Peer-to-Peer Networking and Applications (2022) 15:1653–1668
https://doi.org/10.1007/s12083-022-01316-5

Blockchain-based access control for dynamic device management in microgrid

Kai Zhang[1] · Jinhu Yu[1] · Chao Lin[2,3] · Jianting Ning[2,4]

Received: 31 December 2021 / Accepted: 16 March 2022 / Published online: 25 March 2022

Abstract
Microgrid is a self-sufficient grid system that covers one or more kinds of distributed energy, where a variety of terminal devices collect, transmit and store electricity data based on fog-based network infrastructure. Due to security and privacy concerns, efficient and secure access control over terminal devices in microgrid is the primary way to prevent unauthorized access and data breach. Therefore, a number of solutions of device management are proposed. However, they are usually prone to single point of failure, decision-centralized, over-manual intervened. To address the problem, we introduce a blockchain-based fast and dynamic access control (FDAC) system for device management in fog-assisted microgrid. In particular, we adopt an attribute-based access control formula to model a flexible, dynamic and fast fine-grained access control system. FDAC deploys four smart contracts that dynamically manages devices, which includes user authentication, subject/object attributes, access policy, decision-making and credit assessment of user behavior. In addition, FDAC employs a Cuckoo filter to speed up policy search in smart contracts and proposes new credit verification algorithm to improve credit rewards and punishments. To clarify practical performance, we build a private blockchain platform to simulate FDAC. Compared to classic traversal approaches for policy search, FDAC maintains higher accuracy and lower time delay.

Keywords Microgrid · Blockchain · Device management · Access control

1 Introduction

Microgrid is a self-sufficient grid system that relies on various distributed energy sources to generate electricity, such as solar panels, wind turbines and cogeneration. To collect, transmit and store electronic data accurately and efficiently, a number of terminal devices (e.g., smart meter, wireless sensors) are widely deployed in microgrid. In particular, the Internet of Things (IoT) technology [1, 2] enables numerous devices with cost-effectively implementation and connected via distributed network infrastructure. Hence, organizations can access two-way of flows of energy electricity and communication information that provided by intelligent IoT devices [3]. Once these large-scale connected IoT devices suffer malicious intrusion, the reliable running of microgrid will be severely affected.

In December 2015, a cyberattack accident "Ukraine power grid hack" leads to a large area of power failure [4], which severely influenced power supply for roughly 230,000 users. According to a released report by Ukrainian TSN television, the main reason for this accident is that the distributed plant's terminal devices are exposed to unauthorized

✉ Jianting Ning
jtning88@gmail.com

Kai Zhang
kzhang@shiep.edu.cn

Jinhu Yu
Coolfish355@gmail.com

Chao Lin
linchao91@fjnu.edu.cn

1 College of Computer Science and Technology, Shanghai University of Electric Power, 201306 Shanghai, China

2 College of Computer and Cyber Security, Fujian Normal University, 350117 Fuzhou, China

3 Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, Wuhan University, 430072 Wuhan, China

4 State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, 100093 Beijing, China

access and illegal control. The attackers can thus make wrong orders, even power equipment shutdown. Different to classic smart grid, the microgrid composed up of distributed energy include numerous diverse and complex terminal devices. Once these devices are hacked due to unauthorized access, the security and privacy of sensitive information may be leaked to outsiders. Therefore, the primary goal of device management in microgrid is to provide authorized, secure and flexible access control over terminal devices.

Architechture under blockchain-based paradigm To enhance the security guarantee, trust transferring and ownership authenticity, the microgrid has been evolved into under a blockchain-based infrastructure that managed by distributed peer-to-peer network [5], which is shown in Fig 1. Generally, the process of accessing devices and tracking records in a microgrid device management can be well facilitated. In concrete, the blockchain-based microgrid provides a trusted environment and manages a variety of nodes that include terminal device nodes, user nodes and resource computation nodes. Thus, the information of resource and devices are recorded into blocks and later accessed by peer nodes, such as device fingerprint, address, attributes and modes. Since there are massive and complicated terminal devices spread over in the blockchain environment, enabling secure, dynamic and fine-grained access control for device management in microgrid is highly appreciated.

Models of access control There have been many traditional access control models proposed : *discretionary access control* (DAC) [6], *role-based access control* (RBAC) [7], and *capability-based access control* (CapBAC) [8]. In particular, the DAC builds an authorization list for each object where the subjects without identifiers or sufficient resources can not access the object, which only works for simple systems. In RBAC, the entities that described as different roles may access the resources of the principal. As a result, an amount of relationship/connection between access rights and subjects are established [9], but it may increase exponentially due to the growth of entities. In CapBAC, the model distributes corresponding rights for different entities based on their capabilities, such as a transferable and unforgeable authorization token [10, 11]. In practical applications, the access objects cannot be trusted as access verification entities, since they are usually vulnerable to attacks due to lightweight capabilities. To summarize, these models can only provide basic access control characteristics in resource-constrained systems. In addition, they are prone to single point of failure due to microgrid's underlying decentralized and dynamic architecture characteristics.

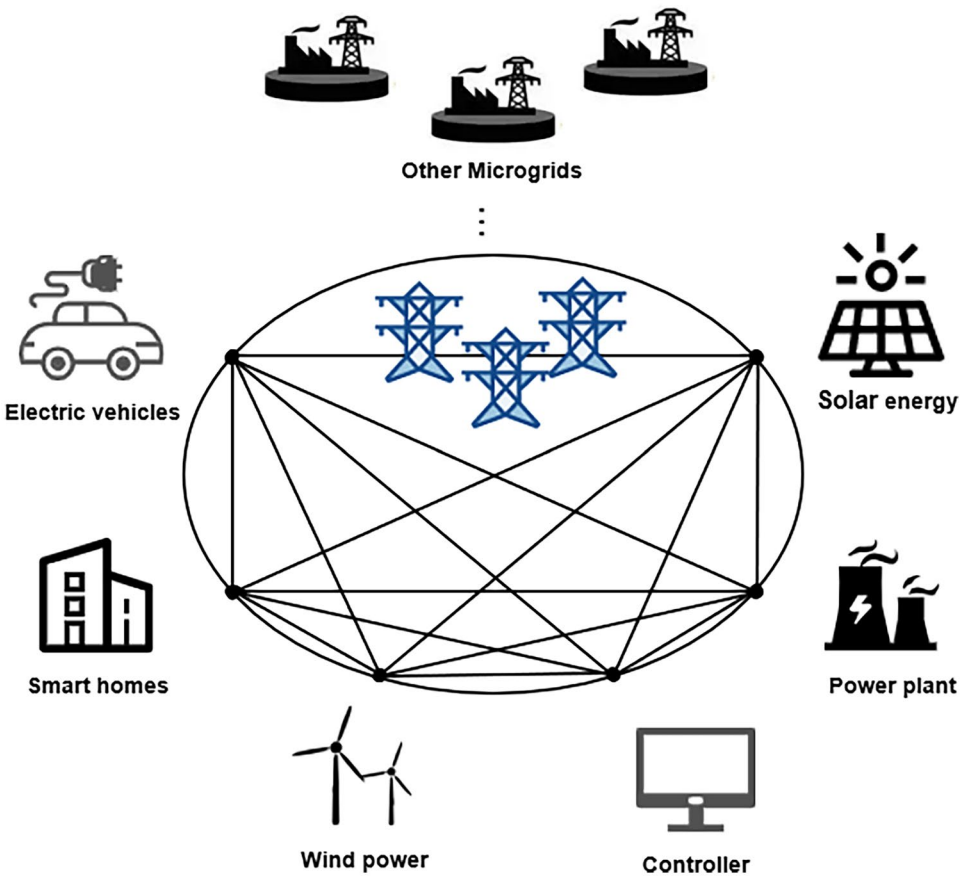

Fig. 1 The Blockchain-based Microgrid

To address the problem, Hu et. al [12] introduced the model of *attribute-based access control* (ABAC). In ABAC, each entity is described by a set of attributes, and the policy is specified by different attributes and a set of rules. Thus, only parties whose attributes satisfy the specified policy can access information [13, 14]. Therefore, the ABAC model has been widely deployed in practical applications that require decentralized access control, which can provide fine-grained access control for device management in microgrid.

Blockchain-based access control To adapt to the blockchain-based network infrastructure, there are a number of blockchain-based access control systems [15–19] introduced. In particular, the work [15–17] employed the distributed and immutable blockchain paradigm to store access control policies, but its computation capability is unfortunately limited. Recently, Zhang et.al, [18] implements smart contract-based access control in the Internet of Things, which provide efficient and secure trustworthy managements over devices. Later, [19] extended [18] to further consider flexible authorization and authentication of users, and presented a novel blockchain based access control system that achieves privacy, efficiency, decentralization and scalability of IoT network services. Nevertheless, it has not formally considered a good tradeoff between effectiveness, credibility and authentication of the system. To detect illegal access behavior (such as frequent access), [20] proposed an access control system that can efficiently check user's behavior and punish the access user but fail to assess user's credit. Very recently, Zhang et.al, [21] introduced a dynamic attribute-based access control framework based on well-designed smart contracts, which only focus on managing the processing time period. However, the state-of-the-art solutions of blockchain-based access control have not formally considered credit assessment for behaviors of nodes and efficiently searching for access control policies.

1.1 Our results

To enable such a dynamic, efficient and secure access control framework for IoT devices, we propose FDAC, a fast and dynamic access control system for device management in fog-assisted microgrid. Generally, we revisit blockchain-based ABAC model with introducing credit assessment methodology and fast policy searching function. In particular, we propose new smart contracts to automatically give a fully dynamic managements over terminal devices. In addition, we employ Cuckoo filter to speed up the policy retrieval process that determines whether one entity's operations match a specified policy. Concretely, the main characteristics of our FDAC can be summarized as follows:

1. Achieving secure, scalable and fine-grained attribute-based access control for device management, in which only authenticated and authorized user nodes can access information of devices. Based on an underlying blockchain-based framework, FDAC has well captured the risk of single point access control and centralized policy decision-making.
2. Deploying dynamic and efficient smart contracts that realizes user authorization and authentication, attribute management and access decision-making. By employing Cuckoo filter to fast search target policies, the running efficiency of algorithm-functions in FDAC are obviously improved.
3. Configuring credit assessment to monitor malicious behavior of user nodes, where the decision-making is dynamically adapted to the sensitivity of resources. With introducing new credit assessment algorithms, the historical transaction records about access process between user nodes and terminal nodes are stored in edge nodes.

To illustrate practical utility of our FDAC, we build an Ethereum blockchain-based platform, and deploy corresponding nodes distribution and smart contracts.

The results show that FDAC can automatically execute access decision and malicious behavior monitoring. In addition, the policy searching in smart contracts is much faster than classic traversal approaches, where the accuracy of the policy search maintains 85% above and the delay of a single policy search is almost 0.2 millisecond. As our FDAC considers effective and efficient credit assessment and policy search, the cost of natural gas consumed by the deployment of smart contracts and the execution of the corresponding ABI are slightly increased by 15%. Although the performance has increased by nearly 30%, which is highly acceptable in practical applications.

Comparison Table 1 shows a general analysis between state-of-the-art solutions and our FDAC, which includes feature and cost comparison. As can be seen, the work and FDAC support tamper-proof [15, 18, 20–22] and privacy preservation [15, 20, 22]. Although several smart contracts are all deployed in current solutions, our FDAC provides a fully dynamic managements of user nodes. In addition, [21, 22] and FDAC adopt the ABAC model to give a more fine-grained access control for policy managements. Compared with [18, 20–22], FDAC introduces a credit assessment algorithm to measure the credit rewards and punishments of user nodes. Furthermore, our FDAC achieves a constant cost of policy search, while that of other solutions increases with the number of items in a specified policy. In particular, the policy searching cost in state-of-the-art solutions [15, 18, 20–22] is linear with the policy scale (i.e., $\mathcal{O}(n)$); while our FDAC system achieves a constant policy searching cost due to the adoption of Cuckoo filter in the smart contract PMC.

Table 1 Comparison between State-of-the-art Work and Our FDAC

	[15]	[18]	[20]	[21]	[22]	Ours
Tamper-Proof	✓	✓	✓	✓	✓	✓
Privacy Preservation	✓	×	✓	×	✓	✓
Usage of Smart Contract	×	✓	✓	✓	✓	✓
Attribute-Based	×	×	×	✓	✓	✓
Credit Assessment	×	×	×	×	×	✓
Policy Searching Cost	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$

1.2 Related Work

Blockchain-based access control models Ouaddah et al. [23] presented fairaccess, a blockchain-based authorization scheme where smart contracts are used to perform access control policies to exchange access tokens, but the assumed number of transactions is limited. And [24] adopted smart contracts to achieve access control for only data records management. Different from [24], Zhang et al. [18] utilized Ethereum smart contracts to store ACLs, and thus proposed an ACL-based access control framework. However, the system usually brings about large monetary cost in deploying the contract since each contract is only designed for each subject-object pair. Later, Dorri et al. [15] considered the access control problem in IoT and conducted a case study with smart home. In the scheme, each block uses a policy header to store the access control policy to deal with all access requests that interacted with home devices, which the critical proof-of-work process in blockchain technology is not formally considered. Compared to local private chain, a novel control chain [25] is proposed for providing user transparent, user-friendly, completely decentralized and fault-tolerant. Nevertheless, there are four different blockchains to perform access control, which are cost-expensively maintained with. To consider distributed access control in cloud services [26], a blockchain-based decentralized runtime access monitoring system (DRAMS) is introduced for federated cloud cooperation. As a result, it provides data privacy and data secure sharing. In recent years, Novo et al. [27] proposed an authorization scheme for managing IoT devices based on distributed blockchain. However, the two solutions are less practical since the assumed IoT devices usually suffer from limited computing power. To effectively protect large-scale IoT devices, Xu et al. [28] introduced a decentralized and joint capability-based delegation model (FCDM). However, the large storage cost leads to a high latency to access requests.

ABAC-based access control Qashlan et al. [22] and Zhang et al. [21] considered attribute-based access control and smart contracts for practical IoT applications. In particular, [22] proposed an authentication scheme for secure IoT device management in fog-assisted smart home. Nevertheless, it cannot effectively monitor illegal behaviors of user nodes, which limits its practical deployments in privacy-enhanced environments. Zhang et al. [21] presented some smart contracts for access control framework for smart city, which achieves functionalities of managing the attributes of the subject and object, adding and deleting access policies, and decision-making. However, the designed access control only works for dynamic time management, while not formally considering dynamic managements over terminal devices. In addition, [29] introduced an attribute-based access control model for blockchain in the open IoT environment, where key attribute information is stored on the blockchain. The system judges the access request through the attribute-based access control method in the smart contract, and makes access control judgment. However, the analysis of safety and operating costs is lacking in the scheme. Later, Zhang et al. [30] formulated an effective attribute-based collaborative access control scheme to realize controlled access authorization in IoT applications. Although the scheme guarantees the security of authorized access, the data integrity and privacy threats of access control have not formally considered. Recently, Rouhani et al. [31] proposed an attribute-based access control system in a blockchain to provide trusted auditing of access attempts. Additionally, the system provided a degree of transparency that benefited both access requesters and resource owners, but the user authentication is not integrated into the user authorization phase.

Organization We review some background knowledge in Sect. 2 and describe the problem formulation in Sect. 3. Section 4 presents our FDAC system and Sect. 5 gives its function and security analysis. We implement the system and show its performance in Sect. 6 and finally concludes the work in Sect. 7.

2 Background knowledge

In this section, we review some background knowledge that includes blockchain technology, smart contract, Cuckoo filter and attribute-based signature.

2.1 Blockchain

Blockchain [32] is a distributed ledger that allows data to be recorded, stored and updated in a distributed manner. In blockchain, the transaction is the most fundamental activity that created, recorded and approved in the block by the miner. Based on a consensus algorithm, the miner who wins bookkeeping rights sends its created block to the system to each peer node. Other nodes later verify the hash value, signature and transaction validity of the block, and finally join in the local after a consensus. Therefore, blockchain is considered to be a promising architecture that guarantees security of distributed transactions for all participants in a public peer-to-peer environment.

2.2 Smart contract

Smart contract [33] is a special account with associated codes (i.e. functions) and data (i.e. states) in blockchain. Generally, it is compiled into the bytecode of the specific binary format of blockchain platform (i.e., Ethereum), and the account is deployed into the blockchain. Smart contract provides a number of functions or interactive application binary interfaces (ABIs). These ABIs are usually executed through transactions between accounts or the transmission of messages between contracts. Moreover, they can also be executed by simply calling functions without sending transactions and messages.

2.3 Cuckoo filter

Cuckoo Filter [34] is a variant of Cuckoo hash table, which supports dynamic addition and deletion of items. For each item inserted, only its fingerprint (bit string generated by the hash function) is stored instead of a key-value pair. A Cuckoo hash table consists of an array of buckets, and each insertion item is two candidate buckets determined by hash functions. Among that, one bucket records the item and the other bucket backups the item. When to construct a Cuckoo filter, its fingerprint size is determined by the target false positive rate decision. Note that a smaller false positive rate requires a longer fingerprint to reject more false positive queries.

2.4 Attribute-based signature

The concept of attribute-based signature (ABS) was first proposed by [35], which allowed a user to sign messages with any predicate with its attributes that issued from an authority. The signed message only reveals that the specified policy but preserves user's identity information privacy. Therefore, ABS not only preserves privacy information anonymity for users, but also provides fine-grained access control over the users.

- ABS.Setup: The setup algorithm inputs a security parameter λ, and outputs a system public parameter PP and a master key MK.
- ABS.KeyGen: The key generation algorithm inputs PP, MK and a user's attribute set Γ, and finally outputs a private key SK_Γ for the user.
- ABS.Sign: The signature algorithm input PP, a message M, a user's private key SK_Γ and a predicate Λ that accepts Γ, and outputs a signature δ of M.
- ABS.Verify: The verification algorithm inputs PP and a signature δ along with an attribute set Γ, and finally outputs 1 if δ is valid. Otherwise, it outputs 0.

3 Problem formulation

In this section, we formalize the system model, design goals and threat model of our blockchain-based FDAC for device management in microgrid.

3.1 System model

Our blockchain-based FDAC system consists of the following four entities: *Device nodes*, *User nodes*, *Edge nodes* and *Cloud*, which is formally described as in Fig. 2.

- **Device nodes:** The terminal devices include a variety of power devices, such as smart meters, wireless sensors, and communication boxes. These devices are used to collect and store electricity consumption information, instrument temperature, and network data.
- **User nodes:** The semi-honest users include different types of lightweight users, such as clients and organizations. Based on blockchain-based FDAC system, they access device information and resource information.
- **Edge nodes:** The edge node records incoming and outgoing transactions of device running operations, which usually considered as miners in the blockchain infrastructure. Note that its capability is stronger than terminal devices but weaker than cloud server.
- **Cloud:** It provides highly-efficient, cost-effective and secure storage services, where its computation and storage resources can be flexibly configured as a number of nodes on the blockchain.

3.2 System running flow

The running flow of the system is formalized as follows:

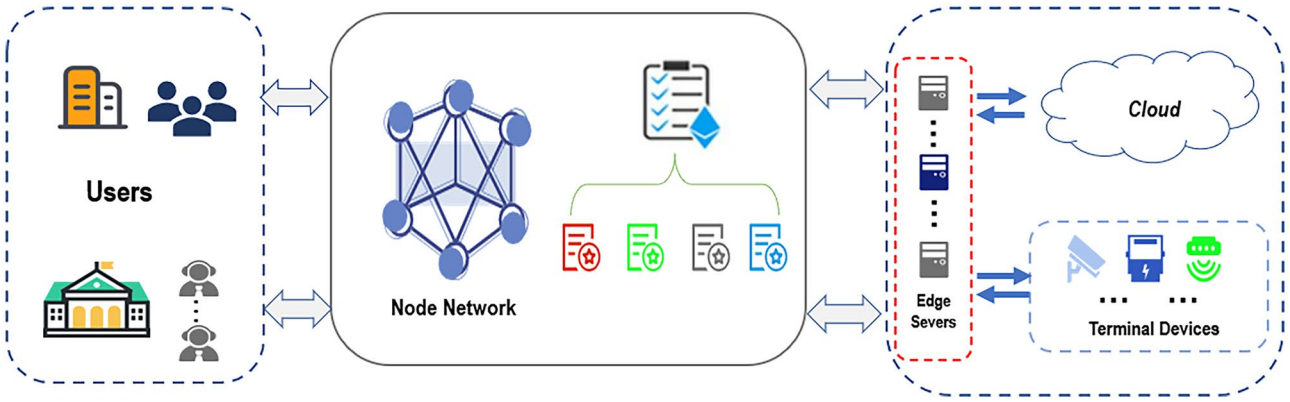

Fig. 2 The system model of FDAC system in fog-assisted microgrid

1. First, a user node is authenticated by the attribute-based signature(ABS) algorithm in AMC. If it is authenticated, the AMC returns a unique identification number *ID* and blockchain address *addr* to the user node.
2. Later, a user sends an access request to the smart contract ACC, for accessing device information of terminal devices that stored in the cloud. If determined as a legal request by ACC, the ACC returns access authorization or access block to the user.
3. Then, the ACC sends a user's misbehavior report *MR* to a smart contract CC, where CC assess the credit of user behavior in *MR* and returns a credit value cv_i to ACC. Based on different levels of credit, ACC specifies corresponding punishments.
4. Finally, receiving a user's access information records, the edge nodes package them as transactions into blocks to get system rewards.

3.3 Design goals

The design goals of our FDAC system are as follows:

- **Fine-grained access control.** The access rights of each user is conducted as a subject-object pair, where the devices and users are all described by a set of attributes. Note that only users whose attributes satisfy the specified policy can access the information.
- **Automatically-running.** Following the inherent policy of smart contracts, the system can run automatically without manual intervention and provide fairness for all peer nodes.
- **Privacy-Preserving.** The signer's private information is hidden from verifies and the public when it registers in the system. That is, the signed request only reveals that its attributes, while not leaking its identity information.
- **Constant policy searching cost.** The cost of policy searching in smart contracts is constant, where the query time overhead and storage overhead remains highly efficient.

3.4 Threat model

- **Collusion Attack.** The terminal device nodes may collude together to access unauthorized information and lead to the disclosure of attribute information. That is, the colluded attackers may try to maliciously collect attributes and satisfy a specified access policy.
- **Replay Attack.** This attack indicates that the attacker may intercept the access request that sent by the user and replay the access request to gain access of device resources.
- **Modification attack.** This attack indicates that an attacker may try to modify or delete stored data of a specific user or device. The attacker compromise the local storage and learns confidential data information to launch such attack.
- **Masquerade Attack.** A masquerading attack is performed by an adversary to gain unauthorized access to the system. This attack usually includes stealing passwords, snooping login names and finding loopholes of the system.

4 The FDAC scheme

In this section, we formally present FDAC, a fast and dynamic access control for blockchain-based device management system in fog-assisted microgrid.

4.1 Blockchain structure

The nodes in the blockchain-based FDAC system includes the following four types of nodes:

- **Light nodes:** The light nodes consist of access user nodes and device nodes, where they only keep transactions information that related to themselves.
- **Admin nodes:** The admin nodes manage the attributes of the subject and object, and audit the data in the blockchain that related to user nodes.
- **Edge nodes:** The edge nodes, usually acted as miners, record incoming and outgoing transactions of device running operations. Note that its capability is stronger than terminal devices but weaker than cloud server.
- **Cloud node:** The cloud node provide a secure, flexible and user-configured data storage services for the system, which manages the records of user information and data collected by terminal devices.

4.2 Running flow

There are four phases included in our FDAC system, that is, *initialization, access control management, credit assessment,* and *access transaction consensus*. In particular, Fig. 3 shows concrete running flow of our FDAC.

In addition, we give a more careful explanation on the interactive process between each smart contract that deployed used in our FDAC system. In particular, there are the following four smart contracts (as depicted in Fig. 4) that are introduced for FDAC: Attribute Management Contract (AMC), Policy Management Contract (PMC), Access Control Contract (ACC) and Credit Contract (CC). In general, the AMC manages the attributes of subjects and objects, and the PMC requests that required set of attributes from the AMC to generate corresponding policies. When ACC determines whether the user's access policy is legal, it should match the policy that existed in the PMC. If ACC meets illegal access, it may send a misbehavior report to the CC, and CC calculates and returns the user's credit value to the ACC. Finally, the system formulates the corresponding punishment according to the credit value.

4.2.1 Initialization

The attribute-based signature (ABS) is deployed in the smart contract AMC to assist users to perform user authentication (signature verification) during the system initialization phase. Generally, an ABS consists of four algorithms ABS.Setup, ABS.KeyGen, ABS.Sign and ABS.Verify. In particular, the admin node first generates the system parameters *PP* and a master key *MK* via running ABS.Setup algorithm. Later, it computes a private key SK_Γ based on a user node's attribute set Γ via running ABS.KeyGen algorithm. Then, the user node calls ABS.Sign algorithm and outputs a signature δ of its message *M*. Finally, AMC verifies δ by calling ABS.Verify algorithm. As a result, the user node is successfully registered in the system.

1. **Registration.** A new user completes the registration and authentication based on the involving ABS algorithm in AMC. After that, the system grants a corresponding *ID* and a unique blockchain account address *addr* for a user node, where the user's *ID* associates with an attribute

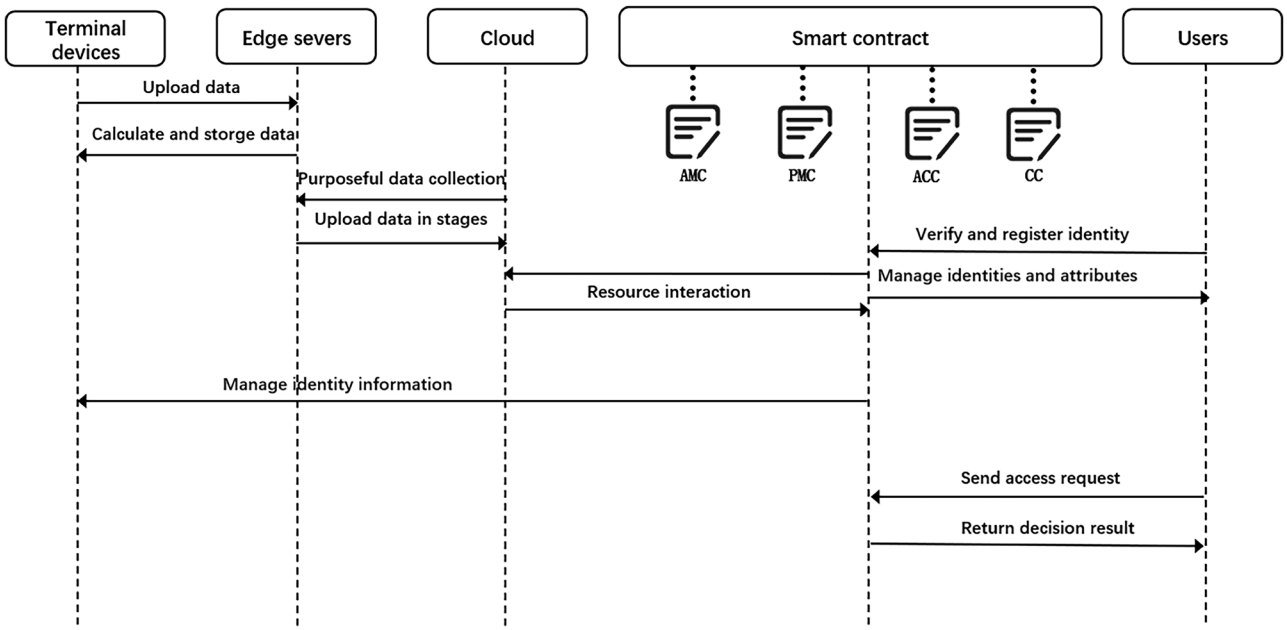

Fig. 3 The running flow of the blockchain-based FDAC system

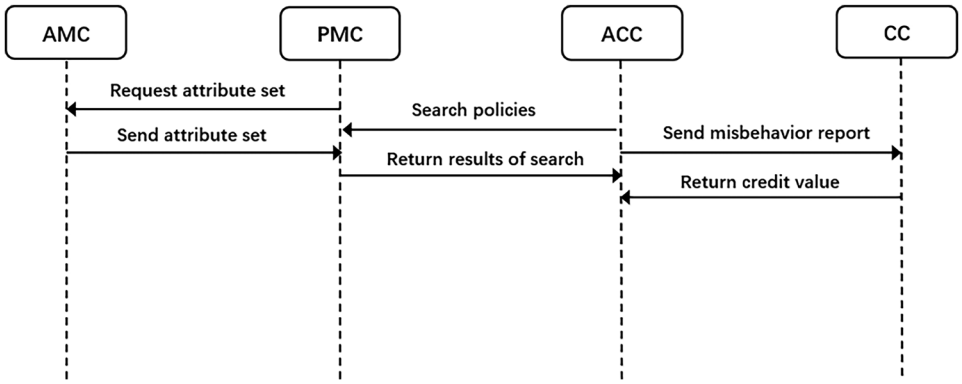

Fig. 4 Interactive process between smart contract

set that implies the subject of *ID* is equipped with these attributes. Hence, a registered user can verify its identity with submitting own *ID*, *addr* and the validation of identity algorithm (as shown in Algorithm 1). Once a user logs out of the system with canceling its *ID*, as a result, its corresponding address is also cleared from the blockchain network (but the cancellation record is maintained). In particular, the ABIs of registration in AMC includes:

- idValidate: verifies whether a user is the member of the system.
- idLogout: a user logs out of the system.

Algorithm 1 Validation of identity

Require: a user's unique ID and its corresponding blockchain address $addr$
Ensure: result of validation
1: Select($entity \leftarrow$ type of user) **then** $Info \leftarrow$ entity[name, unique ID, blockchain address $addr$]
2: **if** "$info$==True" **then**
3: validation succeeded
4: **else if** "$info$==False" **then**
5: validation failed
6: **end if**
7: retrun reslut of validation

2. **Attribute Management.** In AMC, the administrator can execute different ABIs to add, update, and delete attributes of subjects and objects. Generally, the subject denotes the accessing user, and the object represents the resources that provided by terminal devices. The ABIs of attribute management in AMC includes:

- subjectAdd: adds the attributes of the subject (user nodes).
- objectAdd: add the attributes of the object (device nodes).
- subjectUpdate: update the attributes of the subject.
- objectUpdate: update the attributes of the object.
- subjectDelete: delete the attributes of the subject.
- objectDelete: delete the attributes of the object.

4.2.2 Access control management

The access control management (ACC) consists of policy management, policy dynamic search, and access control execution. In particular, the policy administrator calls corresponding ABIs to add, update and delete policies in the literature. Based on the hash index structure of Cuckoo filter, the policy search achieves a dynamic and real-time policy retrieval. The access control execution is used to control the user's access request to the resources. In addition, ACC determines whether the access request satisfies the specified policies. If it is a legal request, ACC sends *"access authorized"* and returns the corresponding resource information to the users. Otherwise, it sends *"request is blocked"* and rejects the user's access request.

1. **Policy Management.** In the system, an access policy is defined as a logical combination of subject attributes, object attributes, operations, and access time. In particular, Table 2 gives an example explanation. That is, the subject attribute consists of user name, user role, and unique user blockchain address; the object attrib-

Table 2 An example of an employed ABAC policy in our FDAC system

Subject Attributes	Object Attributes	Action	Time
Name:"Alice" Role:"home user" addr:"0x19c..." ...	Name:"Smartmeter" Number:"002" addr:"0x48b..." ...	Read Write Execute...	startTime:23456 endTime:23800

ute includes terminal device name, serial number, and device blockchain address. Note that if the user name or device name is consistent, we can use other attributes to distinguish individual users or devices. The action of the subject on the object can be divided into read, write and execute; the time includes start time and end time, which is used to control the of time for the user to access the device. Moreover, we remark that the policy administrator can give a dynamic policy management, such as adding, updating, and deleting policies. For example, the policy administrator may add and configure a new policy when some resources are newly deployed.

- policyAdd: adds the new access policy.
- policyUpdate: updates the access policy.
- policyDelete: removes an inapplicable policy.

Only there is no conflict between existing policies and newly added policies, the new policies can be successfully included in the system. Similarly, a policy conflict verification is also managed for the condition of updating and deleting policies.

2. **Dynamic Policies Search.** To improve the efficiency of policy search and reduce time consumption, our FDAC introduces a dynamic retrieval methodology based on Cuckoo filter that supports both insertion and deletion policy functions. When a system administrator inserts, searches, or deletes an access policy, the corresponding ABIs in the PMC may invoke the Cuckoo filter to complete corresponding operations. In the Cuckoo filter, the process of policy insertion, search and deletion proceeds as follows.

   - **Policy Insertion.** When inserting an item x in a policy, we use two hash functions to calculate the indexes of the two candidate buckets, i.e., $h_1(x) = hash(x), h_2(x) = h_1(x) \oplus \text{hash}\left(x's \text{ fingerprint}\right)$. If there is a conflict between the fingerprint information and a previous one, the previously inserted policy item is moved to the spare bucket. If there is no room in both buckets, a candidate bucket may be chosen to kick out the existing item and re-insert the kicked item into its spare position. Note that this process may be repeated until an empty bucket is found. If there is no empty bucket, it implies that the hash table cannot be inserted with any new item. When the number of insertions and the number of buckets are sufficient, the probability of insertion failure can certainly be reduced. Figure 5 shows the process of inserting an item in a policy.
   - **Policy Search.** The search process of the Cuckoo filter is shown in Algorithm 2. The algorithm first calculates the fingerprint of policy item x and two candidate buckets. If any of the existing fingerprints in the two buckets match, the cuckoo filter returns true, otherwise the filter returns false.

Algorithm 2 Search an item x of policy

Require: items x
Ensure: true or false
1: **while** Perform search x operation **do**
2: $f = \text{fingerprint}(x)$;
3: $i = \text{hash}(x)$;
4: $j = i \oplus hash(f)$;
5: **if** if f exists in bucket$[i]$ or bucket$[j]$ **then**
6: return true;
7: **else**
8: return false;
9: **end if**
10: **end while**

   - **Policy Deletion.** When deleting an item, the system automatically checks whether a given policy item exists in two candidate buckets. If the fingerprints of x exists in any bucket, the system deletes a copy of the matching fingerprint from the bucket, as shown in Algorithm 3. To delete a policy securely and accurately, the policy item should have been inserted before. Otherwise, deleting a non-existent policy item may result in unintentionally deleting a different policy item that shares the same fingerprint.

Algorithm 3 Delete an item x of policy

Require: items x
Ensure: true or false
1: **while** Perform delete x operation **do**
2: $f = \text{fingerprint}(x)$;
3: $i = \text{hash}(x)$;
4: $j = i \oplus hash(f)$;
5: **if** if f exist in bucket$[i]$ or bucket$[j]$ **then**
6: remove a copy of f from this bucket;
7: return true;
8: **else**
9: return false;
10: **end if**
11: **end while**

3. **Access Control.** The main function of ACC is to perform access control and make decisions on access requests, where the access control algorithm is shown in Algorithm 4. Receiving a request (such as user *ID*, resource *ID*, and action), the accessControl ABI checks if access request is leagl. Access request includes legal access and illegal access, which are monitored by ACC. Legal access refers to the normal access of users according to the access policy, while illegal access refers to the malicious behavior of users in the access, mainly including access prohibition of resource information and frequent access. To allow or deny an access request, ACC finally outputs the access decision to the access user node. Note that if malicious access behavior of user is detected,

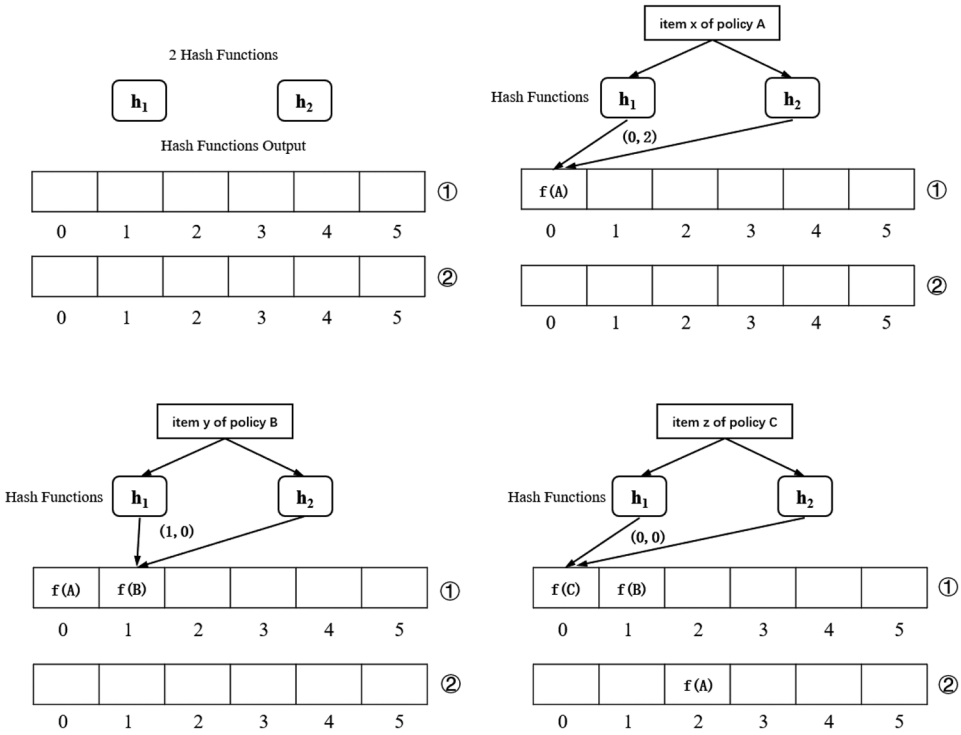

Fig. 5 Access control policy insertion based on Cuckoo Filter

ACC sends a misbehavior report(*MR*) to CC. Then, the user will be punished at different levels according to the credit value cv_i returned by CC. The *MR* mainly contains the type of misbehavior, number of misbehavior, time of occurrence and historical record of access behavior.

Algorithm 4 Access Control

Require: subject, object, resource, action
Ensure: access result
1: **if** access request is from the subject **then**
2: sa $\leftarrow getma(subject) + getsoa(subject, object)$
3: oa $\leftarrow getoa(object)$
4: p $\leftarrow getpolicy(object, resource, action)$
5: $r[\,] \leftarrow getrule(subject, object, resource, action)$
6: **while** each $r[i]$ in r **do**
7: **if** sa,oa and ma satisfies $r[i].sa$,$r[i].oa$,$r[i].ma$ respectively **then**
8: $result[i]$=$r[i].result$ //Policy attribute matching
9: **else**
10: $result[i]$=NotMach
11: **end if**
12: **end while**
13: **while** p.*algorithm*=deny overrides **do**
14: **if** there is a $result[i]$=deny **then**
15: access result=deny //Access denied
16: **end if**
17: **end while**
18: **while** p.*algorithm*=allow overrides **do**
19: **if** there is a $result[i]$ whose *result* is allow **then**
20: access result=allow //Access permission
21: **end if**
22: **end while**
23: **else if** misbehavior is detected **then**
24: Send MR to CC
25: **end if**
26: **return** access result

- accessControl: Make a decision on access request according to the policies and rules defined in the contract.
- misbehaviorCheck: Check for misbehavior and write relevant records into *MR* report.

4.2.3 Credit assessment

There are several inherent malicious behaviors existed in the implementation of access control, such as frequent requests for a short period of time, and access to prohibited resources. However, these malicious behaviors unfortunately generate excessive blockchain transactions. Thus, this may reduce the probability of legal transactions that collected by the blockchain, or prolong the confirmation time. To address it, when CC receives *MR* from ACC, it calculates the user's credibility based on the *MR* which involves the types of misbehavior, number of misbehavior and history of access behavior. The credit assessment algorithm is shown in Algorithm 5 .

1. **Credit Calculate Scheme.** According to the behavior of each user node labeled with i, its credit value cv_i is defined by

$$\mathsf{cv}_i = \lambda_1 \mathsf{cv}_i^P - \lambda_2 \mathsf{cv}_i^N.$$

In particular, cv_i^P denotes the positive impact factor where the access nodes have always followed the specified access policies. And cv_i^N represents the nega-

tive impact factor where the access node has malicious behavior, which consists of accessing prohibited information or sending access requests in a short period of time. In addition, λ_1 and λ_N respectively represents a corresponding weight that can be dynamically updated by the system as

$$\lambda_2 = \lambda_N \cdot \frac{\beta(t_i)}{\sum_{i=1}^{h} \beta(t_i)},$$

where $\beta(t_i) = 1/\gamma^{t_i}$. Note that λ_2, λ_N and γ are preset values, t_i represents the total time of the access node, and $\beta(t_i)$ represents the time attenuation factor. As t_i increases, the time decay factor $\beta(t_i)$ decreases, where weight λ_2 also decreases. That is, the influence of malicious behavior on the node gradually decreases, where the node may thus be punished. The positive influence function cv_i^P of reputation value is positively correlated with the number of legal behaviors (i.e. access request), which is defined as

$$\mathsf{cv}_i^P = \min\left(\mathsf{cv}_i^P \max, (T_i - I_l)\omega\right).$$

Note that cv_i^P max is the upper bound of cv_i^P, which is a system-defined threshold (preliminarily set to 0.95) to prevent the infinite increase of cv_i^P. And ω is the weight of legal behavior, T_i is the total number of legal behaviors of user node i and I_l is the last legal behavior index of the last punishment. For the negative influence function, m_i is the total number of malicious behaviors of node i, k represents the total number of malicious behaviors of each type, $\alpha(\eta)$ denotes the penalty coefficient of malicious behavior, which ranges from 1 to 10. Moreover, the value can be adjusted according to the requirements of sensitivity to malicious behavior, which is defined as

$$\mathsf{cv}_i^N = \sum_{k=1}^{m_i} \alpha(\eta) \cdot \frac{1}{m_i - k},$$

where

$$\alpha(\eta) = \begin{cases} \alpha_l & \text{if prohibited information accessed} \\ \alpha_d & \text{if frequent requests sent} \end{cases}$$

Generally, the user's credit value falls into between 0 and 1, and the punishment varies with the size of the credit value.

2. **Credit Contract.** When receiving a misbehavior report *MR* from ACC, CC reads the information in *MR* and gives the required value to creditCalculate ABI, then calls it to carry out the credit calculation according to the credit calculation scheme in the previous section, and finally returns the credit value to ACC.

- creditCalculate: Calculate credit value of user.

Algorithm 5 Credit Assessment

Require: positive and negative impact value
Ensure: credit value
1: **while** CC receives ACC misbehavior report MR **do**
2: credit assessment
3: **if** behavior is positive **then**
4: $T_i \leftarrow$ the total number of legal behaviors
5: $I_l \leftarrow$ the last legal behavior index
6: calculate cv_i^P *//positive impact value*
7: **else** behavior is negative
8: $m_i \leftarrow$ total number of misbehavior
9: $k \leftarrow$ the total number of misbehavior of a certain type
10: calculate cv_i^N *//negative impact value*
11: **end if**
12: **end while**
13: System distribute some weight coefficients and calculate $\lambda_1 \mathsf{cv}_i^P + \lambda_2 \mathsf{cv}_i^N$
14: **return** cv_i *//credit value of user node i*

4.3 Access transaction consensus

FDAC employs the proof of work (POW) algorithm to model the access transaction consensus. In particular, an edge node retrieves a number of transactions in the smart contracts and tries to gain a block reward for packaging transactions. Later, the miner broadcasts its solution to the blockchain for letting other peer node to reach a consensus.

5 Function and security analysis

In this section, we give a function and security analysis of FDAC respectively according to the design goals and threat model in Sect. 3.

5.1 Function analysis

- **Fine-grained access control.** In FDAC, the user nodes are described by a set of attributes, and the specified authorization consists of a variety of different attributes and thresholds. As a result, the subject and the object of our FDAC system associate with attributes set, which provides an inherent fine-grained attribute-based access control over device management.
- **Automatically-running.** In FDAC, there are several smart contracts introduced to manage system running. Our blockchain-based FDAC can be automatically-running to implement access control according to the logical order without excessive manual intervention. Hence, the fairness for all peer nodes is certainly achieved in FDAC.
- **Privacy-Preserving.** The anonymity of ABS utilized in FDAC can protect the true identities of users from being leaked. When to verify the attribute-based signature, the relevant identity information is kept in privacy from public.

5.2 Security analysis

- **Collusion Resistant.** The device nodes may collude together to exchange their attributes. To satisfy an unauthorized access policy, the adversary tries to collect additional attributes from other devices. However, our FDAC records the attributes of each user node and device node in the blockchain, and provides trustworthy for these digital credentials. As a result, a requester can not use other attributes that not belongs. If a user maliciously access unauthorized resources, it may be detected and get a punishment by the system. For example, S_i utilizes S_j's attributes to maliciously construct the attributes map$\left\{ ID_{S_i} : (atti)_{i \in S_i}, (attj)_{j \notin S_j} \right\}$.
- **Replay attack Resistant.** The adversary may intercept the access request sent by the user and replay the access request to gain access to device resources. In FDAC, the access requests are only sorted by *RequestID*, thus the adversary cannot send multiple requests with a same request ID. Therefore, the malicious repeated replays are eventually blocked permanently according to the conditions built into ACC and CC.
- **Modification attack Resistant.** The adversary tries to change or delete the stored data of a particular user or device. To launch such an attack, the attacker should compromise the security of the local storage. In FDAC, only the administrator node can delete and update the policy according to the designed smart contract. If any attacker tries to modify the block, the malicious modification is certainly detected. Since each block includes the hash value of its previous block *Hash*(*PreBlock*), and any modification on one block may lead to severe break to the blockchain.
- **Masquerade Attack Resistant.** The adversary tries to gain unauthorized access by stealing passwords, login names, finding program loopholes, and later launch a forgery attack. However, our FDAC can resist such attacks since the adversary cannot be registered as a legal entity. Moreover, each entity has a unique *ID* and blockchain address(*addr*), as a result, they are verified during system registration.

6 Performance analysis

To clarify practical performance of our FDAC, we implement it and employ an Ethereum platform to build an experiment environment. In particular, we install the geth client [36] on a laptop, a desktop and a server, and set up multiple Ethereum nodes. We list the specifications and configurations of employed equipments in Table 3. To well write and compile smart contracts, we use the Remix integrated development environment (IDE) [37], where Solidity is implemented and conducted in a browser-based IDE for implementing. Moreover, we use web3.js [38] to simulate the communications among corresponding geth clients. In particular, the access messages and results are received based on smart contracts via an HTTP interface. In addition, we conduct an extensive experimental analysis from the sides of smart contracts, strategy retrieval and credit respectively.

6.1 Performance of smart contracts

Here, we give a cost-consuming and time-consuming analysis of the introduced smart contracts in our FDAC system.

- **Gas Cost.** To measure the workload that perform various operations in the Ethereum platform, we employ the well-known gas to count it. Based on the collected data in our experiment, the gas that required to deploy AAC, PMC, ACC, and RC are respectively 2,354,612, 3,232,167, 2,939,883 and 2571606, which is shown in Fig. 6.
- **Average Latency.** To analyze the average latency of the deployment and execution of smart contracts, we give a couple of experiments with simulating different number of user nodes. As shown in Fig. 7, the consumed time for smart contract deployment and execution increases, with the increase of user nodes. However, the time delay between them also gradually increases.
- **Access Result.** When receiving a user's access request, ACC makes a corresponding access decision and returns the result to the user. In particular, Table 4 and Table 5 respectively shows the returned legal results and illegal results from the system. As a result, the system grants the access rights for a user when the user's access request is determined as legal. Otherwise, the user's access request will be blocked.

6.2 Performance of policy search

Based on the discovered findings of [39], the inherent false positive rate can reach the best or near the best case under the condition that: there are two candidate buckets in the

Table 3 Device specifications

Device Model	Processor	Memory
Laptop	AMD Ryzen7 4800H with Graphics2.90 GHz	16GB
Desktop	Intel(R)Core™ i5-7500 CPU @3.40GHz	16GB
Server	Intel(R)Core™ i5-9100 CPU @3.80GHz	32GB

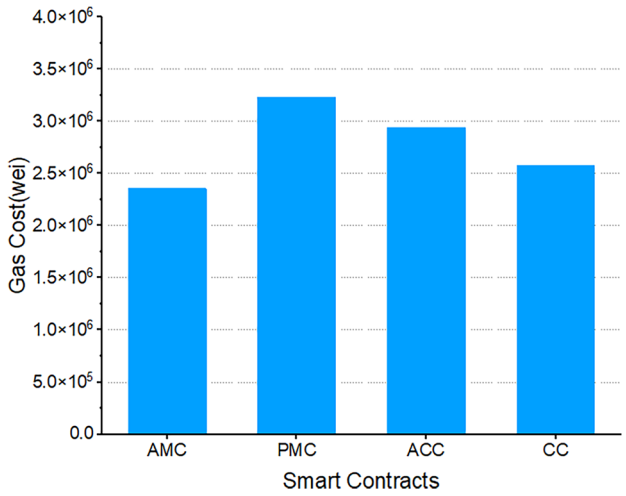

Fig. 6 Cost of deploying smart contracts

cuckoo filter and four fingerprints of each bucket. Note that the fingerprint information of the policy item is only stored in the bucket. As shown in Table 6, the false positive rate and the time delay of single policy search increase with the increase of the number of policies.

In addition, we conduct a comparison of the policy search efficiency between our FDAC and classic traversal search-based solutions. In the experiment, we measure the policy retrieval time based on different policy scales, which is shown in Fig. 8. Generally, our FDAC certainly reduces time consumption and greatly improves roughly 3× faster search efficiency compared to classic solutions.

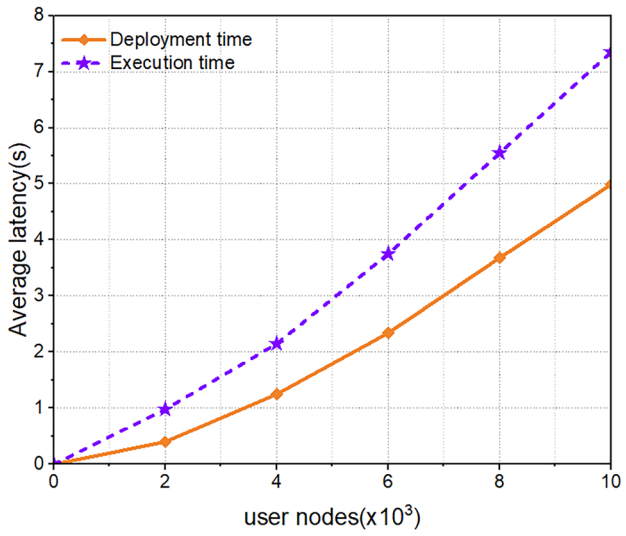

Fig. 7 Average latency during the deployment and execution of smart contracts

Table 4 Result of legal

Contract address	0x9ecEA68DE55F316B702f27eE389D-10C2EE0dde84
Block Number	2301
Tx Hash	0xe3009fcbb791b5b9192a493a44f114a7cf-5bfeb58033417a89036bb534e7c79a
Block Hash	0x4256997b953a30d7ebf5288fa2a2f297d-0ca7b088b66bc51ba1454827731d4a0
Message	Access authorized!
Result	true

6.3 Credit assesment analysis

To further clarify behaviors of users nodes, we measure the proportion of malicious behaviors of all the historical behaviors. As shown in Fig. 9, the credit value of user nodes and the degree of reduction gradually reduces with the increase of malicious behaviors. Therefore, we may conclude that the malicious behaviors has a much higher influence on the reputation of nodes than positive impact factors.

In our experiments, the time factor of credit value changes from t1 to t10, whose influence is shown in Fig. 10. Figure 10 shows that the positive impact of credit increases regularly, and the negative impact also increases with a small range. Finally, the overall credit value can be gradually improved. In particular, the influence of negative impact has gradually reduced over time. The primary goal of the system is to provide misbehavior-resistance and enhanced security against malicious behaviors. A misbehaved user's credit value gradually increases if its misbehavior is corrected, as a result, it can thus access the resources and information of terminal devices.

Table 5 Result of illegal access

Contract address	0x9ecEA68DE55F316B702f27eE389D-10C2EE0dde84
Block Number	2500
Tx Hash	0xdd09133ca4416ad3a648bc066a27d082a4e-0c1f3e6831b957bc0b008336ff1fd
Block Hash	0x347b7cf102df4e6511cad37beed-6a22d7b2928b9b5fbee11978710b0f1ecf3d1
Message	Requests are blocked!
Result	false

Table 6 Performance of policy search

Rounds	false positive rate(%)	Single time delay (ms)
1000	0.0402	0.115
2000	0.0402	0.115
3000	0.1533	0.116
4000	0.1165	0.125
5000	0.1644	0.134

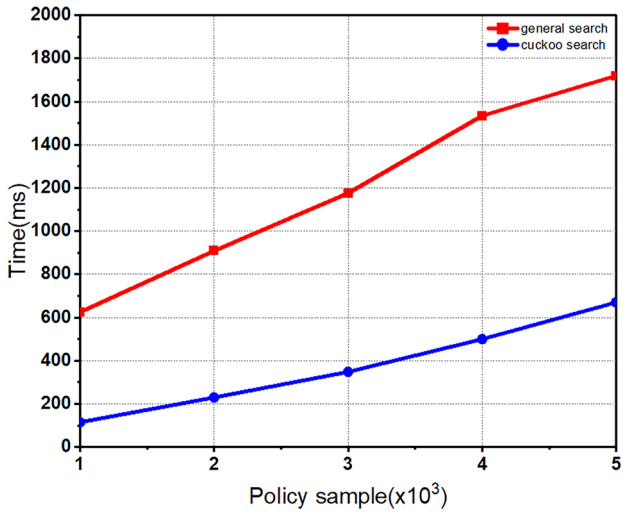

Fig. 8 Time efficiency comparison of policy retrieval time

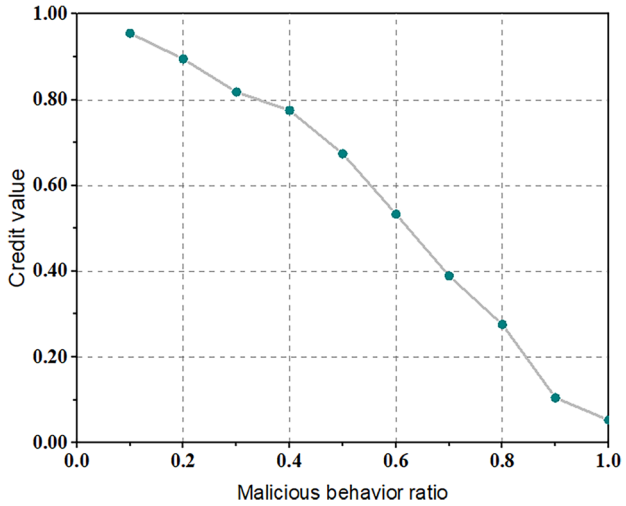

Fig. 9 Changes of credit value with different malicious behavior ratio

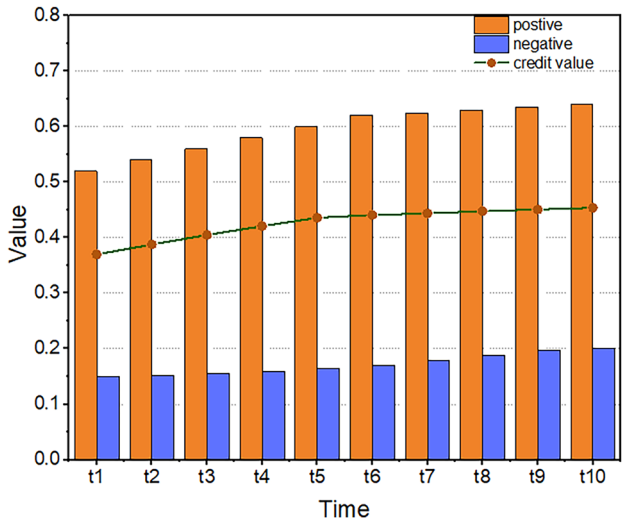

Fig. 10 Changes of credit value with different time

7 Conclusion

This paper has introduced a FDAC system for terminal device management in blockchain-based microgrid, which achieves distributed, efficient and fine-grained access control. In particular, we proposed four smart contracts to effectively and dynamically mange numerous devices, introduced Cuckoo filters and new algorithm of credit assessment to enhance the fine-grained flexibility of the system. A well-implemented experiment shows that our FDAC achieves high policy search efficiency, low time delay of smart contracts running, and dynamic, fine-grained access control over devices. Nevertheless, the storage cost of policies is a little high due to the discontinuous access space addresses, which seems an interesting future work to further improve policy storage efficiency.

Funding information This work was supported by National Natural Science Foundation of China (61802248, 61972094, 62032005, 62102089), the "Chenguang Program" supported by Shanghai Municipal Education Commission (No.18CG62), Program of Shanghai Academic Research Leader (No.21XD1421500), the Fundamental Research Funds for the Central Universities (2042021kf1030).

Declarations

Conflict of interest The authors declare that they do not have any commercial or associative interest that represents a conflict of interest in connection with the work submitted.

References

1. Yaqoob I, Ahmed E, Hashem IAT, Ahmed AIA, Gani A, Imran M, Guizani M (2017) Internet of things architecture: Recent advances, taxonomy, requirements, and open challenges. IEEE Wirel Commun 24(3):10–16
2. Palattella MR, Dohler M, Grieco A, Rizzo G, Torsner J, Engel T, Ladid L (2016) Internet of things in the 5g era: Enablers, architecture, and business models. IEEE J Sel Areas Commun 34(3):510–527
3. D'Orazio CJ, Choo KKR, Yang LT (2016) Data exfiltration from internet of things devices: ios devices as case studies. IEEE Internet of Things J 4(2):524–535
4. https://www.sans.org/industrial-control-systems-security/
5. Underwood S (2016) Blockchain beyond bitcoin. Commun ACM 59(11):15–17
6. Osborn S, Sandhu R, Munawer Q (2000) Configuring role-based access control to enforce mandatory and discretionary access control policies. ACM Transactions on Information and System Security (TISSEC) 3(2):85–106
7. Sandhu R (1998) Role-based access control. vol 46, Elsevier, pp 237–248
8. Sandhu RS, Samarati P (1994) Access control: principle and practice. IEEE Commun Mag 32(9):40–48
9. Yavari A, Panah AS, Georgakopoulos D, Jayaraman PP, van Schyndel R (2017) Scalable role-based data disclosure control for the internet of things. In: 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), IEEE, pp 2226–2233

10. Gusmeroli S, Piccione S, Rotondi D (2013) A capability-based security approach to manage access control in the internet of things. Math Comput Model 58(5–6):1189–1205
11. Mahalle PN, Anggorojati B, Prasad NR, Prasad R (2013) Identity authentication and capability based access control (iacac) for the internet of things. J Cyber Sec Mobility 1(4):309–348
12. Hu VC, Ferraiolo D, Kuhn R, Friedman AR, Lang AJ, Cogdell MM, Schnitzer A, Sandlin K, Miller R, Scarfone K et al (2013) Guide to attribute based access control (abac) definition and considerations (draft). NIST Spec Publ 800(162):1–54
13. Ye N, Zhu Y, Wang Rc, Malekian R, Lin Qm (2014) An efficient authentication and access control scheme for perception layer of internet of things
14. Bhatt S, Patwa F, Sandhu R (2017) Access control model for aws internet of things. In: International Conference on Network and System Security, Springer, pp 721–736
15. Dorri A, Kanhere SS, Jurdak R, Gauravaram P (2017) Blockchain for iot security and privacy: The case study of a smart home. In: 2017 IEEE international conference on pervasive computing and communications workshops (PerCom workshops), IEEE, pp 618–623
16. Zyskind G, Nathan O et al (2015) Decentralizing privacy: Using blockchain to protect personal data. In: 2015 IEEE Security and Privacy Workshops, IEEE, pp 180–184
17. Maesa DDF, Mori P, Ricci L (2017) Blockchain based access control. In: IFIP international conference on distributed applications and interoperable systems, Springer, pp 206–220
18. Zhang Y, Kasahara S, Shen Y, Jiang X, Wan J (2018) Smart contract-based access control for the internet of things. IEEE Internet Things J 6(2):1594–1605
19. Sifah EB, Xia Q, Agyekum KOBO, Amofa S, Gao J, Chen R, Xia H, Gee JC, Du X, Guizani M (2018) Chain-based big data access control infrastructure. J Supercomput 74(10):4945–4964
20. Saini A, Zhu Q, Singh N, Xiang Y, Gao L, Zhang Y (2020) A smart-contract-based access control framework for cloud smart healthcare system. IEEE Internet Things J 8(7):5914–5925
21. Zhang Y, Yutaka M, Sasabe M, Kasahara S (2020) Attribute-based access control for smart cities: A smart-contract-driven framework. IEEE Internet Things J 8(8):6372–6384
22. Qashlan A, Nanda P, He X (2020) Security and privacy implementation in smart home: Attributes based access control and smart contracts. 2020 IEEE 19th International Conference on Trust. Security and Privacy in Computing and Communications (TrustCom), IEEE, pp 951–958
23. Ouaddah A, Abou Elkalam A, Ait Ouahman A (2016) Fairaccess: a new blockchain-based access control framework for the internet of things. Sec Com Netw 9(18):5943–5964
24. Azaria A, Ekblaw A, Vieira T, Lippman A (2016) Medrec: Using blockchain for medical data access and permission management. In: 2016 2nd international conference on open and big data (OBD), IEEE, pp 25–30
25. Pinno OJA, Gregio ARA, De Bona LC (2017) Controlchain: Blockchain as a central enabler for access control authorizations in the iot. In: GLOBECOM 2017-2017 IEEE Global Communications Conference, IEEE, pp 1–6
26. Ferdous MS, Margheri A, Paci F, Yang M, Sassone V (2017) Decentralised runtime monitoring for access control systems in cloud federations. In: 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), IEEE, pp 2632–2633
27. Novo O (2018) Blockchain meets iot: An architecture for scalable access management in iot. IEEE Internet Things J 5(2):1184–1195
28. Xu R, Chen Y, Blasch E, Chen G (2018) Blendcac: A smart contract enabled decentralized capability-based access control mechanism for the iot. Computers 7(3):39
29. Song L, Li M, Zhu Z, Yuan P, He Y (2020) Attribute-based access control using smart contracts for the internet of things. Proc Comp Sci 174:231–242
30. Zhang Y, Li B, Liu B, Wu J, Wang Y, Yang X (2020) An attribute-based collaborative access control scheme using blockchain for iot devices. Electronics 9(2):285
31. Rouhani S, Belchior R, Cruz RS, Deters R (2021) Distributed attribute-based access control system using permissioned blockchain. World Wide Web 24(5):1617–1644
32. Nakamoto S (2008) Bitcoin: A peer-to-peer electronic cash system. Decentralized Business Review p 21260
33. Introduction to smart contracts, https://solidity.readthedocs.io/en/v0.5.6/introduction-to-smart-contracts.html/, [online]
34. Yang XS, Deb S (2010) Engineering optimisation by cuckoo search. Int J Math Model Numer Optim 1(4):330–343
35. Maji HK, Prabhakaran M, Rosulek M (2011) Attribute-based signatures. In: Cryptographers track at the RSA conference, Springer, pp 376–392
36. geth-go implementaion of ethereum protocol, https://github.com/ethereum/go-ethereum
37. remix- ide for smart contract deployment provided by ethereum, https://remix.ethereum.org/
38. web3 javascript api to interact with ethreum nodes, https://github.com/ethereum/wiki/wiki/javascript-api
39. Fan B, Andersen DG, Kaminsky M, Mitzenmacher MD (2014) Cuckoo filter: Practically better than bloom. In: Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies, pp 75–88

Kai Zhang received the Bachelor's degree with Computer Science and Technology from Shandong Normal University, China, in 2012, and the Ph.D. degree with Computer Science and Technology from East China Normal University, China, in 2017. He visited Nanyang Technological University in 2017. He is currently an Associate Professor with Shanghai University of Electric Power, China. His research interest includes data-driven privacy enhanced techniques and information security.

Jinhu Yu received the bachelor's degree from the School of Computer Engineering, Jiangsu Ocean University, China, in 2020. He is currently pursuing his master degree in Department of Computer Science and Technology, Shanghai University of Electric Power, China. His research interests include blockchain and access control.

Chao Lin received the Ph.D. degree from the School of Cyber Science and Engineering, Wuhan University, in 2020. He currently works with the College of Mathematics and Informatics, Fujian Normal University, China. His research interests mainly include applied cryptography and blockchain technology.

Jianting Ning received the Ph.D. degree from the Department of Computer Science and Engineering, Shanghai Jiao Tong University, in 2016.He is currently a Professor with Fujian Provincial Key Laboratory of Network Security and Cryptology, College of Computer and Cyber Security, Fujian Normal University, China. Previously, he was a Research Scientist at the School of Computing and Information Systems, Singapore Management University. He has published papers in major conferences/journals, such as ACM CCS, ASIACRYPT, ESORICS, ACSAC, IEEE TIFS, and IEEETDSC. His research interests include applied cryptography and information security.