applied sciences

Article

A Traceable and Authenticated IoTs Trigger Event of Private Security Record Based on Blockchain

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

[1] School of Computer and Information Engineering, Xiamen University of Technology, Xiamen 361024, China; clc@mail.cyut.edu.tw
[2] School of Information Engineering, Changchun Sci-Tech University, Changchun 130600, China
[3] Department of Computer Science and Information Engineering, Chaoyang University of Technology, 168 Jifeng East Road, Taichung 41349, Taiwan; allendeng@cyut.edu.tw
* Correspondence: zylim@cyut.edu.tw(Z.-Y.L.); hcliao@cyut.edu.tw (H.-C.L.)

Abstract: Recently, private security services have become increasingly needed by the public. The proposed scheme involves blockchain technology with a smart contract. When a private security company signs a contract with a client, they install an Internet of Things (IoTs) device in the client's house and connect it with the IoT main controller; then, the IoT main controller connects to the security control center (SCC). Once there is an event triggered (e.g., a break-in or fire incident) by the IoTs device, the controller sends a message to the SCC. The SCC allocates a security guard (SG) to the incident scene immediately. After the task is accomplished, the SG sends a message to the SCC. All of these record the messages and events chained in the blockchain center. The proposed scheme makes security event records have the following characteristics: authenticated, traceable, and integral. The proposed scheme is proved by a security analysis with mutual authentication, traceability, integrity, and non-repudiation. The known attacks (e.g., man-in-the-middle attack, replay attack, forgery attack) are avoided by message encryption and a signing mechanism. Threat models in the communication phase can also be avoided. Finally, computation cost, communication performance, and comparison with related works are also discussed to prove its applicability. We also provide an arbitration mechanism, so that the proposed scheme can reduce disputes between private security companies and the client.

Keywords: private security guard; blockchain; Internet of Things (IoTs), traceable

check for updates

Citation: Chen, C.-L.; Lim, Z.-Y.; Liao, H.-C.; Deng, Y.-Y. A Traceable and Authenticated IoTs Trigger Event of Private Security Record Based on Blockchain. *Appl. Sci.* **2021**, *11*, 2843. https://doi.org/10.3390/app11062843

Academic Editor: Lin-Huang Chang

Received: 12 January 2021
Accepted: 19 March 2021
Published: 22 March 2021

1. Introduction

1.1. Background

In recent years, the economy has become more and more prosperous, but the security of the city has not improved along with economic growth. To strengthen the security of the city, except for the police placed by the government in the city, the general company or the community will sign a contract with a private security company, and that company provides the security service to ensure the property of the client. Nalla [1] wrote that "There has been a steady increase in security guard employment in the United States in recent decades. Data estimates for 2010 suggest that there are over 10,000 security companies in the United States, employing 1,046,760 guards." The above description indicates that the demand to hire private security services is increasing and the social needs of the security industry have become very important.

Furthermore, the Federal Bureau of Investigation (FBI) in the United States analyzes offense statistics every year [2]. We sorted out the annual statistics from 2015 to 2019 regarding cases of burglary and larceny-theft in buildings. Table 1 shows the number of the two types of cases and the average US dollar value of the property loss. Except for the United States, the burglary record in England and Wales has come to 417,416 cases [3].

Therefore, private security services are important to prevent the occurrence of criminals. When crime continues to exist, people need the service to avoid it.

Table 1. FBI offense statistics from 2015 to 2019 [2].

Year	Burglary (Property Value)	Larceny-Theft (Property Value)
2015	1,395,913 ($2316)	582,055 ($1394)
2016	1,354,920 ($2361)	533,553 ($1449)
2017	1,250,983 ($2416)	523,223 ($1381)
2018	1,047,388 ($2799)	448,439 ($1610)
2019	917,464 ($2661)	404,734 ($1663)

There are several types of services of private security companies: community security, patrol security, bodyguards, armored cars, etc. In this research, we focus on community security. The operation of community security comprises several cars with security guards patrol in a specific area; if a client's house or company sends an alarm to the security control center, the control center needs to notify the security guards and solve the problem to protect the property of the client.

Most security companies include security systems with their services, such as video surveillance cameras, access control keys, and virtual fencing systems [4]. Nowadays, many Internet of Things (IoTs)-related products have been developed, e.g., door/window sensors, smoke detectors, motion sensors, and alarm sirens. Those devices can also be integrated into the security system which makes the system automated. Although the ability to automatically alert has been achieved, there are still some problems. Firstly, the security of the Internet of Things is constantly being challenged [5,6]. Secondly, the security company is unable to keep trackable records; it is insufficient to clarify the responsibility between the security company and the client. However, if it needs real-time processing, records can be verified and traceable.

Blockchain is a technology that can solve issues. Blockchain technology is widely applied to the cryptocurrency Bitcoin. Bitcoin was developed and deployed by Nakamoto in 2008 [7]; it is a digital virtual currency that is secured by cryptography [8]. The notion of blockchain is decentralized networks, and every data chained in the network is irreversible. Blockchains are also authentic, consistent, able to provide consensus, and traceable. Because of the characters and advantages, blockchain technology has been affected and applied in various fields in recent years; for example, in healthcare [9–11], payment [12–14], digital content [15–17], etc. For all the benefits mentioned above, blockchain technology is very suitable to be involved in security systems. There are at least three types of blockchains: public blockchains, private blockchains, and hybrid blockchains. In our scheme, private blockchains are more suitable for protecting customer privacy, and it is maintainable by the security service provider. Those security events are stored in an immutable ledger, and the ledger will be mainly stored with the security company, official agency, and client, to ensure that the information will not be arbitrarily altered.

To sum up, our research aims to accomplish the following goals: an automated alarm system that communicates with the security control center to propose an authentication protocol that supports real-time event status report/record, data traceability, data integrity, data non-repudiation, and proposes an arbitration mechanism when the dispute occurs.

1.2. Related Works

The related works are surveyed and shown in Table 2. In this table, it can be seen that some of the related literature are compared with their characteristics. There are lots of security services applied within a smart home, a city, campuses, and industries; the most important element of these smart services is using large amounts of IoT devices to achieve automated security services. Some of these authors also used blockchain to ensure the architecture of communication between IoTs is more secure.

Table 2. The related works survey.

Authors	Year	Objective	Technologies	1	2	3	4	5
Smith et al. [18]	2010	Security services with a central alarm station and dispatch operations	Alarm system, access control system	N	Y	Y	N	N
Liu et al. [19]	2020	Blockchain-based access control system	Blockchain, IoT	Y	N	Y	N	N
Lin et al. [20]	2018	Blockchain-based access control to access control for industry 4.0	Blockchain, access control system	Y	N	Y	Y	Y
Mbarek et al. [21]	2020	Blockchain-based access control to controls the smart home devices	Blockchain, smart home, IoT	Y	N	Y	N	N
Alkhammash et al. [22]	2020	Smart campus with the Internet of Things and blockchain	Blockchain, smart campus, IoT	Y	N	Y	N	N
Lyu et al. [23]	2019	Accessing private smart home with smart devices through an IFTTT Gateway	Smart home, IFTTT, IoT	N	N	Y	N	Y
Fakroon et al. [24]	2020	Remote anonymous authentication for smart home	Smart home, IoT	N	N	Y	Y	Y

Notes: (1) Focus on a blockchain, (2) security services by a third-party, (3) proposing an architecture or framework, (4) method classification, (5) security analysis, (Y) yes, (N) no.

Smith et al. [18] introduced a security service with central alarm stations and dispatch centers. The monitoring system has various sensors that cooperate with the central station, (the central station is known as the security control center); if there an alarm is triggered and transmitted to the monitoring system, the center will dispatch an official to resolve the problem. These information automated security services can ensure the safety of the protected person or property, but if there is no information security technology that is involved and analyzed, the service may be subject to cyber-attacks.

Liu et al. [19] proposed a blockchain-based access control system with IoT in 2020. The system contains three types of smart contracts, which are Device Contract (DC), Policy Contract (PC), and Access Contract (AC). The author used a decentralized management framework, but they did not prove the security analysis of the scheme. Lin et al. [20] proposed a blockchain-based access control system in 2018. The system was designed to be used in industry 4.0, and the information security technology is also included in the proposed scheme; however, there is no security control center dispatch mechanism to keep monitoring and to dispatch professionals to handle the problem.

Other applications, such as smart homes and smart campuses, also implement blockchain and IoT technology. Mbarek et al. [21] proposed a blockchain-based access control to control smart home devices, but the main disadvantage of their research is the lack of information security analyses. Alkhammash et al. [22] applied blockchain technology in smart campuses; they proposed a new smart campus framework with some layers, such as a physical layer, communication layer, platform layer, data layer, business layer, and application layer, but the research lacked a traceable mechanism and security analysis.

There is some literature involved in the research of smart homes and information security analyses. Lyu et al. [23] were involved in information security with an IFTTT (if this then that) gateway in the smart home; the proposed method implemented authentication and key agreement schemes when the user accessed the gateway remotely. Furthermore, Fakroon et al. [24] proposed remote anonymous authentication for smart homes. These studies lack record traceability and integrity.

To summarize the shortcomings of the current research, there is no literature proposing complete security services by a third-party; for example, private security guards, and community security companies. Properly storing the content of the incident is also important for trust between the services provider and client, so it is also important to store complete records of the event processing to achieve traceability, integrity, non-repudiation, etc. Therefore, we propose smart contracts that enable chaining the event's record on the blockchain center.

The remaining sections of this paper are organized as follows: Section 2 introduces the techniques that are used in our proposed scheme. Section 3 presents our proposed scheme. The security analysis and comparison discussion are given in Sections 4 and 5, respectively. Finally, Section 6 concludes this paper.

2. Preliminary

2.1. Smart Contract and Blockchain

In 2014, Buterin implemented the first blockchain-based smart contracts technologies of Ethereum [25]. As an event occurs, a smart contract is able to be executed by the predesigned program automatically; every smart contract that is chained to the blockchain also inherits the characteristics of the blockchain. Therefore, we applied private blockchains with the Ethereum framework. Commonly, there are many types of consensus algorithms in blockchain technologies, e.g., Proof-of-Work (PoW), Proof-of-Stake (PoS), and Proof-of-Authority (PoA). Most private enterprise blockchain applications use a PoA consensus. F. Þ. Hjálmarsson et al. proposed a blockchain-based e-voting system with PoA in 2018 [26]. Neo also proposed an enterprise blockchain that implements supply chain anti-counterfeiting and traceability [27]; the proposed scheme also applied the PoA-based consensus in the blockchain. The PoA consensus algorithm uses fewer computing resources to quickly validate and upload the block to the blockchain. According to the advantages of PoA, our proposed scheme is implemented with PoA. By applying PoA in our private blockchain, only authorized accounts can verify transactions and blocks; therefore the information of the client is kept confidential.

2.2. ECDSA

Vanstone [28] proposed the Elliptic Curve Digital Signature Algorithm (ECDSA) in 1992, which is a derived type of the Digital Signature Algorithm (DSA) that uses Elliptic Curve Cryptography (ECC). ECC is a type of public-key cryptography based on the algebraic structure of elliptic curves over finite fields. The characteristics of ECC make the ECDSA require a significantly smaller key size with the same level of security which offers faster computations and less storage space.

Johnson et al. [29] introduced the ECDSA in detail, including its acceptance in any global standards. ECDSA is accepted in the following standards:

(1) ISO 14888-3: International Standards Organization standard in 1998.
(2) ANSI X9.62: American National Standards Institute standard in 1999.
(3) IEEE 1363-2000: Institute of Electrical and Electronics Engineers standard in 2000.
(4) FIPS 186-4: Federal Information Processing standard in 2013.
(5) ANSI X9.142-2020: American National Standards Institute standard in 2020.

Next, we briefly describe the three phases of ECDSA key generation for verification:

(1) Key generation phase: We assume that any participant must apply to our blockchain center for public and private keys, and the key generation with the ECDSA is as follows: $Q_X = d_X G$, where X is the participant ID, Q_X is the public key, d_X is the private key, and G is a generating point based on the elliptic curve. The public key Q_X and private key d_X are sent to the participant and stored. Q_X is also stored in the blockchain center.
(2) Signature phase: There is a message that needs to be sent by participant X to Y. X choose a random number k between 1 and $n-1$, and calculates a point on the curve

as follows $(x, y) = k \times G$. Then, it calculates $R_X = x \bmod n$. Next, X signs a message M as follows: $Sig_X = k^{-1}(H(M) + R_X d_X)$, sends (M, R_X, Sig_X) to Y.

(3) Verification phase: When Y receives (M, R_X, Sig_X), it then calculates the parameters as follows: $(x', y') = (H(M){Sig_X}^{-1} \bmod n)G + (\mathrm{R}_X {Sig_X}^{-1} \bmod n)Q_X$. Then, it verifies $x' \overset{?}{=} R_X \bmod n$ to determine if the signature is valid.

In general, some key parameters need to be defined: in our proposed scheme, the length of bits and Secure Hash Algorithm (SHA), according to the NIST's minimum security-strength requirement [30], set the length of n with 224 bits and SHA-512/224 for digital signature generation.

2.3. BAN Logic

Burrows et al. [31] proposed BAN logic, a defining logic for the analysis of security protocols in 1990. BAN logic is a logic of beliefs, and the purpose of BAN logic is to analyze authentication protocols by deriving the beliefs [32]. Therefore, BAN logic can be used for validating the mutual authentication with the communication protocols.

2.4. Threat Model

The threat model involves the understanding of latent threats to the system; it is an important security analysis that needs to be done to determine the method of defense. When the security services system is implemented in the security company, various threats may be encountered. The threats are as follows:

(1) The integrity of data: The data sent by the sender must not be modified or destroyed by other parties so as to guarantee privacy and security [33]. When the receiver receives the message, they are able to verify the completeness of the data to guarantee that the data will not be missed.

(2) Untraceable record: Every record that is generated from a security services company is saved in the database; the record can be freely modified by the service provider, and customers that apply the security service are unable to trace their records correctly. An automated and traceable record included at every stage of the case processing is important to various applications for making evidence in the arbitration phase (for example, medical, criminal, smart home, etc.) [34–36].

(3) Message repudiation: A digital signature can ensure the non-repudiation of the message [37]. Every message that sends from the sender should be signed with the sender's private key; it makes the receiver able to prove that the message is definitely from the sender.

(4) Third-party arbitration: The responsibility clarification is important when there is a doubt that occurs. In the traditional security service, the client only can doubt the security company, and the company is able to falsify the record to protect the company's reputation. So the third-party fair arbitration is important for clarification [38].

(5) Cyber-attacks: There are some cyber-attacks by attackers that must be considered; for example, a man-in-the-middle attack [39], replay attack [40], forgery attack [41], etc.

(6) Unstable communication: There is some internet communication between IoT devices, such as the IoT main controller, clients, and security company. The unstable communication will cause data loss and the message will not be completed. Therefore, the stabilization of communication must be taken seriously [42].

(7) Side-channel attack: This is a physical security attack. There are various kinds of side-channel attacks, such as Simple Power Analysis (SPA) and Differential Power Analysis (DPA) [43]. This type of attack monitors the power consumption of the electronic device to extract important data like secret keys [44]. Daniel et al. demonstrated an ECDSA key extraction from mobile devices with a side-channel attack [45].

3. The Proposed Scheme

The proposed system consists of the following six parties: the client, the IoT main controller of the client's house, the security control center, the security guard, the blockchain center, and the official agency. The system framework is shown in Figure 1.

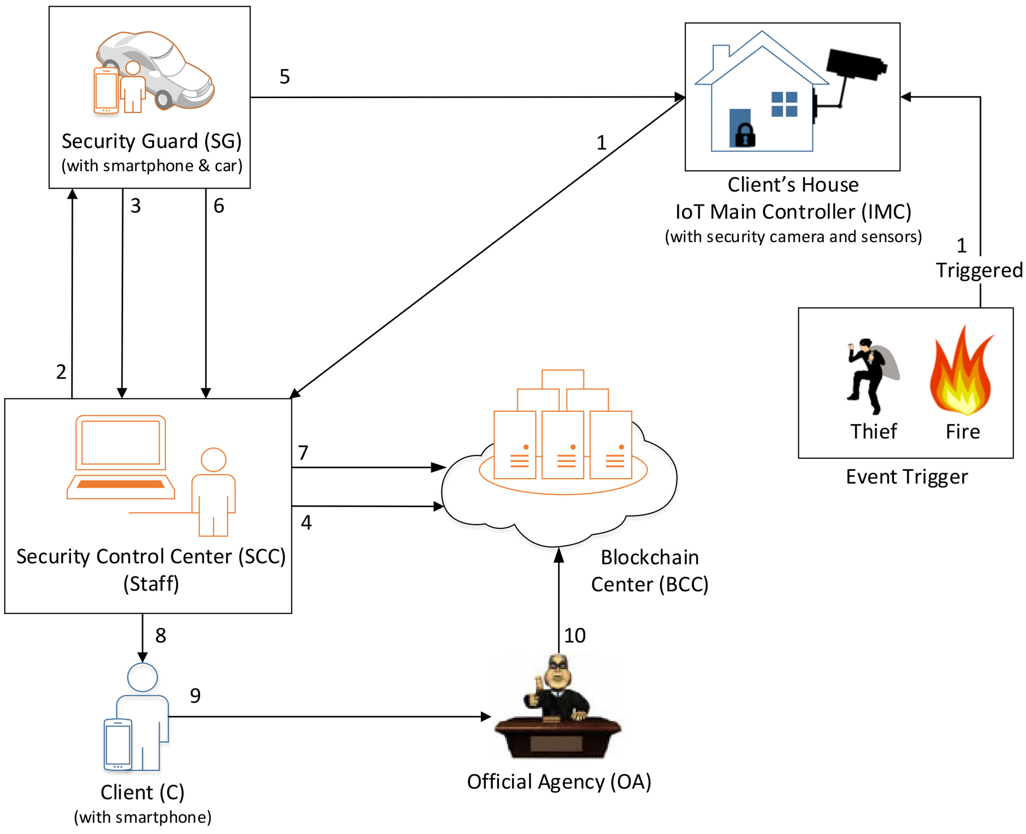


Figure 1. System framework of the proposed scheme.

3.1. System Architecture

1. Client (C): The customer needs a security service from the security company. He/she carries a smartphone with the security company's application that enables them to get the notification when an event happens at his/her house.
2. IoT main controller (IMC): A controller that is installed in a client's house, and connects to every Internet of Things (IoT) device in the house, e.g., camera, alarm sensors, magnetic contact, switched, motion detection devices, and fire detection devices. All security events are controlled by the IoT main controller. Each sensor can be triggered by an event that triggers the security problem with the client's house; for example, a thief breaking into the house, or a fire incident.
3. Security control center (SCC): This is the security company's control center that can monitor the client's home security situation. If there is an event triggered by the client's house, the control center's staff needs to notify a security guard to take action and go to the client's house to solve the problem.
4. Security guard (SG): He/she drives a car and carries a smartphone. He/she needs to take action when they receive the task/notification from the SCC to solve the security problem in the first scene.
5. Blockchain center (BCC): This is the blockchain that records the event. When an event from the client's house is triggered, the BCC creates a record with an event ID, and sends an active notification to the SCC; every message reply to the event needs to be chained with the event ID.

6. Official agency (OA): This is the third-party official agency. The client can ask to prove the contract and event from the BCC. The official agency has the right to prove that is there any problem when executing the contract from the security company.

All staff in the security company (including security control center staff, and the security guard) needs to register an account from the BCC to get a unique ID. When the client needs a security service from the security company, after the contract is signed, the client, the client's house with the IoT main controller, and the IoT sensors that are inside the house also need to register to the BCC to get a unique ID, a public key, and a private key pair. Figure 1 presents the scenarios which are triggered by an event by some IoT devices (IoT) of the client's house.

Step 1. A thief breaks into the client's house or a fire incident happens. The client's house security system (with sensors) is triggered to the IMC, and the IMC notifies the SCC with a new event ID.
Step 2. The SCC's staff will connect and check the recorded video from the digital video recorder (DVR) before and after the occurrence event time manually. Every client's house needs to install a digital video recorder (DVR) and connect it with multiple cameras to capture the critical area. Those devices are classified as IoT devices. If there are no abnormal movements or objects (e.g., human break-in, or fire) in the video, then that means it is a false alarm. The SCC's staff checks if it is a false alarm or not and confirms it. Then, SCC searches for available SGs that are nearby the event client's house. If there is an event at the client's house, a notification will be sent to the specific SG.
Step 3. If SG accepts the task, then the SG sends a confirmation message to SCC.
Step 4. SCC generates and sends the confirmation message to BCC and chains the confirmation record in the BCC.
Step 5. The SG goes to the incident scene and solves the security event.
Step 6. After the event is solved, the security guard reports the detailed records and sends the report to the SCC.
Step 7. SCC sends the record to BCC and chains the report to BCC.
Step 8. The client receives the latest status notification from the SCC.
Step 9. If the client does not satisfy the service of the security company, the client can propose an arbitration request to the official agency (arbiter).
Step 10. Since the processing details (included the event timestamp and signature) are chained in the record, the OA can retrieve and verify the event record from the BCC.

3.2. Notation

The following is an explanation of these symbols.

ID_X	X's identity
Pwd_X	X's password
ID_{Event}	Event ID that generates automatically while an event is triggered. The ID will be named in the format: (five digits of the postcode) + (five digits of the IMC ID) + (four digits of sequence value)
$Token_i$	The ith token issued by the BCC, which is used to prove whether the identity is legal
Puk_X	The public key of party X, which is issued by the BCC
Prk_X	The private key of party X, which is issued by the BCC
Sig_{X_i}	The ith signature of X
C_{X_i}	The ith ciphertext of X
h_{X_i}	The ith hash value of X
T_i	The ith timestamp that sender sends a message to the receiver
T_{i+1}	The i + 1th timestamp that the receiver received a message from the sender
Seq_i	The sequential number i from the sender increased to i + 1 in the next message
M_{Event_i}	The ith message of any party event information
M_{Access}	The message of any party that accessed SCC's request message
$E_{Puk_X}(M)/D_{Prkx_X}(M)$	Encrypt/decrypt message M with a public key or private key of party X
$h(M)$	The hash value of a message M is calculated by a one-way hash function
$A \overset{?}{=} B$	Determine if A is equal to B or not
τ	The threshold that checks the validity of the send and receives a timestamp

3.3. Initialization Phase

All parties need to register with the SCC, and the public and private keys of all parties will be generated through the blockchain center; a public key Puk_X and private key Prk_X will be issued to parties. Figures 2 and 3 are the smart contract structure of our scheme. The enumeration of the IoT type is shown in Figure 2, e.g., alarm, camera, fire detector, motion detector. The information structure of the security control center's staff, security guard, client, and the IoT device information inside the client's house are shown in Figure 3; all parties need to register their information with those defined data fields. The staff (including the control center's staff, security guard) needs to login into the security control application when starting to work and log out when off-duty; those actions will connect to BCC and set the "OnDuty" variable to true or false.

```
enum IoTTypes{
        Alarm,
        Camera,
        IRCamera,
        Door_Locker,
        Window_Locker,
        Infrared,
        Temperature_Detector,
        Fire_Detector,
        Smoke_Detector,
        Motion_Detector,
        Fence_Detector,
        others
    }
```

Figure 2. The enumeration of IoT types.

```
struct SecurityControlCenterStaff{
    uint ID;
    string Name;
    string Detail;
    bool OnDuty;
}
struct SecurityGuard{
    uint ID;
    string Name;
    string Detail;
    string GPS_Location;
    string Carplatte_No;
    bool OnDuty;
}
struct IoT{
    IoTTypes IOT_Type;
    uint ID;
    string Name;
    string Position_At_Home;
    string Detail;
    uint Timestamp_installed;
}
```

```
struct Client{
    uint ID;
    string Name;
    string Detail;
    string HP_Number;
    string Home_Address;
    string Home_GPS_Location;
    string Home_Telephone_Number;
    string IoT_Main_Controller_IP;
    IoT[] Devices;
    uint Contract_Timestamp_start;
    uint Contract_Timestamp_end;
    uint Maximum_Time_Of_Notify;
    uint Maximum_Time_Of_Process;
    uint PaymentAmount;
    string[] PaymentRecord;
    string BankAccount;
}
```

Figure 3. Smart contract initialization.

3.4. Communication Phase

The SCC keeps a connection with the IoT main controller in the CH. When the connection is lost, the SCC will automatically send a notification to SG to solve the communication problem.

The scenario of the communication phase is shown in Figure 4. The situation of a loss of connection from the SCC to the client's house, or if the IoT devices do not work, is detailed in the following steps:

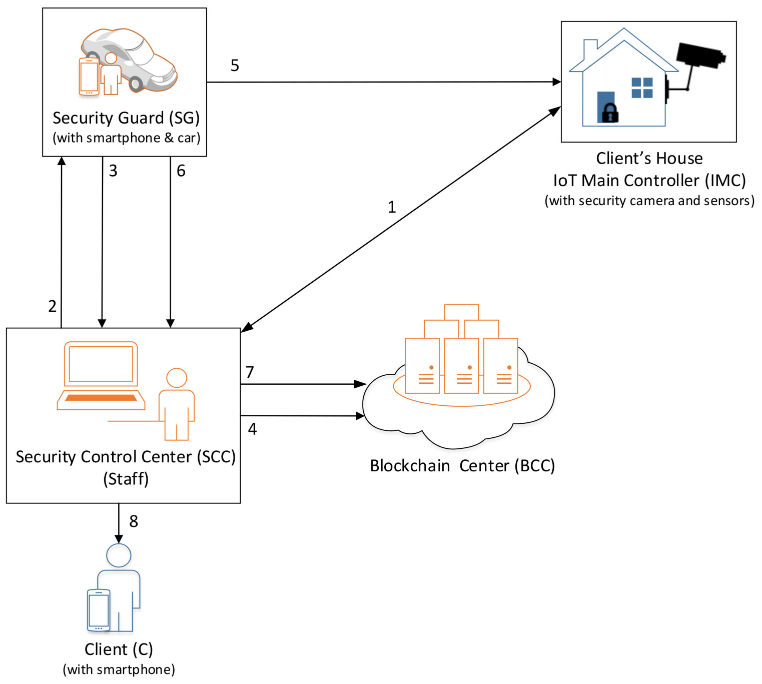


Figure 4. The mechanism in the communication phase.

Step 1. The SCC keeps communicating for the connection of the IMC in the client's house every minute (depending on the contract). If the IoT devices or the controller of the client's house does not respond, it means that the security mechanism is not working. SCC will trigger a lost connection event and record it.
Step 2. The SCC staff check if it is a false alarm or not and confirm it. After confirmation, the SCC will search for available SGs who are nearby the client's house location automatically. The event is authenticated by the SCC, and a notification will be sent to the specific SG who is near the house.
Step 3. The SG confirms and accepts the task, and sends a confirmation message to the SCC.
Step 4. The SCC sends the event confirmation to the BCC and chains the confirmation record in the BCC.
Step 5. The SG will go to the incident scene and solve the communication problem.
Step 6. After the problem is solved, the security guard will report the problem with details and send the content to the SCC.
Step 7. The SCC sends the record to the BCC and chains the report in the BCC.
Step 8. The client gets the latest status notification from the SCC.

3.5. Authentication Phase

In this phase, access parties (AP) need to be authenticated by the SCC. The AP means any party who wants to access the information of the SCC, for example, the client (C), the IoT main controller (IMC), and security guard (SG). After the verification, the party will get a token and the AP will use the token to access the information of the SCC. Figure 5 is the flowchart of the authentication phase.

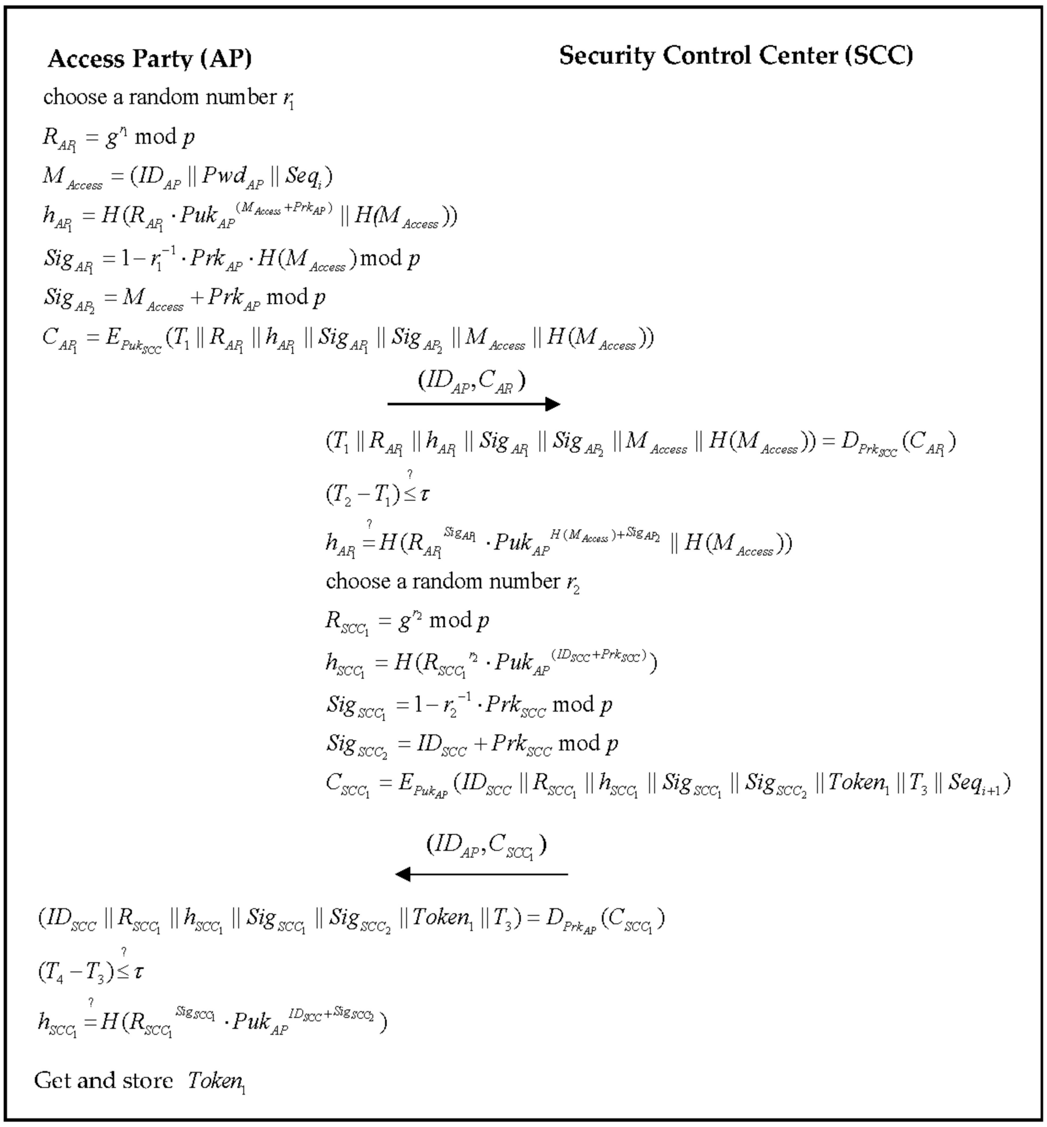

Access Party (AP)
Security Control Center (SCC)
choose a random number r_1
$R_{AP_1} = g^{r_1} \bmod p$
$M_{Access} = (ID_{AP} || Pwd_{AP} || Seq_i)$
$h_{AP_1} = H(R_{AP_1} \cdot Puk_{AP}^{(M_{Access}+Prk_{AP})} || H(M_{Access}))$
$Sig_{AP_1} = 1 - r_1^{-1} \cdot Prk_{AP} \cdot H(M_{Access}) \bmod p$
$Sig_{AP_2} = M_{Access} + Prk_{AP} \bmod p$
$C_{AP_1} = E_{Puk_{SCC}}(T_1 || R_{AP_1} || h_{AP_1} || Sig_{AP_1} || Sig_{AP_2} || M_{Access} || H(M_{Access}))$
(ID_{AP}, C_{AP_1})
$(T_1 || R_{AP_1} || h_{AP_1} || Sig_{AP_1} || Sig_{AP_2} || M_{Access} || H(M_{Access})) = D_{Prk_{SCC}}(C_{AP_1})$
$(T_2 - T_1) \overset{?}{\le} \tau$
$h_{AP_1} \overset{?}{=} H(R_{AP_1}^{Sig_{AP_1}} \cdot Puk_{AP}^{H(M_{Access})+Sig_{AP_2}} || H(M_{Access}))$
choose a random number r_2
$R_{SCC_1} = g^{r_2} \bmod p$
$h_{SCC_1} = H(R_{SCC_1}^{r_2} \cdot Puk_{AP}^{(ID_{SCC}+Prk_{SCC})})$
$Sig_{SCC_1} = 1 - r_2^{-1} \cdot Prk_{SCC} \bmod p$
$Sig_{SCC_2} = ID_{SCC} + Prk_{SCC} \bmod p$
$C_{SCC_1} = E_{Puk_{AP}}(ID_{SCC} || R_{SCC_1} || h_{SCC_1} || Sig_{SCC_1} || Sig_{SCC_2} || Token_1 || T_3 || Seq_{i+1})$
(ID_{AP}, C_{SCC_1})
$(ID_{SCC} || R_{SCC_1} || h_{SCC_1} || Sig_{SCC_1} || Sig_{SCC_2} || Token_1 || T_3) = D_{Prk_{AP}}(C_{SCC_1})$
$(T_4 - T_3) \overset{?}{\le} \tau$
$h_{SCC_1} \overset{?}{=} H(R_{SCC_1}^{Sig_{SCC_1}} \cdot Puk_{AP}^{ID_{SCC}+Sig_{SCC_2}})$
Get and store $Token_1$

Figure 5. The flowchart of the authentication phase.

Step 1. The AP selects a random number r_1, and computes the following parameter:

$$R_{AP_1} = g^{r_1} \bmod p \tag{1}$$

AP computes the access message:

$$M_{Access} = (ID_{AP} || Pwd_{AP} || Seq_i) \tag{2}$$

AP uses the hash function to compute the parameter h_{AP_1}:

$$h_{AP_1} = H(R_{AP_1} \cdot Puk_{AP}^{(M_{Access}+Prk_{AP})} || H(M_{Access})) \tag{3}$$

and uses its private key Prk_{AP} and M_{Access} to compute the signatures as follows:

$$Sig_{AP_1} = 1 - r_1{}^{-1} \cdot Prk_{AP} \cdot H(M_{Access}) \bmod p \tag{4}$$

$$Sig_{AP_2} = M_{Access} + Prk_{AP} \bmod p \tag{5}$$

After that, AP uses the SCC's public key Puk_{SCC} to encrypt the message into a ciphertext C_{AP_1}:

$$C_{AP_1} = E_{Puk_{SCC}}(T_1 || R_{AP_1} || h_{AP_1} || Sig_{AP_1} || Sig_{AP_2} || M_{Access} || H(M_{Access})) \tag{6}$$

Then, AP sends the (ID_{AP}, C_{AP_1}) to *SCC*.

Step 2. Once the *SCC* receives the *AP*'s *ID* and cipher message C_{AP_1} at T_2, *SCC* uses the private key Prk_{SCC} to decrypt the message:

$$(T_1||R_{AP_1}||h_{AP_1}||Sig_{AP_1}||Sig_{AP_2}||M_{Access}||H(M_{Access})) = D_{Prk_{SCC}}(C_{AP_1}) \tag{7}$$

Then, *SCC* checks the validity of the timestamp:

$$(T_2 - T_1) \overset{?}{\leq} \tau \tag{8}$$

If the timestamp interval is valid, then *SCC* needs to verify *AP*'s hash parameter:

$$h_{AP_1} \overset{?}{=} H({R_{AP_1}}^{Sig_{AP_1}} \cdot {Puk_{AP}}^{H(M_{Access})+Sig_{AP_2}}||H(M_{Access})) \tag{9}$$

If Equation (9) holds, the *SCC* selects a random number r_2, computes the following parameters:

$$R_{SCC_1} = g^{r_2} \text{mod} p \tag{10}$$

SCC uses the hash function to compute the parameter h_{SCC_1}:

$$h_{SCC_1} = H({R_{SCC_1}}^{r_2} \cdot {Puk_{SCC}}^{(ID_{SCC}+Prk_{SCC})}) \tag{11}$$

SCC uses its private key Prk_{SCC} and ID_{SCC} to compute the signatures as follows:

$$Sig_{SCC_1} = 1 - {r_2}^{-1} \cdot Prk_{SCC} \text{mod} p \tag{12}$$

$$Sig_{SCC_2} = ID_{SCC} + Prk_{SCC} \text{mod} p \tag{13}$$

SCC generates $Token_1$, then *SCC* uses the *AP*'s public key Puk_{AP} to encrypt the message into cipher message C_{SCC_1}:

$$C_{SCC_1} = E_{Puk_{AP}}(ID_{SCC}||R_{SCC_1}||h_{SCC_1}||Sig_{SCC_1}||Sig_{SCC_2}||Token_1||T_3||Seq_{i+1}) \tag{14}$$

Then, *SCC* sends the cipher message to *AP*.

Step 3. The *AP* receives the *SCC*'s ID_{AP} and cipher message C_{SCC_1} at T_4, *AP* uses its private key Prk_{AP} to decrypt the messages:

$$(ID_{SCC}||R_{SCC_1}||h_{SCC_1}||Sig_{SCC_1}||Sig_{SCC_2}||Token_1||T_3) = D_{Prk_{AP}}(C_{SCC_1}) \tag{15}$$

Then, *AP* checks the validity of the timestamp:

$$(T_4 - T_3) \overset{?}{\leq} \tau \tag{16}$$

If the timestamp interval is valid, then *AP* needs to verify *SCC*'s hash parameter:

$$h_{SCC_1} \overset{?}{=} H({R_{SCC_1}}^{Sig_{SCC_1}} \cdot {Puk_{AP}}^{ID_{SCC}+Sig_{SCC_2}}) \tag{17}$$

If it is valid, *AP* gets the $Token_1$ that can access the information of *SCC* in other phases.

In the authentication phase, when any parties need to access the information of *SCC*, they must send their identity authentication $Token_1$ to the *SCC* before communication.

3.6. Event–Trigger Phase

In this phase, any IoT devices can send trigger signals to the *IMC* in the client's house. For example, the client installs some IoT devices (sensors) on doors and windows; when a thief enters the house silently and triggers the sensors, the sensors will send an event signal to the *IMC*. The event initialization structure and scenarios are presented in Figures 6 and 7, respectively.

```
struct Event{
        uint ID;
        uint Client_ID;
        uint IoT_ID;
        uint Trigger_Timestamp;
        uint Timestamp_Notified;
        uint Timestamp_Client_Received;
        uint Staff_ID;
        uint Timestamp_Officier_Received;
        uint Timestamp_Officier_SendJob;
        uint SecurityGuard_ID;
        uint Timestamp_SecurityGuard_Notified;
        uint Timestamp_SecurityGuard_Received;
        uint Timestamp_SecurityGuard_Departed;
        uint Timestamp_SecurityGuard_Arrived;
        uint Timestamp_SecurityGuard_Accomplished;
        string SecurityGuard_Remark;
        string Officier_Remark;
        uint Timestamp_Officier_Confirmed;
    }
```

Figure 6. Event initialization structure.

IoT device **(Cameras or Sensors)** | **IoT Main Controller (IMC)** **(Client's House)** | **Security Control Center (SCC)**

Event triggered

$\xrightarrow{(ID_{IoT}, T_5)}$

Generate ID_{Event}

choose a random number r_3

$R_{IMC_1} = g^{r_3} \bmod p$

$M_{Event_1} = (ID_{Event} \| ID_{IMC} \| ID_{IoT} \| T_5 \| T_6 \| Seq_i)$

$h_{IMC_1} = H(R_{IMC_1}{}^{r_3} \cdot Puk_{IMC}{}^{(M_{Event_1} + Prk_{IMC})} \| H(M_{Event_1}))$

$Sig_{IMC_1} = 1 - r_3^{-1} \cdot Prk_{IMC} \cdot H(M_{Event_1}) \bmod p$

$Sig_{IMC_2} = M_{Event_1} + Prk_{IMC} \bmod p$

$C_{IMC_1} = E_{Puk_{SCC}}(T_6 \| R_{IMC_1} \| h_{IMC_1} \| Sig_{IMC_1} \| Sig_{IMC_2} \| M_{Event_1} \| H(M_{Event_1}))$

$\xrightarrow{(ID_{IMC}, ID_{IoT}, C_{IMC_1})}$

$(T_6 \| R_{IMC_1} \| h_{IMC_1} \| Sig_{IMC_1} \| Sig_{IMC_2} \| M_{Event_1} \| H(M_{Event_1})) = D_{Prk_{SCC}}(C_{IMC_1})$

$(T_7 - T_6) \overset{?}{\le} \tau$

$h_{IMC_1} \overset{?}{=} H(R_{IMC_1}{}^{Sig_{IMC_1}} \cdot Puk_{IMC}{}^{H(M_{Event_1}) + Sig_{IMC_2}} \| H(M_{Event_1}))$

Send $(R_{IMC_1}, h_{IMC_1}, Sig_{IMC_1}, Sig_{IMC_2}, M_{Event_1})$ to BCC.

choose a random number r_4

$R_{SCC_2} = g^{r_4} \bmod p$

$M_{Event_2} = (ID_{Event} \| ID_{IMC} \| ID_{SCC} \| T_7 \| Seq_{i+1})$

$h_{SCC_2} = H(R_{SCC_2}{}^{r_4} \cdot Puk_{SCC}{}^{(M_{Event_2} + Prk_{SCC})} \| H(M_{Event_2}))$

$Sig_{SCC_3} = 1 - r_4^{-1} \cdot Prk_{SCC} \cdot H(M_{Event_2}) \bmod p$

$Sig_{SCC_4} = M_{Event_2} + Prk_{SCC} \bmod p$

Send $(R_{SCC_2}, h_{SCC_2}, Sig_{SCC_3}, Sig_{SCC_4}, M_{Event_2})$ to BCC.

Figure 7. The flowchart of the event-trigger phase.

Step 1. The IoT device sends a trigger signal with ID_{IoT} and the triggering time T_5 to the *IMC*.

Step 2. The *IMC* generates a unique event ID_{Event}, to bind the event to record and trace the event status.

Then, the *IMC* selects a random number r_3, and computes the following parameter:

$$R_{IMC_1} = g^{r_3} \mathrm{mod} p \tag{18}$$

The triggered *IoT*'s *ID* and *IMC*'s *ID* is written to the event message with a timestamp T_6:

$$M_{Event_1} = (ID_{Event} || ID_{IMC} || ID_{IoT} || T_5 || T_6 || Seq_i) \tag{19}$$

The *IMC* uses the hash function to compute the parameter h_{CH_1}:

$$h_{IMC_1} = H(R_{IMC_1}{}^{r_3} \cdot Puk_{IMC}{}^{(M_{Event} + Prk_{IMC})} || H(M_{Event_1})) \tag{20}$$

The *IMC* uses its private key Prk_{IMC} and M_{Event_1} to compute the signatures as follows:

$$Sig_{IMC_1} = 1 - r_3{}^{-1} \cdot Prk_{IMC} \cdot H(M_{Event_1}) \mathrm{mod} p \tag{21}$$

$$Sig_{IMC_2} = M_{Event_1} + Prk_{IMC} \mathrm{mod} p \tag{22}$$

After that, *IMC* uses the *SCC*'s public key Puk_{SCC} to encrypt the message into a ciphertext C_{CH_1}:

$$C_{IMC_1} = E_{Puk_{SCC}}(T_6 || R_{IMC_1} || h_{IMC_1} || Sig_{IMC_1} || Sig_{IMC_2} || M_{Event_1} || H(M_{Event_1})) \tag{23}$$

Then, *IMC* sends $(ID_{IMC}, ID_{IoT}, C_{IMC_1})$ to *SCC*.

Step 3. The *SCC* receives the *IMC*'s *ID*, *IoT*'s *ID* and cipher message C_{IMC_1} at T_7, *SCC* uses its private key Prk_{SCC} to decrypt the message:

$$(T_6 || R_{IMC_1} || h_{IMC_1} || Sig_{IMC_1} || Sig_{IMC_2} || M_{Event} || H(M_{Event})) = D_{Prk_{SCC}}(C_{IMC_1}) \tag{24}$$

Then, *SCC* checks the validity of the timestamp:

$$(T_7 - T_6) \overset{?}{\leq} \tau \tag{25}$$

If the timestamp interval is valid, then *SCC* needs to verify *IMC*'s hash parameter:

$$h_{IMC_1} \overset{?}{=} H(R_{IMC_1}{}^{Sig_{IMC_1}} \cdot Puk_{IMC}{}^{H(M_{Event_1}) + Sig_{IMC_2}} || H(M_{Event_1})) \tag{26}$$

If Equation (26) holds, *SCC* will check the situation of the client's house via cameras and sensors; if there is an available incident in the client's house, the next step will be to search for a nearby *SG* and construct the connection to send the event message to the selected *SG*.

In the meanwhile, *SCC* sends and chains $(R_{IMC_1}, h_{IMC_1}, Sig_{IMC_1}, Sig_{IMC_2}, M_{Event_1})$ to *BCC*.

Step 4. Then, *SCC* selects a random number r_4, and computes the following parameter:

$$R_{SCC_2} = g^{r_4} \mathrm{mod} p \tag{27}$$

The event's *ID*, *IMC*'s *ID*, and *SCC*'s staff *ID* are written to the event message with a timestamp T_7:

$$M_{Event_2} = (ID_{Event} || ID_{IMC} || ID_{SCC} || T_7 || Seq_{i+1}) \tag{28}$$

SCC uses the hash function to compute the parameter h_{SCC_2}:

$$h_{SCC_2} = H(R_{SCC_2}{}^{r_4} \cdot Puk_{SCC}{}^{(M_{Event_2} + Prk_{SCC})} || H(M_{Event_2})) \tag{29}$$

SCC uses its private key Prk_{SCC} and M_{Event_2} to compute the signatures as follows:

$$Sig_{SCC_3} = 1 - {r_4}^{-1} \cdot Prk_{SCC} \cdot H(M_{Event_2})\mathrm{mod}p \tag{30}$$

$$Sig_{SCC_4} = M_{Event_2} + Prk_{SCC}\mathrm{mod}p \tag{31}$$

Then, *SCC* sends and chains $(R_{SCC_2}, h_{SCC_2}, Sig_{SCC_3}, Sig_{SCC_4}, M_{Event_2})$ to *BCC*.

3.7. Task Allocating Phase

After receiving the event trigger message, the *SCC*'s staff will search for the nearest *SG* to the client's house. Then, a task will be allocated to the selected *SG* by the *SCC*'s staff. Once the *SG* receives a task from the *SCC*'s staff, the *SG* will need to reply to the *SCC*'s staff that the job was accepted or rejected. If the job is accepted, the *SG* should go to the client's house rapidly and solve the incident scene immediately. The *SG* also needs to report to the *SCC*'s staff in the task accomplished phase. The flowchart of the allocating task phase is illustrated in Figure 8.

Step 1. The *SCC* searches for an idle *SG* automatically by the nearest distance to the client's house and the *SG* gets allocated ID_{SG}.

Step 2. The *SCC* selects a random number r_5, and computes the following parameter:

$$R_{SCC_3} = g^{r_5}\mathrm{mod}p \tag{32}$$

The allocated ID_{SG} will be appended to the event message with a timestamp T_8:

$$M_{Event_3} = (ID_{IMC}||ID_{IoT}||ID_{Event}||ID_{SCC}||ID_{SG}||T_8||Seq_i) \tag{33}$$

The *SCC* uses the hash function to compute the parameter h_{SCC_2}:

$$h_{SCC_3} = H({R_{SCC_3}}^{r_5} \cdot {Puk_{SCC}}^{(M_{Event_3} + Prk_{SCC})}||H(M_{Event_3})) \tag{34}$$

The *SCC* uses its private key Prk_{SCC} and M_{Event_3} to compute the signatures as follows:

$$Sig_{SCC_5} = 1 - {r_5}^{-1} \cdot Prk_{SCC} \cdot H(M_{Event_3})\mathrm{mod}p \tag{35}$$

$$Sig_{SCC_6} = M_{Event_3} + Prk_{SCC}\mathrm{mod}p \tag{36}$$

Then, the *SCC* uses the *SG*'s public key Puk_{SG} to encrypt the message into a ciphertext C_{SCC_2}:

$$C_{SCC_2} = E_{Puk_{SG}}(T_8||R_{SCC_3}||h_{SCC_3}||Sig_{SCC_5}||Sig_{SCC_6}||M_{Event_3}||H(M_{Event_3})) \tag{37}$$

Then, the *SCC* sends and chains $(R_{SCC_3}, h_{SCC_3}, Sig_{SCC_5}, Sig_{SCC_6}, M_{Event_3})$ to the *BCC*. Meanwhile, the *SCC* sends the cipher message to the allocated *SG*.

Step 3. Once the *SG* receives the *SCC*'s *ID*, the *IMC*'s *ID*, the *SG*'s *ID*, and cipher message C_{SCC_2} at T_9, the SG uses the private key Prk_{SG} to decrypt the message:

$$(T_8||R_{SCC_3}||h_{SCC_3}||Sig_{SCC_5}||Sig_{SCC_6}||M_{Event_3}||H(M_{Event_3})) = D_{Prk_{SG}}(C_{SCC_2}) \tag{38}$$

Then, the *SG* checks the validity of the timestamp:

$$(T_9 - T_8) \overset{?}{\leq} \tau \tag{39}$$

If the timestamp interval is valid, then the *SG* needs to verify the *SCC*'s hash parameter:

$$h_{SCC_2} \overset{?}{=} H({R_{SCC_3}}^{Sig_{SCC_5}} \cdot {Puk_{SCC}}^{H(M_{Event_3}) + Sig_{SCC_6}}||H(M_{Event_3})) \tag{40}$$

If Equation (40) holds, then the *SG* needs to reply to the *SCC* that they are accepting the task. If the *SG* does not accept the task, then they reply no, and otherwise reply yes. The *SG* selects a random number r_6, and computes the following parameter:

$$R_{SG_1} = g^{r_6} \bmod p \tag{41}$$

A timestamp T_{10} is appended to the event message:

$$M_{Event_4} = (ID_{Event}||ID_{SG}||ID_{SCC}||T_{10}||Seq_{i+1}) \tag{42}$$

SG uses the hash function to compute the parameter h_{SG_1}:

$$h_{SG_1} = H(R_{SG_1}{}^{r_6} \cdot Puk_{SG}{}^{(M_{Event_4}+Prk_{SG})}||H(M_{Event_4})) \tag{43}$$

SG uses the private key Prk_{SG} and M_{Event_4} to compute the signatures as follows:

$$Sig_{SG_1} = 1 - r_6{}^{-1} \cdot Prk_{SG} \cdot H(M_{Event_4}) \bmod p \tag{44}$$

$$Sig_{SG_2} = M_{Event_4} + Prk_{SG} \bmod p \tag{45}$$

Then the *SG* uses the *SCC*'s public key Puk_{SCC} to encrypt the message into cipher message C_{SG_1}:

$$C_{SG_1} = E_{Puk_{SCC}}(T_{10}||R_{SG_1}||h_{SG_1}||Sig_{SG_1}||Sig_{SG_2}||M_{Event_4}||H(M_{Event_4})||(Y/N)) \tag{46}$$

Then, the *SG* responds to the cipher message to the *SCC*. In the message includes Y/N, "Y" means to accept the task, and "N" means to reject the task. If the *SG* accepts the task, then the *SG* must go to the incident scene immediately and solve the security problem.

Step 4. Once the *SCC* receives the cipher message C_{SG_1} at T_{11}, the *SCC* uses the private key Prk_{SCC} to decrypt the message:

$$(T_{10}||R_{SG_1}||h_{SG_1}||Sig_{SG_1}||Sig_{SG_2}||M_{Event_4}||H(M_{Event_4})||(Y/N)) = D_{Prk_{SCC}}(C_{SG_1}) \tag{47}$$

Then, the *SCC* checks the validity of the timestamp:

$$(T_{11} - T_{10}) \overset{?}{\leq} \tau \tag{48}$$

If the timestamp interval is valid, then the *SCC* needs to verify the *SG*'s hash parameter:

$$h_{SG_1} \overset{?}{=} H(R_{SG_1}{}^{Sig_{SG_1}} \cdot Puk_{SG}{}^{H(M_{Event_4})+Sig_{SG_2}}||H(M_{Event_4})) \tag{49}$$

If Equation (49) holds, the *SCC* sends and chains $(R_{SG_1}, h_{SG_1}, Sig_{SG_1}, Sig_{SG_2}, M_{Event_4})$ to the *BCC*.

Then the *SCC* needs to check the reply message; if the task is not accepted, then it repeats from the first step and searches the *SG* again. If the *SG* accepts it, then it goes to the next step.

Step 5. Then, the *SCC* selects a random number r_7, and computes the following parameter:

$$R_{SCC_4} = g^{r_7} \bmod p \tag{50}$$

The *event ID*, *SG*'s *ID*, and *SCC*'s staff *ID* are written to the event message with a timestamp T_{12}:

$$M_{Event_5} = (ID_{Event}||ID_{SG}||ID_{SCC}||T_{12}||Seq_{i+2}) \tag{51}$$

The *SCC* uses the hash function to compute the parameter h_{SCC_4}:

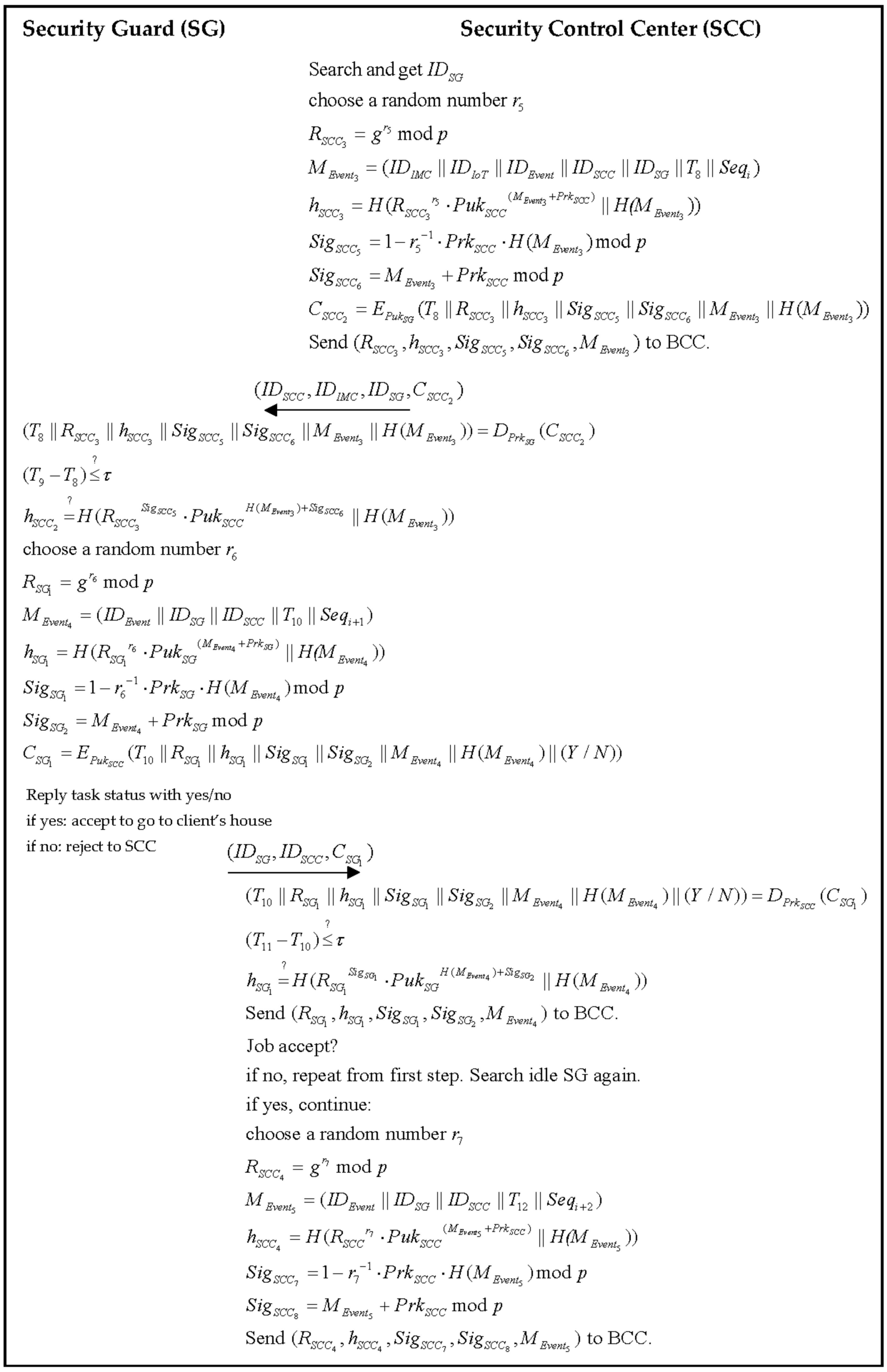
$$h_{SCC_4} = H(R_{SCC_4}{}^{r_7} \cdot Puk_{SCC}{}^{(M_{Event_5}+Prk_{SCC})}||H(M_{Event_5})) \tag{52}$$

The *SCC* uses its private key Prk_{SCC} and M_{Event_5} to compute the signatures as follows:

$$Sig_{SCC_7} = 1 - r_7{}^{-1} \cdot Prk_{SCC} \cdot H(M_{Event_5}) \mathrm{mod} p \tag{53}$$

$$Sig_{SCC_8} = M_{Event_5} + Prk_{SCC} \mathrm{mod} p \tag{54}$$

Then, the *SCC* sends and chains $(R_{SCC_4}, h_{SCC_4}, Sig_{SCC_7}, Sig_{SCC_8}, M_{Event_5})$ to the *BCC*.



Security Guard (SG) **Security Control Center (SCC)**

Search and get ID_{SG}

choose a random number r_5

$R_{SCC_3} = g^{r_5} \bmod p$

$M_{Event_3} = (ID_{IMC} \| ID_{IoT} \| ID_{Event} \| ID_{SCC} \| ID_{SG} \| T_8 \| Seq_i)$

$h_{SCC_3} = H(R_{SCC_3}{}^{r_5} \cdot Puk_{SCC}{}^{(M_{Event_3}+Prk_{SCC})} \| H(M_{Event_3}))$

$Sig_{SCC_5} = 1 - r_5^{-1} \cdot Prk_{SCC} \cdot H(M_{Event_3}) \bmod p$

$Sig_{SCC_6} = M_{Event_3} + Prk_{SCC} \bmod p$

$C_{SCC_2} = E_{Puk_{SG}}(T_8 \| R_{SCC_3} \| h_{SCC_3} \| Sig_{SCC_5} \| Sig_{SCC_6} \| M_{Event_3} \| H(M_{Event_3}))$

Send $(R_{SCC_3}, h_{SCC_3}, Sig_{SCC_5}, Sig_{SCC_6}, M_{Event_3})$ to BCC.

$\xleftarrow{(ID_{SCC}, ID_{IMC}, ID_{SG}, C_{SCC_2})}$

$(T_8 \| R_{SCC_3} \| h_{SCC_3} \| Sig_{SCC_5} \| Sig_{SCC_6} \| M_{Event_3} \| H(M_{Event_3})) = D_{Prk_{SG}}(C_{SCC_2})$

$(T_9 - T_8) \overset{?}{\leq} \tau$

$h_{SCC_2} \overset{?}{=} H(R_{SCC_3}{}^{Sig_{SCC_5}} \cdot Puk_{SCC}{}^{H(M_{Event_3})+Sig_{SCC_6}} \| H(M_{Event_3}))$

choose a random number r_6

$R_{SG_1} = g^{r_6} \bmod p$

$M_{Event_4} = (ID_{Event} \| ID_{SG} \| ID_{SCC} \| T_{10} \| Seq_{i+1})$

$h_{SG_1} = H(R_{SG_1}{}^{r_6} \cdot Puk_{SG}{}^{(M_{Event_4}+Prk_{SG})} \| H(M_{Event_4}))$

$Sig_{SG_1} = 1 - r_6^{-1} \cdot Prk_{SG} \cdot H(M_{Event_4}) \bmod p$

$Sig_{SG_2} = M_{Event_4} + Prk_{SG} \bmod p$

$C_{SG_1} = E_{Puk_{SCC}}(T_{10} \| R_{SG_1} \| h_{SG_1} \| Sig_{SG_1} \| Sig_{SG_2} \| M_{Event_4} \| H(M_{Event_4}) \| (Y/N))$

Reply task status with yes/no

if yes: accept to go to client's house

if no: reject to SCC

$\xrightarrow{(ID_{SG}, ID_{SCC}, C_{SG_1})}$

$(T_{10} \| R_{SG_1} \| h_{SG_1} \| Sig_{SG_1} \| Sig_{SG_2} \| M_{Event_4} \| H(M_{Event_4}) \| (Y/N)) = D_{Prk_{SCC}}(C_{SG_1})$

$(T_{11} - T_{10}) \overset{?}{\leq} \tau$

$h_{SG_1} \overset{?}{=} H(R_{SG_1}{}^{Sig_{SG_1}} \cdot Puk_{SG}{}^{H(M_{Event_4})+Sig_{SG_2}} \| H(M_{Event_4}))$

Send $(R_{SG_1}, h_{SG_1}, Sig_{SG_1}, Sig_{SG_2}, M_{Event_4})$ to BCC.

Job accept?

if no, repeat from first step. Search idle SG again.

if yes, continue:

choose a random number r_7

$R_{SCC_4} = g^{r_7} \bmod p$

$M_{Event_5} = (ID_{Event} \| ID_{SG} \| ID_{SCC} \| T_{12} \| Seq_{i+2})$

$h_{SCC_4} = H(R_{SCC}{}^{r_7} \cdot Puk_{SCC}{}^{(M_{Event_5}+Prk_{SCC})} \| H(M_{Event_5}))$

$Sig_{SCC_7} = 1 - r_7^{-1} \cdot Prk_{SCC} \cdot H(M_{Event_5}) \bmod p$

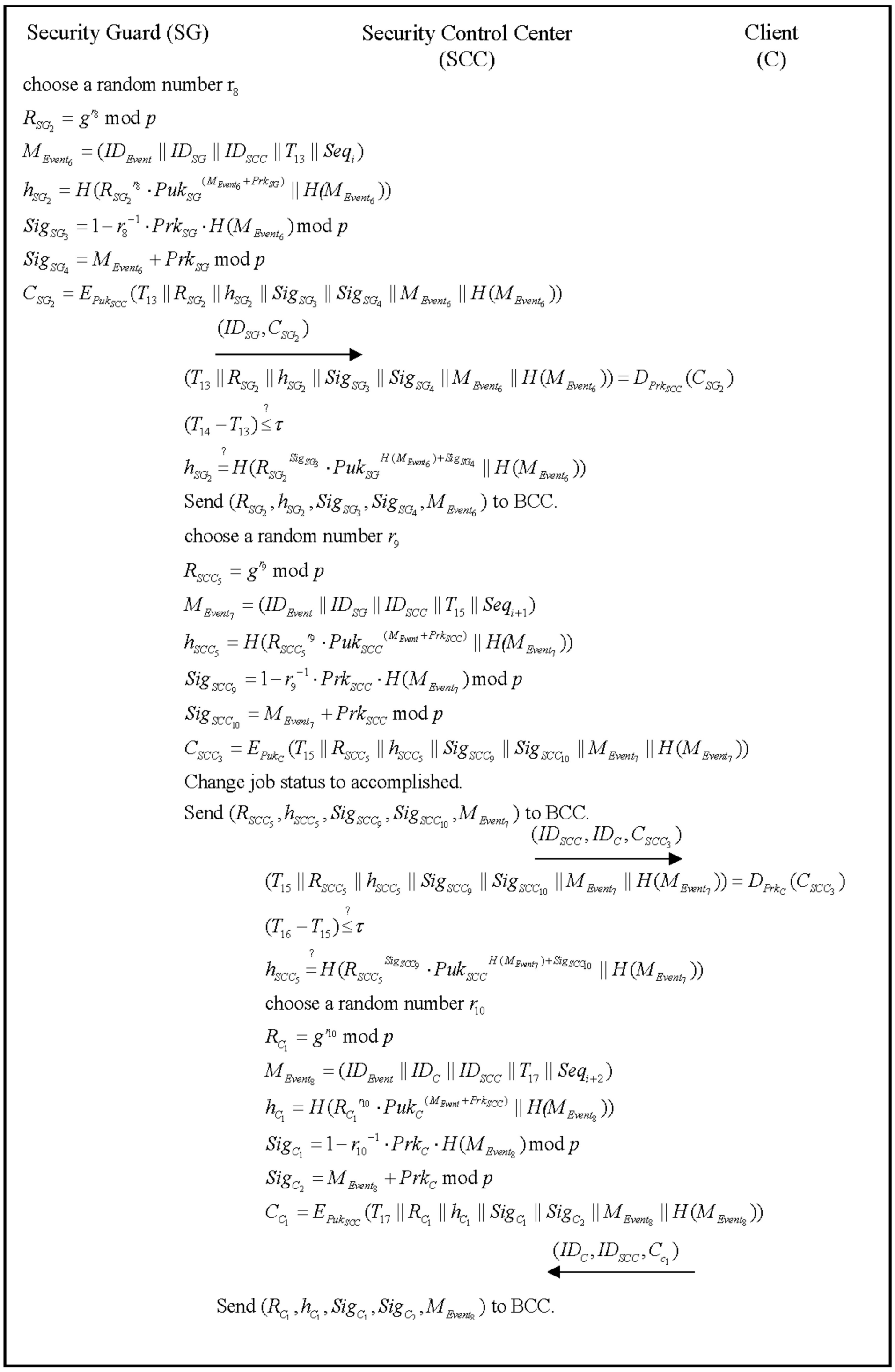
$Sig_{SCC_8} = M_{Event_5} + Prk_{SCC} \bmod p$

Send $(R_{SCC_4}, h_{SCC_4}, Sig_{SCC_7}, Sig_{SCC_8}, M_{Event_5})$ to BCC.

Figure 8. The flowchart of the allocating task phase.

3.8. Task Accomplished Phase

After the task is accomplished by the *SG*, the *SG* needs to report the event status to the *SCC*. Then, the *SCC* will send a notification including the detailed process and remark automatically to the client. Figure 9 shows the flowchart of the task accomplished phase.



Security Guard (SG) | Security Control Center (SCC) | Client (C)

choose a random number r_8

$R_{SG_2} = g^{r_8} \bmod p$

$M_{Event_6} = (ID_{Event} \| ID_{SG} \| ID_{SCC} \| T_{13} \| Seq_i)$

$h_{SG_2} = H(R_{SG_2}{}^{r_8} \cdot Puk_{SG}{}^{(M_{Event_6} + Prk_{SG})} \| H(M_{Event_6}))$

$Sig_{SG_3} = 1 - r_8^{-1} \cdot Prk_{SG} \cdot H(M_{Event_6}) \bmod p$

$Sig_{SG_4} = M_{Event_6} + Prk_{SG} \bmod p$

$C_{SG_2} = E_{Puk_{SCC}}(T_{13} \| R_{SG_2} \| h_{SG_2} \| Sig_{SG_3} \| Sig_{SG_4} \| M_{Event_6} \| H(M_{Event_6}))$

$\xrightarrow{(ID_{SG}, C_{SG_2})}$

$(T_{13} \| R_{SG_2} \| h_{SG_2} \| Sig_{SG_3} \| Sig_{SG_4} \| M_{Event_6} \| H(M_{Event_6})) = D_{Prk_{SCC}}(C_{SG_2})$

$(T_{14} - T_{13}) \overset{?}{\le} \tau$

$h_{SG_2} \overset{?}{=} H(R_{SG_2}{}^{Sig_{SG_3}} \cdot Puk_{SG}{}^{H(M_{Event_6}) + Sig_{SG_4}} \| H(M_{Event_6}))$

Send $(R_{SG_2}, h_{SG_2}, Sig_{SG_3}, Sig_{SG_4}, M_{Event_6})$ to BCC.

choose a random number r_9

$R_{SCC_5} = g^{r_9} \bmod p$

$M_{Event_7} = (ID_{Event} \| ID_{SG} \| ID_{SCC} \| T_{15} \| Seq_{i+1})$

$h_{SCC_5} = H(R_{SCC_5}{}^{r_9} \cdot Puk_{SCC}{}^{(M_{Event} + Prk_{SCC})} \| H(M_{Event_7}))$

$Sig_{SCC_9} = 1 - r_9^{-1} \cdot Prk_{SCC} \cdot H(M_{Event_7}) \bmod p$

$Sig_{SCC_{10}} = M_{Event_7} + Prk_{SCC} \bmod p$

$C_{SCC_3} = E_{Puk_C}(T_{15} \| R_{SCC_5} \| h_{SCC_5} \| Sig_{SCC_9} \| Sig_{SCC_{10}} \| M_{Event_7} \| H(M_{Event_7}))$

Change job status to accomplished.

Send $(R_{SCC_5}, h_{SCC_5}, Sig_{SCC_9}, Sig_{SCC_{10}}, M_{Event_7})$ to BCC.

$\xrightarrow{(ID_{SCC}, ID_C, C_{SCC_3})}$

$(T_{15} \| R_{SCC_5} \| h_{SCC_5} \| Sig_{SCC_9} \| Sig_{SCC_{10}} \| M_{Event_7} \| H(M_{Event_7})) = D_{Prk_C}(C_{SCC_3})$

$(T_{16} - T_{15}) \overset{?}{\le} \tau$

$h_{SCC_5} \overset{?}{=} H(R_{SCC_5}{}^{Sig_{SCC_9}} \cdot Puk_{SCC}{}^{H(M_{Event_7}) + Sig_{SCC_{10}}} \| H(M_{Event_7}))$

choose a random number r_{10}

$R_{C_1} = g^{r_{10}} \bmod p$

$M_{Event_8} = (ID_{Event} \| ID_C \| ID_{SCC} \| T_{17} \| Seq_{i+2})$

$h_{C_1} = H(R_{C_1}{}^{r_{10}} \cdot Puk_C{}^{(M_{Event} + Prk_{SCC})} \| H(M_{Event_8}))$

$Sig_{C_1} = 1 - r_{10}^{-1} \cdot Prk_C \cdot H(M_{Event_8}) \bmod p$

$Sig_{C_2} = M_{Event_8} + Prk_C \bmod p$

$C_{C_1} = E_{Puk_{SCC}}(T_{17} \| R_{C_1} \| h_{C_1} \| Sig_{C_1} \| Sig_{C_2} \| M_{Event_8} \| H(M_{Event_8}))$

$\xleftarrow{(ID_C, ID_{SCC}, C_{c_1})}$

Send $(R_{C_1}, h_{C_1}, Sig_{C_1}, Sig_{C_2}, M_{Event_8})$ to BCC.

Figure 9. The flowchart of the task accomplished phase.

Step 1. The *SG* selects a random r_8, and computes the following parameter:

$$R_{SG_2} = g^{r_8} \bmod p \tag{55}$$

The timestamp T_{13} will be appended to the event message:

$$M_{Event_6} = (ID_{Event}||ID_{SG}||ID_{SCC}||T_{13}||Seq_i) \tag{56}$$

The *SG* uses the hash function to compute the parameter h_{SG_2}:

$$h_{SG_2} = H({R_{SG_2}}^{r_8} \cdot {Puk_{SG}}^{(M_{Event_6}+Prk_{SG})}||H(M_{Event_6})) \tag{57}$$

The *SG* uses its private key Prk_{SG} and M_{Event_6} to compute the signatures as follows:

$$Sig_{SG_3} = 1 - {r_8}^{-1} \cdot Prk_{SG} \cdot H(M_{Event_6})\text{mod}p \tag{58}$$

$$Sig_{SG_4} = M_{Event_6} + Prk_{SG}\text{mod}p \tag{59}$$

After that, the *SG* uses *SCC*'s public key Puk_{SCC} to encrypt the message into a ciphertext C_{SG_2}:

$$C_{SG_2} = E_{Puk_{SCC}}(T_{13}||R_{SG_2}||h_{SG_2}||Sig_{SG_3}||Sig_{SG_4}||M_{Event_6}||H(M_{Event_6})) \tag{60}$$

Then, the *SG* sends $(ID_{SG}, ID_{event}, C_{SG_2})$ to *SCC*.

Step 2. Once the *SCC* receives the cipher message C_{SG_2} at T_{14}, the *SCC* uses its private key Prk_{SCC} to decrypt the message:

$$(T_{13}||R_{SG_2}||h_{SG_2}||Sig_{SG_3}||Sig_{SG_4}||M_{Event_6}||H(M_{Event_6})) = D_{Prk_{SCC}}(C_{SG_2}) \tag{61}$$

Then, the *SCC* checks the validity of the timestamp:

$$(T_{14} - T_{13}) \overset{?}{\leq} \tau \tag{62}$$

If the timestamp interval is valid, then the *SCC* needs to verify the *SG*'s hash parameter:

$$h_{SG_2} \overset{?}{=} H({R_{SG_2}}^{Sig_{SG_3}} \cdot {Puk_{SG}}^{H(M_{Event_6})+Sig_{SG_4}}||H(M_{Event_6})) \tag{63}$$

If Equation (63) holds, the event status will be changed to "accomplished". Then, the *SCC* sends and chains $(R_{SG_2}, h_{SG_2}, Sig_{SG_3}, Sig_{SG_4}, M_{Event_6})$ to the *BCC*.

Step 3. Next, a notification will be sent to the client to let the client know the current state of the security situation of the *CH*. The *SCC* selects a random r_9, and computes the following parameter:

$$R_{SCC_5} = g^{r_9}\text{mod}p \tag{64}$$

The timestamp T_{15} will be appended to the event message:

$$M_{Event_7} = (ID_{Event}||ID_{SG}||ID_{SCC}||T_{15}||Seq_{i+1}) \tag{65}$$

The *SCC* uses the hash function to compute the parameter h_{SCC_3}:

$$h_{SCC_5} = H({R_{SCC_5}}^{r_9} \cdot {Puk_{SCC}}^{(M_{Event}+Prk_{SCC})}||H(M_{Event_7})) \tag{66}$$

The *SCC* uses its private key Prk_{SCC} and M_{Event} to compute the signatures as follows:

$$Sig_{SCC_9} = 1 - {r_9}^{-1} \cdot Prk_{SCC} \cdot H(M_{Event_7})\text{mod}p \tag{67}$$

$$Sig_{SCC_{10}} = M_{Event_7} + Prk_{SCC}\text{mod}p \tag{68}$$

After that, the *SCC* uses the client's public key Puk_C to encrypt the message into a ciphertext C_{SCC_3}:

$$C_{SCC_3} = E_{Puk_C}(T_{15}||R_{SCC_5}||h_{SCC_5}||Sig_{SCC_9}||Sig_{SCC_{10}}||M_{Event_7}||H(M_{Event_7})) \tag{69}$$

Then, the *SCC* sends and chains $(R_{SCC_5}, h_{SCC_5}, Sig_{SCC_9}, Sig_{SCC_{10}}, M_{Event_7})$ to the *BCC*.

Meanwhile, the *SCC* sends $(ID_{SCC}, ID_C, C_{SCC_3})$ to the client, and a notification will be sent to the client to let the client know the current state of the security situation of the *CH*.

Step 4. Once the client receives the cipher message C_{SCC_3} at T_{16}, the client uses its private key Prk_C to decrypt the message:

$$(T_{15}||R_{SCC_5}||h_{SCC_5}||Sig_{SCC_9}||Sig_{SCC_{10}}||M_{Event_7}||H(M_{Event_7})) = D_{Prk_C}(C_{SCC_3}) \tag{70}$$

Then, the client checks the validity of the timestamp:

$$(T_{16} - T_{15}) \overset{?}{\leq} \tau \tag{71}$$

If the timestamp interval is valid, then the client needs to verify *SCC*'s hash parameter:

$$h_{SCC_5} \overset{?}{=} H({R_{SCC_5}}^{Sig_{SCC_9}} \cdot {Puk_{SCC}}^{H(M_{Event_7})+Sig_{SCC_{10}}}||H(M_{Event_7})) \tag{72}$$

If Equation (72) holds, then the client will reply to the *SCC* that the notification is received.

Step 5. The client selects a random r_{10}, and computes the following parameter:

$$R_{C_1} = g^{r_{10}} \bmod p \tag{73}$$

The timestamp T_{17} will be appended to the event message:

$$M_{Event_8} = (ID_{Event}||ID_C||ID_{SCC}||T_{17}||Seq_{i+2}) \tag{74}$$

The client uses the hash function to compute the parameter h_{C_1}:

$$h_{C_1} = H({R_{C_1}}^{r_{10}} \cdot {Puk_C}^{(M_{Event}+Prk_{SCC})}||H(M_{Event_8})) \tag{75}$$

The client uses the private key Prk_C and M_{Event_8} to compute the signatures as follows:

$$Sig_{C_1} = 1 - {r_{10}}^{-1} \cdot Prk_C \cdot H(M_{Event_8}) \bmod p \tag{76}$$

$$Sig_{C_2} = M_{Event_8} + Prk_C \bmod p \tag{77}$$

After that, the client uses the *SCC*'s public key Puk_{SCC} to encrypt the message into a ciphertext C_{C_1}:

$$C_{C_1} = E_{Puk_{SCC}}(T_{17}||R_{C_1}||h_{C_1}||Sig_{C_1}||Sig_{C_2}||M_{Event_8}||H(M_{Event_8})) \tag{78}$$

Then, the *C* sends $(ID_C, ID_{SCC}, C_{C_1})$ to the *SCC*.

Step 6. Once receiving the cipher message C_{C_1} at T_{18}, the *SCC* uses its private key Prk_{SCC} to decrypt the message:

$$(T_{17}||R_{C_1}||h_{C_1}||Sig_{C_1}||Sig_{C_2}||M_{Event_8}||H(M_{Event_8})) = D_{Prk_C}(C_{C_1}) \tag{79}$$

Then, the *SCC* checks the validity of the timestamp:

$$(T_{18} - T_{17}) \overset{?}{\leq} \tau \tag{80}$$

If the timestamp interval is valid, then the client needs to verify the client's hash parameter:

$$h_{C_1} \overset{?}{=} H({R_{C_1}}^{Sig_{C_1}} \cdot {Puk_C}^{H(M_{Event_8})+Sig_{C_2}}||H(M_{Event_8})) \tag{81}$$

If Equation (81) holds, the latest message will be sent and chained $(R_{C_1}, h_{C_1}, Sig_{C_1}, Sig_{C_2}, M_{Event_8})$ to the *BCC*.

Step 7. Then, the *SCC* selects a random number r_{11}, and computes the following parameter:

$$R_{SCC_6} = g^{r_{11}} \bmod p \tag{82}$$

The *event ID*, client's *ID*, and *SCC*'s staff *ID* will be written to the event message with a timestamp T_{19}:

$$M_{Event_9} = (ID_{Event}||ID_C||ID_{SCC}||T_{19}||Seq_{i+3}) \tag{83}$$

The *SCC* uses the hash function to compute the parameter h_{SCC_6}:

$$h_{SCC_6} = H({R_{SCC_6}}^{r_{11}} \cdot Puk_{SCC}{}^{(M_{Event_9}+Prk_{SCC})}||H(M_{Event_9})) \tag{84}$$

The *SCC* uses its private key Prk_{SCC} and M_{Event_9} to compute the signatures as follows:

$$Sig_{SCC_{11}} = 1 - {r_{11}}^{-1} \cdot Prk_{SCC} \cdot H(M_{Event_9}) \bmod p \tag{85}$$

$$Sig_{SCC_{12}} = M_{Event_9} + Prk_{SCC} \bmod p \tag{86}$$

Then, the *SCC* sends and chains $(R_{SCC_6}, h_{SCC_6}, Sig_{SCC_{11}}, Sig_{SCC_{12}}, M_{Event_9})$ to the *BCC*.

3.9. Arbitration Phase

In this phase, if the client doubts the legality of a security event, the client needs to provide his/her *ID*, *IMC ID*, and *event ID* to the official agency (*OA*) to verify the validity of the event. The flowchart of the arbitration phase is shown in Figure 10.

The detailed steps of this phase are as follows:

Step 1. *AP* send $(ID_{AP}, ID_{IMC}, ID_{Event}, Sig_{AP_3}, Sig_{AP_4})$ to the *OA*.

Step 2. The *OA* sends a request message with $(ID_{AP}, ID_{IMC}, ID_{Event}, ID_{OA}, Sig_{AP_3}, Sig_{AP_4}, Sig_{OA_1}, Sig_{OA_2})$.

Step 3. The *BCC* checks the signature from the *OA*; if it is valid, the *BCC* sends the validation data of the specific *event ID* to the *OA*. In every phase, the *SCC* needs to send the validation data to the *BCC*, and let the *OA* verify whether the event is legal or not. Every phase of the validation data is shown as follows:

Event trigger phase:

$$(R_{IMC_1}, h_{IMC_1}, Sig_{IMC_1}, Sig_{IMC_2}, M_{Event_1}) \tag{87}$$

$$(R_{SCC_2}, h_{SCC_2}, Sig_{SCC_3}, Sig_{SCC_4}, M_{Event_2}) \tag{88}$$

Task allocating phase:

$$(R_{SCC_3}, h_{SCC_3}, Sig_{SCC_5}, Sig_{SCC_6}, M_{Event_3}) \tag{89}$$

$$(R_{SG_1}, h_{SG_1}, Sig_{SG_1}, Sig_{SG_2}, M_{Event_4}) \tag{90}$$

$$(R_{SCC_4}, h_{SCC_4}, Sig_{SCC_7}, Sig_{SCC_8}, M_{Event_5}) \tag{91}$$

Task accomplished phase:

$$(R_{SG_2}, h_{SG_2}, Sig_{SG_3}, Sig_{SG_4}, M_{Event_6}) \tag{92}$$

$$(R_{SCC_5}, h_{SCC_5}, Sig_{SCC_9}, Sig_{SCC_{10}}, M_{Event_7}) \tag{93}$$

$$(R_{C_1}, h_{C_1}, Sig_{C_1}, Sig_{C_2}, M_{Event_8}) \tag{94}$$

$$(R_{SCC_6}, h_{SCC_6}, Sig_{SCC_{11}}, Sig_{SCC_{12}}, M_{Event_9}) \tag{95}$$

Step 4. The *OA* verifies the legality of the hash parameter h_{C_1}:

$$h_{C_1} \overset{?}{=} H({R_{C_1}}^{Sig_{C_1}} \cdot {Puk_C}^{Sig_{C_2}}||H(M_{Event_8})) \tag{96}$$

If the equation holds, we go to the next step.

Otherwise, the event is illegal (hash value is not equal to the right, which means that the signature is not valid) when the *C* reads the event notification, so the *C* is illegal if the customer does not read the notification.

Step 5. The *OA* verifies the legality of the hash parameter h_{SCC_5}:

$$h_{SCC_5} \overset{?}{=} H({R_{SCC_5}}^{Sig_{SCC_9}} \cdot {Puk_{SCC}}^{Sig_{SCC_{10}}} || H(M_{Event_7})) \tag{97}$$

If the equation holds, go to the next step.

Otherwise, the event is illegal when the *SCC* sends a notification to the *C*, so the *SCC* is illegal.

Step 6. The *OA* verifies the legality of the hash parameter h_{SG_2}:

$$h_{SG_2} \overset{?}{=} H({R_{SG_2}}^{Sig_{SG_3}} \cdot {Puk_{SG}}^{Sig_{SG_4}} || H(M_{Event_6})) \tag{98}$$

If the equation holds, we go to the next step.

Otherwise, the event is illegal when the *SG* reports the tasks accomplished to the *SCC*, so the *SG* is illegal.

Step 7. The *OA* verifies the legality of the hash parameter h_{SCC_4}:

$$h_{SCC_4} \overset{?}{=} H({R_{SCC_4}}^{Sig_{SCC_7}} \cdot {Puk_{SCC}}^{H(M_{Event_5})+Sig_{SCC_8}} || H(M_{Event_5})) \tag{99}$$

If the equation holds, we go to the next step.

Otherwise, the event is illegal when the *SCC* receives the accepted task message, so the *SCC* is illegal.

Step 8. The *OA* verifies the legality of the hash parameter h_{SG_1}:

$$h_{SG_1} \overset{?}{=} H({R_{SG_1}}^{Sig_{SG_1}} \cdot {Puk_{SG}}^{H(M_{Event_4})+Sig_{SG_2}} || H(M_{Event_4})) \tag{100}$$

If the equation holds, we go to the next step.

Otherwise, the event is illegal when the *SG* accepts the task and replies to the *SCC*, so the *SG* is illegal.

Step 9. The *OA* verifies the legality of the hash parameter h_{SCC_2}:

$$h_{SCC_2} \overset{?}{=} H({R_{SCC_3}}^{Sig_{SCC_5}} \cdot {Puk_{SCC}}^{H(M_{Event_3})+Sig_{SCC_6}} || H(M_{Event_3})) \tag{101}$$

If the equation holds, we go to the next step.

Otherwise, the event is illegal when the *SCC* allocates the task to the *SG*, so the *SCC* is illegal.

Step 10. The *OA* verifies the legality of the hash parameter h_{SCC_1}:

$$h_{SCC_1} \overset{?}{=} H({R_{SCC_2}}^{Sig_{SCC_3}} \cdot {Puk_{SCC}}^{H(M_{Event_1})+{Sig_{SCC_4}}} || H(M_{Event_2})) \tag{102}$$

If the equation holds, we go to the next step.

Otherwise, the event is illegal when the *SCC* receives the message from the *IMC*, so the *SCC* is illegal.

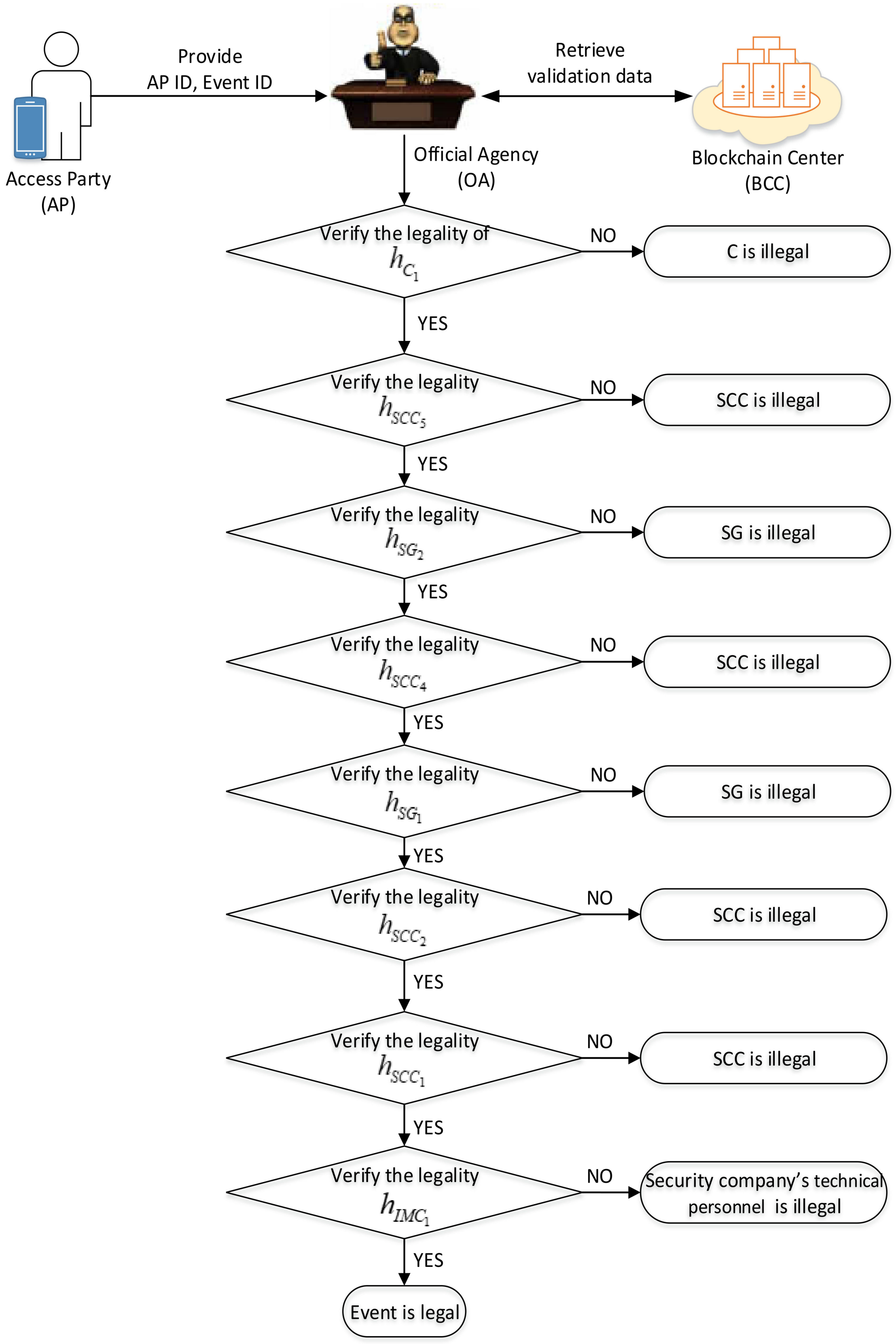
Step 11. The *OA* verifies the legality of the hash parameter h_{IMC_1}:

$$h_{IMC_1} \overset{?}{=} H({R_{IMC_1}}^{Sig_{IMC_1}} \cdot {Puk_{IMC}}^{H(M_{Event_1})+{Sig_{IMC_2}}} || H(M_{Event_1})) \tag{103}$$

If the equation holds, we go to the next step.

Otherwise, the event is illegal when the *IMC* in the client's house sends an event trigger message to the *SCC*, so the security company's technical personnel is illegal.

Step 12. The *OA* judges that the event is legal, and sends the judgment to *AP*.



Provide
AP ID, Event ID
Retrieve
validation data
Access Party
(AP)
Official Agency
(OA)
Blockchain Center
(BCC)
Verify the legality of h_{C_1}
NO
C is illegal
YES
Verify the legality h_{SCC_5}
SCC is illegal
Verify the legality h_{SG_2}
SG is illegal
Verify the legality h_{SCC_4}
SCC is illegal
Verify the legality h_{SG_1}
SG is illegal
Verify the legality h_{SCC_2}
SCC is illegal
Verify the legality h_{SCC_1}
SCC is illegal
Verify the legality h_{IMC_1}
Security company's technical personnel is illegal
Event is legal

Figure 10. The flowchart of the arbitration phase.

3.10. Key Recovery Phase

Access parties (including the client, security guard, and staff) need to register with the *SCC*, and the public key Puk_X and private key Prk_X will be issued to the parties. If the parties lose their private key, they can send a key recovery request to the *SCC*; the scenario is as follows:

Step 1. The access party sends a key recovery request to the *SCC*, and the request message includes *ID* and details information.
Step 2. The *SCC* receives the request, then checks the consistency of the information from the request message and smart contract automatically.
Step 3. If the information from the accessing party is equal to the smart contract, then they re-register a new key pair from the *BCC* and update the new key pairs to the corresponding party's information structure.
Step 4. The *SCC* responds with a new key pair to the accessing party.
Step 5. The access party receives the new key pair and the key recovery is done.

4. Security Analysis

4.1. Mutual Authentication

In this research, BAN logic is used to prove mutual authentication in the authentication algorithm [31], mainly to ensure that data is not modified during the phases.

The notation of BAN logic is described as below:

$P\mid\equiv X$	*P* believes *X*.
$P \lhd X$	*P* sees *X*.
$P\mid\sim X$	*P* once said *X*.
$P\mid\Rightarrow X$	*P* has jurisdiction over *X*.
$\#(X)$	Formula *X* is new.
$P\mid\overset{K}{\rightarrow} X$	*P* has *X* as a public key.
$P\overset{K}{\leftrightarrow} Q$	*P* and *X* may use the session key *K* to communicate with each other.
$\{X\}_K$	The formula *X* is encrypted by *K*.

In the initializing phase, the scheme makes sure that the legality is authenticated by the accessing party *(AP)* and the *SCC*. The main goals of the scheme are:

G1: $AP\mid\equiv AP \overset{SK_{AP-BCC}}{\leftrightarrow} SCC$
G2: $AP\mid\equiv SCC\mid\equiv AP \overset{SK_{AP-BCC}}{\leftrightarrow} SCC$
G3: $SCC\mid\equiv AP \overset{SK_{AP-BCC}}{\leftrightarrow} SCC$
G4: $SCC\mid\equiv AP\mid\equiv AP \overset{SK_{AP-BCC}}{\leftrightarrow} SCC$
G5: $AP\mid\equiv ID_{SCC}$
G6: $AP\mid\equiv SCC\mid\equiv ID_{SCC}$
G7: $SCC\mid\equiv ID_{AP}$
G8: $SCC\mid\equiv AP\mid\equiv ID_{AP}$

According to the authenticate algorithm, BAN logic is used to produce an idealized form as follows:

M1: $AP\rightarrow SCC(\{ID_{AP}, Pwd_{AP}, R_{AP}, h_{AP_1}, M_{Access}, T_1\}_{Puk_{SCC}})$
M2: $SCC\rightarrow AP(\{ID_{SCC}, R_{SCC}, h_{SCC_1}, Token_1, T_3\}_{Puk_{AP}})$

To analyze the proposed scheme, the following assumptions are made:

A1: $AP\mid\equiv\#(R_{AP})$
A2: $SCC\mid\equiv\#(R_{AP})$
A3: $AP\mid\equiv\#(R_{SCC})$
A4: $SCC\mid\equiv\#(R_{SCC})$
A5: $AP\mid\equiv SCC\mid\Rightarrow SCC \overset{SK_{AP-BCC}}{\leftrightarrow} AP$

A6: $SCC| \equiv AP| \Rightarrow AP \overset{SK_{AP-BCC}}{\leftrightarrow} SCC$
A7: $AP| \equiv SCC| \Rightarrow ID_{SCC}$
A8: $SCC| \equiv AP| \Rightarrow ID_{AP}$

a. Security control center authenticates access party.

By *M1* and the seeing rule, derive:

Statement 1. $SCC \lhd (\{ID_{AP}, Pwd_{AP}, R_{AP}, h_{AP_1}, M_{Access}, T_1\}_{Puk_{SCC}})$

By *A2* and the freshness rule, derive:

Statement 2. $SCC| \equiv \#(\{ID_{AP}, Pwd_{AP}, R_{AP}, h_{AP_1}, M_{Access}, T_1\}_{Puk_{SCC}})$

By (Statement 1) and the message meaning rule derive:

Statement 3. $SCC| \equiv AP| \sim (ID_{AP}, Pwd_{AP}, R_{AP}, h_{AP_1}, M_{Access}, T_1)$

By (Statement 2), (Statement 3) and the nonce verification rule, derive:

Statement 4. $SCC| \equiv AP| \equiv (ID_{AP}, Pwd_{AP}, R_{AP}, h_{AP_1}, M_{Access}, T_1)$

By (Statement 4) and the belief rule, derive (*G4*):

Statement 5. $SCC| \equiv AP| \equiv AP \overset{SK_{AP-SCC}}{\leftrightarrow} SCC$

By (Statement 5), *A6*, and the jurisdiction rule, derive (*G3*):

Statement 6. $SCC| \equiv AP \overset{SK_{AP-BCC}}{\leftrightarrow} SCC$

By (Statement 4) and the belief rule, derive (*G8*):

Statement 7. $SCC| \equiv AP| \Rightarrow ID_{AP}$

By (Statement 7), *A8*, and the belief rule, derive (*G7*):

Statement 8. $SCC| \equiv ID_{AP}$

b. Access party authenticates security control center.

By *M1* and the seeing rule, derive:

Statement 9. $AP \lhd (\{ID_{SCC}, R_{SCC}, h_{SCC_1}, Token_1, T_3\}_{Puk_{AP}})$

By *A3* and the freshness rule, derive:

Statement 10. $AP| \equiv \#(\{ID_{SCC}, R_{SCC}, h_{SCC_1}, Token_1, T_3\}_{Puk_{AP}})$

By (Statement 9) and the message meaning rule derive:

Statement 11. $AP| \equiv SCC| \sim (ID_{SCC}, R_{SCC}, h_{SCC_1}, Token_1, T_3)$

By (Statement 10), (Statement 11) and the nonce verification rule, derive:

Statement 12. $AP| \equiv SCC| \equiv (ID_{SCC}, R_{SCC}, h_{SCC_1}, Token_1, T_3)$

By (Statement 12) and the belief rule, derive (*G2*):

Statement 13. $AP| \equiv SCC| \equiv SCC \overset{SK_{AP-SCC}}{\leftrightarrow} AP$

By (Statement 13), *A5* and the jurisdiction rule, derive (*G1*):

Statement 14. $AP| \equiv SCC \overset{SK_{AP-BCC}}{\leftrightarrow} AP$

By (Statement 12) and the belief rule, derive (*G6*):

Statement 15. $AP| \equiv SCC| \Rightarrow ID_{SCC}$

By (Statement 15), *A7*, and the belief rule, derive (*G5*):

Statement 16. $AP| \equiv ID_{SCC}$

By (Statement 6), (Statement 8), (Statement 14), and (Statement 16), the accessing party and the *SCC* authenticate each other in the proposed scheme. The *SCC* authenticates the accessing party by:

$$h_{AP_1} \overset{?}{=} H({R_{AP}}^{Sig_{AP_1}} \cdot {Puk_{AP}}^{H(M_{Access})+Sig_{AP_2}} || H(M_{Access})) \tag{104}$$

The accessing party authenticates the *SCC* by:

$$h_{SCC_1} \overset{?}{=} H({R_{SCC}}^{Sig_{SCC_1}} \cdot {Puk_{AP}}^{ID_{SCC}+Sig_{SCC_2}} || H(M_{Access})) \tag{105}$$

Scenario: The attacker pretends to be the *SCC* and fetches data from the user.

Analysis: The attacker cannot obtain the message from the accessing party because our message transmission process is encrypted with the receiver's public key. Only the person with the private key can decrypt the data. Both parties can check the correctness of the signature. Therefore, the attacker will not be able to pretend to be the *SCC* in the proposed method.

4.2. Traceable

Figure 11 shows the characteristics of the blockchain structure. Any event message received by *SCC* automatically chains the message with the sender's signatures to the blockchain. The chaining flow is shown as follows:

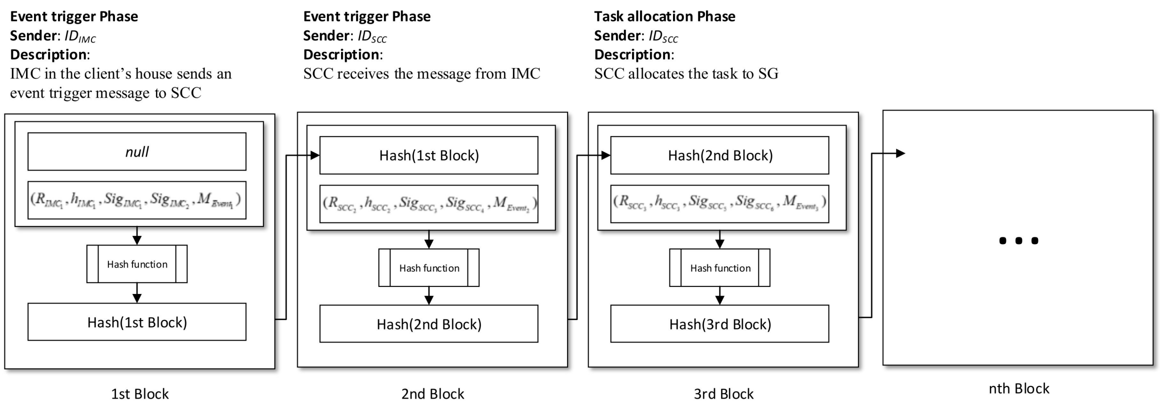

Event trigger Phase
Sender: ID_{IMC}
Description:
IMC in the client's house sends an event trigger message to SCC
Event trigger Phase
Sender: ID_{SCC}
Description:
SCC receives the message from IMC
Task allocation Phase
Sender: ID_{SCC}
Description:
SCC allocates the task to SG
null
$(R_{IMC_1}, h_{IMC_1}, Sig_{IMC_1}, Sig_{IMC_2}, M_{Event_1})$
Hash function
Hash(1st Block)
Hash(1st Block)
$(R_{SCC_2}, h_{SCC_2}, Sig_{SCC_3}, Sig_{SCC_4}, M_{Event_2})$
Hash function
Hash(2nd Block)
Hash(2nd Block)
$(R_{SCC_3}, h_{SCC_3}, Sig_{SCC_5}, Sig_{SCC_6}, M_{Event_3})$
Hash function
Hash(3rd Block)
...
1st Block
2nd Block
3rd Block
nth Block

Figure 11. The event message chaining process in the blockchain.

(1) In the event trigger phase, once the event is triggered from the *IMC* in the client's house, the *SCC* receives the message and sends $(R_{IMC_1}, h_{IMC_1}, Sig_{IMC_1}, Sig_{IMC_2}, M_{Event_1})$ to the *BCC*, then the *BCC* calculates the hash with the message, and creates that information as the first block.

(2) Next, the *SCC* sends $(R_{SCC_2}, h_{SCC_2}, Sig_{SCC_3}, Sig_{SCC_4}, M_{Event_2})$ to the *BCC*, then the *BCC* calculates the hash value of the first block and chains that information as the second block.

(3) In the task allocation phase, the *SCC* sends $(R_{SCC_3}, h_{SCC_3}, Sig_{SCC_5}, Sig_{SCC_6}, M_{Event_3})$ to the *BCC*, then the *BCC* calculates the hash value with the message and hash value in the second block and chains that information as the third block.

Continuously, when any event message needs to chain to the *BCC*, the *SCC* needs to send the parameters, the signatures, and the message from the sender, then calculate the current block's hash value with the message and the previous block's hash value. Therefore, the processing of the event can be recorded in detail and can be traced.

The phase of creating the blockchain with signatures is defined as follows:

(1) Event-trigger phase

The message from the *IMC* in the client's house is:

$$Sig_{IMC_1} = 1 - {r_3}^{-1} \cdot Prk_{IMC} \cdot H(M_{Event_1})\mathrm{mod}p \tag{106}$$

$$Sig_{IMC_2} = M_{Event_1} + Prk_{IMC}\mathrm{mod}p \tag{107}$$

SCC's staff confirm received the message from the *IMC*:

$$Sig_{SCC_3} = 1 - {r_4}^{-1} \cdot Prk_{SCC} \cdot H(M_{Event_2})\mathrm{mod}p \tag{108}$$

$$Sig_{SCC_4} = M_{Event_2} + Prk_{SCC}\mathrm{mod}p \tag{109}$$

(2) Task allocating phase

The messages are sent to the *SG*:

$$Sig_{SCC_5} = 1 - {r_5}^{-1} \cdot Prk_{SCC} \cdot H(M_{Event_3})\mathrm{mod}p \tag{110}$$

$$Sig_{SCC_6} = M_{Event_3} + Prk_{SCC}\mathrm{mod}p \tag{111}$$

The message from the *SG* is:

$$Sig_{SG_1} = 1 - {r_6}^{-1} \cdot Prk_{SG} \cdot H(M_{Event_4})\mathrm{mod}p \tag{112}$$

$$Sig_{SG_2} = M_{Event_4} + Prk_{SG}\mathrm{mod}p \tag{113}$$

SCC's staff confirm received the messages from the *SG*:

$$Sig_{SCC_7} = 1 - {r_7}^{-1} \cdot Prk_{SCC} \cdot H(M_{Event_5})\mathrm{mod}p \tag{114}$$

$$Sig_{SCC_8} = M_{Event_5} + Prk_{SCC}\mathrm{mod}p \tag{115}$$

(3) Task accomplished phase

The message from the *SG* is:

$$Sig_{SG_3} = 1 - {r_8}^{-1} \cdot Prk_{SG} \cdot H(M_{Event_6})\mathrm{mod}p \tag{116}$$

$$Sig_{SG_4} = M_{Event_6} + Prk_{SG}\mathrm{mod}p \tag{117}$$

The messages are sent to the *C*:

$$Sig_{SCC_9} = 1 - {r_9}^{-1} \cdot Prk_{SCC} \cdot H(M_{Event_7})\mathrm{mod}p \tag{118}$$

$$Sig_{SCC_{10}} = M_{Event_7} + Prk_{SCC}\mathrm{mod}p \tag{119}$$

The message from the *C* is:

$$Sig_{C_1} = 1 - {r_{10}}^{-1} \cdot Prk_{C} \cdot H(M_{Event_8})\mathrm{mod}p \tag{120}$$

$$Sig_{C_2} = M_{Event_8} + Prk_{C}\mathrm{mod}p \tag{121}$$

SCC's staff confirm the received messages from the *C*:

$$Sig_{SCC_{11}} = 1 - {r_{11}}^{-1} \cdot Prk_{SCC} \cdot H(M_{Event_9})\mathrm{mod}p \tag{122}$$

$$Sig_{SCC_{12}} = M_{Event_9} + Prk_{SCC}\mathrm{mod}p \tag{123}$$

Therefore, the accessing party can easily track and verify the transactions via the blockchain.

4.3. Integrity

To preserve the integrity of the data in this paper, we used a public key, a private key, and digital signatures. When a malicious person modifies the data, the receiver can verify whether the data has been modified or not by verifying the signatures. The user can use the receiver's public key to encrypt data for transmission. Malicious people are not able to obtain the encrypted data because they do not have a valid private key, so he/she is unable to access and tamper with the data.

In the information transmission process, preventing the modification of transmitted data is as follows: Equations (21), (22), (30), (31), (35), (36), (44), (45), (53), (54), (58), (59), (67), (68), (76), (77), (85) and (86).

Scenario: The attacker intercepts the message from the SCC authority to the accessing party and tampers with the message before sending it to the accessing party.

Analysis: The attacker will fail because our proposed method uses the receiver's public key to encrypt the data, and only the receiver's private key can decrypt the data. Therefore, when the attacker intercepts the message, the attacker cannot decrypt the message and maliciously modify the data.

4.4. Non-Repudiation

Non-repudiation is achieved in our proposed method. In every communication phase, the sender must sign their private key with the message; when the receiver receives the message, they must verify the hash value from the sender. If the equation is equal, the signatures are valid. So the message that is sent by the sender will have the ability of non-repudiation. Table 3 shows the non-repudiation of the proposed scheme.

Table 3. Non-repudiation of the proposed scheme.

Phase	**Party**		**Verification**
	Issuer	**Holder**	
Authentication phase	AP	SCC	$h_{AP_1} \overset{?}{=} H({R_{AP_1}}^{Sig_{AP_1}} \cdot {Puk_{AP}}^{H(M_{Access})+Sig_{AP_2}} \| \| H(M_{Access}))$
	SCC	AP	$h_{SCC_1} \overset{?}{=} H({R_{SCC_1}}^{Sig_{SCC_1}} \cdot {Puk_{AP}}^{ID_{SCC}+Sig_{SCC_2}})$
Event-trigger phase	IMC	SCC	$h_{IMC_1} \overset{?}{=} H({R_{IMC_1}}^{Sig_{IMC_1}} \cdot {Puk_{IMC}}^{H(M_{Event_1})+Sig_{IMC_2}} \| \| H(M_{Event_1}))$
Task allocating phase	SCC	SG	$h_{SCC_2} \overset{?}{=} H({R_{SCC_3}}^{Sig_{SCC_5}} \cdot {Puk_{SCC}}^{H(M_{Event_3})+Sig_{SCC_6}} \| \| H(M_{Event_3}))$
	SG	SCC	$h_{SG_1} \overset{?}{=} H({R_{SG_1}}^{Sig_{SG_1}} \cdot {Puk_{SG}}^{H(M_{Event_4})+Sig_{SG_2}} \| \| H(M_{Event_4}))$
Task accomplished phase	SG	SCC	$h_{SG_2} \overset{?}{=} H({R_{SG_2}}^{Sig_{SG_3}} \cdot {Puk_{SG}}^{H(M_{Event_6})+Sig_{SG_4}} \| \| H(M_{Event_6}))$
	SCC	C	$h_{SCC_5} \overset{?}{=} H({R_{SCC_5}}^{Sig_{SCC_9}} \cdot {Puk_{SCC}}^{H(M_{Event_7})+Sig_{SCC_{10}}} \| \| H(M_{Event_7}))$
	C	SCC	$h_{C_1} \overset{?}{=} H({R_{C_1}}^{Sig_{C_1}} \cdot {Puk_{C}}^{H(M_{Event_8})+Sig_{C_2}} \| \| H(M_{Event_8}))$

4.5. Against Known Attacks

4.5.1. Man-in-the-Middle Attack

In this attack, the attacker may try to pretend to be any accessing party in every phase. The attacker will try to intercept and tamper with the data. However, all messages that are created or updated from the main role from the client's house and security company can be effectively authenticated through signatures in our proposed method, so it is difficult to tamper with the data during communication.

For example, in the authentication phase, the *AP* encrypts the message with the *SCC*'s public key in Equation (6), then generates the chipper message C_{AP_1}. Next, the *AP* sends C_{AP_1} to the *SCC*, and the receiver decrypts C_{AP_1} with the *SCC*'s private key in Equation (7). The related formula is as follows:

(1) Authentication phase

- *AP->SCC*

$$C_{AP_1} = E_{Puk_{SCC}}(T_1||R_{AP_1}||h_{AP_1}||Sig_{AP_1}||Sig_{AP_2}||M_{Access}||H(M_{Access})) \quad (124)$$

$$(T_1||R_{AP_1}||h_{AP_1}||Sig_{AP_1}||Sig_{AP_2}||M_{Access}||H(M_{Access})) = D_{Prk_{SCC}}(C_{AP_1}) \quad (125)$$

- *SCC->AP*

$$C_{SCC_1} = E_{Puk_{AP}}(ID_{SCC}||R_{SCC_1}||h_{SCC_1}||Sig_{SCC_1}||Sig_{SCC_2}||Token_1||T_3||Seq_{i+1}) \quad (126)$$

$$(ID_{SCC}||R_{SCC_1}||h_{SCC_1}||Sig_{SCC_1}||Sig_{SCC_2}||Token_1||T_3) = D_{Prk_{AP}}(C_{SCC_1}) \quad (127)$$

(2) Event-trigger phase

- *IMC->SCC*

$$C_{IMC_1} = E_{Puk_{SCC}}(T_6||R_{IMC_1}||h_{IMC_1}||Sig_{IMC_1}||Sig_{IMC_2}||M_{Event_1}||H(M_{Event_1})) \quad (128)$$

$$(T_6||R_{IMC_1}||h_{IMC_1}||Sig_{IMC_1}||Sig_{IMC_2}||M_{Event}||H(M_{Event})) = D_{Prk_{SCC}}(C_{IMC_1}) \quad (129)$$

(3) Task allocating phase

- *SCC->SG*

$$C_{SCC_2} = E_{Puk_{SG}}(T_8||R_{SCC_3}||h_{SCC_3}||Sig_{SCC_5}||Sig_{SCC_6}||M_{Event_3}||H(M_{Event_3})) \quad (130)$$

$$(T_8||R_{SCC_3}||h_{SCC_3}||Sig_{SCC_5}||Sig_{SCC_6}||M_{Event_3}||H(M_{Event_3})) = D_{Prk_{SG}}(C_{SCC_2}) \quad (131)$$

- *SG->SCC*

$$C_{SG_1} = E_{Puk_{SCC}}(T_{10}||R_{SG_1}||h_{SG_1}||Sig_{SG_1}||Sig_{SG_2}||M_{Event_4}||H(M_{Event_4})||(Y/N)) \quad (132)$$

$$(T_{10}||R_{SG_1}||h_{SG_1}||Sig_{SG_1}||Sig_{SG_2}||M_{Event_4}||H(M_{Event_4})||(Y/N)) = D_{Prk_{SCC}}(C_{SG_1}) \quad (133)$$

(4) Task accomplished phase

- *SG->SCC*

$$C_{SG_2} = E_{Puk_{SCC}}(T_{13}||R_{SG_2}||h_{SG_2}||Sig_{SG_3}||Sig_{SG_4}||M_{Event_6}||H(M_{Event_6})) \quad (134)$$

$$(T_{13}||R_{SG_2}||h_{SG_2}||Sig_{SG_3}||Sig_{SG_4}||M_{Event_6}||H(M_{Event_6})) = D_{Prk_{SCC}}(C_{SG_2}) \quad (135)$$

- *SCC->C*

$$C_{SCC_3} = E_{Puk_C}(T_{15}||R_{SCC_5}||h_{SCC_5}||Sig_{SCC_9}||Sig_{SCC_{10}}||M_{Event_7}||H(M_{Event_7})) \quad (136)$$

$$(T_{15}||R_{SCC_5}||h_{SCC_5}||Sig_{SCC_9}||Sig_{SCC_{10}}||M_{Event_7}||H(M_{Event_7})) = D_{Prk_C}(C_{SCC_3}) \quad (137)$$

- *C->SCC*

$$C_{C_1} = E_{Puk_{SCC}}(T_{17}||R_{C_1}||h_{C_1}||Sig_{C_1}||Sig_{C_2}||M_{Event_8}||H(M_{Event_8})) \quad (138)$$

$$(T_{17}||R_{C_1}||h_{C_1}||Sig_{C_1}||Sig_{C_2}||M_{Event_8}||H(M_{Event_8})) = D_{Prk_C}(C_{C_1}) \quad (139)$$

Therefore, the encryption and decryption scheme in every phase can effectively prevent man-in-the-middle attacks.

4.5.2. Replay Attack

An attacker intercepts the data from the communication, then sends the same data to the receiver, pretending to be the sender. In our method, we use a timestamp mechanism in every communication between the sender and receiver to prevent this kind of attack.

For example, in the authentication phase, the accessing party is encrypted at a timestamp T_1 in the cipher message C_{AP_1} and sends it to the *SCC*. The *SCC* decrypts the message and generates a timestamp T_2 that represents the received time, then checks the time difference between T_2 and T_1. If the time is over the parameter τ, that means that the message sending time is too long and it may have been intercepted by an attacker. The related formula is as follows:

$$C_{AP_1} = E_{Puk_{SCC}}(T_1||R_{AP_1}||h_{AP_1}||Sig_{AP_1}||Sig_{AP_2}||M_{Access}||H(M_{Access})) \tag{140}$$

$$(T_1||R_{AP_1}||h_{AP_1}||Sig_{AP_1}||Sig_{AP_2}||M_{Access}||H(M_{Access})) = D_{Prk_{SCC}}(C_{AP_1}) \tag{141}$$

$$(T_2 - T_1) \overset{?}{\leq} \tau \tag{142}$$

Our method also added a sequential number in the message from a sender; for example, in the authentication phase, a sequential *ID* Seq_i generated randomly from the *AP*, in the next round of messages from the *SCC* will increase the Seq_i to Seq_{i+1}. It is bound with the message and sent to the *AP*; the *AP* receives the message and validates the sequential *ID* and checks if it is in the same sequence.

4.5.3. Side-Channel Attack

In the proposed scheme, it is hard to retrieve the private key via a side-channel attack. For every message that sends from the sender, the sender needs to select a random number and compute a parameter. For example, in the authentication phase, the *AP* selects a random number r_1, and computes the following parameter:

$$R_{AP_1} = g^{r_1} \bmod p \tag{143}$$

Then *AP* uses the hash function and R_{AP_1} to compute the parameter h_{AP_1}:

$$h_{AP_1} = H(R_{AP_1} \cdot Puk_{AP}{}^{(M_{Access}+Prk_{AP})}||H(M_{Access})) \tag{144}$$

The *AP* also uses r_1 to compute the signatures as follows:

$$Sig_{AP_1} = 1 - r_1{}^{-1} \cdot Prk_{AP} \cdot H(M_{Access}) \bmod p \tag{145}$$

$$Sig_{AP_2} = M_{Access} + Prk_{AP} \bmod p \tag{146}$$

Therefore, it is hard for the attacker to attack due to the randomized number.

4.5.4. Forgery Attack

In our scheme, it is difficult for an attacker to forge the message to the *SCC*; the received message will verify the identity of the accessing party. Table 3 in Section 4.4 shows all the verification equations of the proposed method. We have implemented and modified the ECDSA; the verification of the ECDSA can be verified by using the method introduced in Section 2.2. Based on the security of the Elliptic Curve Digital Signature Algorithm (ECDSA), it is a discrete logarithm problem. The ECDSA for the details of the proof refers to Section 8.1 of [29].

We also give two examples of verification derived for checking the hash value. They are shown as follows:

(1) In the authentication phase, the verification is verified as follows:

$$\begin{aligned} h_{AP_1} &= H({R_{AP_1}}^{Sig_{AP_1}} \cdot {Puk_{AP}}^{H(M_{Access})+Sig_{AP_2}} || H(M_{Access})) \\ &= H({R_{AP}}^{Sig_{AP_1}} \cdot {Puk_{AP}}^{H(M_{Access})} \cdot {Puk_{AP}}^{Sig_{AP_2}} || H(M_{Access})) \\ &= H(g^{r_1 \cdot Sig_{AP_1}} \cdot g^{Prk_{AP} \cdot H(M_{Access})} \cdot g^{Prk_{AP} \cdot Sig_{AP_2}} || H(M_{Access})) \\ &= H(g^{r_1 \cdot Sig_{AP_1} + Prk_{AP} \cdot H(M_{Access}) + Prk_{AP} \cdot Sig_{AP_2}} || H(M_{Access})) \\ &= H(g^{r_1 (1 - {r_1}^{-1} \cdot Prk_{AP} \cdot H(M_{Access})) + Prk_{AP} \cdot H(M_{Access}) + Prk_{AP} \cdot Sig_{AP_2}} || H(M_{Access})) \\ &= H(g^{r_1 - Prk_{AP} \cdot H(M_{Access}) + Prk_{AP} \cdot H(M_{Access}) + Prk_{AP} \cdot Sig_{AP_2}} || H(M_{Access})) \\ &= H(g^{r_1 + Prk_{AP} \cdot Sig_{AP_2}} || H(M_{Access})) \\ &= H(g^{r_1 + Prk_{AP}{}^{(M_{Access} + Prk_{AP})}} || H(M_{Access})) \\ &= H(g^{r_1} \cdot (g^{Prk_{AP}})^{(M_{Access} + Prk_{AP})} || H(M_{Access})) \\ &= H(R_{AP_1} \cdot {Puk_{AP}}^{(M_{Access} + Prk_{AP})} || H(M_{Access})) \end{aligned}$$

(2) In the event-trigger phase, the verification is derived as follows:

$$\begin{aligned} h_{IMC_1} &= H({R_{IMC}}^{Sig_{IMC_1}} \cdot {Puk_{IMC}}^{H(M_{Event})+Sig_{IMC_2}} || H(M_{Event})) \\ &= H({R_{IMC}}^{Sig_{IMC_1}} \cdot {Puk_{IMC}}^{H(M_{Event})} \cdot {Puk_{IMC}}^{Sig_{IMC_2}} || H(M_{Event})) \\ &= H(g^{r_3 \cdot Sig_{IMC_1}} \cdot g^{Prk_{IMC} \cdot H(M_{Event})} \cdot g^{Prk_{IMC} \cdot Sig_{IMC_2}} || H(M_{Event})) \\ &= H(g^{r_3 \cdot Sig_{IMC_1} + Prk_{IMC} \cdot H(M_{Event}) + Prk_{IMC} \cdot Sig_{IMC_2}} || H(M_{Event})) \\ &= H(g^{r_3 (1 - {r_3}^{-1} \cdot Prk_{IMC} \cdot H(M_{Event})) + Prk_{IMC} \cdot H(M_{Event}) + Prk_{IMC} \cdot Sig_{IMC_2}} || H(M_{Event})) \\ &= H(g^{r_3 - Prk_{IMC} \cdot H(M_{Event}) + Prk_{IMC} \cdot H(M_{Event}) + Prk_{IMC} \cdot Sig_{IMC_2}} || H(M_{Event})) \\ &= H(g^{r_3 + Prk_{IMC} \cdot Sig_{IMC_2}} || H(M_{Event})) \\ &= H(g^{r_3 + Prk_{IMC}{}^{(M_{Event} + Prk_{IMC})}} || H(M_{Event})) \\ &= H(g^{r_3} \cdot (g^{Prk_{IMC}})^{(M_{Event} + Prk_{IMC})} || H(M_{Event})) \\ &= H(R_{IMC} \cdot {Puk_{IMC}}^{(M_{Event} + Prk_{IMC})} || H(M_{Event})) \end{aligned}$$

4.6. Threat Models in the Communication Phase

Next, we will discuss the scenarios of some threat models in the communication phase.

(1) The IoT is disconnected from the IoT main controller: The IoT devices in the client's house send a "keep-alive" signal to the main controller every number of seconds. Therefore, the main controller is able to confirm that the IoT is still alive. Some situations will cause IoT devices to disconnect from the main controller. For example, power loss, low battery, and device aging. If the main controller does not receive the "keep-alive" signal from one of the IoT devices, the controller will send an event automatically to the *SCC*, and the *SCC* will allocate the task to the security guard to solve the problem immediately.

(2) The *IMC* in the client's house is disconnected from the *SCC*: The *SCC* will ask the *IMC* in the client's house automatically in a routine job; the main controller needs to respond with an alive message to the *BCC*. If the main controller does not receive the response message, the *SCC* will generate an event to ask the *SG* to solve the problem instantly.

(3) Fake message impersonated from the *IMC*: Every communication between the IoT main controller and the *SCC* can be forged by a third party. To prevent this forgery security problem, the message sent between the *IMC* and the *SCC* communicate with the session key via a secure channel. Details will be described in the next section.

(4) False alarm with the IoT: If IoT devices are tuned in the *IMC*, the probability of a false alarm in the IoT is very low. If it unfortunately happens, the alarm will be checked by the *SCC*, and *SCC*'s staff will check the camera manually to determine if there is a problem.

5. Discussion

5.1. Computation Cost

In Table 4, we analyze the computational costs of each phase. We use the asymmetrical, symmetrical, hash function, addition/subtraction operation, multiplication/division operation, and exponential as the basis for calculating the cost.

Table 4. Computation costs of the proposed scheme.

	Security Control Center	Other Party
Authentication phase	$2T_{asy} + 1T_v + 1T_h + 3T_{add} + 2T_{sub} + 3T_{mul} + 1T_{div} + 5T_{exp}$	Accessing Party: $2T_{asy} + 1T_v + 3T_h + 3T_{add} + 2T_{sub} + 3T_{mul} + 1T_{div} + 4T_{exp}$
Event-trigger phase	$1T_{asy} + 1T_v + 3T_h + 3T_{add} + 2T_{sub} + 3T_{mul} + 1T_{div} + 5T_{exp}$	Client House's IoT Main Contoller: $1T_{asy} + 2T_h + 2T_{add} + 1T_{sub} + 2T_{mul} + 1T_{div} + 3T_{exp}$
Task allocating phase	$2T_{asy} + 1T_v + 5T_h + 5T_{add} + 3T_{sub} + 5T_{mul} + 2T_{div} + 8T_{exp}$	Security Guard: $2T_{asy} + 1T_v + 3T_h + 3T_{add} + 2T_{sub} + 3T_{mul} + 1T_{div} + 5T_{exp}$
Task accomplished phase	$3T_{asy} + 2T_v + 5T_h + 5T_{add} + 4T_{sub} + 5T_{mul} + 2T_{div} + 8T_{exp}$	Security Guard: $1T_{asy} + 2T_h + 2T_{add} + 1T_{sub} + 2T_{mul} + 1T_{div} + 3T_{exp}$ Client: $2T_{asy} + 1T_v + 3T_h + 3T_{add} + 2T_{sub} + 3T_{mul} + 1T_{div} + 5T_{exp}$

Notes: T_{asy}: the time required for an asymmetrical encryption/decryption. T_v: the time required for verifying the hash value. T_h: the time required for a one-way hash function. T_{add}: the time required for an additional operation. T_{sub}: the time required for a subtraction operation. T_{mul}: the time required for a multiplication operation. T_{div}: the time required for a division operation. T_{exp}: the time required for an exponential operation.

5.2. Communication Performance

In Table 5, we analyze the communication costs at every phase. In a 4G environment, the maximum transmission speed is 100 Mbps [46]. In a 5G environment, the maximum transmission speed is 20 Gbps [47]. In our analysis, it was assumed that an access or event message required 304 bits, a hash operation required 160 bits, a signature operation required 1024 bits, and the other message required 80 bits. The 304 bits of an access or event message is assumed by the maximum length of the message (five IDs and one timestamp) that sends at the allocating task phase, which is 5 × 80 bits + 1 × 32 bits = 432 bits.

Table 5. Communication costs of the proposed scheme.

	Message Length	4G (100 Mbps)	5G (20 Gbps)
Authentication phase	$1L_m + 2L_h + 4L_{sig} + 8L_{other} =$ 1 × 432 + 2 × 160 + 4 × 1024 + 8 × 80 = 5488 bits	5488/102,400 = 0.054 ms	5488/20,480,000 = 0.27 us
Event-trigger phase	$1L_m + 2L_h + 2L_{sig} + 4L_{other} =$ 1 × 432 + 2 × 160 + 2 × 1024 + 4 × 80 = 3120 bits	3120/102,400 = 0.030 ms	3120/20,480,000 = 0.15 us
Task allocating phase	$2L_m + 4L_h + 4L_{sig} + 10L_{other} =$ 2 × 432 + 4 × 160 + 4 × 1024 + 10 × 80 = 6400 bits	6400/102,400 = 0.065 ms	6400/20,480,000 = 0.31 us
Task accomplished phase	$3L_m + 6L_h + 6L_{sig} + 11L_{other} =$ 3×432 + 6 × 160 + 6 × 1024 + 11 × 80 = 9280 bits	9280/102,400 = 0.091 ms	9280/20,480,000 = 0.45 us

Notes: L_m: The message length of an access or event message (432 bits). L_h: The message length of a hash value (160 bits). L_{sig}: The message length of a signature (1024 bits). L_{other}: The message length of other (80 bits).

The communication costs are calculated depending on the number of messages. For example, in the authentication phase, the AP sends one access message, two hash values, two signatures, and three other messages to the SCC. After that, the SCC sends two signatures and five other messages. In total, it requires 1 × 432 bits + 2 × 160 bits + 4 × 1024 bits + 8 × 80 bits = 5488 bits. In a 4G environment, the maximum transmission speed is 100 Mbps and it only takes 0.054 ms to transfer all the messages. In a 5G environment, the maximum transmission speed is 20 Gbps; the transmission time needed is only 0.27 us to complete the authentication phase.

5.3. Comparison

In this section, we compare related works which involve the security analysis in Table 6. Compared to the related works, some of the proposed schemes lack complete functionality (for example, blockchain, smart contract, or control center). Our scheme focuses on proposing a security service that involves blockchain and smart contract technologies to ensure that the event messages that generate and send from the accessing party to the security control center are fully recorded to the blockchain center. Therefore, compared to other schemes, our scheme achieves the full safety security protocol and we also make a full security analysis including mutual authentication, traceability, integrity, non-repudiation, and threat models.

Table 6. Comparison of related works and the proposed study.

Authors	Year	Objective	1	2	3	4	5	6	7	8	9
Liu et al. [19]	2020	Blockchain-based access control system	Y	Y	Y	N	N	N	N	N	N
Lin et al. [20]	2018	Blockchain-based access control to access control for industry 4.0	Y	Y	N	N	Y	Y	Y	N	N
Alkhammash et al. [22]	2020	Smart campus with the Internet of Things and blockchain	Y	Y	N	N	N	Y	N	N	N
Lyu et al. [23]	2019	Accessing private smart home with smart devices through an IFTTT Gateway	N	N	Y	Y	Y	N	N	Y	N
Fakroon et al. [24]	2020	Remote anonymous authentication for smart home	N	N	Y	Y	N	Y	N	Y	N
Ours	2021	Proposed a traceable and authenticated security company record on the blockchain	Y	Y	Y	Y	Y	Y	Y	Y	Y

Notes: 1: Focus on a blockchain, 2: Smart contract, 3. Protocol, 4: Proof of mutual authentication, 5: Traceability, 6: Integrity, 7: Non-repudiation, 8: Threat models, 9: Control center, Y: Yes, N: No.

6. Conclusions

This study aimed to propose a traceable security system. We involved smart contracts and blockchain technologies in this scheme. Every record from the accessing party to the blockchain center is sent and chained. We used the characteristics of the blockchain center to solve the trust problem between the security company and the client. We also proposed a fair and clear arbitration mechanism for the clarification of responsibilities, which was not available in the security industry in the past.

The proposed method achieves the following goals. Firstly, the characteristic data through the blockchain can be publicly verified and the information will not be modified. Secondly, we used BAN logic to prove mutual authentication. Thirdly, the official agency is able to query the information with the signature from the accessing party and validate the legality of the event record. Fourthly, the proposed method achieves traceability, integrity, and non-repudiation. The method can also resist man-in-the-middle attacks, replay attacks, and forgery attacks. In this study, we also considered the threat models during the communication phase. Every message from the IoT main controller in the client's house will not be able to send a false alarm to the security control center. It also allows the control center to stay connected with the IoT main controller at any time.

In future works, the proposed method of a blockchain center can be expanded to become a blockchain alliance with multiple security companies able to connect to blockchain alliance services with traceability and authentication.

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

Funding: The APC was funded by Chaoyang University of Technology.

Institutional Review Board Statement: This study is based entirely on theoretical basic research. It does not involve humans.

Informed Consent Statement: This study is based entirely on theoretical basic research. It does not involve humans.

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. 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]
2. FBI UCR: Crime in the U.S. Available online: https://ucr.fbi.gov/crime-in-the-u.s/ (accessed on 6 January 2021).
3. Crime in England and Wales: Year Ending June 2019. Available online: https://www.ons.gov.uk/peoplepopulationandcommunity/crimeandjustice/bulletins/crimeinenglandandwales/yearendingjune2019 (accessed on 6 January 2021).
4. Nemeth, C.P. *Private Security: An Introduction to Principles and Practice*, 1st ed.; CRC Press: Boca Raton, FL, USA, 2017.
5. Lin, H.; Bergmann, N. IoT Privacy and Security Challenges for Smart Home Environments. *Information* **2016**, *7*, 44. [CrossRef]
6. Hassija, V.; Chamola, V.; Saxena, V.; Jain, D.; Goyal, P.; Sikdar, B. A Survey on IoT Security: Application Areas, Security Threats, and Solution Architectures. *IEEE Access* **2019**, *7*, 82721–82743. [CrossRef]
7. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. Available online: https://bitcoin.org/bitcoin.pdf (accessed on 29 December 2020).
8. Feroz Ahmad, A.; Prashant, K.; Gulshan, S.; Med Salim, B. Bitcoin: Digital Decentralized Cryptocurrency. In *Handbook of Research on Network Forensics and Analysis Techniques*; Gulshan, S., Prabhat, K., Gupta, B.B., Suman, B., Nilanjan, D., Eds.; IGI Global: Hershey, PA, USA, 2018; pp. 395–415. [CrossRef]
9. Khatoon, A. A Blockchain-Based Smart Contract System for Healthcare Management. *Electronics* **2020**, *9*, 94. [CrossRef]
10. 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]
11. Zarour, M.; Ansari, M.T.J.; Alenezi, M.; Sarkar, A.K.; Faizan, M.; Agrawal, A.; Kumar, R.; Khan, R.A. Evaluating the Impact of Blockchain Models for Secure and Trustworthy Electronic Healthcare Records. *IEEE Access* **2020**, *8*, 157959–157973. [CrossRef]
12. Hu, Y.; Manzoor, A.; Ekparinya, P.; Liyanage, M.; Thilakarathna, K.; Jourjon, G.; Seneviratne, A. A Delay-Tolerant Payment Scheme Based on the Ethereum Blockchain. *IEEE Access* **2019**, *7*, 33159–33172. [CrossRef]
13. Lin, C.; He, D.; Huang, X.; Khan, M.K.; Choo, K.R. DCAP: A Secure and Efficient Decentralized Conditional Anonymous Payment System Based on Blockchain. *IEEE Trans. Inf. Forensics Secur.* **2020**, *15*, 2440–2452. [CrossRef]
14. 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]
15. Chen, Q.; Srivastava, G.; Parizi, R.M.; Aloqaily, M.; Ridhawi, I.A. An incentive-aware blockchain-based solution for internet of fake media things. *Inf. Process. Manag.* **2020**, *57*, 102370. [CrossRef]
16. Li, J.; Grintsvayg, A.; Kauffman, J.; Fleming, C. LBRY: A Blockchain-Based Decentralized Digital Content Marketplace. In Proceedings of the 2020 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS), Oxford, UK, 3–6 August 2020; pp. 42–51.
17. Ma, Z.; Jiang, M.; Gao, H.; Wang, Z. Blockchain for digital rights management. *Future Gener. Comput. Syst.* **2018**, *89*, 746–764. [CrossRef]
18. Smith, S.; Ellis, J.; Abrams, R. Central Alarm Stations and Dispatch Operations. In *The Professional Protection Officer*; Butterworth-Heinemann: Oxford, UK, 2010; pp. 89–103. [CrossRef]
19. Liu, H.; Han, D.; Li, D. Fabric-iot: A Blockchain-Based Access Control System in IoT. *IEEE Access* **2020**, *8*, 18207–18218. [CrossRef]
20. Lin, C.; He, D.; Huang, X.; Choo, K.-K.R.; Vasilakos, A.V. BSeIn: A blockchain-based secure mutual authentication with fine-grained access control system for industry 4.0. *J. Netw. Comput. Appl.* **2018**, *116*, 42–52. [CrossRef]
21. Mbarek, B.; Ge, M.; Pitner, T. *Blockchain-Based Access Control for IoT in Smart Home Systems*; Springer: Cham, Switzerland, 2020; pp. 17–32. [CrossRef]
22. Alkhammash, M.; Beloff, N.; White, M. *An Internet of Things and Blockchain Based Smart Campus Architecture*; Springer: Cham, Switzerland, 2020; pp. 467–486.
23. Lyu, Q.; Zheng, N.; Liu, H.; Gao, C.; Chen, S.; Liu, J. Remotely Access "My" Smart Home in Private: An Anti-Tracking Authentication and Key Agreement Scheme. *IEEE Access* **2019**, *7*, 41835–41851. [CrossRef]
24. Fakroon, M.; Alshahrani, M.; Gebali, F.; Traore, I. Secure remote anonymous user authentication scheme for smart home environment. *Internet Things* **2020**, *9*, 100158. [CrossRef]
25. Buterin, V. A next-generation smart contract and decentralized application platform. *Ethereum White Pap.* **2014**, *3*, 36.

26. Hjálmarsson, F.Þ.; Hreiðarsson, G.K.; Hamdaqa, M.; Hjálmtýsson, G. Blockchain-Based E-Voting System. In Proceedings of the 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), San Francisco, CA, USA, 2–7 July 2018; pp. 983–986.
27. Yiu, N.C.K. An Empirical Analysis of Implementing Enterprise Blockchain Protocols in Supply Chain Anti-Counterfeiting and Traceability. *Cryptogr. Secur.* **2021**. [CrossRef]
28. Vanstone, S. Responses to NIST's proposal. *Commun. ACM* **1992**, *35*, 50–52.
29. Johnson, D.; Menezes, A.; Vanstone, S. The Elliptic Curve Digital Signature Algorithm (ECDSA). *Int. J. Inf. Secur.* **2001**, *1*, 36–63. [CrossRef]
30. Elaine Barker, A.R. *Transitioning the Use of Cryptographic Algorithms and Key Lengths*; National Institute of Standards and Technology: Washington, DC, USA, 2019.
31. Burrows, M.; Abadi, M.; Needham, R. A logic of authentication. *ACM Trans. Comput. Syst.* **1990**, *8*, 18–36. [CrossRef]
32. Sierra, J.M.; Hernández, J.C.; Alcaide, A.; Torres, J. *Validating the Use of BAN LOGIC*; Springer: Berlin/Heidelberg, Germany; pp. 851–858.
33. Zhao, Q.; Chen, S.; Liu, Z.; Baker, T.; Zhang, Y. Blockchain-based privacy-preserving remote data integrity checking scheme for IoT information systems. *Inf. Process. Manag.* **2020**, *57*, 102355. [CrossRef]
34. Tasnim, M.A.; Omar, A.A.; Rahman, M.S.; Bhuiyan, M.Z.A. CRAB: Blockchain Based Criminal Record Management System. In *Proceedings of Security, Privacy, and Anonymity in Computation, Communication, and Storage*; Springer: Cham, Switzerland, 2018; pp. 294–303.
35. Servida, F.; Casey, E. IoT forensic challenges and opportunities for digital traces. *Digit. Investig.* **2019**, *28*, S22–S29. [CrossRef]
36. Alblooshi, M.; Salah, K.; Alhammadi, Y. Blockchain-based Ownership Management for Medical IoT (MIoT) Devices. In Proceedings of the 2018 International Conference on Innovations in Information Technology (IIT), Al Ain, United Arab Emirates, 18–19 November 2018; pp. 151–156.
37. 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]
38. Jin, H.; Jiang, H.; Zhou, K. Dynamic and Public Auditing with Fair Arbitration for Cloud Data. *IEEE Trans. Cloud Comput.* **2018**, *6*, 680–693. [CrossRef]
39. Conti, M.; Dragoni, N.; Lesyk, V. A Survey of Man In The Middle Attacks. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 2027–2051. [CrossRef]
40. 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.
41. Saadeh, M.; Sleit, A.; Qatawneh, M.; Almobaideen, W. Authentication Techniques for the Internet of Things: A Survey. In Proceedings of the 2016 Cybersecurity and Cyberforensics Conference (CCC), Amman, Jordan, 2–4 August 2016; pp. 28–34.
42. Samuel, S.S.I. A review of connectivity challenges in IoT-smart home. In Proceedings of the 2016 3rd MEC International Conference on Big Data and Smart City (ICBDSC), Muscat, Oman, 15–16 March 2016; pp. 1–4.
43. Kocher, P.; Jaffe, J.; Jun, B.; Rohatgi, P. Introduction to differential power analysis. *J. Cryptogr. Eng.* **2011**, *1*, 5–27. [CrossRef]
44. Mamiya, H.; Miyaji, A.; Morimoto, H. *Efficient Countermeasures against RPA, DPA, and SPA*; Springer: Berlin/Heidelberg, Germany, 2004; pp. 343–356.
45. Genkin, D.; Pachmanov, L.; Pipman, I.; Tromer, E.; Yarom, Y. ECDSA Key Extraction from Mobile Devices via Nonintrusive Physical Side Channels. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016; pp. 1626–1638.
46. Khan, A.H.; Qadeer, M.A.; Ansari, J.A.; Waheed, S. 4G as a Next Generation Wireless Network. In Proceedings of the 2009 International Conference on Future Computer and Communication, Kuala Lumpur, Malaysia, 3–5 April 2009; pp. 334–338.
47. Kaur, K.; Kumar, S.; Baliyan, A. 5G: A new era of wireless communication. *Int. J. Inf. Technol.* **2020**, *12*, 619–624. [CrossRef]