Hindawi
Wireless Communications and Mobile Computing
Volume 2023, Article ID 4778967, 13 pages
https://doi.org/10.1155/2023/4778967

*Research Article*

# A Multi-Blockchain-Based Cross-Domain Authentication and Authorization Scheme for Energy Internet

**Donglan Liu, Xin Liu, Rui Wang, Hao Zhang, Fangzhe Zhang, Lili Sun, Honglei Yao, and Hao Yu**

*State Grid Shandong Electric Power Research Institute, Jinan 250003, China*

Correspondence should be addressed to Donglan Liu; liudonglan2006@126.com

Received 6 August 2022; Revised 2 October 2022; Accepted 24 November 2022; Published 29 April 2023

Academic Editor: Zengpeng Li

The expansion of the scale of the Power Internet of Things stimulated by the development of the Energy Internet makes the growth in demand for the effective authentication and access control technologies in the cross-domain data exchange. Based on the cross-chain technology of the blockchain and the cuckoo filter, this paper proposes a cross-domain authentication scheme for Power Internet of Things. Firstly, a cross-chain authentication architecture is established. Combined with the existing authentication technologies used in intra-domain authentication, a cross-domain authentication process based on the cross-chain technology is proposed to realize the automatic transmission of the authentication credentials from application domain to authentication domain. The cuckoo filter is deployed on the blockchain as smart contracts, and the user certificate fingerprint is inserted into the filter to realize user registration, query, and revocation, which reduces the cost of the user certificate management. Experimental results show the effectiveness and feasibility of our scheme. Based on the proposed authentication scheme, a cross-domain access control scheme based on roles and object classes is presented, by treating the object classes as controlled objects and then applying the role-based access control to the object classes, on the condition that the heterogeneous domains in the Energy Internet have the same kinds of resources.

## 1. Introduction

In the scenario of Energy Internet and Power Internet of Things, there are a large number of terminal nodes deployed in a wide range, and the physical environments of some of the nodes are uncontrollable. This makes them vulnerable to many threats such as physical hijacking, node replication, signal interception, theft and replay, man in the middle attack, and so on. Therefore, the terminal authentication has a greater and greater impact on the security of the power system. The traditional Power Internet of Things access security mainly depends on the centralized key management mechanism, which has the disadvantages of low authentication efficiency and the risk of single point failure [1]. As many comprehensive services require cross-domain data sharing, centralized authentication and authorization can no longer meet the trust requirements for multiparty cross-domain business systems such as Power Internet of Things source-network-load-storage interaction and accurate material supply. The scenario is shown in Figure 1. Therefore, it is necessary to propose cross-domain authentication and authorizaiton schemes. In addition, in the design of cross-domain schemes, the limited computing resources, storage, and communication capabilities of Power Internet of Things terminals need to be considered.

To support cross-domain authentication, many solutions have been proposed. Generally speaking, these solutions can be divided into four categories: public key infrastructure (PKI) based solutions, identity-based encryption (IBE) based solutions, password-based solutions, and blockchain based solutions. PKI based schemes are only suitable for the Internet scenario rather than the Internet of Things scenario, as the result of the high cost of the certificate management. IBE schemes can eliminate the overhead of the certificate transmission, authentication, and maintenance [2] so that it is a potential solution for cross-domain

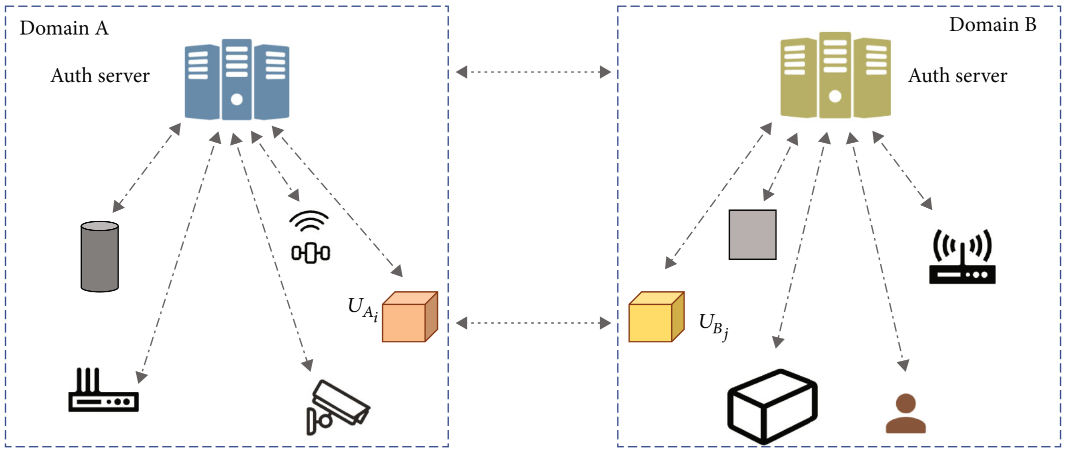


FIGURE 1: Cross domain scenario.

authentication. Blockchain is composed of data blocks in the form of ordered chain and can be treated as a distributed ledger maintained by multiple parties [3]. It uses encryption, Merkel tree, consensus mechanism, and other technologies to realize the transparency, tamper-resilience, and traceability of transaction data. Blockchain also provides a platform to deploy the smart contract [4], which is defined as a computer program that can automatically execute predefined protocols. Applying blockchain to cross-domain authentication can promote the efficiency and security.

*1.1. Related Work.* Cross-domain authentication can be realized based on public key infrastructure. For example, Vaidya et al. proposed the multidomain mechanism of V2G infrastructure [5], which supports point-to-point cross-domain authentication by constructing a model based on hybrid public key infrastructure and establishing the trust relationship between nodes through intra-domain and inter-domain digital identities. However, all PKI-based schemes require certification authority (CA) to store, issue, and manage digital certificates for each user [6], which costs heavy overheads. Since the introduction of the identity-based encryption, which does not need to manage digital certificates, researchers began to take advantage of it to design efficient protocols. In 2010, Cao et al. [7] proposed an efficient two-party authentication protocol reducing the number of interactive rounds. However, the protocol needs a trusted third party, and can not achieve cross-domain authentication between heterogeneous domains which have deployed different authentication mechenisms. Benzarti and Triki [8] designed an authentication framework based on identity encryption and signature, in which group identifier, object identifier, IP address, and a unique tamper-proof RFID tag of a user are combined with a temporary identity to realize authentication. However, the scheme cannot support cross-domain authentication. Shen et al. [9] proposed a cross-domain authentication scheme based on blockchain and identity-based signature.

As an effective method of the trust transmission in decentralized scenarios, blockchain technology has been widely studied in the field of Internet of Things security [10–19]. Ouaddah et al. [11] proposes a blockchain based access control framework for the Internet of Things. In terms of authentication, Fromknecht et al. [10] proposes a blockchain based distributed public key infrastructure (PKI) system, which records user certificates through the public general ledger to solve the single point of failure problem of the traditional PKI systems. In the resource sharing scenario, [12] proposed a cross-domain framework based on consortium blockchain technology. Yao et al. [14] proposed a blockchain assisted lightweight anonymous authentication mechanism. Guo et al. [15] designed a method to support authentication of different systems and domains. These schemes use blockchain to replace TTP, store tamper-proof authentication data, and support cross-domain authentication. However, they are inefficient and inflexible. Zhang et al. [16] proposes a thoroughly cross-domain authentication scheme based on blockchain, but this scheme does not consider the independent deployments of the blockchains in different domains, i.e., cross chain. Besides, it is inappropriate in this scheme that, when the user is about to be removed from the domain, it requires the user to actively ask for being revoked and needs some information generated by the user under his private key. For the blockchain-based authorization and access control, Gauhar et al. [18] proposed a blockchain based IoT authorization framework to realize authorization based on the authorization policies stored on the blockchain. In 2021, Zhu et al. [19] proposed an approach utilizing capability-based cross-domain access control and risk-based access control mechanisms in a domain while taking IoT nodes as data resources. However, these solutions cannot effectively leverage the access control mechanisms that are already adopted within the domains.

*1.2. Our Contribution.* The current researches mainly focus on using a single blockchain to improve the distribution and reliability of the authentication services [20]. However, the single blockchain structure is difficult to meet the requirements of the Energy Internet in terms of operation efficiency, maintenance cost, and privacy protection. To solve the problems of “incompletely cross-domain” [16], we propose a multi-blockchain-based cross-domain authentication scheme supporting heterogeneous domains for the Energy Internet. In our scheme, users coming from different domains can be authenticated by the authentication server of the domain he wants to access with the help of the chaincodes which have been deployed in the blockchain platforms. Each domain which may be an energy company can deploy its own blockchain which stores some authentication data of users in it to realize the isolation and data protection. There is a blockchain deployed to coordinate and audit the cross-domain authentication which is called supervision chain. We design a revocation process which is more suitable for the practical use, i.e., the authentication server in each domain can revoke users adaptively without users' private information. To realize the user revocation, we first initialize a cuckoo filter, and then user information is mapped to the fingerprint and inserted into the cuckoo filter. When the authentication server wants to revoke a user, it only needs to compute the fingerprint information of the user and delete the fingerprint from the cuckoo filter. To support cross-domain authorization, we assume that the heterogeneous domains have the same kinds of resources and treat the object classes as controlled objects. By applying the role-based access control to the object classes, we can extend our blockchain-based cross-domain authentication scheme and propose a cross-domain access control model based on roles and object classes.

The rest of the paper is organized as follows: In Section 2, we give the preliminaries of blockchain and cuckoo filter. In Section 3, we present the multi-blockchain-based cross-domain authentication scheme supporting heterogeneous domains for the Energy Internet. We also analyze the security and performance of the scheme in this section. In Section 4, we show how to extend the proposed authentication scheme to support the cross-domain authorization. Finally, we draw a conclusion in Section 5.

## 2. Preliminaries

We will provide in this section some preliminaries which are necessary for the understanding of the subsequent schemes.

*2.1. Blockchain.* Blockchain is a chain structure that links blocks in order [3]. A block is a collection of data in which relevant information and records are stored. The block is composed of a block header and a block body. The block header stores the version number, the hash value of the previous block, Merkle root, timestamp, etc. The block body contains the information of multiple transactions that have occurred since the previous block. The blockchain realizes the consistency of data through the consensus mechanism of self-trust. It uses encryption, Merkel tree, consensus mechanism, and other technologies to realize the transparency, tamper-resilience, and traceability of transaction data, and can be treated as a distributed ledger maintained by multiple parties. It can use the smart contract to automatically process data and realize the efficient and secure data exchange between entities without the need for a trusted third party. Blockchain provides a solution for trust establishment between entities in the distributed environment.

Smart contract [4] is a sequence of computer program with predefined protocols, which can be deployed on the blockchain. Smart contract can automatically execute the predefined protocols to complete information exchange and asset management. Smart contract is called "chaincode" in hyperledger fabric [21]. Chaincode runs in a secured container isolated from the endorsing peer process. Users can read and write a set of key-value pair status data on the ledger.

*2.2. Cuckoo Filter.* Cuckoo filter is a data structure based on cuckoo hashing algorithm [22]. The algorithm uses two hash functions $h_1$ and $h_2$ to map the element to be inserted to the corresponding position in one of the two maintained hash tables each of which has $m$ elements. Each element $x$ will either be inserted at position $h_1(x)$ in the first hash table or $h_2(x)$ in the second one. When inserting an element $x$ into the $i$-th ($i \in 1, 2$) hash table, it first checks whether there has been an element placed in the posision $h_i(x)$. If there has been no element in $h_i(x)$, then $x$ is placed in the posision $h_i(x)$ of the $i$-th hash table. Otherwise, let $y$ be the element which has been placed in $h_i(x)$ of the $i$-th hash table, it should try to insert $y$ to the other hash table in the position $h_{3-i}(y)$ after placing $x$ in the posision $h_i(x)$ of the $i$-th hash table. The inserting operation will not stop until no element needs to be moved. For a new element, the algorithm can place it to the hash tables by executing the above operation starting from inserting it into the first hash table. Cuckoo hashing improves the load factor and query performance of the hash table. Note that the first hash table and the second hash table can be combined to be one hash table [23].

The cuckoo filter [23] designed by Fan et al. expands the cuckoo hash table to a multidimensional structure by adding several slots at each bucket of the hash table. It can significantly reduce the number of the element relocation operations. The cuckoo filter stores the binary fingerprint information of the elements and defines $h_2(x) = h_1(x) \oplus \text{fingerprint}(x)$ to help distribute the items uniformly in the hash table and complete an insertion only using information in the hash table rather than retrieving the original item $x$. The time complexities of the cuckoo filter for looking up and deleting an element are $O(1)$.

## 3. Multi-Blockchain-Based Cross-Domain Authentication Scheme

*3.1. System Model.* We design a cross-domain authentication system model utilizing the multichain structure as shown in Figure 2, which consists of domains, authentication servers, users, and multichain networks.

Domain: we define a domain as a group in which the users trust each other, i.e., Domain $A$ and $B$ in Figure 2. Note that different domains may adopt different cryptographic settings

Authentication Server: the authentication server provides authentication for the users in the domain and can handle authentication requests from within and across

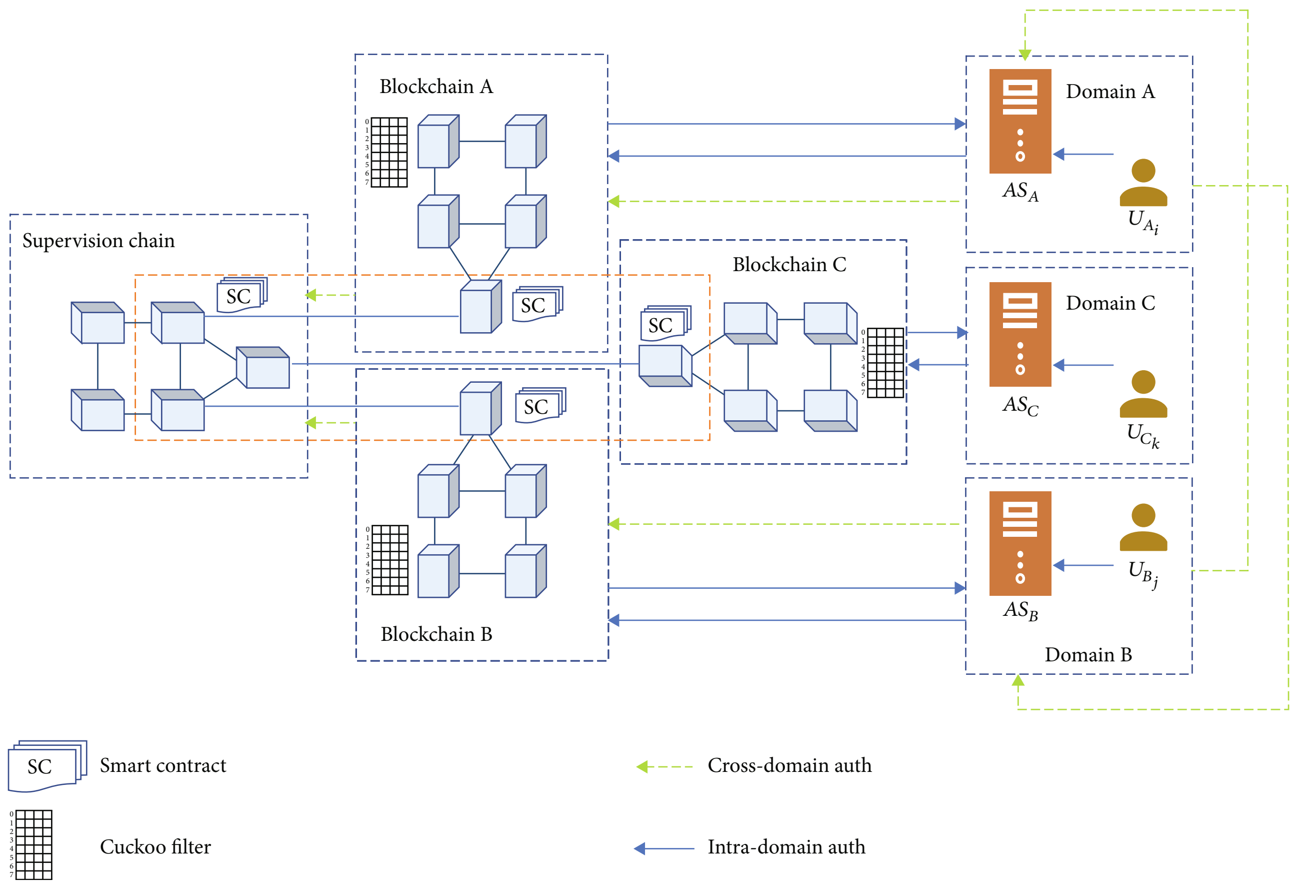


FIGURE 2: Blockchain based cross-domain authentication system model.

domains with the help of the blockchain. As shown in Figure 2, the authentication servers in domain $A$ and domain $B$ are represented by $AS_A$ and $AS_B$, respectively

User: there are two roles of the users, i.e., data owner who owns specific resources within a domain and data user who wants to access resources within the same domain or across domains. As shown in Figure 2, the users in domains $A$ and $B$ are represented by $U_{A_i}$ and $U_{B_j}$

Multichain Networks: each domain deploy a blockchain which stores information of the users used in authentication in the domain. The supervision chain is used to implement cross-domain authentication

*3.2. Details of the Scheme.* In this section, we describe the details of our scheme in the scenario shown in Figure 2. Specifically, each domain should create its blockchain using Hyperledger Fabric, in which the chaincode of the cuckoo filter should be deployed. Also, the supervision chain should be created which help the cross-domain authentication. Then, each domain chooses a peer in its blockchain as the link peer and join it to the channel of the supervision chain as follows:

Actually, in fabric, joining the link peer to the channel of the supervision chain is adding an orgnization to a channel. Channel in fabric is only an abstract concept rather than a real entity. Therefore, configuring the channel is basically the management of *Channel Configuration*. After a blockchain is successfully deployed, the genesis block files in the folder "channel-artifacts" form the *Channel Configuration*. However, these configuration files are binary files. So, we need to turn them into readable configuration files by executing the following operations in Figure 3. The resulting configuration file is shown on the right of Figure 3.

Then, we can modify these configuration files as follows:

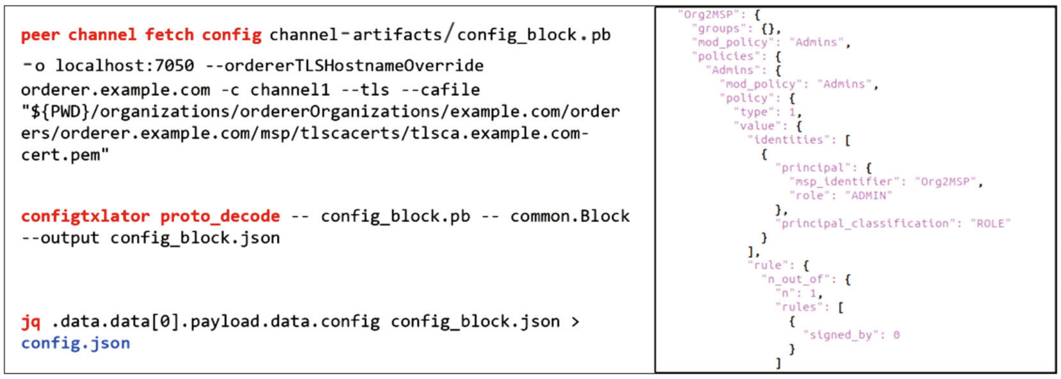
(i) Use the jq command to write the information of the link pear into the above mentioned configuration files

(ii) Use configtxlator proto_encode command packages the original configuration files and the modified configuration files, respectively, to obtain the binary file supported by fabric

(iii) Use configtxlator compute_update command calculates the differences between the original configuration files and the modified configuration files

(iv) Use configtxlator to update the calculated changes to the original configuration files

(v) Each domain and the orgnization of the supervision chain use peer channel signconfigtx to sign the new



```
peer channel fetch config channel-artifacts/config_block.pb
-o localhost:7050 --ordererTLSHostnameOverride
orderer.example.com -c channel1 --tls --cafile
"${PWD}/organizations/ordererOrganizations/example.com/order
ers/orderer.example.com/msp/tlscacerts/tlsca.example.com-
cert.pem"


configtxlator proto_decode -- config_block.pb -- common.Block
--output config_block.json



jq .data.data[0].payload.data.config config_block.json >
config.json
```

```
"Org2MSP": {
  "groups": {},
  "mod_policy": "Admins",
  "policies": {
    "Admins": {
      "mod_policy": "Admins",
      "policy": {
        "type": 1,
        "value": {
          "identities": [
            {
              "principal": {
                "msp_identifier": "Org2MSP",
                "role": "ADMIN"
              },
              "principal_classification": "ROLE"
            }
          ],
          "rule": {
            "n_out_of": {
              "n": 1,
              "rules": [
                {
                  "signed_by": 0
                }
              ]
```

Figure 3: Turn binary files into readable configuration files.

configuration files, respectively. Also, they use peer channel update to enable new configuration files

Now, the link peer is successfully joined into the channel of the supervision chain.

Next, we describe how to deploy the key chaincode about crossing chain in different chains. In fabric, a peer can query and change the world status of other chaincodes in the same channel, while a peer can query the world status of other chaincodes in the different channel rather than changing them. As described above, link peer in each domain's blockchain is joined into the channel of the supervision chain. Therefore, each domain can let the supervision chain obtain the authentication data stored in its blockchain. That is to say, link peer should deploy the chaincode about crossing chain as well as the supervision chain. The multichain structure can be seen in Figure 2.

Algorithm 1 shows the key chaincode about crossing chain. Algorithms 2–5 show the chaincodes about creating the cuckoo filter, insert algorithm, lookup algorithm, and delete algorithm of the cuckoo filter.

At last, we describe the details of our authentication scheme including registration, intra-domain authentication, cross-domain authentication, and revocation.

*Registration*. For a user $U_{A_i}$, firstly, registration is done according to the authentication mechanism of the domain which the user belongs to. Then the authentication information in the domain such as user identification $U_{A_i}$, public key certificate $Cert_{U_{A_i}}$, timestamp, authority identification $AS_A$, etc. are stored in the blockchain created by the domain. The chaincode of the insert algorithm of the cuckoo filter is called to add the user's identity and authentication information to the filter. At last, the link peer of the blockchain calls the chaincode *SendCrossMessage* deployed in the supervision chain to generate a new transaction record on the supervision chain for storing the authentication information.

*Intra-domain authentication*. Upon receiving the authentication request from user $U_{A_i}$, the authentication server $AS_A$ first calls the lookup algorithm of the cuckoo filter to check whether the certificate $Cert_{U_{A_i}}$ exists in the filter, i.e., $U_{A_i}$ is not revoked. Then, $AS_A$ authenticates $U_{A_i}$ according to the authentication mechanism of Domain $A$.

*Cross-domain authentication*. If a user $U_{A_i}$ in Domain $A$ wants to be authenticated by the authentication server $AS_B$ in Domain $B$, they execute an interactive protocol as follows:

(1) $U_{A_i}$ sends the authentication request to $AS_B$ along with its identity and certificate $Cert_{U_{A_i}}$

(2) $AS_B$ checks whether the user $U_{A_i}$ has been revoked by calling the chaincode of the lookup algorithm of the cuckoo filter. If the check passes, go to the next step. Otherwise, the authentication stops

(3) $U_{A_i}$ passes the authentication by $AS_A$ and the authentication information of $U_{A_i}$ is written to the blockchain

(4) $AS_B$ sends the information request to the supervision chain by calling the chaincode *GetCrossMessage* described in Algorithm 1 to get the authentication information of $U_{A_i}$

(5) $AS_B$ sends a random challenge cha to user $U_{A_i}$

(6) $U_{A_i}$ uses the private key to sign the challenge and sends the signature $\mathrm{Sig(cha)}$ to $AS_B$

(7) $AS_B$ verify the signature $\mathrm{Sig(cha)}$ using the public key of $U_{A_i}$

(8) The authentication is completed

Figure 4 Shows the workflow of the cross-domain authentication.

*Revocation*. If the authentication server $AS_A$ needs to revoke a user $U_{A_i}$, it calls the delete algorithm of the cuckoo filter to remove the fingerprint information of $U_{A_i}$'s certificate, and finally the filter is updated in the blockchain. $AS_A$ also needs to complete the revocation process according to the authentication mechanism of Domain $A$.

### *3.3. Security and Performance Analysis*

*3.3.1. Security*. Due to the unforgeability of the blockchain, the user's authentication information on blockchains is hard

```
1: Procedure (s *SmartContract) GetCrossMessage(ctx contractapi.TransactionContextInterface, channel, chaincode, user1, user2 string) (string, error)
2:  params := []string{"GetMessage", user1, user2}
3:  invokeArgs: = make([][]byte, len(params))
4:  For i, arg: = range params {invokeArgs[i] = []byte(arg)}
5:  response := ctx.GetStub().InvokeChaincode(chaincode, invokeArgs, channel)
6:  If response.Status! = shim.OK then
7:      Return shim.Error(err.Error())
8:  End if
9:  users := user1 + ":" + user2
10: messages := Messages{Users: users, Message: string(response.Payload)}
11: msgJSON, err: = json.Marshal(messages)
12: If err! = nil then
13:     Return shim.Error(err.Error())
14: End if
15: Err = ctx.GetStub().PutState(users, msgJSON)
16: If err! = nil then
17:     Return shim.Error(err.Error())
18: End if
19: Return string(response.Payload), nil
20: End procedure
```

Algorithm 1: The key chaincode about crossing chain.

```
1: Procedure (s *SmartContract) InitCFilter(ctx contractapi.TransactionContextInterface) string
2:  var cf CFilter
3:  configure(&cf)
4:  cf.Buckets = make([]bucket, cf.Size, cf.Size)
5:  i := range cf.Buckets
6:  cf.Buckets[i] = make([]fingerprint, cf.BSize, cf.BSize)
7:  cfJSON, _: = json.Marshal(&cf)
8:  err := ctx.GetStub().PutState("cuckfilter", cfJSON)
9:  If err! = nil then
10:     Return shim.Error(err.Error())
11: End if
12: Return string(cfJSON)
13: End procedure
```

Algorithm 2: The chaincode about creating the cuckoo filter.

to tamper and forge, which effectively ensures the integrity of data and the validity of the user's identity in the cross-domain authentication. In the traditional cross-domain authentication scheme based on PKI, Certificate Authority (CA) is vulnerable to attacks. In our scheme, the user's certificate is stored on the blockchain and cannot be tampered so that it is more resistant to denial of service attacks and other attacks affecting system availability. For the privacy of the user's authentication information stored in the blockchain, we utilize Hyperledger Fabric multiple channels mechanism to separate the information between different channels. Only nodes in the same channel can share the data [24].

*3.3.2. Performance.* We mainly analyze the time overheads costed when checking whether the user is revoked while obtaining the user's authentication information in the cross-domain authentication. Table 1 Summarizes the basic information of our experiment.

The test blockchain network runs on two hosts belonging to two different channels. Each host contains a total of eight nodes, four MSPs, and three sorting services belonging to four domains and being located in the same channel.

(i) Time overhead of checking whether the user is revoked. This is the execution time of the chaincode of the lookup algorithm of the cuckoo filter. We send 1000 transactions to the blockchain network and observe the changes of average latency and throughput (TPS) of the system under different transaction sending rates as shown in Figure 5

(ii) Time overhead of obtaining the user's authentication information. This is the execution time of the chaincode of *GetCrossMessage* described in Algorithm 1. We send 5000 transactions to the blockchain network and observe the changes of average latency

```
Procedure func (s *SmartContract) Insert(ctx contractapi.TransactionContextInterface, id string, item string) error
    cfJSON, err: = ctx.GetStub().GetState(id)
    If err! = nil then
        Return shim.Error(err.Error())
    End if
    var cf CFilter
    err = json.Unmarshal(cfJSON, &cf)
    If err! = nil then
        Return shim.Error(err.Error())
    End if
    f := fprint([]byte(item), cf.FpSize, Hashfn)
    j: = hashfp([]byte(item))
    k := (j XOR hashfp(f)) % cf.Size
    If cf.Buckets[j].insert(f) || cf.Buckets[k].insert(f) then
        cf.Count++
    Else
        i: = [2]uint{j, k}[rand.Intn(2)]

        For n := uint(0); n<cf.Kicks; n++ do
            f = cf.Buckets[i].swap(f)
            i = (i XOR hashfp(f)) % cf.Size

            If cf.Buckets[i].insert(f) then
                cf.Count++
            End if
        End for
    End if
    cfJSON, err = json.Marshal(&cf)
    If err! = nil then
        Return shim.Error(err.Error())
    End if
    CF_: = CF{id: Id, cfilter: cfJSON}
    err = ctx.GetStub().PutState(cf_.id, cf_.cfilter)
    If err! = nil then
        Return shim.Error(err.Error())
    End if
    Return nil
End procedure
```

Algorithm 3: The chaincode about the insert algorithm of the cuckoo filter.

```
Procedure func (s *SmartContract) Lookup(ctx contractapi.TransactionContextInterface, id string, item string) bool
    cfJSON, err: = ctx.GetStub().GetState(id)
    If err! = nil then
        Return shim.Error(err.Error())
    End if
    var cf CFilter
    err = json.Unmarshal(cfJSON, &cf)
    If err! = nil then
        Return shim.Error(err.Error())
    End if
    f := fprint([]byte(item), cf.FpSize, Hashfn)
    j := hashfp([]byte(item)) % cf.Size
    k := (j XOR hashfp(f)) % cf.Size
    Return cf.Buckets[j].lookup(f) || cf.Buckets[k].lookup(f)
End procedure
```

Algorithm 4: The chaincode about the lookup algorithm of the cuckoo filter.

```
1: Procedure func (s *SmartContract) Delete(ctx contractapi.TransactionContextInterface, id string, item string) error
2:   cfJSON, err: = ctx.GetStub().GetState(id)
3:   **If** err! = nil then
4:     **Return** shim.Error(err.Error())
5:   **End if**
6:   var cf CFilter
7:   err = json.Unmarshal(cfJSON, &cf)
8:   **If** err! = nil then
9:     **Return** shim.Error(err.Error())
10:  **End if**
11:  f := fprint([]byte(item), cf.FpSize, Hashfn)
12:  j := hashfp([]byte(item)) % cf.Size
13:  k := (j XOR hashfp(f)) % cf.Size
14:  If cf.Buckets[j].remove(f) || cf.Buckets[k].remove(f) then
15:      cf.Count–
16:  **End if**
17:  cfJSON, err = json.Marshal(&cf)
18:  If err! = nil **then**
19:      **Return** shim.Error(err.Error())
20:  **End if**
21:  cf_ := CF{id: id, cfilter: cfJSON}
22:  err = ctx.GetStub().PutState(cf_.id, cf_.cfilter)
23:  If err! = nil **then**
24:      **Return** shim.Error(err.Error())
25:  **End if**
26:  **Return** nil
27: **End procedure**
```

Algorithm 5: The chaincode about the delete algorithm of the cuckoo filter.

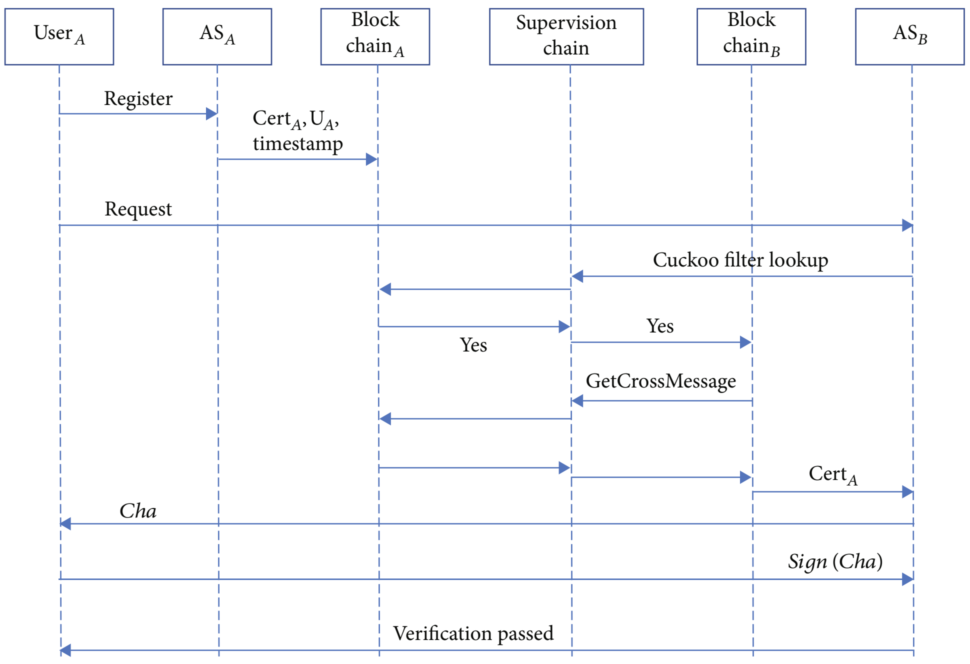


Figure 4: Process of the cross-domain authentication.

Table 1: Basic information.

| CPU series | Intel(R) Core(TM) i7-8700 |
|---|---|
| RAM | 16.0 GB (15.8 GB is available) |
| Operating system | Ubuntu 20.04.2 LTS |
| Software version | Hyperledger fabric 2.4 |
| Environment | Go version: Go 1.16.7 linux/amd64 |
|  | Docker version 20.10.8 |

and throughput (TPS) of the system under different transaction sending rates which is shown in Figure 6

## 4. Cross-Domain Authorization

*4.1. The Mapping Based Solution.* The key issue that should be solved in cross-domain authorization is how to convert an access control policy in the authorization domain to that in the access domain. In the IoT and Energy Internet scenarios, different domains may apply different access control mechanisms such as role-based access control (RBAC) [25, 26], access control list (ACL), attribute-based access control (ABAC) [27, 28], and capability-based access control (CapBAC) [29]. Moreover, even if the same access control mechanism is used, the access control policy may be different. For example, the roles of two domains applying the RBAC mechanism may be different. To make the authorization by the user's own domain be accepted by other domains, it is necessary to solve the problem of interoperability of access control policies between heterogeneous domains.

A straightforward solution is the mapping-based approach. In the initial phase, mappings are done between different authorization bases (user groups in ACL, roles in RBAC, attributes in ABAC) within domains and the mapping results are recorded in the blockchain. In the following, we briefly describe the access control policy interoperability method based on roles mapping between domains using RBAC mechanism, while the mapping methods between other authorization bases are similar which are omitted.

Role-based access control mechanism includes user set USERS, role set ROLES, and permission set PERMS. The authorization center assigns permissions to the roles, and the users obtain the permissions owned by the roles which he is assigned. Role is the core of RBAC mechanism, so the access control policy interoperability between two domains using RBAC mechanism can be achieved through role mapping. That is, mapping the role in the authorization domain to that in the access domain. The mapping should be one-way. For example, let the role sets in the authorization Domain $A$ and the access Domain $B$ be $\mathrm{ROLES}_A$ and $\mathrm{ROLES}_B$, respectively. $\{Ra_1, Ra_2, \cdots\} \mapsto \{Rb_1, Rb_2, \cdots\}$, where $\{Ra_1, Ra_2, \cdots\} \subseteq \mathrm{ROLES}_A$ and $\{Rb_1, Rb_2, \cdots\} \subseteq \mathrm{ROLES}_B$ means that $\{Ra_1, Ra_2, \cdots\}$ is mapped to $\{Rb_1, Rb_2, \cdots\}$ and the mapping results is written to the blockchain. In the process of cross-domain authorization, the user performs identity authentication and obtains the corresponding role according to the above cross-chain authentication process. When a user in Domain $A$ wants to access the resources in the access Domain $B$, the user's roles $\in \mathrm{ROLES}_A$ can be converted into the roles $\in \mathrm{ROLES}_B$ according to the role mapping, and finally the authorization center of the access Domain $B$ can assign permissions to the user according to his corresponding roles $\in \mathrm{ROLES}_B$.

Although the approach of mapping between different authorization bases can achieve interoperability of access control policies between heterogeneous domains, it requires a lot of overheads of implementing the mapping by authorization centers of the domains in the initial stage. Moreover, the mapping between different authorization bases also needs to consider the tradeoff of access control granularity for all of the domains. To our knowledge, most of the IoT domains currently use RBAC and ABAC mechanisms, so a simpler approach [30] proposed to turn the access control policies in ABAC mechanism into the access control rules in RBAC mechanism can be used for access control policy interoperability when we focus on these two kinds of access control mechanisms. As a result, the access control policies of the domains using ABAC mechanism can be converted into the RBAC rules first, and then only the mapping from roles to roles can be applied globally for RBAC rule's interoperability among various domains. Thus, access control policies interoperability among heterogeneous domains can be achieved for cross-domain access control. However, in general, the existing mapping methods have low practicality in the real scenario with large scale and complex relationships.

*4.2. Role and Object Class Based Access Control.* Inspired by the capability-based access control (CapBAC) model [31], we propose a cross-domain access control model based on roles and object classes.

Currently, CapBAC model is used in many IoT domains. CapBAC model is an implementation of the access control matrix (ACM) model, while the access control list (ACL) model is another implementation. In the ACL model, each object is associated with an access control list, which records the access permissions of the subjects to it. Conversely, in the CapBAC model, each subject is associated with a list of capabilities (permissions) that record the subject's access permissions to the objects. As shown in Figure 7(a), from the row perspective, each subject is assigned the access rights to the objects, and from a column perspective, it is an ACL model, where $O$ is the subject's access permissions to the object, such as readability and writability, and $C$ is a set of context-aware information, such as time and location.

We assume that the heterogeneous domains in the Energy Internet have the same kinds of resources, such as computing resources, printer, camera, devices in smart grid, various types of energy data, and so on. By this assumption and using the access control matrix, the objects in the domains can be represented in the unified forms. Then, regardless of the access control mechanism used by the domains, we can use the same classes of the objects to link their access control policies. Figure 7(b) shows that the access control matrix in CapBAC model allows for object classification and the assignment of roles to subjects. Also, ACL, RBAC, and ABAC mechanisms can be transformed to be role and object class based mechanisms. For example,

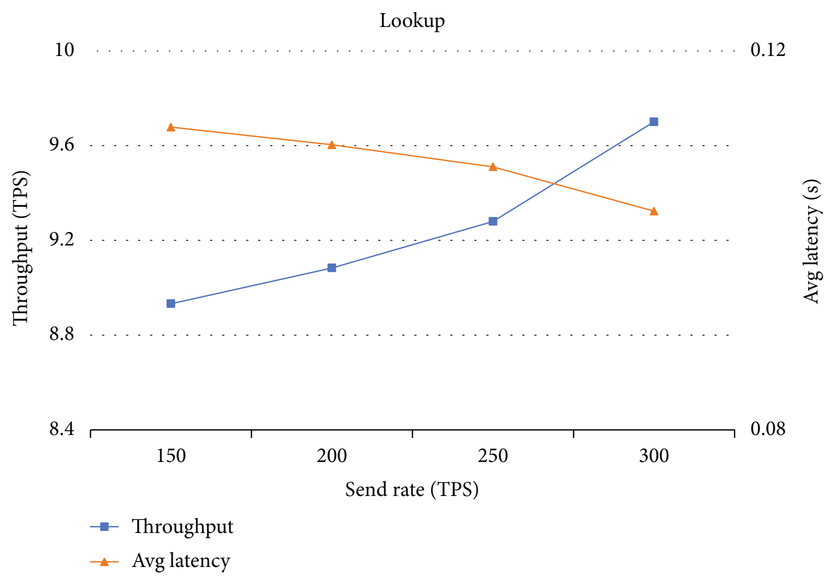


FIGURE 5: Performance of invoking lookup chainchode of the cuckoo filter.

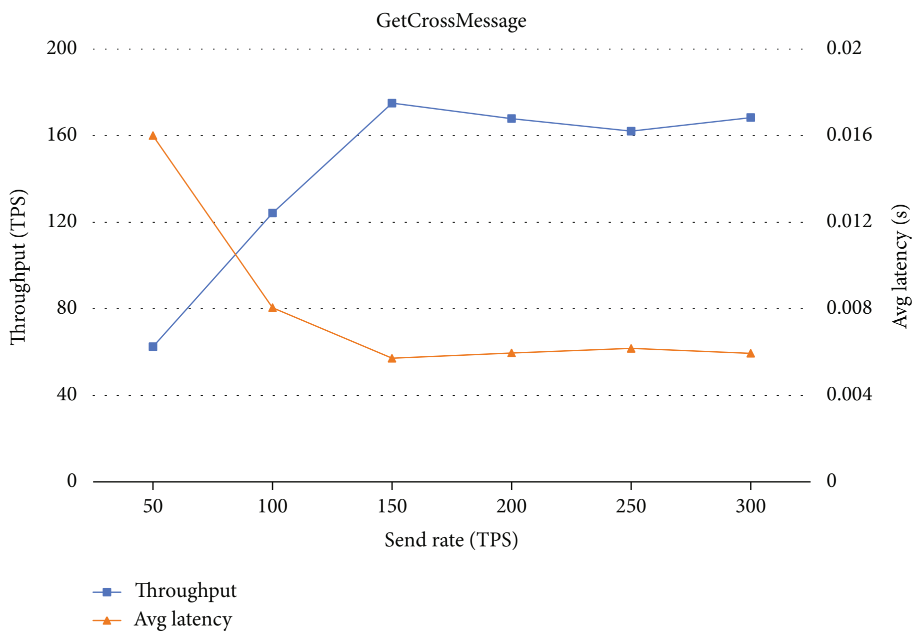
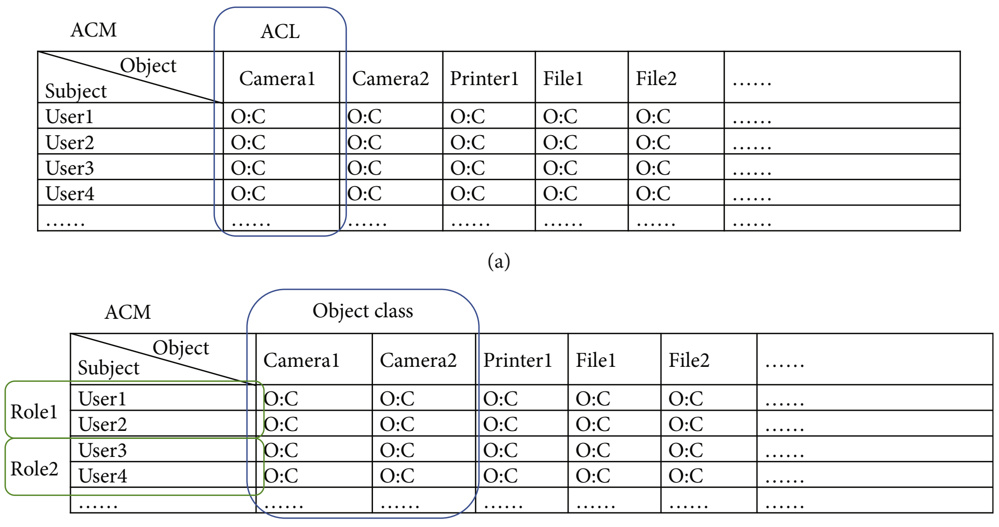


FIGURE 6: Invoke GetCrossMessage chaincode performance.

in the RBAC model, permissions (PRMS) represent the operational permission for roles to access the resources, which can be divided into operations and objects. The objects can be categorized to some object classes and the role-to-object access control rules can be transformed to be role-to-object class rules. In the ABAC model, both subjects and objects are represented by a set of attributes and the corresponding attribute values. Permissions consist of the object descriptors and operations, and the authorizations are defined between subject descriptors and object descriptors, or consist of the attribute conditions on the subject or object. As described above, ABAC policies can be turned to the RBAC rules, and thus turned to the role-to-object class rules.

Considering the characteristics of various kinds of access control mechanisms, we propose to ensure the interoperability of access control rules based on the object categories. In the Energy Internet scenario, we unify the objects among domains based on the resource types, treating the classes of the resources as the objects to be controlled. Each class of the resources is no longer specifically subdivided. For example, all the cameras will be treated as the same class. As a result, if a user with a role in Domain *A* can access the Camera class, he can access the cameras in Domain *B* after the authentication and authorization in Domain *A*. (Note that for some resources with special access control requirements, access control can be performed independently). In summary, we propose a cross-domain authorization mechanism based on blockchain, roles


ACM
ACL
Object class
Role1
Role2

| Subject \ Object | Camera1 | Camera2 | Printer1 | File1 | File2 | …… |
|---|---|---|---|---|---|---|
| User1 | O:C | O:C | O:C | O:C | O:C | …… |
| User2 | O:C | O:C | O:C | O:C | O:C | …… |
| User3 | O:C | O:C | O:C | O:C | O:C | …… |
| User4 | O:C | O:C | O:C | O:C | O:C | …… |
| …… | …… | …… | …… | …… | …… | …… |

(a)

| Subject \ Object | Camera1 | Camera2 | Printer1 | File1 | File2 | …… |
|---|---|---|---|---|---|---|
| User1 | O:C | O:C | O:C | O:C | O:C | …… |
| User2 | O:C | O:C | O:C | O:C | O:C | …… |
| User3 | O:C | O:C | O:C | O:C | O:C | …… |
| User4 | O:C | O:C | O:C | O:C | O:C | …… |
| …… | …… | …… | …… | …… | …… | …… |

(b)

Figure 7: From CapBAC to role and object class.

ACM

| Role \ Object class | Camera | Tablet | Printer | FileClass1 | FileClass2 | …… |
|---|---|---|---|---|---|---|
| Role1 | O:C | O:C | O:C | O:C | O:C | …… |
| Role2 | O:C | O:C | O:C | O:C | O:C | …… |
| Role3 | O:C | O:C | O:C | O:C | O:C | …… |
| Role4 | O:C | O:C | O:C | O:C | O:C | …… |
| …… | …… | …… | …… | …… | …… | …… |

Figure 8: Example of role and object class based access control model.

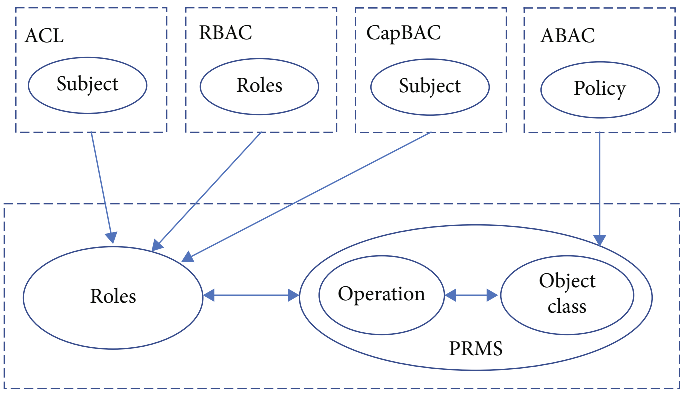


Figure 9: Role and object class based cross-domain authorization.

and object categories. In the initial phase, the authorization centers of each domain negotiate the object classes and the role classes according to the access control methods in their respective domains. First, the objects are categorized to various classes which is a many-to-many mapping, i.e., an object can be regarded as different classes. Then the object classes are authorized to the negotiated roles, and the authorization rules are written to the blockchain. As shown in Figure 8, this is essentially an access control mechanism based on roles and controlled object classes.

To make cross-domain authorization, the access control policies of each heterogeneous domain can be converted into RBAC access control rules using the mapping methods described above. As shown in Figure 9, users can then be granted permissions based on the role-object class based authorization rules negotiated and written to the blockchain.

In the following, we describe the details of our cross-domain authorization scheme including initial phase, registration, intra-domain authorization, and cross-domain authorization.

*Initial phase*. As described above, the authorization centers of each domain negotiate the object classes, the role classes according to the access control methods in their respective domains, and transform their own access control policies to the role and object class based rules. The transformation information is written to the blockchain.

*Registration*. When a user registers in its own domain, the role assignment procedure is added at the end of the registration phase of the authentication process described in the Section III to complete the registration.

*Intra-domain authorization*. Upon receiving the authorization request from user $U_{A_i}$, the authentication server $AS_A$ first calls the lookup algorithm of the cuckoo filter to check whether the certificate $Cert_{U_{A_i}}$ exists in the filter, i.e., $U_{A_i}$ is not revoked. Then, $AS_A$ authorizes $U_{A_i}$ according to the access control mechanism of Domain $A$.

*Cross-domain authorization*. If a user $U_{A_i}$ in Domain $A$ wants to be authorized by the authorization server $AS_B$ in Domain $B$, they execute an interactive protocol as follows:

(1) $U_{A_i}$ and $AS_B$ first complete the cross-domain authentication process as described above

(2) $AS_B$ obtains the access control rules for $U_{A_i}$ by coverting the role and object class based access control rules in the cross-domain authorization token to that of its own access control mechanism according to the transformation in the initial phase

(3) The authorization is completed

## 5. Conclusion

In this paper, based on the cross-chain technology of the blockchain and the cuckoo filter, we propose a multi-blockchain-based cross-domain authentication scheme supporting heterogeneous domains for the Energy Internet. In our scheme, a cross-chain authentication architecture is established and the users coming from different domains can independently perform the authentication with the help of the deployed chaincodes. The cuckoo filter is deployed on the blockchain as chaincodes and used to effectively manage the user certificate for the user registration, query, and revocation. We analyze the security of our scheme and design experiments to analyze the performance of our scheme such as the costs of user revocation and cross-domain authentication. Experimental results show the effectiveness and feasibility of our scheme. Basd on the authentication scheme, we propose a cross-domain access control model based on roles and object classes, by assuming that the heterogeneous domains have the same kinds of resources, treating the object classes as controlled objects and then applying the role-based access control to the object classes. It should be pointed out that we only implemented our scheme on the Hyperledger Fabric platform, and we leave it a future work to implement our scheme on other blockchain platforms. Besides, we will try to construct cross-domain authentication and authorization schemes based on the trusted execution environment.

## Data Availability

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

## Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

## Acknowledgments

This work was supported by the State Grid Shandong Electric Power Company Science and Technology Project funded "Research on Smart IoT Active Security Protection and Simulation and Detection Technology for Energy Internet" (5206002000V5).

## References

[1] T. Bei, D. Shi, F. Li, B. Chen, and Y. Yuan, "Research on security assessment scheme to smart grid based on digital twin," *Shandong Electric Power*, vol. 49, pp. 25–30, 2022.

[2] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Advances in Cryptology*, CRYPTO 1984, G. R. Blakley and D. Chaum, Eds., pp. 47–53, Springer, Berlin, Heidelberg, 1985.

[3] D. Zhang, J. Le, X. Lei, T. Xiang, and X. Liao, "Exploring the redaction mechanisms of mutable blockchains: a comprehensive survey," *International Journal of Intelligent Systems*, vol. 36, no. 9, pp. 5051–5084, 2021.

[4] N. Szabo, "Formalizing and securing relationships on public networks," *First Monday*, vol. 2, no. 9, 1997.

[5] B. Vaidya, D. Makrakis, and H. T. Mouftah, "Security mechanism for multi-domain vehicle-to-grid infrastructure," in *2011 IEEE Global Telecommunications Conference - GLOBECOM 2011*, pp. 1–5, Houston, TX, USA, 2011.

[6] W. Libing, J. Wang, K.-K. R. Choo, Y. Li, and D. He, "An efficient provably-secure identity-based authentication scheme using bilinear pairings for ad hoc network," *Journal of Information Security and Applications*, vol. 37, pp. 112–121, 2017.

[7] X. Cao, W. Kou, and D. Xiaoni, "A pairing-free identity-based authenticated key agreement protocol with minimal message exchanges," *Information Sciences*, vol. 180, no. 15, pp. 2895–2903, 2010.

[8] S. Benzarti and B. Triki, "Drone authentication using id-based signcryption in LoRaWAN network," in *Intelligent Systems Design and Applications*, ISDA 2019. Advances in Intelligent Systems and Computing, P. Siarry, K. Ma, and A. Kaklauskas, Eds., pp. 205–216, Springer, Cham, 2021.

[9] M. Shen, H. Liu, L. Zhu et al., "Blockchain-assisted secure device authentication for cross-domain industrial iot," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 5, pp. 942–954, 2020.

[10] C. Fromknecht, D. Velicanu, and S. Yakoubov, "A decentralized public key infrastructure with identity retention," Cryptology ePrint Archive, 2014, Report 2014/803.

[11] A. Ouaddah, A. A. El Kalam, and A. A. Ouahman, "Fairaccess: a new blockchain-based access control framework for the internet of things," *Security and cCommunication Networks*, vol. 9, no. 18, pp. 5943–5964, 2016.

[12] W. Wang, N. Hu, and X. Liu, "Blockcam: a blockchain-based cross-domain authentication model," in *2018 IEEE Third International Conference on Data Science in Cyberspace (DSC)*, pp. 896–901, Guangzhou, China, 2018.

[13] H. Wang, S. Wanyan, G. Shuxian, and Y. Jia, "Research on blockchain technology in power system," *Shandong Electric Power*, vol. 46, pp. 8–12, 2019.

[14] Y. Yao, X. Chang, J. Misic, V. B. Misic, and L. Li, "BLA: blockchain-assisted lightweight anonymous authentication for distributed vehicular fog services," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 3775–3784, 2019.

[15] S. Guo, F. Wang, N. Zhang, F. Qi, and X. Qiu, "Master-slave chain based trusted cross-domain authentication mechanism in IoT," *Journal of Network and Computer Applications*, vol. 172, article 102812, 2020.

[16] H. Zhang, X. Chen, X. Lan, H. Jin, and Q. Cao, "BTCAS: a blockchain-based thoroughly cross-domain authentication scheme," *Journal of Information Security and Applications*, vol. 55, article 102538, 2020.

[17] Y. Hao and L. Qing, "Block chain based distributed resources participating in power market transactions under ubiquitous power internet of things," *Shandong Electric Power*, vol. 47, pp. 42–48, 2020.

[18] G. Ali, N. Ahmad, Y. Cao et al., "xDBAuth: Blockchain based cross domain authentication and authorization framework for internet of things," *IEEE Access*, vol. 8, pp. 58800–58816, 2020.

[19] X. Zhu, J. Zheng, B. Ren, X. Dong, and Y. Shen, "Microthingschain: blockchain-based controlled data sharing platform in multi-domain iot," *Journal of Networking and Network Applications*, vol. 1, no. 1, pp. 19–27, 2021.

[20] X. Ma, W. Ma, and X. Liu, "A cross domain authentication scheme based on blockchain technology," *Acta Electronica Sinica*, vol. 46, no. 11, pp. 2571–2579, 2018.

[21] "Hyperledger fabric," https://www.hyperledger.org/use/fabric.

[22] R. Pagh and F. F. Rodler, "Cuckoo hashing," *Journal of Algorithms*, vol. 51, no. 2, pp. 122–144, 2004.

[23] B. Fan, D. G. Andersen, M. Kaminsky, and M. D. Mitzenmacher, "Cuckoo filter: practically better than bloom," in *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies*, pp. 75–88, New York, NY, USA, 2014.

[24] C. Ma, X. Kong, Q. Lan, and Z. Zhou, "The privacy protection mechanism of hyperledger fabric and its application in supply chain finance," *Cybersecur*, vol. 2, 2019.

[25] B. Gwak, J.-H. Cho, D. Lee, and H. Son, "TARAS: Trust-aware role-based access control system in public internet-of-things," in *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pp. 74–85, New York, NY, USA, 2018.

[26] M. Amoon, T. Altameem, and A. Altameem, "RRAC: role based reputed access control method for mitigating malicious impact in intelligent IoT platforms," *Computer Communications*, vol. 151, pp. 238–246, 2020.

[27] H. Nasiraee and M. Ashouri-Talouki, "Anonymous decentralized attribute-based access control for cloud-assisted IoT," *Future Generation Computer Systems*, vol. 110, pp. 45–56, 2020.

[28] M. Gupta, F. M. Awaysheh, J. Benson, M. Alazab, F. Patwa, and R. Sandhu, "An attribute-based access control for cloud enabled industrial smart vehicles," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 6, pp. 4288–4297, 2021.

[29] Y. Nakamura, Y. Zhang, M. Sasabe, and S. Kasahara, "Capability-based access control for the internet of things: an ethereum blockchain-based scheme," in *2019 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, Waikoloa, HI, USA, 2019.

[30] G. Batra, V. Atluri, J. Vaidya, and S. Sural, "Enabling the deployment of ABAC policies in RBAC systems," in *Data and Applications Security and Privacy XXXII. DBSec 2018*, Lecture Notes in Computer Science, F. Kerschbaum and S. Paraboschi, Eds., pp. 51–68, Springer, Cham, 2018.

[31] Y. Liu, L. Qinghua, S. Chen et al., "Capability-based IoT access control using blockchain," *Digital Communications and Networks*, vol. 7, no. 4, pp. 463–469, 2021.