sensors

MDPI

Article

An Autonomous Log Storage Management Protocol with Blockchain Mechanism and Access Control for the Internet of Things

Chien-Lung Hsu [1,2,3,4,5,*], Wei-Xin Chen [1] and Tuan-Vinh Le [2]

1. Department Information Management, Chang Gung University, Taoyuan 33302, Taiwan; codychen0704@gmail.com
2. Graduate Institute of Business and Management, Chang Gung University, Taoyuan 33302, Taiwan; tvle.cgu@gmail.com
3. Healthy Aging Research Center, Chang Gung University, Taoyuan 33302, Taiwan
4. Department Visual Communication Design, Ming Chi University of Technology, New Taipei 24301, Taiwan
5. Department Nursing, Taoyuan Chang Gung Memorial Hospital, Taoyuan 333, Taiwan

* Correspondence: clhsu@mail.cgu.edu.tw

Received: 11 October 2020; Accepted: 10 November 2020; Published: 12 November 2020

check for updates

Abstract: As the Internet of Things (IoT) has become prevalent, a massive number of logs produced by IoT devices are transmitted and processed every day. The logs should contain important contents and private information. Moreover, these logs may be used as evidences for forensic investigations when cyber security incidents occur. However, evidence legality and internal security issues in existing works were not properly addressed. This paper proposes an autonomous log storage management protocol with blockchain mechanism and access control for the IoT. Autonomous model allows sensors to encrypt their logs before sending it to gateway and server, so that the logs are not revealed to the public during communication process. Along with blockchain, we introduce the concept "*signature chain*". The integration of blockchain and signature chain provides efficient management functions with valuable security properties for the logs, including robust identity verification, data integrity, non-repudiation, data tamper resistance, and the legality. Our work also employs attribute-based encryption to achieve fine-grained access control and data confidentiality. The results of security analysis using AVSIPA toolset, GNY logic and semantic proof indicate that the proposed protocol meets various security requirements. Providing good performance with elliptic curve small key size, short BLS signature, efficient signcryption method, and single sign-on solution, our work is suitable for the IoT.

Keywords: attribute-based access control; digital forensics; evidence legality; sensor log; signature chain

1. Introduction

With the popularization of computers and rapid development of mobile network technologies, Internet of Things (IoT) has become prevalent. Various devices and entities can wirelessly be connected to the internet as long as they are equipped with sensors. Enabled with fifth generation (5G) technology, communication in IoT environments is performed with super low latency, high-peak data rates and massive network capacity [1]. Data aggregation and transmission in IoT networks have been significantly improved, in order to provide better efficiency of energy consumption, network control overhead, delay time, loss packet and aggregation rates [2]. Due to these advances, IoT has huge potentials to change the information technology, enhance reliability of communication systems, as well as improve our life quality. For example, in wireless body area networks (WBAN) [3], sensing data produced by wearable sensors provides rapid diagnostics, efficient treatments and valuable research data. In addition to healthcare [3–5], IoT applications have been implemented in a lot of domains, such as energy [6], vehicle [7,8], industrial systems [9], etc.

Logs generated by IoT devices contain important contents and sensitive information. The logs can be stored in cloud systems for convenient management. With the management tools, it is allowed to collect, store, analyze, archive, and dispose of the log information [10]. Specific uses of the logs include device monitoring [11], user behavior analysis [12], or digital forensics [13].

1.1. The Problems

Most IoT environments adopt centralized architecture for managing log storage. It suffers from internal threats since the data can be compromised by the management staffs. Moreover, sensitive information of the logs may be revealed to unauthorized persons. The adversary can also tamper with the log for illegal purposes. The integrity of the data needs to be preserved for forensic investigations when security incidents occur [14]. Communicating parties may repudiate data ownership for their own interests or motives, which causes challenges for digital forensics [15]. In addition, the legality of collected evidences must be ensured so that it provides an effective and efficient investigation process. In heterogeneous and distributed IoT environments with various devices and sensors, these concerns become prominent.

For addressing aforesaid problems, it is essential to propose a mechanism which provides integrity, availability, and legality of the logs. Access control to the log data should also be taken into account, which ensures the confidentiality where the log can only be viewed by legitimate parties. Furthermore, the mechanism should bear a rational implementation cost.

1.2. Related Works

Blockchain is a secure decentralized database that can track, verify, and safely protect the data from tampering [16]. It provides open and transparent mechanism that does not require third-party intervention. Blockchain has successfully been used in various sectors, such as transportation systems [17], medical record management [18], and so on. The concept of combining IoT and blockchain promotes the quality of data sharing services with automatic workflows [19]. Blockchain was proposed as a security solution for IoT by various works [20–22]. The research topics include immutable event logs and data access management [23], sensing data transaction [24], or IoT device authentication [25]. The digital forensics in the IoT architecture can be classified into various layers consisting of cloud forensics, network forensics, and device forensics [26]. As the forensics of massive IoT devices require a lot of resources [27], legal evidences helps in improving investigation efficiency in accordance with the demand of law enforcement agencies [28].

Taguchi et al. [29] proposed a distributed management method for logs using a blockchain scheme. The method provides data tamper resistance and increases access availability. Pourmajidi and Miranskyy [30] introduced Logchain, a blockchain-based log system. Their system can avoid log tampering and provides an immutable platform for the log storage. Hang and Kim [31] designed and implemented blockchain platform for ensuring data integrity of the IoT environments. Hang and Kim focused on the integration and management of IoT data and blockchain mechanism. Whereas, the IoT forensics framework designed by Ryu et al. [13] employed the blockchain to satisfy the requirements of IoT forensics. Their work achieves data tamper proof and non-repudiation in third party-less environments. Persistence and privacy of forensic data were also assured. In their design, specific data produced by IoT devices is written into the block for facilitating evidence collection during digital forensic investigations. Aforesaid works have certain strengths that meet several functionality and security requirements. However, internal confidentiality issue was not addressed since they did not introduce access control mechanism. Moreover, the legality of the evidence preservation in their works was uncertain.

Recently, Li et al. [32] proposed a secure fine-grained data sharing scheme for cloud computing. Even though their scheme provides lightweight computation with access control and forward secrecy, it was not introduced with blockchain mechanism. Zheng et al. [33] introduced a new attribute-based encryption scheme using blockchain technique. Their design did not employ digital signature to achieve legal security features. Sowjanya and Dasgupta [34] presented another attribute-based encryption scheme for WBAN. The scheme achieves good performance with elliptic curve cryptography and attribute-based encryption. Zhong et al. [35] also introduced an efficient access control scheme for smart healthcare. Nonetheless, both Sowjanya and Dasgupta [34] and Zhong et al. [35] did not include blockchain mechanism and digital signature technique in their works.

Given the drawbacks of existing works, we are motivated to design a new secure protocol providing log storage management capabilities, fine-grained access control, robust verification, and some other essential security properties. The new design should also meet the forensics requirements as well as the evidence legality.

1.3. Main Contributions

Our work proposes an autonomous log storage management protocol with blockchain mechanism and access control for IoT environments. The proposed protocol allows sensor to perform the signcryption of the log data based on its access policy. With access control mechanism, only the authorized users with appropriate attributes are able to unsigncrypt the message and view the log. Each entity in the system has to sign a signature during communication process so that they can be tracked for potential forensics. We integrate blockchain mechanism and digital signature technique to simultaneously achieve various properties. The contributions made in this paper can be described in the following.

- Autonomous model allows sensors to encrypt the logs before sending them to other entities (gateways and servers). Privacy of the logs therefore is fully protected throughout communication process. In this way, our protocol is even secure for communications via unreliable channels. Typical application of this model is WBAN, where wearable sensors encrypt health data before sending it to coordinators and healthcare providers for specific services.
- Since legality of blockchain signature remains uncertain, whereas digital signature satisfies various requirements with legal security properties [36], we introduce the concept "*signature chain*" in this work. A signature chain is composed by the signatures of all communicating entities of the system including sensors, gateway and server. The integration of blockchain and signature chain achieves valuable properties: robust identity verification, data integrity, tamper proof (insider attack resistance), ownership non-repudiation, and evidence legality. Thus, our work is completely helpful to the purposes of digital forensics.
- In our design, private blockchain is employed as a storage to conveniently and efficiently store and process the signature chain and ciphertexts, with various management functions. We adopt Proof of Work (PoW) [37] as the consensus algorithm in proposed private chain, in order to achieve above-mentioned security properties. Due to its full decentralization mechanism and immutability, public blockchain is integrated in our protocol to assure the trust of the private blockchain.
- Fine-grained access control with ciphertext policy attribute-based encryption is proposed in our work. It provides internal confidentiality in which only the legitimate users with specific appropriate attributes are allows to decrypt the ciphertexts and obtain the log plaintexts.
- We use AVISPA toolset and GNY logic to formally prove security correctness of the proposed protocol. Sematic security proof further indicates that our protocol satisfies various security requirements.
- Our work employs elliptic curve with small key size, short BLS signature, and efficient signcryption method to design the protocol with single sign-on solution. Therefore, our protocol bears low computation and storage overhead, which is suitable to the IoT.

- We provide practical implementation of the proposed protocol with specific use case, system construction and user interface.

1.4. Paper Structure

The paper is structured as follows. We present preliminaries of our work in Section 2. Section 3.1 provides system model of our work including all entities with communicating roles. Security goals are provided in Section 3.2, which are required for providing a secure communication with the proposed system model. Section 3.3 presents specific procedure and algorithms of the protocol. Section 4 presents security analysis of the proposed protocol including GNY logic, AVISPA toolset, and semantic proof. Performance experiment and analysis of the our protocol are provided in Section 5. Section 6 describes the implementation including practical procedures and system construction of our work. Finally, some concluding remarks and future works are given in Section 7 of the paper.

2. Preliminaries

Preliminaries of the paper include linear secret-sharing scheme, attribute-based encryption, signcryption, bilinear map, Boneh-Lynn-Shacham signature, blockchain, and single sign-on.

2.1. Linear Secret-Sharing Scheme

Linear Secret-Sharing Scheme (LSSS) proposed by Lewko and Waters [38] introduced how to use AND and OR gates to generate the matrices. LSSS consists of access policy matrix M and column vector v. The matrix M is composed by m rows and n columns, with the policy defined and stored by Boolean formula [39,40]. Whereas, the vector v is composed by $s, a_1, a_2, \ldots a_n \in_R Z_p$ that are the randomly selected numbers, in which s is the secret value. Multiplying matrix M with vector v will derive a column vector composed by $\lambda_1, \ldots \lambda_n$, where λ is the associated information of the secret value s. Access policy M contains a certain number of attributes. As long as users possess appropriate attributes, they can restore the secret value s.

2.2. Attribute-Based Encryption

Attribute-based encryption (ABE) was proposed by Sahai and Waters in 2005 [41]. In ABE, access policy defined by users considers various attributes. The attributes possessed by users determine whether they can meet the policy of data access. This advantage allows an efficient and flexible encryption process. ABE is categorized into two types: key policy attribute-based encryption (KP-ABE) [42,43] and ciphertext policy attribute-based encryption (CP-ABE) [44–46]. In the CP-ABE scheme, user's key is integrated with the attributes; and the ciphertext is associated with the access policy through the LSSS. When access policy is satisfied, the user can use the attribute key to decrypt the ciphertext. On the other hand, in the KP-ABE scheme, the user's key is associated with the access policy; and the ciphertext is integrated with the attributes. When the ciphertext meets the key's access policy, the user can decrypt the ciphertext.

2.3. Signcryption

Signcryption [47] is the combination of encryption and signature signing. The ciphertext and signature of the message are generated by performing the functions of both encryption and signature at the same time. Compared with the cumulative cost of separate encryption and signing process, this novel method is much more efficient. Signcryption method provides confidentiality, verification and non-repudiation of the given data. Attribute-based signcryption [48] combines the functions of encryption and signature on the attributes. Fine-grained access control can be associated with the signcrypted text to achieve robust message protection. This novel access control mechanism is well suited for data sharing in distributed environments. For example, users outsource their data to cloud storage, and can effectively share the data with other parties. The users who are granted the access can effectively obtain the data from anywhere through the network.

2.4. Bilinear Map

Selects a big number q, we have the elliptic curve: $E : y^2 = x^3 + ax + b \ mod \ q$. Let G_1 be a multiplicative cyclic group of order n, and g, g_1 and g_2 be the generators of G_1, a bilinear map from $G_1 \times G_1$ to G_T is a function $e : G_1 \times G_1 \rightarrow G_T$. The bilinear map provides the following characteristics and assumption [45,49,50]:

- Bilinear: If any two integers $x, y \in Z_p$ and generators $g, g_1, g_2 \in G_1$, then $e\left(g_1^x, g_1^y\right) = e(g_1, g_1)^{xy} = e\left(g_1^y, g_1^x\right)$, and $e(g_1.g_2, g) = e(g_1, g).e(g_2, g)$.
- Non-degenerate: There exists $g_1, g_2 \in G_1$ such that $e(g_1, g_2)$ is the generator of G_T.
- Computable: For any $g_1, g_2 \in G_1$, there exists a polynomial algorithm which can efficiently compute $e(g_1, g_2)$.
- Elliptic Curve Discrete Logarithm Problem (ECDLP): ECDLP is a special case of Discrete Logarithm Problem (DLP), and can be described as follows. Given $g_1, m \in G_1$, the problem is to find integer $x \in Z_p$ such that $g_1^x = m$.

2.5. Boneh-Lynn-Shacham Signature Scheme

Boneh-Lynn-Shacham (BLS) scheme [51] provides shorter signature length than Elliptic Curve Digital Signature Algorithm (ECDSA) [52], but with the same security level. BLS signature scheme can provide batch verification function, which allows to sign and verify multiple signatures at once. Given g_1, G_1, G_T defined in Section 2.4, plaintexts $M_1 : \{0,1\}^*$, $M_2 : \{0,1\}^*$, and hash function $H : \{0,1\}^* \in G_1$, the procedure of BLS scheme is described as follows:

- Key generation: Randomly choose an integer $x \in_R Z_p$, let x be private key, we have $Y = g_1{}^x$ is the corresponding public key.
- Signature generation: Use hash function H and private key x to sign the plaintext M_1 and generate signature $\sigma_1 = H(M_1)^x$.
- Signature verification: Based on plaintext M_1 and signature σ_1, the verification is to confirm the equation $(H(M_1), Y)e(\sigma_1, g_1)$. Correctness of the verification is proved as follows: $e(H(M_1), Y) = e(H(M_1), g_1{}^x) = e\left(H(M_1)^x, g_1\right) = e(\sigma_1, g_1)$.
- Batch signature verification: As stated, $\sigma_1 = H(M_1)^x$ and $\sigma_2 = H(M_2)^x$ are the signatures, the verification is to confirm $e(H(M_1)H(M_2), YY)e(\sigma_1\sigma_2, g_1g_1)$. The verification correctness is proved as follows: $e(H(M_1)H(M_2), YY) = e(H(M_1)H(M_2), g_1{}^x g_1{}^x) = e\left(H(M_1)^x H(M_2)^x, g_1g_1\right) = e(\sigma_1\sigma_2, g_1g_1)$.

2.6. Blockchain

Blockchain was proposed by Nakamoto in 2008 with its first application, Bitcoin [53]. Peer-to-peer (P2P) mechanism of blockchain with distributed ledger is employed to form decentralized networks. Nodes within the networks communicate with each other to confirm the validity of the transactions before they are uploaded to the blockchain. Due to a unique data structure, the content and transaction recorded in blockchain are unalterably protected. Blockchain provides decentralization [54], tamper resistance [55], and user anonymity [56]. There are three types of blockchain: public blockchain, private blockchain and consortium blockchain [57]. In public blockchain, everyone can conduct transactions, verifications and relevant contributions. It is recognized as the concept of completely decentralized open network. Whereas, private blockchain network partly achieves the decentralization since its design allows a single organization to hold central authority. Data access in private blockchain is only granted to a certain number of users based on specific purposes. The consortium blockchain mechanism is similar to the private blockchain. The difference is consortium blockchain includes multiple organizations, which can provide business-to-business (B2B) services.

2.7. Single Sign-on

Single Sign-On (SSO) [58] provides multi-server environment that allows users to use a single password to log in multiple servers. After completing identity authentication with one sever, users can freely access the services on other severs within the network, without having to repeat authentication procedure. The benefits of SSO solution can be summarized as follows: (1) Avoids the confusion of users when they must store massive credentials at the same time in single-server environments; (2) Allows central service provider to conveniently manage the authentication information of users; and (3) Significantly reduces credential storage overhead.

3. The Proposed Log Storage Management Protocol with Blockchain Mechanism and Access Control

In this section, we describe system model and security goals of the proposed protocol. Thereafter, detailed procedure of our protocol is presented. Cryptographic functions and notations used in the protocol are described in Table 1.

Table 1. Cryptographic functions and notations used in this paper.

Notations	Description
PP	Public parameters
MSK	Secret parameters
Y	Public key of the authority
α	Secret key of the authority
C'	Public signcryption key
s	Private signcryption key
M	Log plaintext
C	Log ciphertext
σ_{CT}	Log signature
σ_{IoT}	Sensor signature
σ_{GW}	Gateway signature
σ_{Srv}	Server signature
t	Timestamp
IP	Internet protocol address
$H()$	Secure one-way hash function
$ECDSA(\ )$	ECDSA signature function
$Verify:\ _{ECDSA}(\ )$	Verifying ECDSA signature function
$\vec{v}_e$	Secret vector
BF	Access policy based on Boolean formula
x	Total number of attributes
ID_i	Identity of the user

3.1. System Model

Our system model includes 11 roles: attribute authority, SSO server, timestamp server, sensor (IoT device) and agent, gateway, blockchain server, private blockchain, public blockchain, storage cluster, and user. The attribute authority generates public and secret parameters used in entire communication process. In particular, it sends public parameters to the sensor for log signcryption. The authority also computes private attribute key and transmits it to the user for log unsigncryption. The SSO server provides single sign-on login, allowing users to use a single password to enter multiple servers in multi-server environment. The timestamp server derives timestamp parameters for the system. The sensor is a sensing device which contacts the environment, and generates the logs. The agent is installed inside the sensor, and is responsible for defining the access policy, as well as signcrypting the logs to generate ciphertexts. The gateway verifies the signature included in the ciphertexts to ensure the correctness of the log ciphertext. Blockchain server is responsible for storing the signcrypted text in the storage cluster. Moreover, the server also generates private blocks from single signatures, and public block from multiple signatures, and then writes them into private blockchain and public blockchain respectively. The private blockchain stores signature chains and related information. The public blockchain records corresponding data from the private blockchain, and stores the batch signatures, with fully decentralized nature. The user logs in to the blockchain server through the SSO server, obtains the ciphertext, and uses the attribute private key to unsigncrypt it to view the log plaintext. The user can also verify the validity of the related information stored in private blockchain and public blockchain. System model of the proposed protocol is depicted in Figure 1.

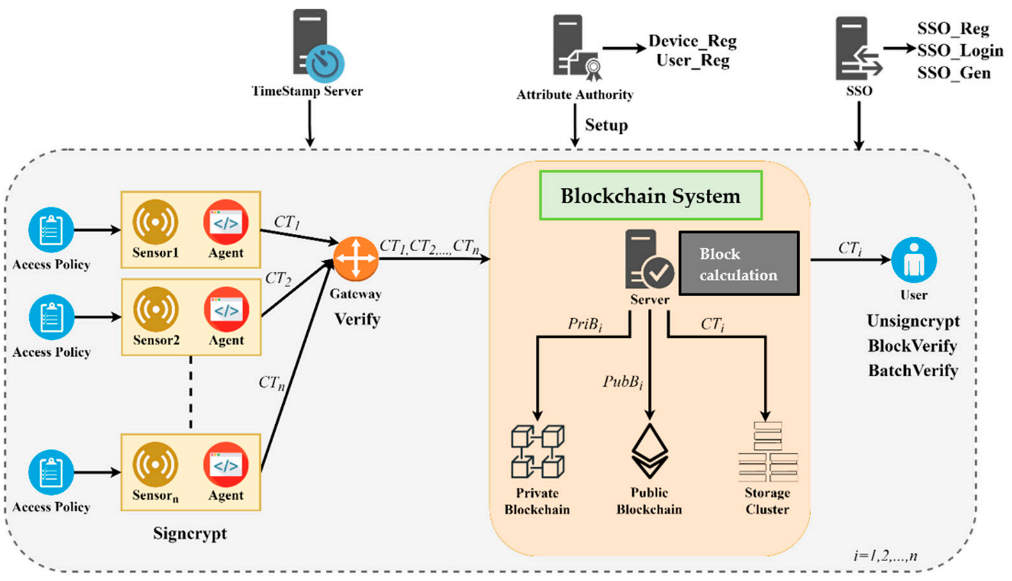


Figure 1. System model of the proposed protocol.

Signature chain is composed by the signatures signed by the sensor, the gateway and the server in each communication session. Data in private blockchain is signed using two types of signature schemes including BLS and ECDSA. Each block contains a single signature chain. These chains are immutably stored in blockchain for further security purposes. Figure 2 depicts the design of private blockchain and signature chain of our work.

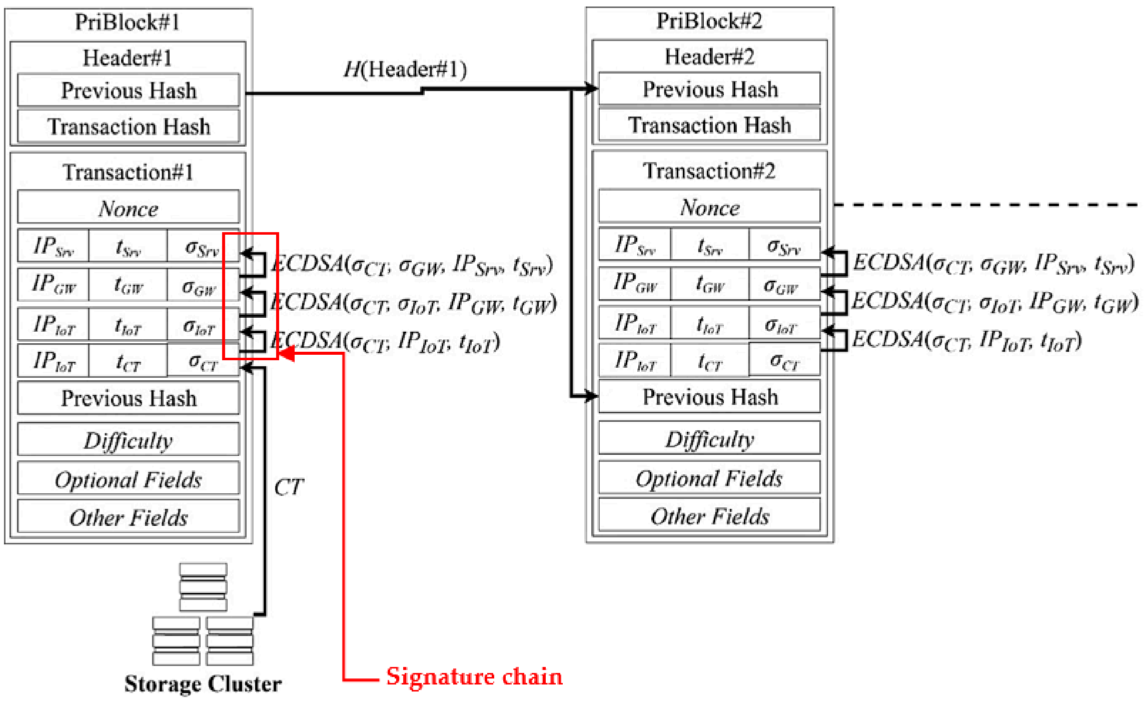

PriBlock#1
Header#1
Previous Hash
Transaction Hash
Transaction#1
Nonce
IP_{Srv} t_{Srv} σ_{Srv}
IP_{GW} t_{GW} σ_{GW}
IP_{IoT} t_{IoT} σ_{IoT}
IP_{IoT} t_{CT} σ_{CT}
Previous Hash
Difficulty
Optional Fields
Other Fields
H(Header#1)
$ECDSA(\sigma_{CT}, \sigma_{GW}, IP_{Srv}, t_{Srv})$
$ECDSA(\sigma_{CT}, \sigma_{IoT}, IP_{GW}, t_{GW})$
$ECDSA(\sigma_{CT}, IP_{IoT}, t_{IoT})$
CT
Storage Cluster
Signature chain
PriBlock#2
Header#2
Previous Hash
Transaction Hash
Transaction#2
Nonce
IP_{Srv} t_{Srv} σ_{Srv}
IP_{GW} t_{GW} σ_{GW}
IP_{IoT} t_{IoT} σ_{IoT}
IP_{IoT} t_{CT} σ_{CT}
Previous Hash
Difficulty
Optional Fields
Other Fields
$ECDSA(\sigma_{CT}, \sigma_{GW}, IP_{Srv}, t_{Srv})$
$ECDSA(\sigma_{CT}, \sigma_{IoT}, IP_{GW}, t_{GW})$
$ECDSA(\sigma_{CT}, IP_{IoT}, t_{IoT})$

Figure 2. Private blockchain and signature chain in our system model.

3.2. Security Goals

Security problems are always big concerns in any information systems. The proposed system model includes various parties in a public communication environment. External invasion and security attacks should also be considered for providing a high security environment. We expect that our protocol can satisfy the following security requirements.

- *Secure decryption key*: After the sensor signcrypts the logs, the user attempts to compute the decryption key to decrypt the ciphertext and access the logs. Only legitimate user possessing appropriate attributes is able to compute the correct key.
- *Robust verification*: The digital signature signed by the sensor makes sure that the log data is truly produced and transmitted by the sensor itself. Any parties participating in the communication can verify the validity of the signature.
- *Dataunforgeability*: Only the sensor with its own private key is able to sign the message. We desire to warrant that the signcryption key of the sensor is kept secret to the sensor only, during communication process. In this way, the adversary cannot forge the signature and impersonate the sensor.
- *Datatampering resistance*: The signatures may be modified for obstruction purposes. In addition, the signer may re-sign the message to tamper with its data. These issues should be addressed so that security properties of digital signature are guaranteed.
- *Data confidentiality*: The log data must be kept confidential to the legal parties under any circumstances. Users within the system are allowed to access the logs only if they possess required attributes.
- *Non-repudiation*: Once the logs are signed, signers cannot repudiate them for any own interests. This property is helpful to digital forensic investigations.
- *Data integrity*: This property makes sure that the logs must be originally sent by the sensor without any modifications to its contents.
- *Perfect forward secrecy*: This security goal is required for the long-term decryption key. It ensures that if the adversaries successfully calculate the current decryption key, they still cannot use it to compromise the logs in previous communications sessions.

3.3. Procedure of the Proposed Protocol

Communication in the proposed protocol is carried out including 13 phases: initialization phase, device registration phase, SSO registration phase, SSO login phase, SSO password generation phase, user registration phase, log signcryption phase, log verification phase, private block calculation phase, private block verification phase, log unsigncryption phase, public block calculation phase and public block verification phase.

3.3.1. System Initialization Phase

In initialization phase, the attribute authority generates public and secret parameters used in entire system. The attribute authority selects a big number q, and determine the elliptic curve: $E : y^2 = x^3 + ax + b \ mod \ q$. It then generates a cyclic group G_1 and bilinear map $e : G_1 \times G_1 \rightarrow G_T$. g is set as the generator of G_1. Next, a set of system attributes is determined by $u_s = \{att_1, att_2, \ldots, att_x\}$. The authority selects the corresponding random numbers $\{Q_1, Q_2, \ldots, Q_x\} \in_R G_1$, based on attribute set u_s, and chooses a secure one-way hash function $H : \{0,1\}^* \in G_1$. It randomly selects $\alpha, \beta \in_R Z_q$, and compute $B = g^{\beta}$ and public key $Y = e(g, g)^{\alpha}$. Finally, the authority generates public parameter $PP = (g, B, Y, H, Q_x, e, G_1, G_T)$ and secret parameter $MSK = (\alpha, \beta, u_s)$. Specific steps of this phase are described in Algorithm 1.

Algorithm 1: System initialization.

```
Input: Initial parameters.
Output: PP, MSK.
1:  Select a big number $q$, and determine the elliptic curve: $E : y^2 = x^3 + ax + b \ mod \ q$.
2:  Generate a cyclic group $G_1$ and bilinear map $e : G_1 \times G_1 \rightarrow G_T$.
3:  Set $g$ as the generator of $G_1$.
4:  Determine system attribute set $u_s = \{att_1, att_2, \ldots, att_x\}$.
5:  Select $\{Q_1, Q_2, \ldots, Q_x\} \in_R G_1$, based on attribute $u_s$.
6:  Choose a secure one-way hash function $H : \{0,1\}^* \in G_1$.
7:  Randomly select $\alpha, \beta \in_R Z_q$.
8:  Compute $B = g^{\beta}$.
9:  Compute public key $Y = e(g, g)^{\alpha}$.
10: Generate $PP = (g, B, Y, H, Q_x, e, G_1, G_T)$ and $MSK = (\alpha, \beta, u_s)$.
```

3.3.2. SSO Registration Phase

In this phase, the user U_i uses a smart card to register with the SSO server for obtaining multiple services. SSO registration procedure is provided in Algorithm 2 as follows. The user U_i enters SID_i and SPW_i into smart card. The smart card generates a random number r_i, and computes $A_i = H(SPW_i) \oplus H(r_i \parallel SID_i)$. The SSO sever then stores r_i and A_i.

Algorithm2: SSO registration.

```
Input: $SID_i$, $SPW_i$.
Output: $r_i$, $A_i$.
1: $U_i$ enters $SID_i$ and $SPW_i$ into smart card.
2: Smart card generates $r_i$.
3: Smart card computes $A_i = H(SPW_i) \oplus H(r_i \parallel SID_i)$.
4: SSO sever stores $r_i$ and $A_i$.
```

3.3.3. SSO Login Phase

The user U_i enters SID_i and SPW_i into SSO sever for verifying his/her legitimacy. The user U_i enters SID_i and SPW_i into the SSO server. The SSO server computes $A_i{}' = H(SPW_i) \oplus H(r_i \parallel SID_i)$. It then compares A_i' and A_i, in order to verify legitimacy of the user U_i. Procedure of this phase is presented by Algorithm 3.

Algorithm 3: SSO login.

```
Input: $SID_i$, $SPW_i$.
Output: True or False.
1:  $U_i$ enters $SID_i$ and $SPW_i$.
2:  SSO server computes $A_i{}' = H(SPW_i) \oplus H(r_i \parallel SID_i)$.
3:  SSO server compares $A_i{}'$ and $A_i$.
4:      if above check holds, then output True, and confirm legitimacy of $U_i$.
5:      otherwise, output False, and terminate the login.
```

3.3.4. SSO Password Generation Phase

The use U_i enters SID_i, SPW_i and ID_i so that the server can generate an SSO password. In this way, the user U_i can obtain services from multiple servers using this single password. The user U_i enters SID_i, SPW_i and ID_i into SSO server. The SSO server generates PW_i from SID_i, SPW_i and ID_i. Procedure of this phase is described by Algorithm 4.

Algorithm 4: SSO password generation.

```
Input: $SID_i, SPW_i, ID_i$.
Output: $PW_i$.
1:  $U_i$ enters $SID_i$, $SPW_i$ and $ID_i$ into SSO server.
2:  SSO server generates $PW_i = SSOgen(SID_i, SPW_i, ID_i)$.
```

3.3.5. Device Registration Phase

In this phase, the $sensor_i$ registers with the attribute authority for further communication. The sensor and the authority perform necessary steps for device registration, as presented in Algorithm 5. The $sensor_i$ sends DID_i to the attribute authority. The authority verifies DID_i, then sends public parameter PP to the $sensor_i$.

Algorithm 5: Device registration.

```
Input: $DID_i$.
Output: $PP$.
1:  $Sensor_i$ sends $DID_i$ to attribute authority.
2:  Attribute authority verifies $DID_i$.
3:  Attribute authority sends $PP$ to $sensor_i$.
```

3.3.6. User Registration Phase

The user U_i uses his/her identity ID_i to register and obtain the attribute private key from the attribute authority. This procedure is performed by the attribute authority, as specified in Algorithm 6. The user U_i first sends his/her identity ID_i to the attribute authority. Upon the received message, the attribute authority verifies ID_i. It then randomly chooses $t_{ID_i} \in_R Z_q$, and uses g, α, β, and t_{ID_i} to compute K^i, $L^i = g^{t_{ID_i}}$ and

$K_j^i = Q_j^{t_{ID_i}} (\forall\ j\ \in x)$. The authority generates attribute private key $SK_{ID_i} = (K^i, L^i, K_j^i)$, and sends SK_{ID_i} to the user U_i.

Algorithm 6: User registration.

```
Input: $ID_i, PP, MSK$.
Output: $SK_{ID_i}$.
1:  Receive $ID_i$ from $U_i$.
2:  Verify $ID_i$.
3:  Choose $t_{ID_i} \in_R Z_q$.
4:  Compute $K^i = g^{\alpha+(\beta t_{ID_i})}$.
5:  Compute $L^i = g^{t_{ID_i}}$.
6:  Compute $K_j^i = Q_j^{t_{ID_i}} (\forall\ j\ \in x)$.
7:  Send $SK_{ID_i} = (K^i, L^i, K_j^i)$ to $U_i$.
```

3.3.7. Log Signcryption Phase

In this phase, the $sensor_i$ is allowed to signcrypt the log, based on attribute-based access policy, Boolean formula BF and public parameters PP. The $sensor_i$ performs specific steps in Algorithm 7 for the log signcryption procedure. It sets LSSS matrix m_e by Boolean formula BF, and randomly generates random number $r_j \in_R Z_q(\forall\ j\ \in x)$ and a secret vector $\overrightarrow{v}_e$ composed by secret signcryption key s and j attributes. Then, the $sensor_i$ uses matrix m_e, vector $\overrightarrow{v}_e$, signcryption key s and parameter g to compute $\lambda_e = m_e \overrightarrow{v}_e$ and $C' = g^s$. Parameters B, r_j, g and λ_e are used to compute $C_j = g^{\beta\lambda_e} Q_j^{-r_j} (\forall\ j\ \in x)$ and $D_j = g^{r_j} (\forall\ j\ \in x)$. Next, the $sensor_i$ computes log ciphertext $C = MY^s$, its hash value $h = H(C)$ and signature $\sigma_{CT} = h^{Y \cdot x^s}$. Thereafter, ECDSA signature $\sigma_{IoT} = ECDSA(\sigma_{CT}, IP_{IoT}, t_{IoT})$ is derived. The $sensor_i$ generates ciphertext $CT = \left(\sigma_{CT}, C, C_j, C', D_j, m_e, IP_{IoT}, t_{CT}\right)$, and then sends it to the gateway. Finally, $\delta_{IoT} = (\sigma_{IoT}, IP_{IoT}, t_{IoT})$ is stored by the $sensor_i$.

Algorithm7: Log signcryption.

```
Input: $BF, PP, M$.
Output: $CT, \delta_{IoT}$.
1:  Set LSSS matrix $m_e$ by $BF$.
2:  Randomly generate $r_j \in_R Z_q(\forall\ j\ \in x)$.
3:  Generate $\overrightarrow{v}_e = \begin{bmatrix} s \\ att_1 \\ att_2 \\ \vdots \\ att_j \end{bmatrix} \in_R Z_q(\forall\ j\ \in x)$.
4:  Use $m_e$ and $\overrightarrow{v}_e$ to compute $\lambda_e = m_e \overrightarrow{v}_e$.
5:  Use $s$ and $g$ to compute $C' = g^s$.
6:  Use $B$, $r_j$ and $\lambda_e$ to compute $C_j = g^{\beta\lambda_e} Q_j^{-r_j} (\forall\ j\ \in x)$.
7:  Use $r_j$ and $g$ to compute $D_j = g^{r_j} (\forall\ j\ \in x)$.
8:  Compute $C = MY^s$.
9:  Compute $h = H(C)$.
10: Compute $\sigma_{CT} = h^{Y \cdot x^s}$.
11: Perform $\sigma_{IoT} = ECDSA(\sigma_{CT}, IP_{IoT}, t_{IoT})$.
12: Send $CT = \left(\sigma_{CT}, C, C_j, C', D_j, m_e, IP_{IoT}, t_{CT}\right) with(m_e, \rho(j))$ to gateway.
13: Store $\delta_{IoT} = (\sigma_{IoT}, IP_{IoT}, t_{IoT})$.
```

3.3.8. Log Verification Phase

The gateway verifies the validity of the signatures and the ciphertext CT, based on public parameters PP. If the verifications are valid, the gateway will send it to the blockchain server. This phase is performed by the gateway with Algorithm 8. The gateway checks $e(h, Y_{.x}C')e(\sigma, g)$ and $ECDSA(\delta_{IoT}, \sigma_{CT})$. If the checks hold, it sends ciphertext CT to the blockchain server. The ciphertext is then stored at cluster storage. The gateway computes signature $\sigma_{GW} = ECDSA(\sigma_{CT}, \sigma_{IoT}, IP_{GW}, t_{GW})$, sets $\delta_{GW} = (\sigma_{GW}, IP_{GW}, t_{GW})$, and stores δ_{GW}.

Algorithm 8: Log Verification.

```
Input: $CT, \delta_{IoT}, PP$.
Output: $\delta_{GW}$.
1:  Verify : $e(h, Y_{.x}C')e(\sigma_{CT}, g)$.
2:  $Verify_{ECDSA}(\delta_{IoT}, \sigma_{CT})$.
3:      if above checks hold, then continue with step 5.
4:      otherwise, terminate the session.5: Send $CT$ to blockchain server, $CT$ is then stored in
    cluster storage.
5:  Compute signature $\sigma_{GW} = ECDSA(\sigma_{CT}, \sigma_{IoT}, IP_{GW}, t_{GW})$.
6:  Store $\delta_{GW} = (\sigma_{GW}, IP_{GW}, t_{GW})$.
```

3.3.9. Log Unsigncryption Phase

In log unsigncryption phase, the user U_i is allowed to unsigncrypt the ciphertext CT to view the log M, using private key SK_{ID_i} (with appropriate attributes) and public parameter PP. The user U_i uses Algorithm 9 to complete this procedure. Parameter w_i is first restored from the matrix m_e with appropriate attributes. The user U_i then uses parameters C, C', D_j and private key $SK_{ID_i} = (K^i, L^i, K^i_j)$ to compute decryption key Y^s. Value $h = H(C)$ is computed for checking $e(h, Y_{.x}C')e(\sigma, g)$ based on parameters g, Y, C and C'. At last, the user U_i uses Y^s to decrypt log ciphertext and obtain the log by $M = C.Y^{s-1}$.

Algorithm 9: Log unsigncryption.

```
Input: $CT, SK_{ID_i}, PP$.
Output: $M$.
1:  Restores $w_i$ from $m_e$ with appropriate attributes: $\begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = m_e w_j$.
2:  Use $C, C', D_j$ and $SK_{ID_i}$ to compute decryption key: $Y^s = e(g,g)^{s\alpha} = \frac{e(C', K^i)}{\Pi_{j \in x}\left(e(C_j, L^i)e\left(D_j K^i_j\right)\right)^{w_j}}$.
3:  Compute $h = H(C)$.
4:  Use $g$, $Y$, $C$ and $C'$ to Verify : $e(h, Y_{.x}C')e(\sigma_{CT}, g)$.
5:      if above check holds, then continue with step 7.
6:      otherwise, terminate the session.
7:  Use $Y^s$ to decrypt $C$ and obtain $M = C.Y^{s-1}$.
```

3.3.10. Private Block Calculation Phase

Based on the ciphertext CT and some other information, the blockchain server calculates private block data, then writes it to the blockchain. At first, the server retrieves previous hash from the private blockchain, and verifies ECDSA signature σ_{GW}. It computes signature $\sigma_{Srv} = ECDSA(\sigma_{CT}, \sigma_{GW}, IP_{Srv}, t_{Srv})$, and sets $\delta_{Srv} = (\sigma_{Srv}, IP_{Srv}, t_{Srv})$ and $\delta_{CT} =$

$(\sigma_{CT}, IP_{Srv}, t_{CT})$. The initial *Nonce* value is set as 0. The server then iteratively compute $H(Nonce \parallel PreviousHash \parallel \delta_{Srv} \parallel \delta_{GW} \parallel \delta_{IoT} \parallel \delta_{CT} \parallel OptionalFields \parallel OtherFields)$, which must be smaller than the $Difficulty$ level. It sets $Nonce = Nonce + 1$ if above check does not hold, and re-compute the hash. The computing is completed if above condition holds. Block data $PriB = (\delta_{Srv}, \delta_{GW}, \delta_{IoT}, \delta_{CT}, Nonce, Difficulty, Optional\ Fields$ is generated and written to the private blockchain. Finally, the server receives a corresponding block number $PriBlockNo_n$. Private blockchain calculation procedure is further specified in Algorithm 10.

3.3.11. Private Block Verification Phase

In this phase, the user U_i verifies the validity of the private block *PriB*. Specific steps of private block verification are performed by the user U_i with Algorithm 11. The user U_i verifies whether $H(Nonce \parallel PreviousHash \parallel \delta_{Srv} \parallel \delta_{GW} \parallel \delta_{IoT} \parallel \delta_{CT} \parallel OptionalFields \parallel OtherFields) < Difficulty$. The validity of ECDSA signatures $(\delta_{IoT}, \sigma_{CT})$, $(\delta_{GW}, \sigma_{CT})$ and $(\delta_{Srv}, \sigma_{CT})$, and $e(h, Y_{.x}C')e(\sigma_{CT}, g)$ are also verified. The system outputs *True* if above verifications hold, otherwise outputs *False*.

Algorithm 10: Private block calculation.

```
Input: $CT, \delta_{IoT}, \delta_{GW}, PreviousHash, OptionalFields, OtherFields$.
Output: $PriB$.
1:  Retrieve previous hash from private blockchain.
2:  Verify$_{ECDSA}(\delta_{GW}, \sigma_{CT})$.
3:      if above check holds, then continue with step 5.
4:      otherwise, terminate the session.
5:  Compute $\sigma_{Srv} = ECDSA(\sigma_{CT}, \sigma_{GW}, IP_{Srv}, t_{Srv})$.
6:  Set $\delta_{Srv} = (\sigma_{Srv}, IP_{Srv}, t_{Srv})$.
7:  Set $\delta_{CT} = (\sigma_{CT}, IP_{Srv}, t_{CT})$.
8:  Set initial $Nonce$ value as 0.
9:      while
    $H(Nonce \parallel PreviousHash \parallel \delta_{Srv} \parallel \delta_{GW} \parallel \delta_{IoT} \parallel \delta_{CT} \parallel OptionalFields \parallel OtherFields) < Difficulty$ do.
10:         $Nonce = Nonce + 1$.
11: end while.
12: Generate $PriB = (\delta_{Srv}, \delta_{GW}, \delta_{IoT}, \delta_{CT}, Nonce, Difficulty, Optional\ Fields)$.
13: Write $PriB$ to private blockchain.
14: Receive $PriBlockNo_n$ from private blockchain.
```

Algorithm 11: Private block verification.

```
Input: $CT, PriB$.
Output: True or False.
1:  Verify if: $H(Nonce \parallel PreviousHash \parallel \delta_{Srv} \parallel \delta_{GW} \parallel \delta_{IoT} \parallel \delta_{CT} \parallel OptionalFields \parallel OtherFields) < Difficulty$.
2:  $Verify_{ECDSA}(\delta_{IoT}, \sigma_{CT})$.
3:  $Verify_{ECDSA}(\delta_{GW}, \sigma_{CT})$.
4:  $Verify_{ECDSA}(\delta_{Srv}, \sigma_{CT})$.
5:  $Verify: e(h, Y_{.x}C')e(\sigma, g)$.
6:      if verifications hold, then output True.
7:      otherwise, output False.
```

3.3.12. Public Block Calculation Phase

The blockchain sever computes batch signature from multiple signatures, and write it to public blockchain. In this. way, credibility of the signatures is enhanced with immutability feature. The procedure is performed by the blockchain server with Algorithm 12 as follows. The server retrieves block number $PriBlockNo_n$ from the corresponding block $PriB_n$, and multiple signatures σ_{CT_n} from multiple private blocks $PriB_n$. Batch signature $BSig = \sigma_{CT_1}\sigma_{CT_2}\ldots\sigma_{CT_n}$ is computed to generate public block $PubB = (PriBlockNo_n, BSig, Optional\ Fields)$. Next, the server writes $PubB$ to the public blockchain, and receives the corresponding block number $PubBlockNo_n$.

3.3.13. Public Block Verification Phase

In this phase, the user U_i verifies the batch verification of data recorded in public blockchain, based on $CT_n, PubB$ and PP. Algorithm 13 is performed by the user U_i to complete this procedure. The user U_i confirms if $PriBlockNo_n$ is available on the private chain, then obtains the corresponding blocks $PriB_n$. Next, the user U_i verifies whether $BSig$ matches the signatures in the private blocks $PriB_n$. Value $h_n = H(C_n)$ is computed based on n log ciphertext C_n. Finally. the user U_i verifies the validity of the batch signature: $e(h_n,\ Y_{.xn}C'_n)e(BSig, g_n)$. The system outputs $True$, meaning the verification is successful if the check holds, otherwise outputs $False$.

Algorithm 12: Public block calculation.

```
**Input**: $PriBlockNo_n$.
**Output**: $PubB$.
1:  Retrieve $PriBlockNo_n$ from corresponding block $PriB_n$.
2:  Retrieve $\sigma_{CT_n}$ from $PriB_n$.
3:  Compute $BSig = \sigma_{CT_1}\sigma_{CT_2}\ldots\sigma_{CT_n}$.
4:  Generate $PubB = (PriBlockNo_n, BSig, Optional\ Fields)$.
5:  Write $PubB$ to public blockchain.
6:  Receive $PubBlockNo_n$ from public blockchain.
```

Algorithm 13: Public block verification.

```
**Input**: $PubB, CT_n, PP$.
**Output**: $True\ or\ False$.
1:  Confirm if $PriBlockNo_n$ is available on private blockchain.
2:  $Verify : BSig\sigma_{CT_1}\sigma_{CT_2}\ldots\sigma_{CT_n}$.
3:      **if** above check holds, **then** continue with step 6.
4:      **otherwise**, terminate current session.
5:  Compute $h_n = H(C_n)$, based on $n$ log ciphertext $C_n$.
6:  $Verify : e(h_i,\ Y_{.xn}C'_n)e(BSig,\ g_n)$.
7:      **if** verification holds, **then** output $True$.
8:      **otherwise**, output $False$.
```

4. Security Analysis

In this section, we use AVISPA toolset and GNY logic to verify security correctness of the proposed protocol. In addition, we prove that our protocol meets various security requirements based on the semantic proof.

4.1. Protocol Simulation Using AVISPA Toolset

We verify the security properties of the proposed protocol by employing Automated Validation of Internet Security Protocols and Applications (AVISPA) [59]. AVISPA tool uses HLPSL [60] as its formal language, and integrates different back-ends in the verification techniques. The back-ends includes On-the-fly Model-Checker (OFMC), Constraint Logic based Attack Searcher (CL-AtSe), SAT-based ModelChecker (SATMC), and Tree Automata based on automatic approximations for the analysis of security protocols (TA4SP). However, the SATMC and TA4SP back-ends are not frequently used since they cannot verify the protocols using algebraic properties of modular exponentiation and XOR operator. In this simulation, AVISPA tool is integrated with Security Protocol Animator (SPAN) for providing a user-friendly application interface.

In our protocol, single signature verifications conducted by the gateway, the user and the blockchain sever are identical. We therefore only simulate the verification of the user in the log unsigncryption phase. Moreover, since the gateway and the sever just merely receives and verifies the signature, and they do not make any changes to the signature, we assume that the user directly receives signature from the sensor. We use the similar arguments of single signature verification for verifying the correctness of the batch signature.

HLPLS codes in the simulation was specified with three roles: authority (A), sensor (S), and user (U). The symmetric key *Kau* is used to protect the information in user registration phase. Based on our protocol, (α, β, u_s) are secret keys of the authority, but for simplicity of the simulation, we only include α. Since AVISPA only support three types of operators (concatenation, exclusive or, and exponentiation), the multiplication and paring function can be performed as hash functions. We also assume ECDSA and inv(ECDSA) are public key and private key respectively, for performing ECDSA algorithms. Specific HLPSL specifications of the user, the authority, and the sensor are provided in Figures 3–5 respectively.

```
role user (A, S, U: agent, Kau: symmetric_key, ECDSA: public_key, H, Add, Mul, Pair: hash_func, SND, RCV: channel (dy))
played_by U def=
local State: nat,
IDi, TIDi, Ys, Alpha, SK, Ss, Ce, C, Hs, Sig, G, M, Y, X, IP, T: text
init State:= 0
transition
% User registration phase
1.   State = 0 /\ RCV(start) =|>
State':= 1
/\ IDi':= new() /\ SND({IDi'}_Kau) /\ secret(IDi',idi,{A,U})
2.   State = 1 /\ RCV({exp(G,Add(Alpha'.TIDi'))}_Kau) =|>
State':= 2
%/\ Store SK % Log verification phase
3.   State = 0 /\ RCV(exp(G,Ss').Mul(M.exp(Y,Ss')).({X'.IP.T}_inv(ECDSA))) =|>
State':= 1
%/\ Use ECDSA to verify ECDSA signature %/\ Use Y and C = Mul(M.exp(Y,S)) to verify log signature
/\ request(S,U,ss,Ss')
```

Figure 3. *Cont.*

```
% Log unsigncryption phase
/\ Ys’:= Pair(SK.exp(G,Ss’))
%/\ Use Ys to decytp C and obtain view the log data M
end role
```

Figure 3. HLPLS specification of user role.

```
role authority (A, S, U: agent, Kau: symmetric_key, ECDSA: public_key, H, Add, Mul, Pair: hash_func, SND, RCV: channel
(dy))
played_by A def=
local State: nat,
IDi, TIDi, Ys, Alpha, SK, Ss, Ce, C, Hs, Sig, G, M, Y, X, IP, T: text
init State:= 0
transition
% User registration phase
1. State = 0 /\ RCV({IDi’}_Kau) =|>
State’:= 1
%/\ Verify IDi
/\ Alpha’:= new() /\ TIDi’:= new() /\ SK’:= exp(G,Add(Alpha’.TIDi’))
%/\ The value SK’ is computed for three attributes A, B, C
/\ SND({SK’}_Kau)
/\ secret(SK’,sk,{U,A}) /\ secret(Alpha’,alpha,{A})
end role
```

Figure 4. HLPLS specification of attribute authority role.

```
role sensor (A, S, U: agent, Kau: symmetric_key, ECDSA: public_key, H, Add, Mul, Pair: hash_func, SND, RCV: channel (dy))
played_by S def=
local State: nat,
IDi, TIDi, Ys, Alpha, SK, Ss, Ce, C, Hs, Sig, G, M, Y, X, IP, T: text
init State:= 0
transition
% Log signcryption phase
1. State = 0 /\ RCV(start) =|>
State’:= 1
/\ Ss’:= new() %/\ Choose Ss for computation of three attributes A, B, C; M is the log data
/\ Ce’:= exp(G,Ss’) /\ C’:= Mul(M.exp(Y,Ss’)) /\ Hs’:= H(C’) /\ X’:= exp(Hs’,Mul(Y.Ss’)) %/\ X: log signature
/\ Sig’:= {X’.IP.T}_inv(ECDSA) %/\ IP: IP adrress of the Sensor; T: timestamp
/\ SND(Ce’.C’.Sig’)
/\ secret(Ss’,ss,{S})
/\ witness(S,U,ss,Ss’)
end role
```

Figure 5. HLPLS specification of sensor role.

In addition, Figure 6 provides the specification of the session role where its composition consisting of all main roles is specified. Environment role illuminated in Figure 7 specifies all relevant components within the communication environment including symmetric keys, functions, protocol id, and intruder knowledge. In simulated environment, we can see that the intruder in turn replaces the roles of the authority, the sensor and the user in respective sessions in which, he/she attempts to compromise the simulated system. We consider four secrecy goals and one authentication goal described in the following:

- "*secrecy_of idi*" represents the identity ID_i that the user uses to register with the authority via a secure channel, it is kept secret to the user and the authority.
- "*secrecy_of sk*" represents SK_{ID_i} that the authority sends to the user, it is also kept secret to the user and the authority.
- "*secrecy_of alpha*" represents the secret value α, it is kept secret to the authority.
- "*secrecy_of ss*" represents the secret key s, it is kept secure to the sensor.
- "*authentication_on ss*": the user authenticates the sensor on s.

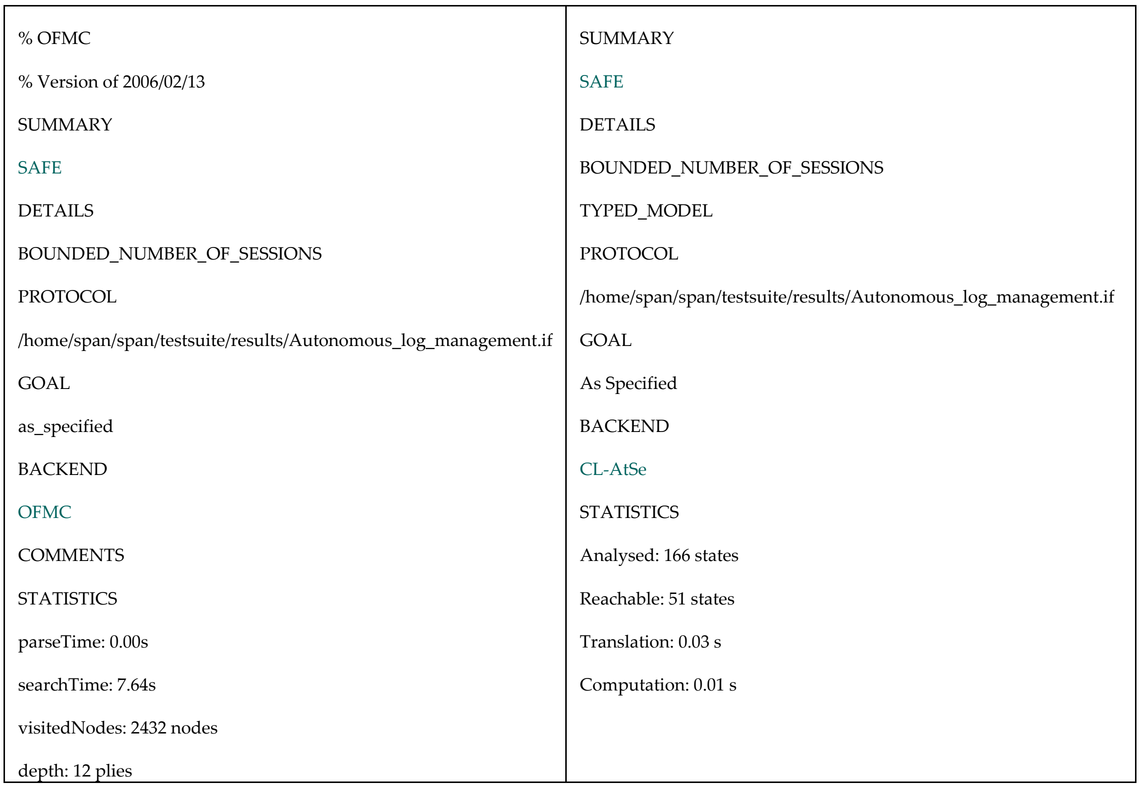
```
role session (A, S, U: agent, Kau: symmetric_key, ECDSA: public_key, H, Add, Mul, Pair: hash_func) def=
local SA, RA, SS, RS, SU, RU: channel (dy)
composition
authority (A,S,U,Kau,ECDSA,H,Add,Mul,Pair,SA,RA) /\ sensor (A,S,U,Kau,ECDSA,H,Add,Mul,Pair,SS,RS)
/\ user (A,S,U,Kau,ECDSA,H,Add,Mul,Pair,SU,RU)
end role
```

Figure 6. HLPLS specification of session role.

```
role environment() def=
const a, s, u, i: agent,
kau, kui: symmetric_key,
ecdsa, ki: public_key,
h, add, mul, pair: hash_func,
idi, sk, alpha, ss: protocol_id
intruder_knowledge = {a,s,u,ki,inv(ki)}
composition
session(a,s,u,kau,ecdsa,h,add,mul,pair) /\ session(i,s,u,kui,ecdsa,h,add,mul,pair)
/\ session(a,i,u,kui,ecdsa,h,add,mul,pair) /\ session(a,s,i,kui,ecdsa,h,add,mul,pair)
end role
goal
secrecy_of idi, sk, alpha, ss authentication_on ss
end goal
environment()
```

Figure 7. HLPLS specification of environment role.

After defining certain communication sessions in environment role, we execute the tool to check the security correctness. The results of OFMC backend and CL-AtSe backend are shown in Figure 8. We claim that the proposed protocol is provably secure under AVISPA simulation.



```
% OFMC
% Version of 2006/02/13
SUMMARY
SAFE
DETAILS
BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
/home/span/span/testsuite/results/Autonomous_log_management.if
GOAL
as_specified
BACKEND
OFMC
COMMENTS
STATISTICS
parseTime: 0.00s
searchTime: 7.64s
visitedNodes: 2432 nodes
depth: 12 plies
```

```
SUMMARY
SAFE
DETAILS
BOUNDED_NUMBER_OF_SESSIONS
TYPED_MODEL
PROTOCOL
/home/span/span/testsuite/results/Autonomous_log_management.if
GOAL
As Specified
BACKEND
CL-AtSe
STATISTICS
Analysed: 166 states
Reachable: 51 states
Translation: 0.03 s
Computation: 0.01 s
```

Figure 8. Verification results using OFMC and CL-AtSe backends.

4.2. Logical Analysis Using GNY Logic

This sub-section proves security completeness and correctness of our proposed protocol through Gong-Needham-Yahalom (GNY) logic [61]. For our protocol, the analysis consists of two phases in the logic sequence: message freshness verification and message origin verification. Based on GNY logic, the assumptions and logical rules of our protocol are described in Tables 2 and 3 respectively [1,62].

Table 2. The assumptions of the proposed protocol.

(A_1) U_i Y^s: The user U_i possesses secret key Y^s
(A_2) $sensor_i$ s: The $sensor_i$ possesses private keys s
(A_3) U_i g^s: The user U_i know of public key g^s
(A_4) $U_i \mid\equiv \varnothing\ \sigma_{CT}$: The user U_i believes that σ_{CT} is recognizable
(A_5) $U_i \mid\equiv \#\ (m_e)$: The user U_i believes that m_e is fresh
(A_6) $U_i \mid\equiv \#\ (IP_{IoT})$: The user U_i believes that IP_{IoT} is fresh
(A_7) $U_i \mid\equiv \#\ (t_{CT})$: The user U_i believes that timestamp t_{CT} is fresh

Table 3. The logical rules of the proposed protocol.

(F) $\frac{U\mid\equiv\#(M)}{U\mid\equiv\#(M,Y),U\mid\equiv\#(F(M))}$: U believes message M is fresh, which means U can believe that any (M, N) including message M is fresh, then U believes $F(M)$, which is computed from message M, is also fresh
(P) $\frac{U\triangleleft M}{U\ni M}$: U can see the message M, indicating that U really possesses the message M
(R) $\frac{U\mid\equiv\varnothing(M)}{U\mid\equiv\varnothing(M,N),\ U\mid\equiv\varnothing(F(M))}$: U believes message M is recognizable, indicating that U can believe that any (M, N) including message M is recognizable, and U believes that any $F(M)$ computed from message M is also recognizable)
(T_1) $\frac{U\triangleleft *M}{U\triangleleft M}$: when U obtains a non-original value $*M$, it means U may obtain the original M
(T_3) $\frac{U\triangleleft\{M\}_K,\ U\ni Y}{U\triangleleft M}$: U uses secret key Y to encrypt, decrypt to obtain message M

Main communication of our protocol can be presented in logic as follows.

$$sensor_i \rightarrow U_i\colon \left(h^{Y.\,xs}, (M)e(g,g)^{\alpha s}, g^{B\lambda_e}Q_j^{-r_j}, g^s, g^{r_j}, m_e, IP_{IoT}, t_{CT}\right)$$
$$sensor_i \rightarrow \text{gateway}: h^{Y.\,xs}, (M)e(g,g)^{\alpha s}, g^{B\lambda_e}Q_j^{-r_j}, g^s, g^{r_j}, m_e, IP_{IoT}, t_{CT}\big)$$

Specific phases and corresponding goals of the protocol are described in the following.

- Phase 1: Message freshness authentication, proving the authenticity of the message.

Goal 1: Other than the authority, only the user U_i can read the content of the message transmitted by the *sensor*$_i$. Goal 1 (G1) is described as follows.

$$U_i \Big|\equiv\ \varnothing \left(h^{Y.\,xs}, (M)e(g,g)^{\alpha s}, g^{B\lambda_e}Q_j^{-r_j}, g^s, g^{r_j}, m_e, IP_{IoT}, t_{CT}\right)$$

- Phase 2: Message origin authentication, proving that the message is transmitted by the legitimate *sensor*$_i$.

Goal 2: The user U_i can verify that only the *sensor*$_i$ can generate the message received by the U_i. Description of Goal 2 (G2) is as follows.

$$U_i \mid\equiv\ sensor_i \mid\sim \left(h^{Y.\,xs}, (M)e(g,g)^{\alpha s}, g^{B\lambda_e}Q_j^{-r_j}, g^s, g^{r_j}, m_e, IP_{IoT}, t_{CT}\right)$$

Goal 3: The gateway can verify that only the *sensor*$_i$ can generate the message received by the gateway. Goal 3 (G3) is described as follows.

$$\text{Gateway} \mid\equiv\ sensor_i \mid\sim \left(h^{Y.\,xs}, (M)e(g,g)^{\alpha s}, g^{B\lambda_e}Q_j^{-r_j}, g^s, g^{r_j}, m_e, IP_{IoT}, t_{CT}\right)$$

Based on the assumptions and logical rules, we have the protocol achieve above goals as follows. Since U_i knows of the message, we have that.

$$U_i \lhd *\left(*\sigma_{CT}, C, C_j, C', D_j, m_e, IP_{IoT}, t_{CT}\right), \tag{1}$$

According to (T_1), we have that.

$$U_i \lhd \left(\sigma_{CT}, C, C_j, C', D_j, m_e, IP_{IoT}, t_{CT}\right), \tag{2}$$

According to (2), (A_1) and (T_3), the user U_i can compute secret key Y^s and use it to decrypt $C = MY^s$, we have that.

$$U_i \lhd \left(h^{Y.\,xs}, (M)e(g,g)^{\alpha s}, g^{B\lambda_e}Q_j^{-r_j}, g^s, g^{r_j}, m_e, IP_{IoT}, t_{CT}\right), \tag{3}$$

According to (3) and (P), we have that.

$$U_i \ni h^{Y.\,xs},\ (M)e(g,g)^{\alpha s},\ g^{B\lambda_e}Q_j^{-r_j},\ g^s,\ g^{r_j},\ m_e,\ IP_{IoT},\ t_{CT}, \tag{4}$$

Based on (4), (A_2) and (A_3), $h^{Y.\,xs}$ is truly recognizable. Therefore, according to (A_4) and (R), we have that.

$$U_i \Big|\equiv\ \varnothing \left(h^{Y.\,xs},\ (M)e(g,g)^{\alpha s},\ g^{B\lambda_e}Q_j^{-r_j},\ g^s,\ g^{r_j},\ m_e,\ IP_{IoT},\ t_{CT}\right), \tag{5}$$

Based on (5), (A_5), (A_6), (A_7), and (F) we achieve G1. Due to (6), (8), (A_3), (A_4), (A_5), (A_6), (A_7), and (F), G2 is achieved. Using similar arguments of G2, we realize G3. As a result, the proposed protocol realizes all goals G1, G2 and G3.

4.3. Semantic Proof

Our proposed protocol provides secure decryption key, signature verification and data integrity, signature unforgeability, data confidentiality, non-repudiation, tamper proof, and perfect forward secrecy. The specific semantic security proof of the protocol is presented in the following.

- ***Secure decryptionkey***: Based on ECDLP, the adversary cannot retrieve the secret values s from C'. The secret value α is also hidden in K^i, and K^i is even kept secret to the user and the authority only. Therefore, the adversary is not able to obtain g^s and g^α for computing the decryption key $Y^s = e(g,g)^{s\alpha}$. The key is successfully computed only when the legitimate user performs correct pairing operation with appropriate attributes. Thus, we claim the proposed protocol achieves secure decryption key.
- ***Robustverification and data integrity***: In our protocol, the signature of the ciphertext is verified to assure the authenticity of the log. The verification correctness of single signature σ_{CT_1} is proved as follows.

$$\begin{aligned} & e(h, Y_{.x}C') \\ = \ & e(h, Y_{.x}g^s) \\ = \ & e\left(h,\ g^{Y_{.x}s}\right) \\ = \ & e\left(h^{Y_{.x}s},\ g\right) \\ = \ & e\left(\sigma_{CT_1}, g\right), \end{aligned} \tag{6}$$

Suppose there have two signatures for the batch signing, the following equation proves the correctness of batch signature verification (including σ_{CT_1} and σ_{CT_2}).

$$\begin{aligned} & e\left(h_1 h_2, Y_{.x} Y_{.x} C_1' C_2'\right) \\ = \ & e(h_1 h_2, Y_{.x} g^s Y_{.x} g^s) \\ = \ & e\left(h_1 h_2,\ g^{Y_{.x}s} g^{Y_{.x}s}\right) \\ = \ & e\left({h_1}^{Y_{.x}s} {h_2}^{Y_{.x}s},\ gg\right) \\ = \ & e\left(\sigma_{CT_1} \sigma_{CT_2}, gg\ \right) \\ = \ & e(BSig, gg), \end{aligned} \tag{7}$$

Therefore, the signature is verifiably correct. After verifying that the log and its signature is originally sent and signed by the sensor, the integrity is achieved. Thus, the conclusion is established.

- ***Signatureunforgeability***: If the adversary wants to forge the signature $\sigma = h^{Y_{.x}s}$, he must obtain the correct s. However, as stated, the value s is protected by ECDLP. The adversary therefore is not able to compute s for forging signature σ. So, our work achieves signature unforgeability.
- ***Dataconfidentiality***: If the adversary wants to restore the log M from the ciphertext, he/she must obtain SK_{ID_i} =(K^i, L^i, K^i_j) and compute Y^s to unsigncrypt C. However, only the user who has registered with the authority possesses correct attributes and the key SK_{ID_i}. As stated, the security of the decryption key Y^s is also guaranteed. Moreover, the adversary does not know of $MSK = (\alpha, \beta, u_s)$ to compromise the system. Thus, the confidentiality of the logs is fully achieved.
- ***Non-repudiation and tamper resistance***: During the communication in our protocol, private key s is only known to the sensor. Therefore, the sensor cannot repudiate the signature signed by itself. The signature is furthermore uploaded to public blockchain. Once recorded, it is not possible for block data to be altered retroactively. In this way, the signature cannot be tampered with. Therefore, we claim non-repudiation and data tampering resistance in the proposed protocol.
- ***Perfect forward secrecy:*** In log signcryption phase, the sensor chooses the key s to compute the ciphertext and generate the decryption key Y^s. Since s is a randomly selected, the key Y^s is

computed as a nonce. Therefore, even though the adversary has obtained the decryption key of the current session, he/she cannot recover keys of the past communication sessions. Thus, perfect forward secrecy is achieved in our protocol.

4.4. Comparison with Related Works

We furthermore indicate contributions of this paper by a comparative study of our work and recently published works discussed in Section 1.2. The comparison is described in Table 4. Symbol √ denotes the protocol achieves the corresponding property, and symbol × denotes the property is not provided by the protocol. Besides, symbol – denotes the property is not available in the protocol. The results show that the proposed protocol satisfies all essential requirements of security and functionality. Especially, only our protocol provides signature chain with evidence legality, which is useful for digital forensic investigations. Public-private blockchain and signcryption method are also not available in all others works except ours. In addition, autonomous model is only introduced in our work, and Hang and Kim [31]'s work.

Table 4. Comparison on security and functionality of our work and related works.

	[13]	[29]	[30]	[31]	[32]	[33]	[34]	[35]	Ours
Autonomous model	×	×	×	√	×	×	×	×	√
Signcryption method	×	×	×	×	×	×	×	×	√
Fine-grained access control with ABE	×	×	×	×	√	√	√	√	√
Blockchain mechanism	√	√	√	√	×	√	×	×	√
Integration of private and public blockchain	×	×	×	×	–	×	–	–	√
Signature chain	×	×	×	×	×	×	×	×	√
Evidence legality	×	–	–	×	–	–	–	–	√
Signature unforgeability	√	–	–	–	–	–	–	–	√
Data non-repudiation	√	–	–	√	–	–	–	–	√
Data integrity	√	–	√	√	–	–	–	–	√
Data tampering resistance	√	√	√	×	×	√	×	×	√
Perfect forward secrecy	–	–	–	–	√	√	√	√	√
Protocol simulation using AVISPA/ProVerif	×	×	×	×	×	×	×	×	√
Protocol implementation	×	×	×	√	×	√	×	×	√

5. Performance Analysis

In this section, we analyze performance of the proposed protocol based on computation cost. Computation times of major cryptographic functions and operations used in the proposed protocol are defined as follows.

- T_E: Time of performing an exponentiation operation in G_1.
- T_{BP}: Time of performing a bilinear paring operation.
- T_H: Time of performing a hash function.
- T_{ECDSA_Gen}: Time of performing an ECDSA generation algorithm.
- T_{ECDSA_Veri}: Time of performing an ECDSA verification algorithm.

Due to SSO solution, computation cost of our protocol is independent of the number of servers. Since generating and verifying the ECDSA are performed using private key and public key, we assume their computation costs are similar to asymmetric encryption and decryption algorithms respectively. Suppose finding the *nonce* value in private blockchain (computing the hash) is straightforward. As shown in Table 5, total cost of the proposed protocol is $(5mT_E + 2m(n+1)T_H + (2mni + 5mn + 2m)T_{BP} + 3mT_{ECDSA\_Gen} + (3mn + 2m)T_{ECDSA\_Veri})$. Especially, the sensors consume only $(5T_E + T_H + T_{ECDSA\_Gen})$ for signcrypting a single log.

Table 5. Execution time complexities of the proposed protocol.

	Sensor	Gateway	User	Server
Log signcryption phase	$(5T_E + T_H + T_{ECDSA\_Gen})m$	–	–	–
Log verification phase	–	$(2T_{BP} + T_{ECDSA\_Gen} + T_{ECDSA\_Veri})m$	–	–
Log unsigncryption phase	–	–	$(2iT_{BP} + 3T_{BP} + T_H)mn$	–
Private block calculation phase	–	–	–	$(T_{ECDSA\_Gen} + T_{ECDSA\_Veri} + T_H)m$
Private block verification phase	–	–	$(T_H + 2T_{BP} + 3T_{ECDSA\_Veri})mn$	–
Total time complexities	$5mT_E + 2m(n+1)T_H + (2mni + 5mn + 2m)T_{BP} + 3mT_{ECDSA\_Gen} + (3mn + 2m)T_{ECDSA\_Veri}$			
Total time estimation (ms)	$0.5244mni + 3.47346mn + 7.72938m$			

m: no. of logs that can be accessed by a single user; n: no. of users; i: no. of attributes; –: not available. Based on [1,63]: $T_E \approx 0.72036ms$, $T_{BP} \approx 0.2622ms$, $T_H \approx 0.00069ms$, $T_{ECDSA\_Gen} \approx 0.72036ms$, and $T_{ECDSA\_Veri} \approx 0.72036ms$.

Based on the data of Table 5, we further conduct experiments of protocol performance with two scenarios: (a) a single user with i attributes signcrypts a single log, and (b) a single user with certain number of attributes (suppose $i = 3$) signcrypts m logs. In the former scenario (depicted in Figure 9a), the cost is slightly increased when i increases. In the latter scenario (depicted in Figure 9b), when m increases, the cost is significantly increased.

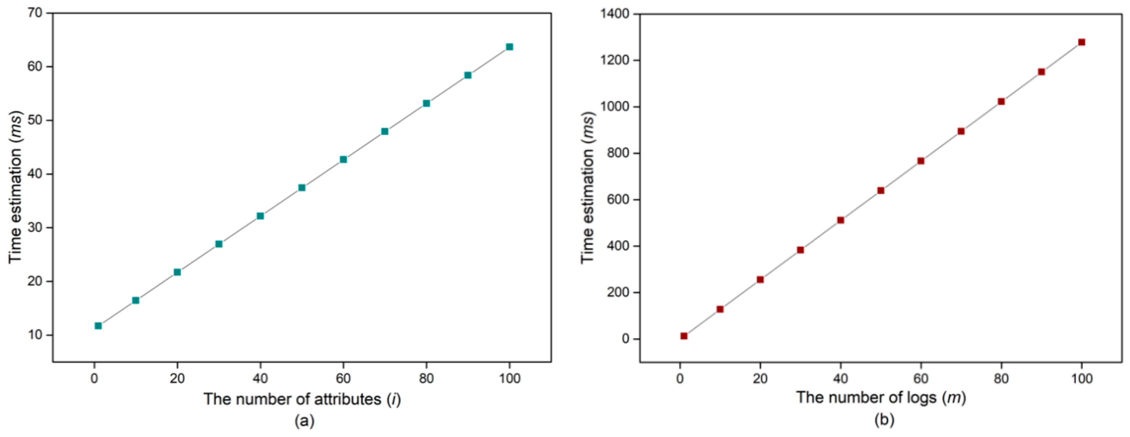


Figure 9. Total computation cost of the proposed protocol: (**a**) $n = 1$ and $m = 1$; (**b**) $n = 1$ and $i = 3$.

It is observed that our protocol bears a reasonable cost with various components and a lot of functionalities. Moreover, it is important to note that the proposed protocol is designed with ECC small key size, rapid BLS signature, signcryption method, and SSO solution. Our work therefore bears low computation and storage cost, and is well suited for the IoT.

6. Implementation

Our implementation simulates log protection of air conditioner $Sensor_1$ in the context of a laboratory in Chang Gung University (Taiwan). $Sensor_1$ is allowed to signcrypt its log based on an access policy. Attributes generated by the attribute authority including *Chang Gung University*, *Department of Information Management*, *Professor* and *Student* are set to att_1, att_2, att_3 and att_4 respectively. User U_1 attempts to access the log produced by $Sensor_1$. The user is able to view the log only if he/she possesses appropriate attributes. We include practical implementation and system construction in the following sub-sections.

6.1. Practical Procedure of the Proposed Protocol

Access policy of the sensors are determined with Boolean formula in the beginning. The access policy of $Sensor_1$ is illuminated in Figure 10.

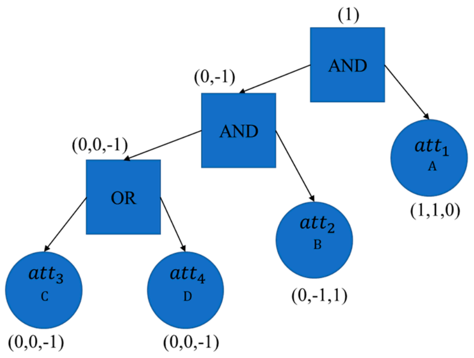


Figure 10. Access policy BF_1 of $Sensor_1$.

At first, the authority initializes the corresponding public parameter $PP = g, B, Y, H, Q_1, Q_2, Q_3, Q_4, e, G_1, G_T$, and secret parameter $MSK = \alpha, \beta, u_s$. $Sensor_1$ uses its identity DID_1 to register with the attribute authority and obtains PP. The user U_1 registers, logs in, and obtains SSO password from the SSO server for accessing specific blockchain server. In this simulation, the user U_1 possesses three attributes: att_1, att_2 and att_3. In the user registration phase, the attribute authority performs the Algorithm 6 to compute the secret key $SK_{ID_1} = (K^1, L^1, K_1^1, K_2^1, K_3^1)$, and send it to the user U_1. Based on attribute-based access policy $BF =$ "$((att_3\ OR\ att_4)\ AND\ att_2)\ AND\ att_1$", $Sensor_1$ uses Algorithm 7 to perform signcryption procedure. It then generates $CT_1{=}(\sigma_{CT_1}, C', C_1, C_2, C_3, D_1, D_2, D_3, C_{CT_1}, IP_{IoT}, t_{CT})\ with\ (m_1, \rho(1))(m_2, \rho(2))(m_3, \rho(3))$ and $\delta^1_{IoT} = \left(\sigma^1_{IoT}, IP_{IoT}, t^1_{IoT}\right)$. The gateway verifies validity of the ciphertext CT_1, sends it to the blockchain server, and stores the signature $\delta^1_{GW} = \left(\sigma^1_{GW}, IP_{GW}, t^1_{GW}\right)$ after performing Algorithm 8. the user U_1 performs Algorithm 9 to compute the correct key Y^s based on the key SK_{ID_1} with appropriate attributes. Thus, the user U_1 is able to unsigncrypt the ciphertext CT_1 and view the log M_1.

In private block calculation phase, based on the ciphertext CT_1 and the previous hash retrieved from the private blockchain, the blockchain server generates block $PriB_1 = \left(Nonce, \delta^1_{CT}, \delta^1_{Srv}, \delta^1_{GW}, \delta^1_{IoT}, PreviousHash, Difficulty, OptionalFields, OtherFields\right)$, and upload it to the chain. This procedure is conducted by Algorithm 10. Thereafter, the server obtains the block number $PriBlockNo_1$ generated by the private blockchain. For verifying the validity of the block data $PriB_1$, the user U_1 performs Algorithm 11. Suppose we have private block $PriB_2$ generated by the similar procedures. In public block calculation phase, the server then calculates batch signature $BSig_1 = \sigma_{CT_1}\sigma_{CT_2}$. Block data $PubB_1$ is then calculated by $PubB_1 = (BSig_1, PriBlockNo_1, PriBlockNo_2)$, and is uploaded to the public blockchain. Thereafter, the sever can obtain the block number $PubBlockNo_1$ generated by the pubic blockchain. The procedure is performed by Algorithm 12. In public block verification phase, the public block data $PubB_1$, the ciphertexts CT_1, CT_2, and public parameter PP are retrieved. Based on these parameters, the user U_1 performs Algorithm 13 to verify validity of the uploaded logs.

6.2. System Construction

In this sub-section, we construct a system deployment for the proposed protocol. The development environment and system interface of our construction are specifically provided in the following.

6.2.1. Development Environment

- Host: We use Ubuntu Server 18.04.02 LTS as the host operating system (OS). The specification includes CPU I7-6820 2.7 GHz, 16 GB RAM memory, and 500 GB hard disk.
- Sensor Configuration: Raspberry Pi is the mainboard used in the system architecture. Raspbian OS is installed with the hardware including Raspberry Pi 3 Model 8 V1.2, CPU ARM Cortex-A53 1.2 GHz Quad Core, 1 GB RAM, and 16 G MicroSD card. Air conditioner sensor YW-51GJ is used as the device of our simulation. The mainboard and the sensor are integrated and assembled in a box so that sensor can contact the ambient air. Overview of our setting is shown in Figure 11.

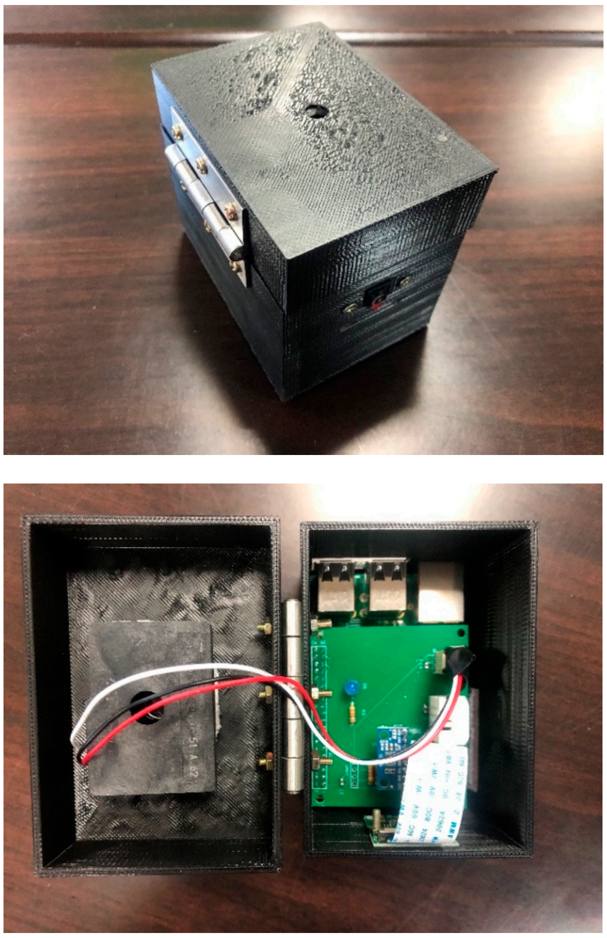


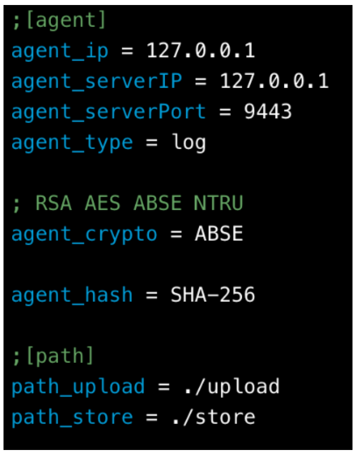
Figure 11. Setting of our implementation.

- Blockchain platform: We employ Hyperledger Fabric v1.0 for private blockchain. This framework provides development foundation, modularity, scalability, and security for the simulated system. We use open source platform of the Ethereum for public blockchain. Smart contract can be fully written and published on Ethereum to develop diversified applications. In this regard, an account with virtual currency is created with Metamask wallet for data transaction. In addition, the contract is written into blockchain through Infura, an Ethereum application programming interface (API).
- Kubernetes: We use Kubernetes as the platform, providing microservices for data management, deployment, and expansion. Kubernetes is compatible to all OS platforms, which provides the proper use of system performance. In our simulation, private blockchain is installed with Kubernetes version 1.13.4.
- Programming language: Languages used in our system includes Golang, HTML and Javascript. Golang library package was developed by Ben Lynn, who developed the library from the original

Pairing-Based Cryptography (PBC) written in C language. According to the recommendation of NIST, the selected key length of ECC for security strength is 224-bit type.

6.2.2. System Interface

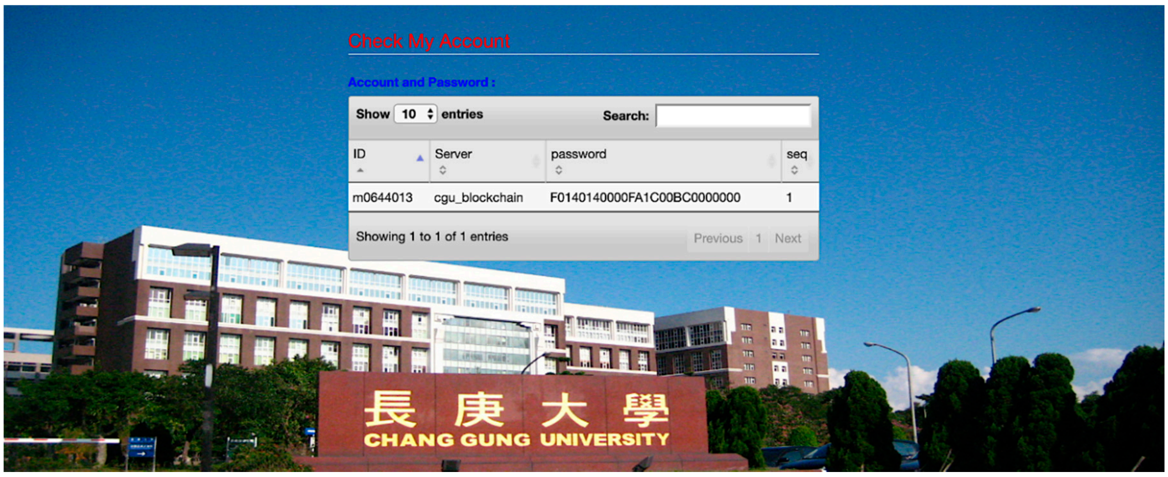
The device can be configured as an agent with related settings. The configuration file includes agent IP address, server IP address, server port, and agent type. In addition, the cryptographic module supports AES, NTRU, RSA and ABSE, based on the choice of specific cryptosystem. Brief description of device configuration is provided in Figure 12.



```
;[agent]
agent_ip = 127.0.0.1
agent_serverIP = 127.0.0.1
agent_serverPort = 9443
agent_type = log

; RSA AES ABSE NTRU
agent_crypto = ABSE

agent_hash = SHA-256

;[path]
path_upload = ./upload
path_store = ./store
```

Figure 12. Device configuration.

The user logs in to the SSO sever using his/her password and smart card. After SSO login, the user has to make another registration with the blockchain sever. To this end, the user enters main identity, password and an additional identity "m0644013" to register with the server of Chang Gung university, named "cgu_blockchain". SSO password is generated in this registration. Figure 13 shows a registered account with its credential generated by the SSO server for targeted service. Using the account generated by the SSO server, the user can log in to the server "cgu_blockchain".



Check My Account

Account and Password :

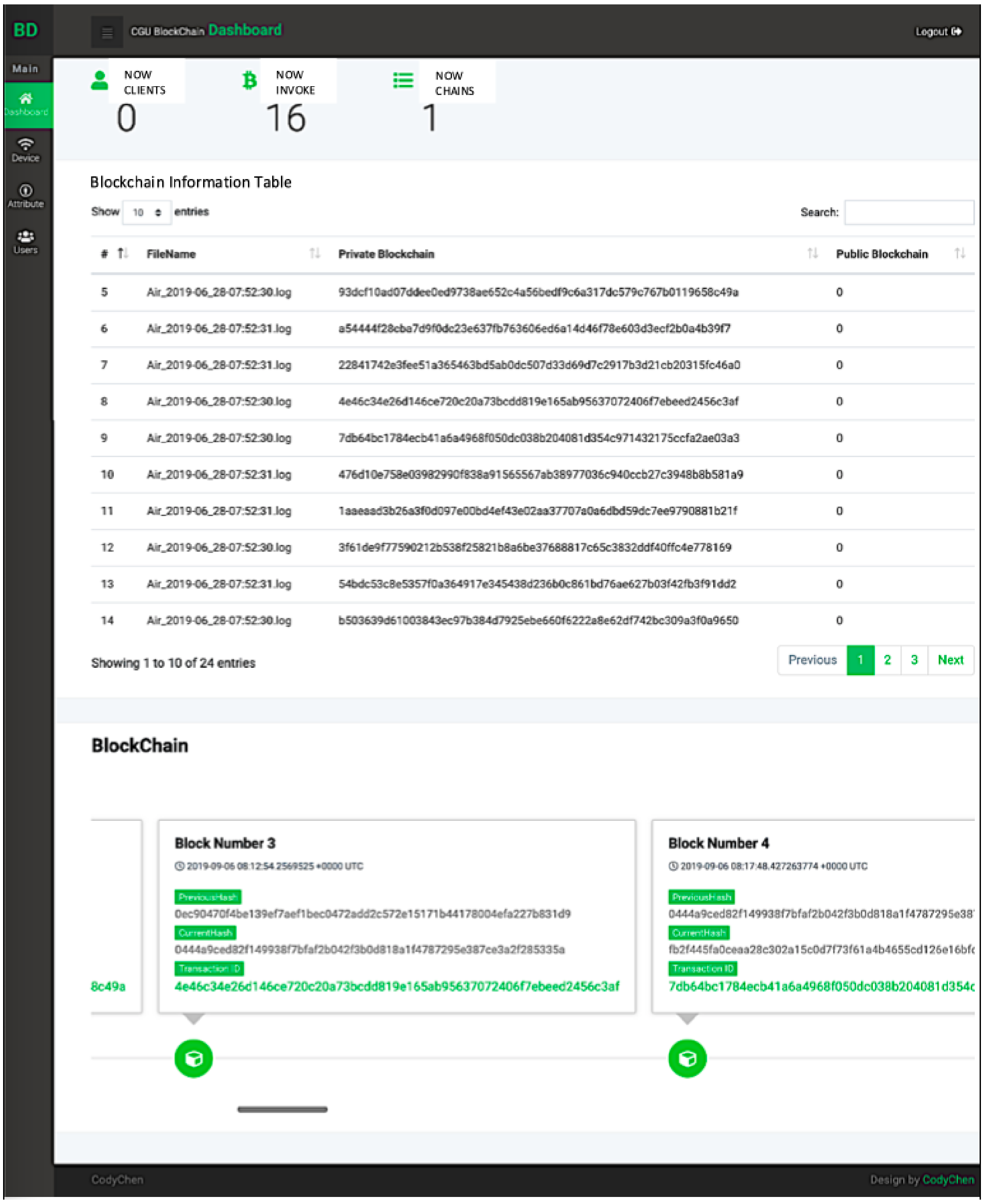
Show 10 entries Search:

ID	Server	password	seq
m0644013	cgu_blockchain	F0140140000FA1C00BC0000000	1

Showing 1 to 1 of 1 entries

Figure 13. Account generated by SSO server.

If an administrator enters the system, he/she can see the blockchain status and overall service through a monitoring interface. The interface is described in Figure 14. Table providing information of registered devices is also shown in Figure 15. The devices signcrypt the log and uploads it to blockchain server.



Blockchain Information Table

Show 10 entries Search:

#	FileName	Private Blockchain	Public Blockchain
5	Air_2019-06_28-07:52:30.log	93dcf10ad07ddee0ed9738ae652c4a56bedf9c6a317dc579c767b0119658c49a	0
6	Air_2019-06_28-07:52:31.log	a54444f28cba7d9f0dc23e637fb763606ed6a14d46f78e603d3ecf2b0a4b39f7	0
7	Air_2019-06_28-07:52:31.log	22841742e3fee51a365463bd5ab0dc507d33d69d7c2917b3d21cb20315fc46a0	0
8	Air_2019-06_28-07:52:30.log	4e46c34e26d146ce720c20a73bcdd819e165ab95637072406f7ebeed2456c3af	0
9	Air_2019-06_28-07:52:30.log	7db64bc1784ecb41a6a4968f050dc038b204081d354c971432175ccfa2ae03a3	0
10	Air_2019-06_28-07:52:31.log	476d10e758e03982990f838a91565567ab38977036c940ccb27c3948b8b581a9	0
11	Air_2019-06_28-07:52:31.log	1aaeaad3b26a3f0d097e00bd4ef43e02aa37707a0a6dbd59dc7ee9790881b21f	0
12	Air_2019-06_28-07:52:30.log	3f61de9f77590212b538f25821b8a6be37688817c65c3832ddf40ffc4e778169	0
13	Air_2019-06_28-07:52:31.log	54bdc53c8e5357f0a364917e345438d236b0c861bd76ae627b03f42fb3f91dd2	0
14	Air_2019-06_28-07:52:30.log	b503639d61003843ec97b384d7925ebe660f6222a8e62df742bc309a3f0a9650	0

Showing 1 to 10 of 24 entries

Previous 1 2 3 Next

BlockChain

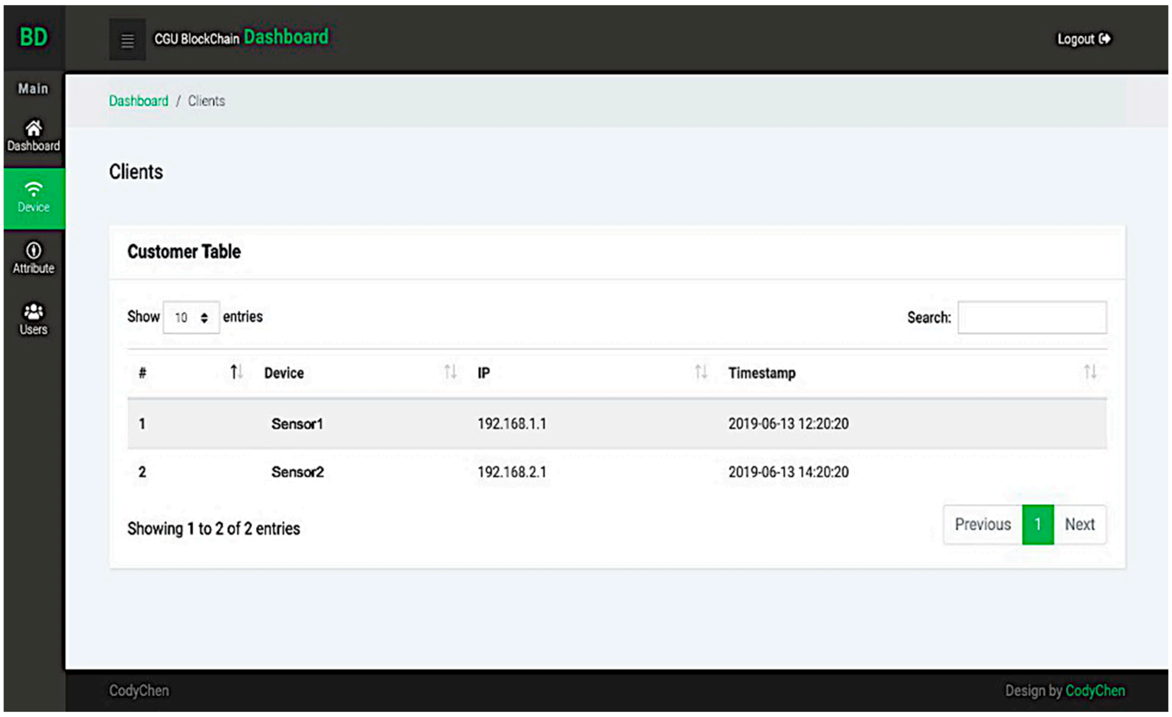
8c49a

Block Number 3
2019-09-06 08:12:54.2569525 +0000 UTC
PreviousHash
0ec90470f4be139ef7aef1bec0472add2c572e15171b44178004efa227b831d9
CurrentHash
0444a9ced82f149938f7bfaf2b042f3b0d818a1f4787295e387ce3a2f285335a
Transaction ID
4e46c34e26d146ce720c20a73bcdd819e165ab95637072406f7ebeed2456c3af

Block Number 4
2019-09-06 08:17:48.427263774 +0000 UTC
PreviousHash
0444a9ced82f149938f7bfaf2b042f3b0d818a1f4787295e38
CurrentHash
fb2f445fa0ceaa28c302a15c0d7f73f61a4b4655cd126e16bf
Transaction ID
7db64bc1784ecb41a6a4968f050dc038b204081d354

CodyChen Design by CodyChen

Figure 14. Blockchain server management interface.



Clients

Customer Table

Show 10 entries Search:

#	Device	IP	Timestamp
1	Sensor1	192.168.1.1	2019-06-13 12:20:20
2	Sensor2	192.168.2.1	2019-06-13 14:20:20

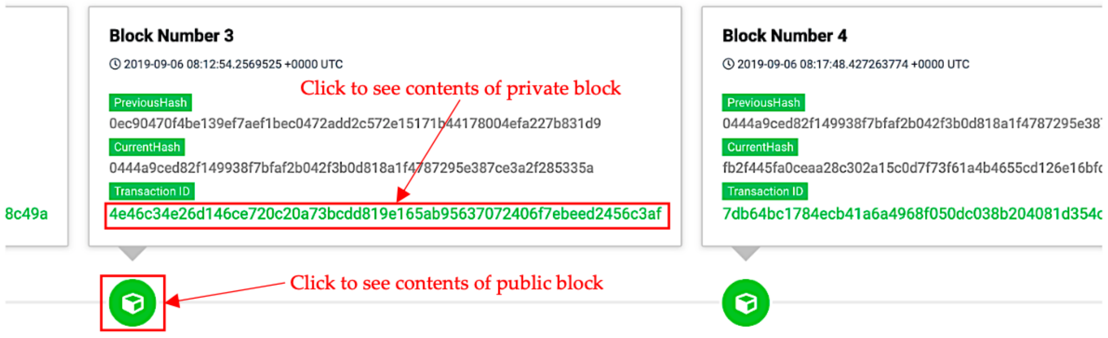
Showing 1 to 2 of 2 entries

Previous 1 Next

Figure 15. Table of registered devices.

In terms of attribute setting, the attribute authority is allowed to add or delete the attributes, in order to initialize the system. In this way, the authority can determine attributes for specific users, and allows them to obtain their attributes during user registration phase.

When the users log in to the system, they can click on the link within private blockchain allocated at the bottom of the monitoring interface (shown in Figure 16) to see block data. The detailed information of private blockchain is provided in Figure 17. In addition, the users (including the administrator and the users who possesses appropriate attributes) can also view the log data at the bottom of this interface. As shown in Figure 17, the log data produced by the sensor includes dust, temperature, humidity, and atmospheric carbon dioxide.

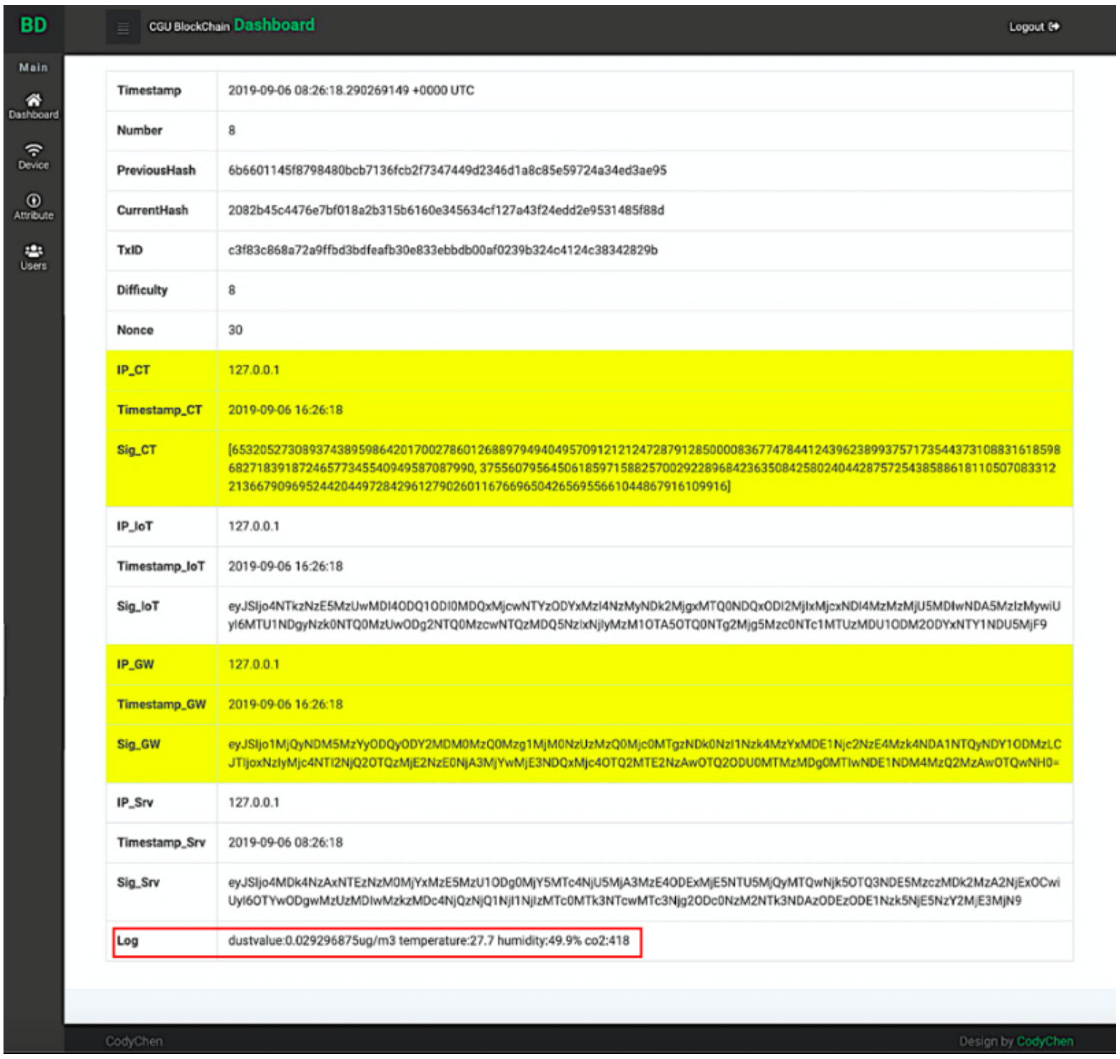

Click to see contents of private block
Click to see contents of public block

8c49a

Block Number 3
2019-09-06 08:12:54.2569525 +0000 UTC
PreviousHash
0ec90470f4be139ef7aef1bec0472add2c572e15171b44178004efa227b831d9
CurrentHash
0444a9ced82f149938f7bfaf2b042f3b0d818a1f4787295e387ce3a2f285335a
Transaction ID
4e46c34e26d146ce720c20a73bcdd819e165ab95637072406f7ebeed2456c3af

Block Number 4
2019-09-06 08:17:48.427263774 +0000 UTC
PreviousHash
0444a9ced82f149938f7bfaf2b042f3b0d818a1f4787295e38
CurrentHash
fb2f445fa0ceaa28c302a15c0d7f73f61a4b4655cd126e16bf
Transaction ID
7db64bc1784ecb41a6a4968f050dc038b204081d354

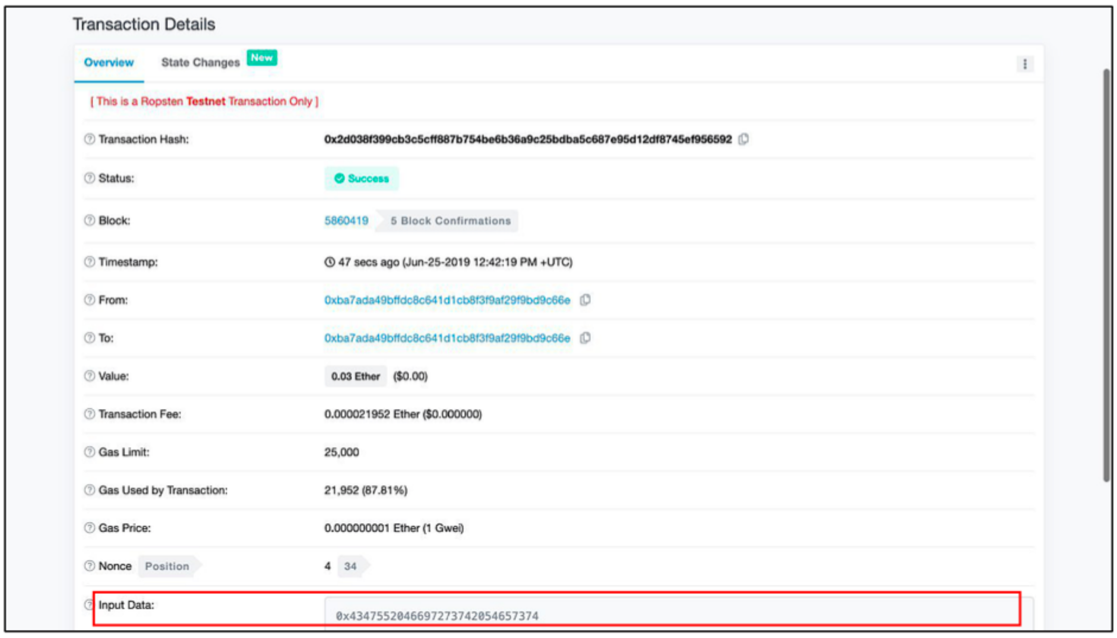
Figure 16. Private blocks within the chain.



Timestamp	2019-09-06 08:26:18.290269149 +0000 UTC
Number	8
PreviousHash	6b6601145f8798480bcb7136fcb2f7347449d2346d1a8c85e59724a34ed3ae95
CurrentHash	2082b45c4476e7bf018a2b315b6160e345634cf127a43f24edd2e9531485f88d
TxID	c3f83c868a72a9ffbd3bdfeafb30e833ebbdb00af0239b324c4124c38342829b
Difficulty	8
Nonce	30
IP_CT	127.0.0.1
Timestamp_CT	2019-09-06 16:26:18
Sig_CT	[653205273089374389598642017002786012688979494049570912121247287912850000836774784412439623899375717354437310883161859868271839187246577345540949587087990, 3755607956450618597158825700292228968423635084258024044287572543858861811050708331221366790969524420449728429612790260116766965042656955661044867916109916]
IP_IoT	127.0.0.1
Timestamp_IoT	2019-09-06 16:26:18
Sig_IoT	eyJSIjo4NTkzNzE5MzUwMDI4ODQ1ODI0MDQxMjcwNTYzODYxMzI4NzMyNDk2MjgxMTQ0NDQxODI2MjIxMjcxNDI4MzMzMjU5MDIwNDA5MzIzMywiUyI6MTU1NDgyNzk0NTQ0MzUwODg2NTQ0MzcwNTQzMDQ5NzIxNjIyMzM1OTA5OTQ0NTg2Mjg5Mzc0NTc1MTUzMDU1ODM2ODYxNTY1NDU5MjF9
IP_GW	127.0.0.1
Timestamp_GW	2019-09-06 16:26:18
Sig_GW	eyJSIjo1MjQyNDM5MzYyODQyODY2MDM0MzQ0Mzg1MjM0NzUzMzQ0Mjc0MTgzNDk0NzI1Nzk4MzYxMDE1Njc2NzE4Mzk4NDA1NTQyNDY1ODMzLCJTIjoxNzIyMjc4NTI2NjQ2OTQzMjE2NzE0NjA3MjYwMjE3NDQxMjc4OTQ2MTE2NzAwOTQ2ODU0MTMzMDg0MTIwNDE1NDM4MzQ2MzAwOTQwNH0=
IP_Srv	127.0.0.1
Timestamp_Srv	2019-09-06 08:26:18
Sig_Srv	eyJSIjo4MDk4NzAxNTEzNzM0MjYxMzE5MzU1ODg0MjY5MTc4NjU5MjA3MzE4ODExMjE5NTU5MjQyMTQwNjk5OTQ3NDE5MzczMDk2MzA2NjExOCwiUyI6OTYwODgwMzUzMDIwMzkzMDc4NjQzNjQ1NjI1NjIzMTc0MTk3NTcwMTc3Njg2ODc0NzM2NTk3NDAzODEzODE1Nzk5NjE5NzY2MjE3MjN9
Log	dustvalue:0.029296875ug/m3 temperature:27.7 humidity:49.9% co2:418

Figure 17. Data stored in a private block.

Furthermore, the user can see detailed information stored in public block by clicking on the icon highlighted in Figure 16. As shown in Figure 18, along with all related information, the data input from private blockchain is available at the bottom of the content table of the public blockchain.



Transaction Details

Overview State Changes New

[This is a Ropsten **Testnet** Transaction Only]

Transaction Hash:	0x2d038f399cb3c5cff887b754be6b36a9c25bdba5c687e95d12df8745ef956592
Status:	Success
Block:	5860419 5 Block Confirmations
Timestamp:	47 secs ago (Jun-25-2019 12:42:19 PM +UTC)
From:	0xba7ada49bffdc8c641d1cb8f3f9af29f9bd9c66e
To:	0xba7ada49bffdc8c641d1cb8f3f9af29f9bd9c66e
Value:	0.03 Ether ($0.00)
Transaction Fee:	0.000021952 Ether ($0.000000)
Gas Limit:	25,000
Gas Used by Transaction:	21,952 (87.81%)
Gas Price:	0.000000001 Ether (1 Gwei)
Nonce Position	4 34
Input Data:	0x43475520466972737420546573 74

Figure 18. Data stored in a public block.

7. Conclusions

The logs produced by IoT devices contain important contents and private information, and can be used as evidences for digital forensic investigations. Our work has introduced an autonomous log storage management protocol with blockchain mechanism and access control for the IoT. Various security properties for the logs including robust identity verification, data integrity, non-repudiation, insider attack resistance, and the legality are achieved by the integration of blockchain and signature chain. Moreover, internal security issue has been addressed by fine-grained attribute-based access control. Security analysis demonstrates that our protocol satisfies various security requirements. Our work is well suited for the IoT with a good performance due to elliptic curve short key length, short BLS signature, efficient signcryption method, and multi-server architecture.

In this paper, we propose a protocol with general IoT applications. Domain applications (for instance, WBANs or smart electricity grids) with specific architectures will be considered for our future works with similar mechanisms. Autonomous model would be modified to other ones in which gateways or servers will perform the signscryption. Additionally, protocol performance should also be taken into account, which further improves communication efficiency of low-power devices in the IoT.

Author Contributions: Conceptualization, W.-X.C., and C.-L.H.; protocol, W.-X.C., and C.-L.H.; GNY logic, T.-V.L.; AVISPA tool, T.-V.L.; semantic security analysis, T.-V.L.; performance analysis, T.-V.L.; implementation, W.-X.C.; supervision, C.-L.H.; and funding acquisition, C.-L.H. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by Chang Gung Memorial Hospital under Grant No. CMRPG5D0183 and CMRPD3D0063, and Ministry of Science and Technology in Taiwan under Grant No. MOST-105-2923-E-182-001-MY3, No. MOST-107-2221-E-182-006, and No. MOST-108-2221-E-182-011. This work was also supported in part by Healthy Aging Research Center, Chang Gung University from the Featured Areas Research Center Program within the Framework of the Higher Education Sprout Project by the Ministry of Education (MOE) in Taiwan under Grant No. EMRPD1I0481, No. EMRPD1H0421, and No. EMRPD1H0551.

Acknowledgments: We would like to thank the support of Chang Gung University, Chang Gung Memorial Hospital, the Ministry of Science and Technology (MOST), and the Ministry of Education (MOE) in Taiwan.

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

References

1. Wong, A.K.; Hsu, C.L.; Le, T.V.; Hsieh, M.C.; Lin, T.W. Three-Factor Fast Authentication Scheme with Time Bound and User Anonymity for Multi-Server E-Health Systems in 5G-Based Wireless Sensor Networks. *Sensors* **2020**, *20*, 2511. [CrossRef] [PubMed]
2. Homaei, M.H.; Salwana, E.; Shamshirband, S. An Enhanced Distributed Data Aggregation Method in the Internet of Things. *Sensors* **2019**, *19*, 3173. [CrossRef] [PubMed]
3. Movassaghi, S.; Abolhasan, M.; Lipman, J.; Smith, D.; Jamalipour, A. Wireless Body Area Networks: A Survey. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 1658–1686. [CrossRef]
4. Guo, X.; Lin, H.; Wu, Y.; Peng, M. A new data clustering strategy for enhancing mutual privacy in healthcare IoT systems. *Future Gener. Comput. Syst.* **2020**, *113*, 407–417. [CrossRef]
5. Abdelmoneem, R.M.; Benslimane, A.; Shaaban, E. Mobility-aware task scheduling in cloud-Fog IoT-based healthcare architectures. *Comput. Netw.* **2020**, *179*, 107348. [CrossRef]
6. Babar, M.; Tariq, M.U.; Jan, M.A. Secure and resilient demand side management engine using machine learning for IoT-enabled smart grid. *Sustain. Cities Soc.* **2020**, *62*, 102370. [CrossRef]
7. Kang, L.; Chen, W.; Zheng, Z.; Li, Z.; Liang, W. A Novel Debt-Credit Mechanism for Blockchain-Based Data-Trading in Internet of Vehicles. *IEEE Internet Things J.* **2019**, *6*, 9098–9111.
8. Praveen, M.; Harini, V. NB-IOT based smart car parking system. In Proceedings of the 2019 International Conference on Smart Structures and Systems (ICSSS), Chennai, India, 14–15 March 2019.
9. Zhang, R.; Cui, S.; Zhao, C. *Design of a Data Acquisition and Transmission System for Smart Factory Based on NB-IoT.*; Springer: Singapore, 2018; pp. 875–880.
10. Yang, C.T.; Kristiani, E.; Wang, Y.T.; Min, G.; Lai, C.H.; Jiang, W.J. On construction of a network log management system using ELK Stack with Ceph. *J. Supercomput.* **2020**, *76*, 6344–6360. [CrossRef]
11. Rochim, A.F.; Aziz, M.A.; Fauzi, A. Design Log Management System of Computer Network Devices Infrastructures Based on ELK Stack. In Proceedings of the ICECOS 2019—3rd International Conference on Electrical Engineering and Computer Science, Batam Island, Indonesia, 2–3 October 2019.
12. Di Tosto, G.; McAlearney, A.S.; Fareed, N.; Huerta, T.R. Metrics for Outpatient Portal Use Based on Log. File Analysis: Algorithm Development. *J. Med. Internet Res.* **2020**, *22*, e16849. [CrossRef]
13. Ryu, J.H.; Sharma, P.K.; Jo, J.H.; Park, J.H. A blockchain-based decentralized efficient investigation framework for IoT digital forensics. *J. Supercomput.* **2019**, *75*, 4372–4387. [CrossRef]
14. Harbawi, M.; Varol, A. An improved digital evidence acquisition model for the Internet of Things forensic I: A theoretical framework. In Proceedings of the 2017 5th International Symposium on Digital Forensic and Security (ISDFS), Tirgu Mures, Romania, 26–28 April 2017.
15. Janjua, K.; Shah, M.A.; Almogren, A.; Khattak, H.A.; Maple, C.; Din, I.U. Proactive forensics in IoT: Privacy-aware log-preservation architecture in fog-enabled-cloud using holochain and containerization technologies. *Electronics* **2020**, *9*, 1172. [CrossRef]
16. Fernández-Caramés, T.M.; Fraga-Lamas, P. A Review on the Use of Blockchain for the Internet of Things. *IEEE Access* **2018**, *6*, 32979–33001. [CrossRef]
17. Yuan, Y.; Wang, F. Towards blockchain-based intelligent transportation systems. In Proceedings of the 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), Rio de Jeneiro, Brazil, 1–4 November 2016.
18. Gordon, W.J.; Catalini, C. Blockchain Technology for Healthcare: Facilitating the Transition to Patient-Driven Interoperability. *Comput. Struct. Biotechnol. J.* **2018**, *16*, 224–230. [CrossRef] [PubMed]
19. Samaniego, M.; Jamsrandorj, U.; Deters, R. Blockchain as a Service for IoT. In Proceedings of the 2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Chengdu, China, 15–18 December 2016.
20. Panarello, A.; Tapas, N.; Merlino, G.; Longo, F.; Puliafito, A. Blockchain and IoT Integration: A Systematic Survey. *Sensors* **2018**, *18*, 2575. [CrossRef]
21. Queiroz, M.M.; Wamba, S.F. Blockchain adoption challenges in supply chain: An empirical investigation of the main drivers in India and the USA. *Int. J. Inf. Manag.* **2019**, *46*, 70–82. [CrossRef]
22. Wang, Y.; Singgih, M.; Wang, J.; Rit, M. Making sense of blockchain technology: How will it transform supply chains? *Int. J. Product. Econom.* **2019**, *211*, 221–236. [CrossRef]

23. Zyskind, G.; Nathan, O.; Pentland, A. Enigma: Decentralized Computation Platform with Guaranteed Privacy. *arXiv* **2015**, arXiv:1506.03471. Available online: https://arxiv.org/abs/1506.03471 (accessed on 10 June 2015).
24. Huang, Z.; Su, X.; Zhang, Y.; Shi, C.; Zhang, H.; Xie, L. A decentralized solution for IoT data trusted exchange based-on blockchain. In Proceedings of the 2017 3rd IEEE International Conference on Computer and Communications (ICCC), Chengdu, China, 13–16 December 2017.
25. Axon, L.; Goldsmith, M. *PB-PKI: A Privacy-Aware Blockchain-Based PKI.*; Oxford University Press: New York, NY, USA, 2017; pp. 311–318.
26. Kebande, V.R.; Ray, I. A Generic Digital Forensic Investigation Framework for Internet of Things (IoT). In Proceedings of the 2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud), Vienna, Austria, 22–24 August 2016.
27. Perumal, S.; Norwawi, N.M.; Raman, V. Internet of Things(IoT) digital forensic investigation model: Top.-down forensic approach methodology. In Proceedings of the 2015 Fifth International Conference on Digital Information Processing and Communications (ICDIPC), Sierre, Switzerland, 7–9 October 2015.
28. MacDermott, A.; Baker, T.; Shi, Q. Iot Forensics: Challenges for the Ioa Era. In Proceedings of the 2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS), Paris, France, 26–28 February 2018.
29. Taguchi, Y.; Kanai, A.; Tanimo, S. A Distributed Log. Management Method using a Blockchain Scheme. In Proceedings of the 2020 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 4–6 January 2020.
30. Pourmajidi, W.; Miranskyy, A. Logchain: Blockchain-Assisted Log. Storage. In Proceedings of the 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), San Francisco, CA, USA, 2–7 July 2018.
31. Hang, L.; Kim, D.-H. Design and Implementation of an Integrated IoT Blockchain Platform for Sensing Data Integrity. *Sensors* **2019**, *19*, 2228. [CrossRef]
32. Li, H.; Lan, C.; Fu, X.; Wang, C.; Li, F.; Guo, H. A Secure and Lightweight Fine-Grained Data Sharing Scheme for Mobile Cloud Computing. *Sensors* **2020**, *20*, 4720. [CrossRef]
33. Zheng, H.; Shao, J.; Wei, G. Attribute-based encryption with outsourced decryption in blockchain. *Peer-to-Peer Netw. Appl.* **2020**, *13*, 1643–1655. [CrossRef]
34. Sowjanya, K.; Dasgupta, M. A ciphertext-policy Attribute based encryption scheme for wireless body area networks based on ECC. *J. Inf. Sec. Appl.* **2020**, *54*, 102559. [CrossRef]
35. Zhong, H.; Zhou, Y.; Zhang, Q.; Xu, Y.; Cui, J. An efficient and outsourcing-supported attribute-based access control scheme for edge-enabled smart healthcare. *Future Gener. Comput. Syst.* **2020**, *115*, 486–496. [CrossRef]
36. Panko, R. Digital Signatures and Electronic Signatures. In *The Internet Encyclopedia*; Bidgoli, H., Ed.; John Wiley and Sons: Hoboken, NJ, USA, 2004.
37. Nguyen, T.; Kim, K. A survey about consensus algorithms used in Blockchain. *J. Inf. Process. Syst.* **2018**, *14*, 101–128.
38. Lewko, A.; Waters, B. New Proof Methods for Attribute-Based Encryption: Achieving Full Security through Selective Techniques. In *Advances in Cryptology—CRYPTO 2012*; Springer: Berlin/Heidelberg, Germany, 2012.
39. Beimel, A. *Secure Schemes for Secret Sharing and Key Distribution*; Technion-Israel Institute of Technology, Faculty of Computer Science: Haifa, Israel, 1996.
40. Shamir, A. Identity-Based Cryptosystems and Signature Schemes. In *Advances in Cryptology*; Springer: Berlin/Heidelberg, Germany, 1985.
41. Sahai, A.; Waters, B. Fuzzy Identity-Based Encryption. In *Advances in Cryptology—EUROCRYPT 2005*; Springer: Berlin/Heidelberg, Germany, 2005.
42. Lai, J.; Deng, R.H.; Guan, C.; Weng, J. Attribute-Based Encryption With Verifiable Outsourced Decryption. *IEEE Trans. Inf. Forensics Sec.* **2013**, *8*, 1343–1354.
43. Han, J.; Susilo, W.; Mu, Y.; Yan, J. Privacy-Preserving Decentralized Key-Policy Attribute-Based Encryption. *IEEE Trans. Parallel Distrib. Syst.* **2012**, *23*, 2150–2162. [CrossRef]
44. Bethencourt, J.; Sahai, A.; Waters, B. Ciphertext-Policy Attribute-Based Encryption. In Proceedings of the 2007 IEEE Symposium on Security and Privacy (SP'07), Berkeley, CA, USA, 20–23 May 2007.
45. Waters, B. Ciphertext-Policy Attribute-Based Encryption: An. Expressive, Efficient, and Provably Secure Realization. In *Public Key Cryptography—PKC 2011*; Springer: Berlin/Heidelberg, Germany, 2011.

46. Han, J.; Susilo, W.; Mu, Y.; Zhou, J.; Au, M.H.A. Improving Privacy and Security in Decentralized Ciphertext-Policy Attribute-Based Encryption. *IEEE Trans. Inf. Forensics Sec.* **2015**, *10*, 665–678.
47. Zheng, Y. Digital signcryption or how to achieve cost(signature & encryption) ≪ cost(signature) + cost(encryption). In *Advances in Cryptology—CRYPTO '97*; Springer: Berlin/Heidelberg, Germany, 1997.
48. Gagné, M.; Narayan, S.; Safavi-Naini, R. Threshold Attribute-Based Signcryption. In *Security and Cryptography for Networks*; Springer: Berlin/Heidelberg, Germany, 2010.
49. Hankerson, D.; Menezes, A. Elliptic Curve Discrete Logarithm Problem. In *Encyclopedia of Cryptography and Security*; van Tilborg, H.C.A., Jajodia, S., Eds.; Springer: Boston, MA, USA, 2011; pp. 397–400.
50. Gordon, D. Discrete Logarithm Problem. In *Encyclopedia of Cryptography and Security*; van Tilborg, H.C.A., Jajodia, S., Eds.; Springer: Boston, MA, USA, 2011; pp. 352–353.
51. Boneh, D.; Lynn, B.; Shacham, H. Short Signatures from the Weil Pairing. In *Advances in Cryptology—ASIACRYPT 2001*; Springer: Berlin/Heidelberg, Germany, 2011.
52. Al-Zubaidie, M.; Zhang, Z.; Zhang, J. Efficient and Secure ECDSA Algorithm and its Applications: A Survey. *arXiv* **2019**, arXiv:1902.10313. Available online: https://arxiv.org/abs/1902.10313 (accessed on 27 February 2019).
53. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. Cryptography Mailing List. 2009. Available online: https://git.dhimmel.com/bitcoin-whitepaper/ (accessed on 10 October 2020).
54. Croman, K.; Decker, C.; Eyal, I.; Gencer, A.E.; Juels, A.; Kosba, A.; Miller, A.; Saxena, P.; Shi, E.; Gün Sirer, E.; et al. On Scaling Decentralized Blockchains. In *Financial Cryptography and Data Security*; Springer: Berlin/Heidelberg, Germany, 2016.
55. Eyal, I.; Sirer, E.G. Majority is not enough: Bitcoin mining is vulnerable. *Commun. ACM* **2018**, *61*, 95–102. [CrossRef]
56. Henry, R.; Herzberg, A.; Kate, A. Blockchain Access Privacy: Challenges and Directions. *IEEE Sec. Priv.* **2018**, *16*, 38–45. [CrossRef]
57. Yeow, K.; Gani, A.; Ahmad, R.W.; Rodrigues, J.J.P.C.; Ko, K. Decentralized Consensus for Edge-Centric Internet of Things: A Review, Taxonomy, and Research Issues. *IEEE Access* **2018**, *6*, 1513–1524. [CrossRef]
58. Nongbri, I.; Hadem, P.; Chettri, S. A Survey on Single Sign-On. *Int. J. Creative Res. Thoughts* **2018**, *6*, 595–602.
59. Team, T.A. Automated Validation of Internet Security Protocols and Applications (AVISPA 1.1). Available online: http://www.avispa-project.org (accessed on 10 November 2020).
60. Von Oheimb, D. The high-level protocol specification language HLPSL developed in the EU project AVISPA. In Proceedings of the APPSEM 2005 Workshop, Frauenchiemsee, Germany, 13 September 2005.
61. Gong, L.; Needham, R.; Yahalom, R. Reasoning about belief in cryptographic protocols. In Proceedings of the 1990 IEEE Computer Society Symposium on Research in Security and Privacy, Oakland, CA, USA, 7–9 May 1990.
62. Arshad, H.; Rasoolzadegan, A. Design of a Secure Authentication and Key Agreement Scheme Preserving User Privacy Usable in Telecare Medicine Information Systems. *J. Med. Syst.* **2016**, *40*, 237. [CrossRef] [PubMed]
63. Hsu, C.L.; Le, T.V.; Hsieh, M.C.; Tsai, K.Y.; Lu, C.F.; Lin, T.W. Three-Factor UCSSO Scheme With Fast Authentication and Privacy Protection for Telecare Medicine Information Systems. *IEEE Access* **2020**, *8*, 196553–196566. [CrossRef]