electronics

MDPI

Article

An Attribute-Based Collaborative Access Control Scheme Using Blockchain for IoT Devices

Yan Zhang [1], Bing Li [1,2,*], Ben Liu [2], Jiaxin Wu [2], Yazhou Wang [2] and Xia Yang [1]

[1] School of Cyber Science and Engineering, Southeast University, Nanjing 210096, China; yanZhang930807@hotmail.com (Y.Z.); destiny_young@seu.edu.cn (X.Y.)

[2] School of Integrated Circuits, Southeast University, Nanjing 210096, China; liuben_ic@seu.edu.cn (B.L.); jx_wu@seu.edu.cn (J.W.); wangyazhou_seu@126.com (Y.W.)

* Correspondence: bernie_seu@seu.edu.cn; Tel.: +86-1536-504-5432

Received: 6 January 2020; Accepted: 4 February 2020; Published: 7 February 2020

Abstract: The Internet of Things (IoT) benefits our lives by integrating physical devices to the real world and offers a crucial internet infrastructure for future civilization. Because IoT devices are widely distributed and restricted in resources, it is difficult for them to adopt traditional security methods to resist malicious attacks. Unauthorized access to IoT devices, which results in severe privacy and security problems, has become a major challenge that has impeded IoT technology from being widely adopted. Therefore, the access control for IoT devices urgently needs to be improved when dealing with authorization issues. In this paper, we propose an attribute-based access control scheme that provides decentralized, flexible, and fine-grained authorization for IoT devices. Blockchain is utilized to provide authentic and reliable credentials. More importantly, a verifiable collaboration mechanism is designed to meet the needs of controlled access authorization in emergencies. Authority nodes are constructed to execute major computation tasks and interact with the blockchain. The security analysis shows that our scheme can reliably guarantee the security of authorized access. More than security assurance, a proof-of-concept prototype has been implemented to prove that our scheme is scalable, efficient, and accommodates IoT devices well.

Keywords: blockchain; attribute-based access control; authorization; IoT; collaboration

1. Introduction

The Internet of Things (IoT) has emerged as a revolutionary type of technology that connects all smart devices together through a distributed capillary networking infrastructure. It enables IoT smart devices to collect and share data more efficiently and autonomously [1], making changes in every corner of our daily lives, including healthcare, transport, environment, energy, business, and culture [2]. Although IoT is promising, security and privacy have been two major issues that have become a bottleneck impeding the application of IoT technology in open environments, where IoT devices are connected to the internet and exposed to unauthorized access [3]. Access control is a technology that can restrict access privileges to a target according to a control rule, thereby helping to solve these security issues. The widely known traditional centralized access control modes include discretionary access control (DAC), mandatory access control (MAC), and role-based access control (RBAC). However, the most widely distributed IoT devices can hardly meet the requirements of traditional security systems due to their limitations in CPU, memory, and battery resources [4], as well as their decentralized and dynamic architectures [5]. DAC assigns an authorization list or matrix to each object, which is impossible for subjects without identifiers or without enough resources. MAC relies on a central authority and is too rigid for IoT scenarios. When adapting RBAC into IoT scenarios, the number of rules that need to be managed increases exponentially with the growth of devices.

Attributed-based Access Control (ABAC) [6] is regarded as one of the most suitable decentralized models for IoT scenarios and offers a large scale, flexibility, and strong dynamicity [3]. In ABAC, access is granted to the requester according to the attributes presented by a target. A target's functions, identities, roles, and other complex features are all abstracted into attributes. These attributes are also used to express specified access policies by a target to decide whether the requester has sufficient privileges for access. When being adopted to IoT environments, ABAC has the ability to provide flexible and fine-grained control over the access requests for every device. For example, to restrict the availability of the data generated by IoT devices, researchers have employed attribute-based encryption (ABE) schemes [7,8] by using each user's attributes to encrypt their information. This ciphertext can be decrypted and obtained by the data's users, who have enough attributes to meet the requirements of the access policy defined by the data owner. However, the computation overhead of this method is too large for IoT devices, and their ABE schemes mainly focused on encryption and storage, not authorization for real-time access. To establish secure real-time access and protect the perception layer, Ye [9] proposed an authentication and authorization scheme for IoT, which restricts the privileges of authenticated users by using an ABAC policy. Hemdi [10] and Sciancalepore [11] also applied ABAC to IoT platforms by utilizing attributes and ABAC policies to restrict access or detect malicious behaviors. However, these schemes require very complex management, and the authorization credentials may be untrustworthy in IoT scenarios.

In addition to the above-mentioned problems, in particular scenarios, unauthorized devices should be able to request collaboration to obtain extra access permissions. The demand for collaboration for IoT was detailed in work [12], proposed by Castiglione et al. They addressed this issue by proposing a novel hierarchical access control model and constructing two schemes that implement the model. The collaboration that prevents the abuse of permissions and separates duties inherent in their work is enlightening. Therefore, the collaboration offered by the scheme that implements the ABAC model should be trustworthy. We use an example to illustrate the concept of collaboration in ABAC model and the construction of a collaboration access policy. In enterprises, we should restrict access to surveillance cameras to security and strictly define the access policy A, as shown in Figure 1. The monitors that have a set of attributes {Security Department, Surveillance, Enterprise A} are able to access the camera and obtain real-time data. However, when a staff member wants to use a phone that has the attribute set {Security Department, Enterprise A} to access the camera to deal with emergencies, policy B (Figure 1) is necessary. That is to say, cameras should be accessible in both policies A and B. Under these conditions, the staff member can use his or her phone to send a collaboration request to the device assigned to the attribute {Manager}. After this extra authorization, the collaboration access will be permitted. In this way, we can construct a new collaboration access policy mixing A and B, as shown in the right side of Figure 1. Xue et al. reported a collaboration scenario concentrated on encryption for cloud storage [13]. The collaboration scenario presented in Figure 1 is similar to the one in [13] but has essential differences. The work of Xue et al. provided an encryption method to restrict and control access to the sharing of data. However, the bilinear maps used in their work are not suitable for resource-constrained IoT devices and cannot be used to deal with authorization for real-time access, which is focused upon in this paper. In general, the main research purpose of this paper is to utilize the ABAC model to improve the access control technology in IoT in order to guarantee the security of real-time IoT authorization. When integrating the ABAC model into IoT scenarios, three major problems need to be solved by the above-mentioned analysis. One involves the storage and computational overheads of the proposed scheme, which should be accepted by IoT devices. Another is that the distribution and acquisition of attributes and access policies should be credible and reliable. The latter urgently requires a trustworthy collaborative access method for IoT devices.

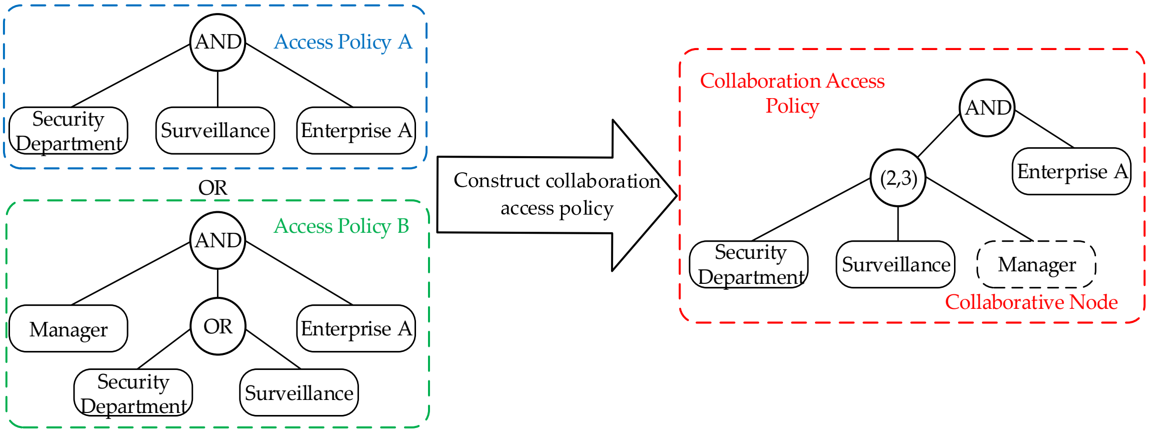


Figure 1. Schematics of the concept of collaboration and the construction of a collaboration access policy.

To solve these problems, a collaborative access control scheme for IoT is proposed by combining the ABAC model and blockchain technology [14]. By adopting blockchain, this solution can build trust among IoT devices and need not rely on third-party authorities. Blockchain is utilized as a key-value database, in which the stored information is distributed and resistant to a single point of failure. The secure database is utilized to provide credible credentials as well as to trustfully transmit access information for authorization on behalf of IoT devices. A verifiable collaboration mechanism is also designed to satisfy the collaboration requirement. Authority nodes are constructed to verify the access credentials by interacting with the blockchain network and afford most of the computing tasks for authorization. Our scheme overcomes the above-mentioned problems by these three methods and the main contributions of our scheme are threefold:

(1) We address the authorization issue of real-time access to IoT devices by proposing a collaborative ABAC scheme using blockchain. Blockchain is utilized to generate a digital account for each device to record the attributes and access policy used for authorization, as well as to forward access information trustfully. The data stored in the blockchain is reliable and credible, guaranteed by the no single point failure and tamper-proof feature offered by the blockchain. Based on these credentials, our scheme can reliably guarantee authorized access and is both efficient and scalable.

(2) A controlled and verifiable collaboration mechanism is also introduced when trustworthy collaboration is needed. This mechanism will ask for extra authorization, reconstruct the access tree, and prevent unwanted collaboration with the help of three novel proposed algorithms. The data structure of the access tree built from the access policy in our scheme was specially designed. Moreover, the access tree is modified by adding a collaborative node so that collaboration is controlled and verifiable.

(3) Authority nodes are constructed to build an access structure from access policy, to make authorization decisions and to interact with the blockchain network. Moreover, IoT devices only need to store a string of access information and perform a constant number of simple cryptographic computations. These factors will make our scheme light-weight and well-adapted to IoT scenarios.

The rest of the manuscript is organized as follows: Section 2 contains background and related work. Section 3 presents the architecture of our proposed system. The implementation of the scheme is detailed in Section 4. The security and performance analysis are presented in Sections 5 and 6. Finally, Section 7 introduces the conclusions and main results of this paper.

2. Background and Related Work

In this section, we first introduce two important technologies utilized in our scheme. Then, related works that concern the authorization issue of IoT devices using the blockchain are summarized.

2.1. Attribute-Based Access Control Model

We detail the construction of the ABAC model that is implemented in our scheme. The ABAC model is governed by the attributes and access policies. The access policy should be translated into the access structure. According to the definition of access structure [15], it is efficient to judge whether the requester's attributes meet the target's access requirements.

The way we translate the access policy to the access structure in our scheme is the same as the process in [12]. The access structure is represented by an access tree made of non-leaf nodes and leaf nodes. Each non-leaf node of the tree represents a threshold gate k and the number of children nodes n, where $0 < k \leq n$. When k = 1, the node can be seen as an OR gate, and when k = n, it becomes an AND gate. Each leaf node is composed of a threshold value k = 1, and one of the attributes described in the access policy, which is denoted as att_i [12,16].

Then, we modify the access tree to meet the collaboration requirements in our work by introducing a collaboration node, as shown in Figure 1. The collaboration node (CN) is designed to be a leaf node. The feature of verifiable and controlled collaboration lies in a key modification, whereby CN additionally stores the identity of the group, denoted as GroupId. The GroupId is used to restrict collaboration in a certain group. In our scheme, devices in the same group are allowed to provide collaborative attributes in order to help the requester obtain secure authorization.

Let T be an access tree with root node γ. Then, we designed an efficient algorithm, Satisfy(L), to compute whether a list of attributes L satisfies access tree T. We execute the algorithm from the root node γ recursively. If x is a non-leaf node, we obtain the result of $\text{Satisfy}_x(L)$ by computing $\text{Satisfy}_{x'}(L)$ for all children x′ of node x. Satisfy(x) evaluates TRUE when at least k_X children nodes returned by $\text{Satisfy}_x{}'(L)$ are TURE. If x is a leaf node, Satisfy_x (L) returns TRUE only if att_i belongs to L. Additionally, for the collaboration node x_C, the GroupId stored in x_C is used to verify whether the provider of att_i belongs to the GroupId. If so, $\text{Satisfy}_{X_c}(L)$ also returns TRUE.

2.2. Blockchain Technology

The blockchain plays an important role as a distributed database that is used to provide credible and public digital credentials. Blockchain technology has the following features:

1. Decentralization: The decentralized architecture of the blockchain has great advantages in its scalability and flexibility. Moreover, there is no need to rely on a central authority to build trust among participators in the blockchain. All transactions recorded by the public ledger will be validated by all peers to reach a consensus.
2. Distribution: All peers in the blockchain preserve a digital and public data ledger, which eliminates the problem of a single-point failure.
3. Security: The blockchain is tamper-proof and secured by cryptographic tools. All information recorded in the blocks cannot be manipulated.

We chose a consortium blockchain [17] project called Hyperledger Fabric [18] to construct our blockchain network. Compared with the public blockchain, the consortium blockchain executes more efficient and less expensive consensus algorithms, such as the Kafka and Raft [19]. These algorithms deal with thousands of transactions per second, which make the consortium blockchain more efficient for validating transactions and forming new blocks. The features of the consortium blockchain are appropriate for the efficiency required in our scheme [20,21].

2.3. Related Works

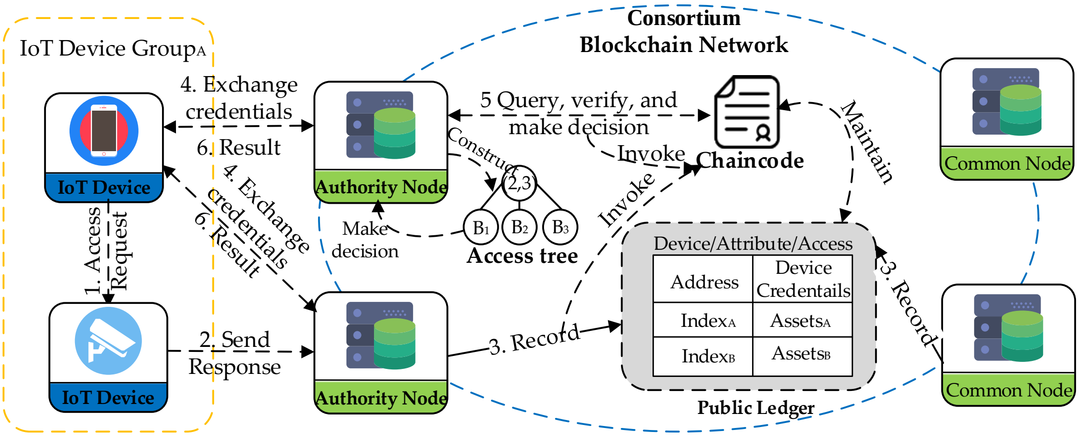
Our proposed method, concentrating on the issue of unauthorized access in IoT environments, requires the collaboration of the blockchain and access control. Therefore, we explore a combination of these two research scopes by presenting the previous research and comparing related works to our proposed scheme.

Novo [5] proposed a distributed blockchain-based authorization scheme to manage devices in IoT. A special design was presented in his work to avoid integrating the blockchain into IoT devices, which provides great inspiration to the design of our scheme. This design extends the application of blockchain technology to more wide-ranging IoT scenarios [5], especially for resource-constrained devices. Ouaddah [22,23] described a blockchain-based authorization scheme named FairAccess. Smart contracts were used to trade fulfillments of access control policies for access tokens. The authors included IoT devices in the blockchain but did pursue the real-time authorization issue or the efficiency of the scheme. Xu [24] proposed a decentralized, federated capability-based access control mechanism using a smart contract. This scheme is scalable, light-weight, and supports hierarchical and multi-hop delegation. The Control Chain in [25] is user transparent, user-friendly, fully decentralized, and fault-tolerant. However, it has to maintain four different blockchains to perform access control, and the efficiency of the scheme was not proven in their work.

To the best of our knowledge, the only previous work that also utilized blockchain and the ABAC model to handle the issue of real-time authorization for IoT is [21]. However, the methods in [21] that are used to integrate the ABAC model into IoT scenarios are totally different from our scheme and the differences are threefold. First, we translate the access policy into an access tree and do not need to forward the attributes list during an authorization request; this makes the authorization process controlled and efficient. Instead, the work in [21] requests the device to select the satisfied subset of the policy by itself, whose performance and reliability have not been proven theoretically. Second, our scheme can meet the demands of collaborative authorization, which was not included in [21]. Moreover, our collaboration process is efficient and secure based on the special design of the access tree. Third, our scheme constructs authority nodes to execute computation tasks and delegate IoT devices to interact with the blockchain. However, the devices used in [21] had to query and invoke the chaincode by themselves.

3. Overview of Our Proposed Access Control System

To facilitate understanding, the system in which we implemented our proposed scheme will be detailed in this section, as presented in Figure 2. A brief introduction is given before the comprehensive description that is provided in Sections 3.1–3.6.



Address	Device Credentails
IndexA	AssetsA
IndexB	AssetsB

Figure 2. The architecture of the internet of things (IoT) system in which we implement our proposed access control scheme.

First, the system is mainly composed of five parts that are marked as bold in Figure 2:

(1) Consortium blockchain network.
(2) Authority node.
(3) IoT devices.

(4) Chaincode and the public ledger.
(5) Access Tree

We explain the relationships between these components. Our blockchain network includes authority nodes and common nodes. The blockchain information denoted as the public ledger will be copied and recorded by all relative participants in the blockchain. IoT devices belong to a certain group and will be allocated with a group ID and an IP address. With this information, the IoT devices in our system can communicate with the devices in any group. The majority of IoT devices are resource-constrained and cannot store the public ledger [5]. Therefore, we separate IoT devices from the blockchain and introduce authority nodes to function as the blockchain clients, which take the responsibility of interacting with the blockchain network on behalf of the IoT devices. Additionally, a chaincode is deployed on the authority nodes in our system. Transactions are sent to query or invoke the chaincode by clients to maintain the distributed public ledger. The access tree is built from the access information collected from the public ledger by the authority node (AN).

Then, the main authorization process supported the five constituents aforementioned is shown in Figure 2 and can be briefly summarized as follows:

1. The requester sends the access request to the target;
2. Then, the target forwards the access information to the AN;
3. The AN will send the transaction to invoke the chaincode to record the access information and transmit the response to the requester.
4. The requester signs the required information using the private key and builds the exchanged access information. This information will be sent to the AN to successfully satisfy the access policy. Every AN can handle the request, due to the fact the access information stored in the public ledger in Step 3 is distributed and available without a single point failure.
5. Afterward, the AN will query the chaincode and retrieve the registered access credentials to verify the validity of the requesters' identity and the target's access policy. Then, the AN constructs the access tree to make authorization.
6. Finally, the AN will record the final access information with the authorization result to the blockchain and send the result to the requester.

Last, we present the organization of our description of our system. Sections 3.1–3.5 describes the above-mentioned five important constituents. Section 3.6 introduces assumptions in our system, including the threat model and the security model. The detailed description is listed as follows.

3.1. Consortium Blockchain Network

The design of the consortium blockchain network is described in this part. The blockchain network constructed by the Hyperledger fabric works as a distributed database to provide reliable digital credentials for IoT devices. The nodes in this certain consortium blockchain can be categorized as Certificate Authority (CA) nodes, Order nodes, and Peer nodes. Peer nodes can be further divided into Committer, Endorser, Leader, and Anchor nodes. All the peer nodes will function as Committers to record a copy of the blockchain. All these nodes have their own duties and work together to sort the transactions, generate new blocks, and finally reach a consensus. We construct ANs as endorser nodes, and the other nodes are denoted as common nodes in our system. The blockchain client is installed on the authority node and utilized to query and invoke the chaincode by sending transactions. All the query and invoke operations will be recorded in the form of transactions in the blocks.

3.2. Authority Nodes

The authority node (AN) has two important functions in our scheme. Firstly, the AN acts as an endorser peer [18] in this consortium blockchain. The chaincode is installed in it to provide a blockchain service that generates trustworthy digital accounts for registered devices by maintaining a key-value state database. The devices' public keys, attributes, access policies, and other digital credentials will all be registered by sending transactions to the blockchain. Once the transactions are validated by other nodes, the devices' accounts will be updated accordingly. Secondly, the AN deployed near IoT networks is also a credible hub. It not only assigns the attributes to devices but also takes responsibility for policy construction and decision making. Moreover, for distribution, we allocate public IP addresses to the ANs so that the ANs can be reached by IoT devices in different groups to transmit the message through the blockchain and provide distributed, reliable, and scalable services. The ANs can be gateways, personal computers, servers, or even smart phones, which are relatively rich in resources.

3.3. IoT Devices

We explain the requirements of IoT devices in our scheme. Each IoT device should possess an Elliptic Curve Cryptography (ECC) key pair. Although separated from the blockchain network, these devices possess their own individual blockchain accounts to record registered information, including attributes and access policies. Only if a requester's authorized attributes satisfy the target device's access policy will access be permitted. The address of a device's account is produced by computing the hash of the device's identity, denoted as Hash (IDs). Accordingly, the device's identity ID is generated from the ECC public key.

3.4. Chainchode and Public Ledgers in the Blockchain

Our proposed system is supported by a chaincode instanced on the authority node. We present the main functions of the chaincode by describing the data structure of the public ledgers that it maintained. The chaincode is invoked or queried by the blockchain client and generates new transactions. Then, the changes of the data brought by these transactions will work together to maintain a key-value state database, which is denoted as the public ledger. There are three closely related key-value databases maintained in our scheme, including Device, Attribute, and Access. Their data structures are marked as red in Figure 3, and the descriptions are listed as following:

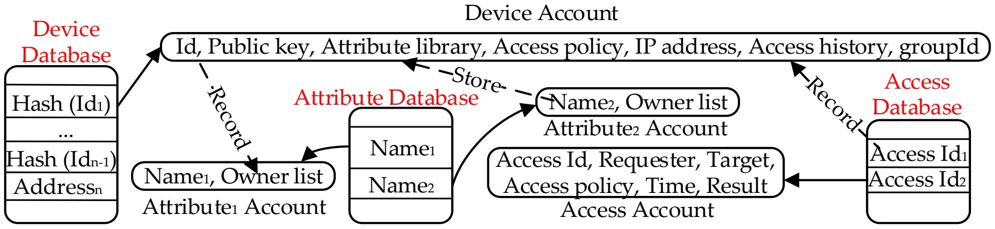


Figure 3. Data structures of the Device Database, Attribute Database, and Access Database maintained by the chaincode.

(1) Device database: When a device is registered, the device's address, denoted as Hash (Id), will be the index in the database. The detailed registration credentials, including the device's public key, Attribute library, device's access policy, IP address, groupId, and Access history, will be recorded as the value in the form of Device Account, as presented in the top of Figure 3.

(2) Attribute Database: When an attribute is registered, the attribute will be indexed by the attribute name. The value, recorded in the form of Attribute Account, is composed of the attribute name and the owner list. When an attribute is assigned to a device, the identity of the device will be added to the owner list. At the same time, the attribute name will be assigned to an owner and stored in its related Attribute library, which is contained in the Device Account mentioned above.

(3) Access Database: The access information sent by the requester will also be stored in the blockchain during the authorization process. The blockchain is utilized to transmit these data reliably and trustfully. The key of the database is the identity of the access request and the data structure of the

value is Access Account, as shown in the middle of Figure 3. When the authorization result is finally recorded, the access Id will be stored in Access history.

In general, the chaincode in blockchain will maintain these three databases, with the purpose of providing credible credentials and transmitting the data trustfully. In addition, the changes in Attribute Database and Access Database will update the Device Database dynamically.

3.5. *Access Tree*

The access tree is detailed from three aspects, including its data structure, the collaborative node, and the reconstruction mechanism. In our scheme, each device can define the access policy according to its own requirements. The access policy is required to be translated into the access tree. For collaboration, the collaborative node is introduced, as well as a mechanism that reconstructs the access tree for verification.

(1) Access Tree: For illustration purposes, we take advantage of an access policy, as shown in the top of Figure 4a. The policy is specially described by a string and will be transformed into an access tree by the AN. The access tree contains three kinds of nodes: the root node, the leaf node, and the non-leaf node. The data structure of the node is shown in the left side and has the threshold (k), the number of the child nodes (n), the groupId, and two flags that are used to judge the state of the node during the process of decision making. In addition, the data structure of Attribute in a node is displayed on the right side. When the access tree is initialized, the policy string is split, and the information of each node is stored in the form of a string array. Every node is in turn constructed based on the array from the leftmost leaf node to the root node. The understanding of the construction process of the access tree should be combined with the definition in Section 2.1.

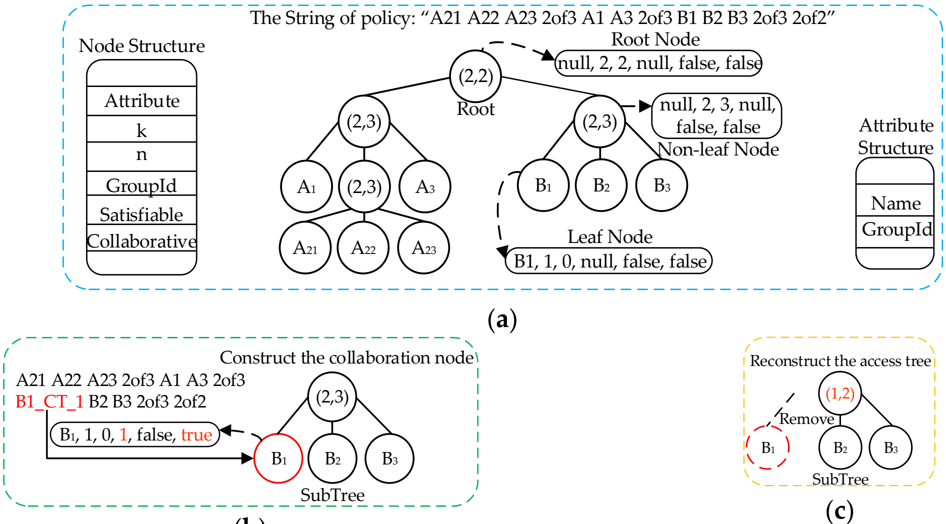


Figure 4. Data structure of the construction of the access tree: (**a**) the data structure for all kinds of nodes and attributes; (**b**) the construction of the collaboration node; (**c**) the reconstruction mechanism of the access policy.

(2) Collaborative Node: When the target needs to set the B1 as the collaborative node, the access policy should be modified as shown in Figure 4b. More specifically, during the construction of the collaboration node, the target will set the collaborative flag to be true and add the required groupId to the node. Thereby, the collaborative node helps to restrict the collaboration in a certain group.

(3) Reconstruction Mechanism: Last, we introduce a reconstruction mechanism that modifies the access tree as presented in Figure 4c. During reconstruction, we will remove the collaboration node from the subtree. Correspondingly, the father of the collaboration node will modify the k and n to

accommodate to the new tree. This mechanism is frequently used to judge whether the requester is allowed to request extra authorization.

3.6. Threat and Security Model

In this part, we detail the models and assumptions in our system, which will enhance the trustworthiness as well as the security of the proposed scheme. We first summarize four types of attacks that aim at obtaining access authorization based on the threat model. Then, we introduce our security model by proposing several security assumptions for our system, in order to achieve the goal of resisting the attacks threating access authorization.

3.6.1. Threat Model

The threat model is based on the widely-used Dolev–Yao model [26]. In our model, each IoT device belongs to a certain group and is able to communicate with other devices and exchange the packets inside or across the group through an open and unreliable channel. However, there are no security methods able to protect an open channel. That is to say, an attacker can read, modify, drop, or inject network messages in an open channel. Our threat model also has similar abilities to the model in [27,28]. The main difference is that the attacker will try to satisfy the access policy by obtaining or using attributes illegally [13].

The main goal of the attacker is to obtain authorization privileges for the target, although the attacker is assumed to control the open channel [26,27] based on the threat model mentioned-above. Therefore, we are interested only in attacks threatening the authorization process. Attacks that block the open channel or deny service (DoS/DDoS) are not under discussion [28]. Our scheme is effectively resistant to various types of attacks:

1. Collusion Attack: Devices may be untrusted and have the chance to collude with other devices. To satisfy the access policy, the attacker may try to collect enough attributes from other devices.
2. Malicious Collaboration Attack: An attacker may use attributes illegally and propose a malicious collaboration, where the behavior is inherently unwanted or the extra attributes requested are from undefined groups.
3. Relay Attack: An attacker may choose to record some packets, such as signatures, during the transmission process and reply to them in another request. Since the messages are generated by valid users, there is a possibility that this information can satisfy the verification and help obtain illegal authorization.
4. Message Substitution Attack: An attacker may create a false identity to impersonate a certain device in the open channel in order to use one's privilege. More problematically, valid messages may be intercepted and altered intentionally so that the target cannot perceive the forgery of the information and will accept them as usual.

3.6.2. Security Model

The security model consists of several reasonable assumptions of the system. The ANs are assumed to be semi-honest [28] and always available. More specifically, the algorithms in our scheme will be executed correctly by the ANs, which are intended to be curious and honest and try their best to infer and obtain sensitive information [13]. In addition, the AN will store its private key securely. IoT devices are usually exposed to malicious attacks and are viewed as untrustworthy participators in our scheme. In our security model, we assume that IoT devices will not suffer from physical attacks under the protection of the solutions proposed in [29,30]. This means that IoT devices can secretly store sensitive information, such as private keys. The assumptions in our scheme has many similarities to the ones proposed in [5,21]. The main difference lies in the blockchain network.

From the perspective of the blockchain network, the scope of the security assumption is extended and is different from the model in [21], which resists Byzantine failures. We assume that the blockchain network in our system is safe, as in [5,31], which means that the security model of the blockchain network can be alternated according to the consensus algorithm that it uses. Transactions proposed by blockchain clients should be correctly recorded to form new blocks and also maintain the state of the database. Based on the assumptions of the system, the scheme implemented should be trustworthy and secure.

4. Proposed Access Control Scheme

In this section, we detail our proposed access control scheme implemented in the above-mentioned system. The integrated authorization process is divided into four different phases, including system initialization, registration, authorization, and collaboration. These phases will be explained as follows.

4.1. System Initialization Phase

We initialize each IoT group and the authority node individually using the following steps:

1. The AN chooses a non-singular elliptic curve Ep(a,b) over a prime finite field $\mathbb{Z}_p$, in which the elliptic curve discrete logarithm problem is difficult. Then, the AN selects a base point P of order n over Ep(a,b) such that n*P equals a zero point or a point in infinity.
2. The AN randomly selects the ECC private key priKey$\in\mathbb{Z}_p^*$ and obtains its public key Q = priKey*P. Moreover, we use a collision-resistant one-way cryptographic hash function, denoted as Hash, which maps a bit string with a flexible size into a new bit string of a fixed size.
3. Lastly, the AN acquires its ECC private key priKey, and the system parameters {Ep(a,b), Q, P, p, Hash} are made public.

4.2. Registration Phase

After the initialization of the system, devices' credentials are registered in the blockchain in this phase. The related three operations should be performed in a secure and private environment.

1. Each device generates its own ECC key pairs and sends its registration parameters {ID, GroupId, L, PK, Policy, IP address} to the AN. L represents a list of the device's attributes, and Policy is denoted as the access policy.
2. Then, the AN generates the address of each device's account in the blockchain by computing the hash of the identity:
$$\text{Address} = \text{Base68Check(ID)}. \quad (1)$$
3. Afterward, the AN authorizes the device's attributes and invokes the chaincode to upload the device's digital credentials to Device database, as explained in Section 3.4. Thus, this information stored in the blockchain is credible and trustworthy.

4.3. Authorization Phase

At first, we will explain how multiple participators authenticate each other and establish session keys to protect open channels before the introduction of our authorization process. We utilize an authentication and key agreement (AKA) scheme based on the blockchain, the ANs' public parameters, the devices' digital accounts, and the Elliptic Curve Integrate Encrypt Scheme (ECIES) [32]. The AKA scheme will establish session keys for each access request, which is specially designed for the system architecture. However, this paper concentrates on the issue of authorization for IoT devices. Thus, we omit detailed descriptions of the AKA scheme and only give a performance analysis in Section 6. Indeed, the AKA process can also be operated using other well-known and outstanding schemes for IoT. Wazid et al. [33] proposed a secure and very light-weight three-factor authentication scheme that applies smart cards, passwords, and personal biometrics, which proves to be secure formally. Aman et al. [34] proposed a mutual authentication scheme using physical unclonable functions (PUFs). The scheme contributes to establishing session keys between a device and a server or two devices.

The identity-based authentication scheme [35] is also efficient upon verification of the device's identity. In summary, our choice of the AKA scheme will not influence the security and performance analysis of our authorization process.

The authorization process between devices S_{Re} and S_{Target} is performed, as shown in Figure 5, where En{*} and De{*} stand for the symmetric encryption activity AES-128, and Sig{*} and Ver{*} are the operations using an elliptic curve digital signature algorithm scheme (ECDSA). The operations conducted by the related objects are itemized in a rectangle. The arrows in Figure 5 show the directions of data flow.

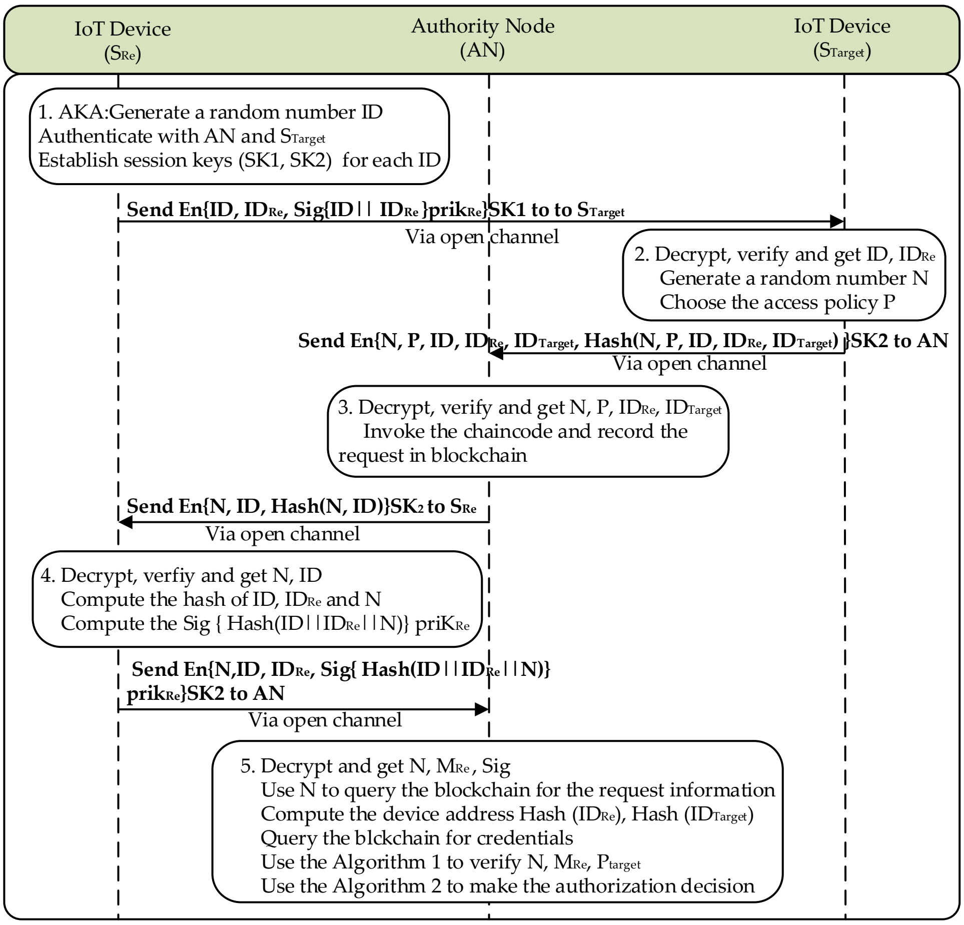

IoT Device
(SRe)
Authority Node
(AN)
IoT Device
(STarget)
1. AKA:Generate a random number ID
Authenticate with AN and STarget
Establish session keys (SK1, SK2) for each ID
Send En{ID, IDRe, Sig{ID | | IDRe }prikRe}SK1 to to STarget
Via open channel
2. Decrypt, verify and get ID, IDRe
Generate a random number N
Choose the access policy P
Send En{N, P, ID, IDRe, IDTarget, Hash(N, P, ID, IDRe, IDTarget) }SK2 to AN
Via open channel
3. Decrypt, verify and get N, P, IDRe, IDTarget
Invoke the chaincode and record the
request in blockchain
Send En{N, ID, Hash(N, ID)}SK2 to SRe
Via open channel
4. Decrypt, verfiy and get N, ID
Compute the hash of ID, IDRe and N
Compute the Sig { Hash(ID| |IDRe| |N)} priKRe
Send En{N,ID, IDRe, Sig{ Hash(ID | |IDRe | |N)}
prikRe}SK2 to AN
Via open channel
5. Decrypt and get N, MRe, Sig
Use N to query the blockchain for the request information
Compute the device address Hash (IDRe), Hash (IDTarget)
Query the blckchain for credentials
Use the Algorithm 1 to verify N, MRe, Ptarget
Use the Algorithm 2 to make the authorization decision

Figure 5. The authorization process of our proposed attribute-based access control scheme between two devices using a blockchain in an IoT scenario.

The authorization process includes five steps:

1. Firstly, the requester generates a random number as the ID to label each access request. Session keys (SK1, SK2) will then be established and stored through the AKA process. SK1 is used to protect the open channel between the IoT devices, and SK2 guarantees communication between the devices and authority nodes. The requester sends its identity ID_{Re} and access number ID, as well as its signature Sig {ID|| ID_{Re}) }$prik_{Re,}$ signed by the requester, to the target:

$$\text{En} \{ \text{ID}, \text{ID}_{Re,} \text{Sig} \{\text{ID}\| \text{ID}_{Re}) \} \text{prik}_{Re} \} \text{SK1}. \quad (2)$$

2. The target decrypts the packet and verifies the signature, generates a random number N, chooses the access policy P, and calculates the hash of all the required information as shown in Equation (3). Then, the target builds the packet and send it to the AN:

$$\text{En}\,\{\, \text{N, P, ID}_{\text{Re}}, \text{ID, ID}_{\text{Target}}, \text{Hash(N, P, ID}_{\text{Re}}, \text{ID, ID}_{\text{Target}})\}\, \text{SK}_{2.} \tag{3}$$

3. The AN decrypts the packet and verifies the hash. Then, the chaincode deployed in the AN will be invoked. The access history forwarded by the target will be recorded temporarily, except for the result. As is known to us, the information stored in the blockchain ledger is tamper-proof and distributed. Thus, the access history can be trustfully and reliably queried by each AN, which will be used in Step 5 later. Moreover, the random number N should be a unique index in the Access Database in the blockchain ledger. A new random N needs to be generated by the target until the access information can be successfully recorded. After finishing these computational tasks, the AN then transmits the packet in Equation (4) to the requester:

$$\text{En}\,\{\, \text{N,ID, Hash(N}\|\text{ID)}\,\}\, \text{SK}_{2.} \tag{4}$$

4. The requester receives, decrypts the packet, and verifies the hash. Then, an access credential will be constructed by computing the hash of ID, ID_{Re} and N, denoted as $\text{Hash(ID}\|\text{N}\ \|\text{ID}_{\text{Re}})$. The requester computes the signature $\text{Sig}\{\ \text{Hash(ID}\|\text{N}\ \|\text{ID}_{\text{Re}})\}\text{priK}_{\text{Re}}$ and sends the packet to the AN:

$$\text{En}\,\{\, \text{ID, N, ID}_{\text{Re}}, \text{Sig}\,\{\text{Hash(ID}\|\text{N}\ \|\text{ID}_{\text{Re}})\,\}\, \text{prik}_{\text{Re}}\,\}\, \text{SK}_{2.} \tag{5}$$

5. The AN decrypts the packet and obtains the relevant information. A random number N is utilized to query the blockchain to obtain the request history { N, ID_{Re}, $\text{ID}_{\text{Target}}$, P_{Target}, time } recorded in Step 3. Then, the AN computes Hash (ID_{Re}) and Hash ($\text{ID}_{\text{Target}}$) and obtains the devices' registered information from related Device Account in Device Database, including { pubK_{Re}, L_{Re} } and { P_{Target} }. Afterward, the AN needs to verify the validity of the access information, including the ID_{Re}' and the access policy $\text{P}_{\text{target}}'$. Algorithm 1 describes the verification process. Operation 1 in Algorithm 1 verifies whether the access Policy received by AN in Step 2 is equal to the one registered by the device. Then, in Operation 2 and 3, we compute the hash and verify the signature mainly to prove the random number N is valid. Through Algorithm 1, we can verify the validity of access credentials.

Algorithm 1 Verify the validity of access credentials.

```
Input: The received $ID_{Re}$, ID and the random number N, the public key of the requester $pubK_{Re}$, the signature result Sig, the target's policy $P_{Target}$, the target's policy in the blockchain $P_{Target}'$
Output: The result of the verification VRes
1: VRes1 = whether $P_{Target}$ equals $P_{Target}'$, VRes2 = TRUE
2: Calculate the hash of the N, $ID_{Re}$, ID and obtain the result $Hash(ID\|N\ \|ID_{Re})$
3: Utilize the $pubK_{Re}$ to verify the signature
VRes2 = VRes2 && $Verify(Sig, Hash(ID\|N\ \|ID_{Re}))pubK_{Re}$
4: VRes = VRes1 && VRes2
```

If Algorithm 1 returns TRUE, the AN will continue calculating Algorithm 2 to make the access control decision. The input of Algorithm 2 includes the attribute list L_{Re} and the policy P_{Target} obtained above. In Operation 1, the access policy described by a string is then transformed into an access tree, as explained in Section 2. The data structure of the node and the tree is shown in Figure 4. Then, Operation 2 in algorithm computes the Satisfy function also described in Section 2.1 and judges whether the attribute list can satisfy the access tree. Finally, the AN will invoke the chaincode to record the authorization result to its history in the blockchain. If Algorithm 1 returns FALSE, the access request will be denied.

Algorithm 2 Make the authorization decision.

```
Input: The attribute list $L_{Re}$, a policy of string $P_{Target}$
Output: The result of the authorization Ares
1: Transform string P to the access tree from the leaf nodes to the root node and construct each node according to Figure 4a.
2: Calculate the function Satisfy (root, $L_{Re}$) recursively, starting from the root node to the leaf node, and finally obtain the root
3: Ares = root.isSatisfiable (one field in the data structure of the node)
```

However, if Algorithm 2 returns FALSE, S_{Re} does not have enough attributes and has no right to access S_{Target} individually. Therefore, we designed an efficient collaboration mechanism to help S_{Re} apply for extra authorization.

4.4. Collaboration Phase

In this phase, we mainly introduce a mechanism that makes contributions to detect malicious collaboration and restrict behaviors to a certain group. When the access request is denied by the aforementioned authorization phase, our scheme comes to the collaboration phase that requires a trustworthy collaboration. Hence, a verifiable and controlled collaboration mechanism is proposed, as presented in Figure 6. The entire mechanism consists of four steps:

1. When Algorithm 2 in the authorization phase returns false, a collaboration request is applied. The AN will reconstruct another access tree by removing the collaborative nodes from the leaf nodes. Afterward, the threshold k and the number n in the non-leaf node should be altered accordingly, as demonstrated in Figure 4c. This means that the only collaborative attributes needed by the requester are located in the removed collaboration nodes, so malicious collaboration can be detected easily. The reconstruction mechanism is explained in Section 3.5. Then, Algorithm 2 is utilized to make the decision again, and AN sends the result {res, L_{co}} to S_{Target}, while L_{Co} is a list composed of the attributes needed during collaboration:

$$\text{En}\ \{\ \text{ID, N, \{res, } L_{Co}\text{\}, Hash(ID}||\text{N}||\text{\{res, } L_{Co}\text{\})}\ \}\ \text{SK2}. \tag{6}$$

2. If the result of the collaboration request is TRUE, the requester will use the attributes received to find the collaborator and perform the AKA process to establish session keys. Or else, the request is denied as well as the access request. Then, the requester transmits the packet in Equation (7) to the collaborator. The L_{Co} is the attribute list that should be provided by the collaborator and N used to label the access request remains unchanged:

$$\text{En}\ \{\text{ID, N, } L_{Co}\text{, Hash(ID}||\text{N}||L_{Co}\text{,)}\}\ \text{SK1}. \tag{7}$$

3. The collaborator obtains the L_{Co} and generates the attribute map M_{Co} {ID_{Co}, L_{Co}}. More importantly, the collaborator will set its groupId for each attribute in L_{Co}. Because the collaboration should be allowed in an assigned group. Then, the collaborator computes the signature Sig{N||Hash(M_{Co})}priK_{Co} and sends the packet to the AN:

$$\text{En}\ \{\ \text{ID, } M_{Co}\text{, N, Sig}\ \{\text{ID}||\text{N}||M_{Co}||\text{Hash(}M_{Co}\text{)\}pri}K_{Co}\ \}\ \text{SK2}. \tag{8}$$

4. The AN decrypts and obtains the N, $M_{Co,}$ and the signature. Then, the collaborator's credentials {L_{Co}′′, groupId$_{Co}$} are collected from the Device Database in the blockchain. We explain the operation in Algorithm 3. After initialization in Operation 1, the Operation 2 in Algorithm 3 traverses the attribute list L_{Co} received from the collaborator. For each attribute, we verify whether it belongs to the attribute list L_{Co}′ stored in the blockchain. Moreover, the groupId in

each attribute should equal to groupId$_{\mathrm{Co}}$. Then, in Operation 3 and 4, the validity of N and M$_{\mathrm{co}}$ is verified in the same way it was in Operation 2 and 3 in Algorithm 1. In Operation 6, we combine the attribute list L$_{\mathrm{Re}}$ with L$_{\mathrm{Co}}$ and construct the new list L$_{\mathrm{Re\,+\,Co}}$ based on the data structure of Attribute in Figure 4a. Finally, we use the Satisfy function again and input the root node as well as the new list L$_{\mathrm{Re\,+\,Co}}$ to get the authorization result.

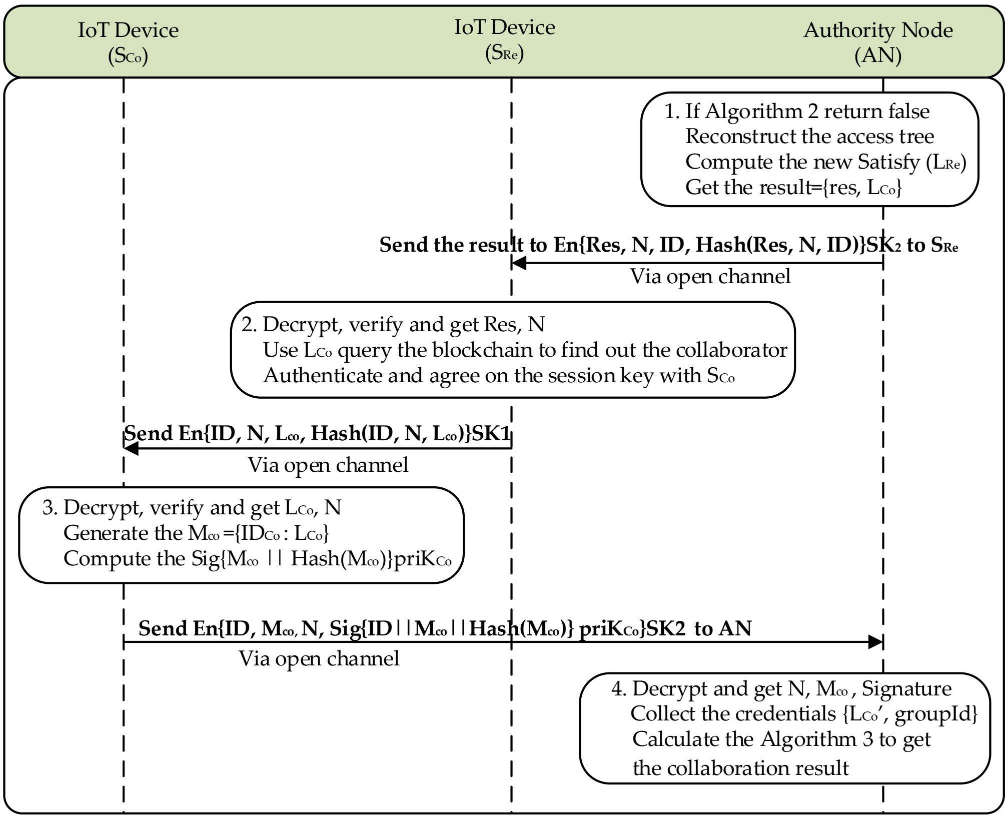

IoT Device (S_{Co})
IoT Device (S_{Re})
Authority Node (AN)
1. If Algorithm 2 return false
Reconstruct the access tree
Compute the new Satisfy (L_{Re})
Get the result={res, L_{Co}}
Send the result to En{Res, N, ID, Hash(Res, N, ID)}SK$_2$ to S_{Re}
Via open channel
2. Decrypt, verify and get Res, N
Use L_{Co} query the blockchain to find out the collaborator
Authenticate and agree on the session key with S_{Co}
Send En{ID, N, L_{co}, Hash(ID, N, L_{co})}SK1
Via open channel
3. Decrypt, verify and get L_{Co}, N
Generate the M_{co} ={ID_{Co} : L_{Co}}
Compute the Sig{M_{co} || Hash(M_{co})}priK_{Co}
Send En{ID, M_{co}, N, Sig{ID||M_{co}||Hash(M_{co})} priK_{Co}}SK2 to AN
Via open channel
4. Decrypt and get N, M_{co}, Signature
Collect the credentials {L_{Co}', groupId}
Calculate the Algorithm 3 to get
the collaboration result

Figure 6. Verifiable and controlled collaboration mechanism.

Algorithm 3 Make the collaboration decision.

```
Input: The received attribute list $L_{Co}$, the attribute list in blockchain $L_{Co}'$, the collaborator's groupId in the
blockchain, the public keys of the collaborator pub$K_{Co}$, the signature Sig, and the random number N
Output: The result of the collaboration CRes
1: Cres = TRUE, res = TRUE
2: for (attribute: $L_{Co}$) do
Calculate res = whether $L_{Co}'$ contains an attribute
Calculate res = res && whether attribute.getGroupId equals to groupId
CRes = CRes && res
end for return CRes
3: Calculate the hash of the attribute map and obtain the result (N || Hash ($M_{Co}$))
4: Utilize the pub$K_{Co}$ to verify the signature
res = Verify(Sig, N || Hash ($M_{Co}$))pub$K_{Co}$
5: CRes = CRes && res
6: Construct the new attribute list $L_{Re+Co}$ using $L_{Re}$, $L_{Co}$
7: Calculate the function Satisfy (root, $L_{Re+Co}$), recursively starting from the root node to the leaf node,
and finally obtain the root. The leaf node returns true if and only if the attribute and the groupId are satisfiable,
as explained in Section 2.1.
8: Cres = CRes && root.isSatisfiable (one certain field of the node)
```

The collaboration mechanism returns the result of the collaboration phase. The result will be recorded in the blokchain and update the Access database accordingly.

5. Security Analysis

After explaining the whole process of our proposed scheme, a security analysis is made in this section. We theoretically analyze how our scheme can efficiently resist the attacks proposed by the thread model in Section 3.6.1 based on the security model in Section 3.6.2. Since the main goal of an attacker is to gain the authorization to access a target, four kinds of attacks that can be efficiently resisted are detailed in this section.

5.1. Collusion Resistant

The access control scheme should be protected from the collusion access request. We record each device's attributes in the blockchain and ensure that these digital credentials are credible. The requester cannot use other devices' attributes (attributes that help satisfy the access tree to obtain real-time access authorization). For example, S_i utilizes S_j's attributes to construct the attributes map $\{ID_{Si}$: $(att\_i)_{i \in Si}$, $(att\_j)_{j \notin S}\}$. The AN will then use ID_{Si} to compute the device's address and query the account in the blockchain to obtain S_i's registered attributes. In this way, collusion can be easily detected.

5.2. Verifiable and Controlled Collaboration

Our access control scheme can also resist malicious collaboration requests. On one hand, collaborative devices should be in the pre-defined group. These devices provide attributes signed by their private keys to help authorization. Thus, devices in other groups are unable to provide a valid signature. On the other hand, during the collaboration phase, the AN will reconstruct the access tree to verify whether the collaboration requester is malicious. We use the example in the introduction section to explain this process. The device {Security Department. Surveillance, Enterprise B} can collaborate with the attributes {Manager, Enterprise A} to gain permission according to the collaboration policy in Figure 1. However, this should be considered malicious behavior from the viewpoint of device security. Therefore, we propose a verifiable mechanism to reconstruct the access tree and remove the collaborative node {Manager} from the tree and modify the threshold value from (2, 3) to (1, 2). Obviously, the device cannot satisfy the new access tree and is not allowed to request collaboration.

5.3. Reply Attack Resistant

Our scheme can defend the relay attack effectively. With the purpose of acquiring authorization, the attacker will try to reply to the signature and attributes obtained in Step 4. However, when preparing the attributes in Step 4, each requester has to use a private key to compute the signature $Sig\{ N||Hash(M_{Re})\}prik_{Re}$. A random number N is unique and generated for each request by the target. Thus, the attacker should calculate $Sig\{ N' \parallel Hash(M_{Re}) \}prik_{Re}$ to pass Algorithm 2, with N′ denoted as the ID of the access sent by the attacker. Due to the fact that the attacker does not have the private key to obtain the required information, the attack is resisted.

5.4. Message Substitution Attack Resistant

Our scheme will not be threatened by the message substitution attack. If a message substitution attack is performed, it is highly possible that that the attacker will intercept the valid message sent by the target in Step 2 and substitute access policy P and the hash result. However, the AN can easily verify the correctness of the policy by querying the chaincode in Algorithm 1 and detect if the altered policy does not belong to the target.

5.5. Supervision and Revocation

The functions of supervision and revocation are also available. For supervision, the AN will send transactions to record the access history in the ledger and update the target's account. In this way, effective actions can be taken instantly to find the attackers and punish them by analyzing this information. Moreover, depriving a user of access rights can be seen as the revocation of attributes. The AN will validate the revocation request and send transactions to update the devices' accounts.

6. Performance Analysis

We implemented a proof-of-concept prototype, conducted an experiment, and evaluated the performance of the proposed scheme. Based on the access policy shown in Figure 4, we chose four cases, which were also utilized by [13], to perform our scheme. Case 1: There is no need to collaborate and the requester's attributes can satisfy the tree. The conditions are the same as those in Figure 4a. Case 2: The target modifies the access policy shown in Figure 4b. Thus, the collaboration node appears in B1. Case 3: The target continues to transform the A1, and the number of collaboration nodes increases to 2. Case 4: The number of the collaboration nodes increases to 3, and these nodes appear in A22, A1, and B1. Following these four different cases, we finished our experiment and obtained the performance results of our proposed scheme.

We first introduce the configuration of our experiments. Then, we evaluate the storage and computation overhead of IoT devices to show that our scheme accommodates IoT scenarios well. Last, the comprehensive analysis of our scheme, including the time cost of our scheme and the performance of the chaincode, is demonstrated to prove the proposed scheme is efficient and scalable.

6.1. Experiment Configuration

The configuration of the experiment is presented in this part. It is necessary to demonstrate how we set up the prototype. We constructed the blockchain network using Hyperledger Fabric v1.1. The network of the prototype is composed of one order node, one CA node, four peer nodes, and one channel. The chaincode that maintains the device accounts was installed and instantiated in all peer nodes. We implemented it on a CentOS 7 virtual machine with 2 GB RAM, which was built from a desktop with an Intel core i7-4510U at 2.80 GHz. In our experiment, we deployed a blockchain client on an authority node using a Java-sdk, and the client connecting the peer node was utilized to query or invoke the chaincode. The chaincode was implemented by Golang and deployed on all peers. The other functions provided by the AN were realized in Java 1.8. According to the system model described in Section 3, the AN should be one of the peer nodes in the blockchain. However, we did not build the AN as a peer node in the blockchain for testing purposes. The AN was implemented on an ASUS laptop with 8 GB RAM and an Intel Core i5-7200U at 2.71 GHz. We used three Raspberry Pi 3B+ units with 1GB RAM and CPU at 1.4GHz to serve as IoT devices, and all their functions were also implemented in Java 1.8. The network configuration of our experiment is presented in Table 1. The devices in our experiment were deployed in the same group and were managed by the same gateway, which has an IP address of 192.168.1.1. Communication across a network segment can be supported by constructing IP tables in the ANs.

In our experiment, various cryptographic algorithms and blockchain operations were performed by ANs and Raspberry Pis. For the evaluation, we used a string of 1082 bytes to build a standard for each algorithm in the experiment. Table 2 presents the time consumptions for each computational task.

6.2. Device Evaluation

The overhead of IoT devices is evaluated in this section. As noted earlier, the majority of the IoT devices are resource-constrained. Thus, our scheme is designed to be light-weight in storage and computation in order to accommodate IoT scenarios.

Table 1. The network configurations of our experiment.

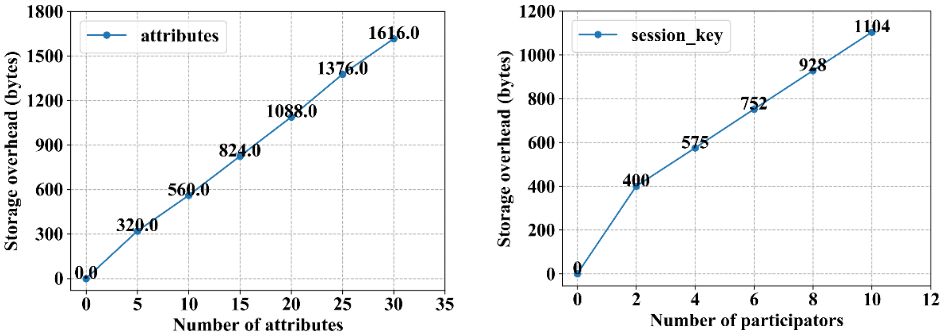
Implement Platform	Role	IP and Port
Laptop	Authority Node	192.168.1.101:8080
Raspberry Pi 3B + 1	Requester	192.168.1.9:8080
Raspberry Pi 3B + 2	Target	192.168.1.10:8080
Raspberry Pi 3B + 3	Collaborator	192.168.1.14:8080
Virtual machine (Consortium blockchain network)	CA	192.168.1.15:7054
	Peer 1–4	192.168.1.15:17053/27053
		192.168.1.15:37053/47053
	Order	192.168.1.15:7050

Table 2. Time consumption for the cryptographic algorithms.

Cryptographic Algorithms	Authority Nodes (ms)	Raspberry Pi (ms)
ECIES Encryption	1.034	7.069
ECIES Decryption	0.618	4.436
ECIES Signature	1.680	6.981
ECIES Verification	2.356	12.224
Hash Algorithm	0.014	0.122
AES Encryption	2.263	11.117
AES Decryption	2.358	11.980
Query Operation	23.087	
Invoke Operation	33.988	

6.2.1. Storage Overhead

We compute the storage overhead of the initial configuration file as well as attributes and session keys required during the authorization process in this part. Each device should store a configuration file that contains an access policy, ID, IP address, groupId, and a pair of ECC keys. A configuration file containing a fixed access policy with 12 nodes is only 1082 bytes. The storage of attributes is presented in Figure 7a. Even if the number of attributes (encoded by UTF-8) reaches 30, the attributes list is only 1616 bytes.



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

Figure 7. Storage overhead for IoT devices: (**a**) storage overhead of the attributes list; (**b**) storage overhead of the session keys.

In our scheme, the target device has to store two AES-128 keys for each access from the requester. When the number of requesters grows, the storage overhead will also increase, as shown in Figure 7b. It takes about 1104 bytes to store session keys when there are 10 requesters. In general, the storage overhead of our scheme for IoT devices is reasonable.

6.2.2. Computation Overhead

The computation overhead of IoT devices is also evaluated in this part. We evaluate the computation overhead in IoT devices by comparing our scheme with that of Ding et al. [21]. Since the work in [21] proved applicable to IoT scenarios and was implemented on the embedded system, the result of the comparison is convincing to demonstrate whether the computation overhead of IoT devices in our scheme is reasonable. Therefore, we translated their scheme into our prototype and only evaluate the computation cost generated by cryptographic algorithms. All the cryptographic algorithms used in these two schemes are denoted with the following symbols: T_{sig} represents the cost to calculate the ECDSA sign, T_{verify} represents the cost to calculate the ECDSA verification, T_{hash} represents the cost to calculate the hash function, T_{aes} t represents the cost to calculate the AES-128 encryption and decryption, T_{aka} represents the cost to perform the AKA process, and T_{query} represents the cost to query the chaincode. T_{aka} in our scheme takes about 77.36 ms, and the other symbols can be found in Table 1.

Thus, it is convenient to calculate each device's computational overhead separately and obtain the total rough time. Table 3 presents a comparison, and we can see that the time costs in [24] is linear to the number of attributes, while the total rough time is (88.12 + 19.2 n) ms, with n indicating the number of attributes. Without collaboration, the computation cost in our proposed scheme takes about 81.53 ms. The scheme is obviously more efficient because we reduce the number of the ECDSA signing and verifying operation to a constant. When collaboration is required, if the number of attributes increases to more than six, the total computation cost in three devices in our scheme will less than the cost in their scheme. In general, the comparison indeed proves that our scheme is efficient and applicable to IoT devices.

Table 3. Comparison of computational costs for IoT devices.

Scheme	Device Cost (Requester)	Device Cost (Target)	Device Cost (Collaborator)	Total Rough Cost (ms)
Ding et al. [21]	n^{1} $T_{sig} + 3T_{aes}$	$T_{hash} + T_{query} + nT_{verify} + 3T_{aes}$		88.12 + 19.2 n
Our scheme (no collaboration)	$2T_{hash} + 2T_{sig} + 3T_{aes}$	$2T_{aes} + T_{verify} + T_{hash}$		81.53
Our scheme (collaboration)	$4T_{hash} + 2T_{sig} + 5T_{aes}$	$2T_{aes} + T_{verify} + T_{hash}$	$2T_{hash} + T_{sig} + 2T_{aes} + T_{aka}$ [2]	210.38

[1] Number of the attributes; [2] Operation time during authentication and key agreement.

6.3. Time Consumption of Our Access Control Scheme

Next, we will evaluate the performance of our proposed access control scheme. The following experiments are composed of four cases, as described in Section 6.1, and the results are shown in Figure 8. As can be seen in Figure 8b, the time cost of the AKA process, in all four cases, remains approximately 310 ms. Moreover, in Figure 8c, the time of the decision-making (Algorithm 2) process and verification (Algorithm 2) process remains steady at 25 and 31 ms, respectively. This is because the number of attributes the requester submits to AN is the same in all four cases. In addition, the time of the co-decision (Algorithm 3) process increases with a change in the collaborative nodes from 1 to 3, caused by the fact that the AN receives more attributes from the collaborator and needs more time to make a decision. Therefore, all the algorithms proposed in our scheme are efficient and valid.

Without collaboration, the authorization time is minimal and reaches only 727.6 ms. The co-addition that records the additional time needed for collaboration increases from 487.5 to 542.2 ms as the number of the collaborative nodes grows. This growth rate is slow, and the time consumption is relatively small. If there are three collaboration nodes, the maximum time to permit authorization is about 1448.1 ms. Relatively, the additional time is only 542.2 ms. In general, this minor delay is worthwhile for collaborative authorization, and our scheme is, therefore, efficient.

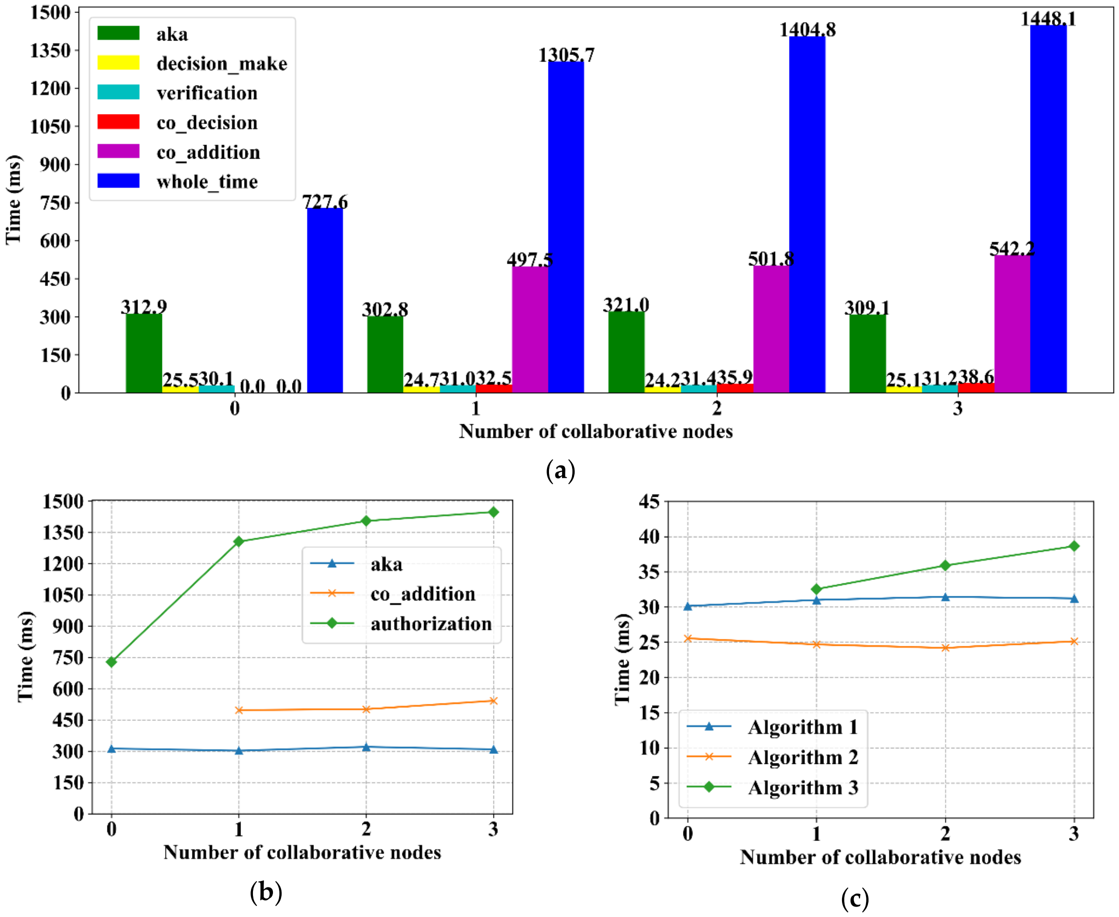


Figure 8. Time cost of our proposed access control scheme: (**a**) an overview of the time cost for our access control scheme; (**b**) the time cost of an authentication and key agreement (AKA), additional time, and the entire access control time; (**c**) the time cost of the three algorithms.

6.4. Performance of the Chaincode

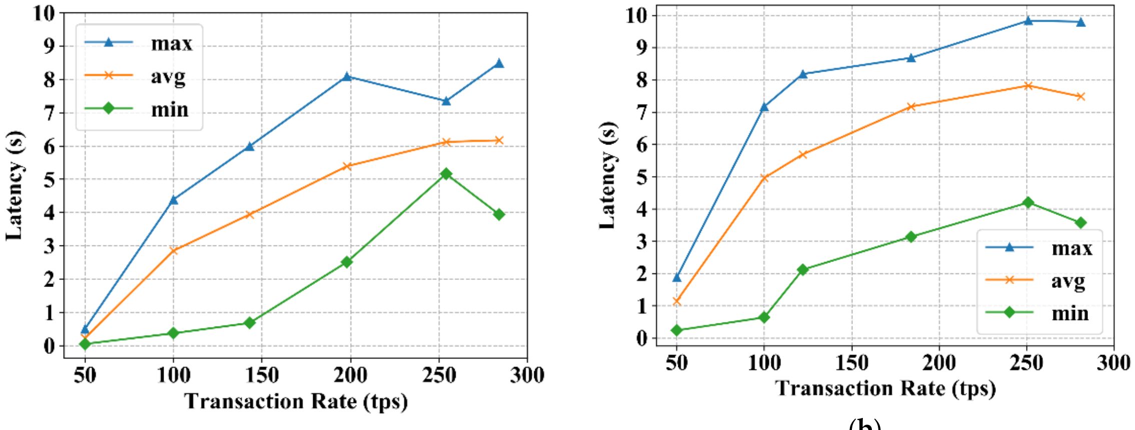
In this section, we evaluate and test our chaincode in a blockchain network, which influences the scalability of our access control scheme. Hyperledger Caliper [36], an open-source benchmark tool that supports measuring the performance of a blockchain network, is utilized in this evaluation. Theoretically, the throughput of Hyperledger fabric can reach 3500 TPS (transactions per second) [18]. However, restricted by our testing environments, we only introduce one order node in our prototype and execute the Solo consensus algorithm. Thus, the throughput of our system only reaches 100 TPS. Once deployed in an industrial environment, the throughput will be largely improved, and the performance of our system will exceed the results we obtained in our prototype.

To test the performance of the chaincode in our experiment, we use variable transaction sending rates, ranging from 50 to 300 TPS, with a fixed block size of 10. The results of the average latency of each transaction and throughput are presented in Table 4. We sent 1000 transactions to test both the query and invoke functions. When the actual send rate reaches 143 and 97 TPS, the throughput is highest, reaching 67 and 62 TPS. From the perspective of the time, the average latency of the query and invoke operations grow gradually, as can be seen in Figure 9. When the send rate increases, the system cannot deal with the transactions, and a delay occurs.

The collaboration process requires four query operations and two invoke operations. Accordingly, we can approximately calculate the number of concurrent requests. If the throughput is 3500, the system is theoretically capable of handling, at most, 583 collaboration access requests per second. This proves that our scheme is scalable and has the potential to meet high concurrent requirements.

Table 4. The chaincode results of our blockchain prototype.

Scheme	Send Rate (tps)	Average Latency (s)	Throughput (tps)
Query User	50	0.22	50
	100	2.85	64
	143	3.94	67
	198	5.39	54
	254	6.12	62
	284	6.17	59
Record History	50	1.15	48
	97	4.96	62
	122	5.69	51
	185	7.17	51
	251	7.82	46
	281	7.48	48



(**a**)

(**b**)

Figure 9. The result of two functions in the chaincode representing latency versus transaction rate: (**a** query the chaincode to get information; (**b**) invoke the chaincode to record information.

7. Conclusions

In this paper, we proposed an attribute-based access control scheme to deal with the problem of unauthorized access, especially for IoT devices. Blockchain technology was utilized to provide credible credentials and transmit the access information trustfully. Furthermore, a verifiable and controlled collaboration mechanism was utilized to detect malicious behaviors and restrict the extra authorization for a certain group. To make our scheme fit IoT devices well, ANs were constructed for computation tasks and to query or invoke the chaincode.

The security analysis shows that our access control scheme can efficiently guarantee authorized access by resisting various attacks and providing a revocation and supervision function. The performance evaluation shows that our scheme is light-weight and appropriate for IoT devices because the storage overhead is acceptable, and the computation overhead was shown to be reasonable after the comparison. Moreover, our proposed access control is efficient and usually only costs 757.6 ms. The extra time generated by the collaboration is only slightly more than 500 ms. This minor delay is worthwhile to satisfy the need for collaborative authorization. The chaincode test further indicates that our scheme is scalable enough to meet high concurrent requirements.

Author Contributions: Conceptualization, Y.Z.; data curation, B.L. (Ben Liu); methodology, Y.Z. and B.L. (Ben Liu); software, Y.Z.; supervision, B.L. (Bing Li); writing—original draft, Y.Z.; writing—review and editing, B.L. (Bing Li), B.L. (Ben Liu), J.W., X.Y., and Y.W. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the national natural science foundation of China (No. 61571116), the basic research (exploration) of science and technology in Shenzhen (JCYJ20170817115538543), and the basic research (layout) of science and technology in Shenzhen (JCYJ20170817115500476).

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

References

1. Sciancalepore, S.; Piro, G.; Caldarola, D.; Boggia, G.; Bianchi, G. OAuth-IoT: An access control framework for the Internet of Things based on open standards. In Proceedings of the 2017 IEEE Symposium on Computers and Communications (ISCC), Heraklion, Crete, Greece, 3–6 July 2017; pp. 676–681.
2. Mahalle, P.N.; Anggorojati, B.; Prasad, N.R.; Prasad, R. Identity authentication and capability based access control (iacac) for the internet of things. *J. Cyber Secur. Mobil.* **2013**, *1*, 309–348.
3. Ouaddah, A.; Mousannif, H.; Elkalam, A.A.; Ouahman, A.A. Access control in the Internet of Things: Big challenges and new opportunities. *Comput. Netw.* **2017**, *112*, 237–262. [CrossRef]
4. Sun, X.; Ansari, N. Dynamic resource caching in the IoT application layer for smart cities. *IEEE Internet Things J.* **2017**, *5*, 606–613. [CrossRef]
5. Novo, O. Blockchain meets IoT: An architecture for scalable access management in IoT. *IEEE Internet Things J.* **2018**, *5*, 1184–1195. [CrossRef]
6. Hu, V.C.; Ferraiolo, D.; Kuhn, R.; Friedman, A.R.; Lang, A.J.; Cogdell, M.M.; Scarfone, K. Guide to attribute based access control (ABAC) definition and considerations (draft). *NIST Spec. Publ.* **2013**, 800. [CrossRef]
7. Yang, Y.; Zheng, X.; Guo, W.; Liu, X.; Chang, V. Privacy-Preserving smart IoT-Based healthcare big data storage and self-Adaptive access control system. *Inf. Sci.* **2019**, *479*, 567–592. [CrossRef]
8. Zhang, Y.; Zheng, D.; Deng, R.H. Security and privacy in smart health: Efficient policy-Hiding attribute-Based access control. *IEEE Internet Things J.* **2018**, *5*, 2130–2145. [CrossRef]
9. Ye, N.; Zhu, Y.; Wang, R.C.; Malekian, R.; Qiao-Min, L. An efficient authentication and access control scheme for perception layer of internet of things. *Appl. Math. Inf. Sci.* **2014**, *8*, 1617. [CrossRef]
10. Sciancalepore, S.; Pilc, M.; Schröder, S.; Bianchi, G.; Boggia, G.; Pawłowski, M.; Weisgrab, H. Attribute-Based access control scheme in federated IoT platforms. In *International Workshop on Interoperability and Open-Source Solutions*; Springer: Cham, Switzerland, 2016; pp. 123–138.
11. Hemdi, M.; Deters, R. Using REST based protocol to enable ABAC within IoT systems. In Proceedings of the 2016 IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), Vancouver, BC, Canada, 13–15 October 2016; pp. 1–7.
12. Castiglione, A.; De Santis, A.; Masucci, B.; Palmieri, F.; Castiglione, A.; Li, J.; Huang, X. Hierarchical and shared access control. *IEEE Trans. Inf. Forensics Secur.* **2015**, *11*, 850–865. [CrossRef]
13. Xue, Y.; Xue, K.; Gai, N.; Hong, J.; Wei, D.S.; Hong, P. An Attribute-Based Controlled Collaborative Access Control Scheme for Public Cloud Storage. *IEEE Trans. Inf. Forensics Secur.* **2019**. [CrossRef]
14. Underwood, S. Blockchain beyond bitcoin. *Commun. ACM* **2016**, *59*, 15–17. [CrossRef]
15. Bethencourt, J.; Sahai, A.; Waters, B. Ciphertext-Policy attribute-Based encryption. In Proceedings of the 2007 IEEE Symposium on Security and Privacy (SP'07), Oakland, CA, USA, 20–23 May 2007; pp. 321–334.
16. Miao, Y.; Ma, J.; Liu, X.; Wei, F.; Liu, Z.; Wang, X.A. m 2-ABKS: Attribute-based multi-keyword search over encrypted personal health records in multi-owner setting. *J. Med. Syst.* **2016**, *40*, 246. [CrossRef] [PubMed]
17. Zheng, Z.; Xie, S.; Dai, H.; Chen, X.; Wang, H. An overview of blockchain technology: Architecture, consensus, and future trends. In Proceedings of the 2017 IEEE International Congress on Big Data (BigData Congress), Honolulu, HI, USA, 25–30 June 2017; pp. 557–564.
18. Androulaki, E.; Barger, A.; Bortnikov, V.; Cachin, C.; Christidis, K.; De Caro, A.; Muralidharan, S. Hyperledger fabric: A distributed operating system for permissioned blockchains. In Proceedings of the Thirteenth EuroSys Conference, Porto, Portugal, 23–26 April 2018; p. 30.
19. Ongaro, D.; Ousterhout, J. In search of an understandable consensus algorithm. In Proceedings of the 2014 {USENIX} Annual Technical Conference ({USENIX}{ATC} 14), Philadelphia, PA, USA, 19–20 June 2014; pp. 305–319.
20. Li, Z.; Kang, J.; Yu, R.; Ye, D.; Deng, Q.; Zhang, Y. Consortium blockchain for secure energy trading in industrial internet of things. *IEEE Trans. Ind. Inform.* **2017**, *14*, 3690–3700. [CrossRef]

21. Ding, S.; Cao, J.; Li, C.; Fan, K.; Li, H. A Novel Attribute-Based Access Control Scheme Using Blockchain for IoT. *IEEE Access* **2019**, *7*, 38431–38441. [CrossRef]
22. Ouaddah, A.; Abou Elkalam, A.; Ait Ouahman, A. FairAccess: A new Blockchain-Based access control framework for the Internet of Things. *Secur. Commun. Netw.* **2016**, *9*, 5943–5964. [CrossRef]
23. Ouaddah, A.; Elkalam, A.A.; Ouahman, A.A. Towards a novel privacy-Preserving access control model based on blockchain technology in IoT. In *Europe and MENA Cooperation Advances in Information and Communication Technologies*; Springer: Cham, Switzerland, 2017; pp. 523–533.
24. Xu, R.; Chen, Y.; Blasch, E.; Chen, G. Blendcac: A smart contract enabled decentralized capability-Based access control mechanism for the iot. *Computers* **2018**, *7*, 39. [CrossRef]
25. Pinno, O.J.A.; Gregio, A.R.A.; De Bona, L.C. Controlchain: Blockchain as a central enabler for access control authorizations in the iot. In Proceedings of the GLOBECOM 2017—2017 IEEE Global Communications Conference, Singapore, 4–8 December 2017; pp. 1–6.
26. Dolev, D.; Yao, A. On the security of public key protocols. *IEEE Trans. Inf. Theory* **1983**, *29*, 198–208. [CrossRef]
27. Hammi, M.T.; Hammi, B.; Bellot, P.; Serhrouchni, A. Bubbles of Trust: A decentralized blockchain-Based authentication system for IoT. *Comput. Secur.* **2018**, *78*, 126–142. [CrossRef]
28. Das, A.K.; Wazid, M.; Yannam, A.R.; Rodrigues, J.J.; Park, Y. Provably Secure ECC-Based Device Access Control and Key Agreement Protocol for IoT Environment. *IEEE Access* **2019**, *7*, 55382–55397. [CrossRef]
29. Evans, D.L.; Bond, P.; Bement, A. *FIPS Pub 140-2: Security Requirements for Cryptographic Modules*; Federal Information Processing Standards Publication: Gaithersburg, MD, USA, 2002.
30. Bong, D.; Philipp, A. Securing the Smart Grid with Hardware Security Modules. In *ISSE 2012 Securing Electronic Business Processes*; Springer Vieweg: Wiesbaden, Germany, 2012; pp. 128–136.
31. Wang, S.; Zhang, Y.; Zhang, Y. A blockchain-Based framework for data sharing with fine-Grained access control in decentralized storage systems. *IEEE Access* **2018**, *6*, 38437–38450. [CrossRef]
32. Martínez, V.G.; Encinas, L.H.; Ávila, C.S. A survey of the elliptic curve integrated encryption scheme. *Ratio* **2010**, *80*, 160–223.
33. Wazid, M.; Das, A.K.; Odelu, V.; Kumar, N.; Conti, M.; Jo, M. Design of secure user authenticated key management protocol for generic IoT networks. *IEEE Internet Things J.* **2017**, *5*, 269–282. [CrossRef]
34. Aman, M.N.; Chua, K.C.; Sikdar, B. Mutual authentication in IoT systems using physical unclonable functions. *IEEE Internet Things J.* **2017**, *4*, 1327–1340. [CrossRef]
35. Salman, O.; Abdallah, S.; Elhajj, I.H.; Chehab, A.; Kayssi, A. Identity-Based authentication scheme for the internet of things. In Proceedings of the 2016 IEEE Symposium on Computers and Communication (ISCC), Messina, Italy, 27–30 June 2016; pp. 1109–1111.
36. Hyperledger Caliper. Available online: https://www.hyperledger.org/projects/caliper (accessed on 15 January 2019).