for healthcare 4.0 applications. *J. Inf. Secur. Appl.* **2020**, *50*, 102407. [CrossRef]
63. Jayaraman, R.; Salah, K.; King, N. Improving Opportunities in Healthcare Supply Chain Processes via the Internet of Things and Blockchain Technology. *Int. J. Healthc. Inf. Syst. Inf.* **2019**, *14*, 49–65. [CrossRef]
64. Mohamed, K.S. Cryptography Concepts: Integrity, Authentication, Availability, Access Control, and Non-repudiation. In *New Frontiers in Cryptography: Quantum, Blockchain, Lightweight, Chaotic and DNA*; Mohamed, K.S., Ed.; Springer International Publishing: Cham, Switzerland, 2020; pp. 41–63. [CrossRef]
65. Yuan, H.; Chen, X.; Wang, J.; Yuan, J.; Yan, H.; Susilo, W. Blockchain-based public auditing and secure deduplication with fair arbitration. *Inf. Sci.* **2020**, *541*, 409–425. [CrossRef]
66. Conti, M.; Dragoni, N.; Lesyk, V. A Survey of Man In The Middle Attacks. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 2027–2051. [CrossRef]
67. Zhao, Y.; Li, Y.; Mu, Q.; Yang, B.; Yu, Y. Secure Pub-Sub: Blockchain-Based Fair Payment With Reputation for Reliable Cyber Physical Systems. *IEEE Access* **2018**, *6*, 12295–12303. [CrossRef]
68. Feng, Y.; Wang, W.; Weng, Y.; Zhang, H. A Replay-Attack Resistant Authentication Scheme for the Internet of Things. In Proceedings of the 2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC), Guangzhou, China, 21–24 July 2017; pp. 541–547.
69. Zaman, A.; Safarinejadian, B.; Birk, W. Security analysis and fault detection against stealthy replay attacks. *Int. J. Control.* **2020**, 1–14. [CrossRef]
70. Johnson, D.; Menezes, A.; Vanstone, S. The Elliptic Curve Digital Signature Algorithm (ECDSA). *Int. J. Inf. Secur.* **2001**, *1*, 36–63. [CrossRef]
71. Khairuddin, M.H.; Shahbudin, S.; Kassim, M. A smart building security system with intelligent face detection and recognition. *IOP Conf. Ser. Mater. Sci. Eng.* **2021**, *1176*, 012030. [CrossRef]