Computers, Materials & Continua
DOI: 10.32604/cmc.2022.027281
Article

Tech Science Press

CWoT-Share: Context-Based Web of Things Resource Sharing in Blockchain Environment

Yangqun Li[1,2,*], Jin Qi[1,2], Lijuan Min[1,2], Hongzhi Yang[1,2], Chenyang Zhou[1,2] and Bonan Jin[3]

[1]School of Internet of Things, Nanjing University of Posts and Telecommunications, Nanjing, 210003, China
[2]Jiangsu Key Laboratory of Broadband Wireless Communication and Internet of Things, Nanjing University of Posts and Telecommunications, Nanjing, 210003, China
[3]Information VIII, University of Würzburg, Würzburg, 97074, Germany
*Corresponding Author: Yangqun Li. Email: yqli@njupt.edu.cn
Received: 13 January 2022; Accepted: 11 March 2022

Abstract: Web of Things (WoT) resources are not only numerous, but also have a wide range of applications and deployments. The centralized WoT resource sharing mechanism lacks flexibility and scalability, and hence cannot satisfy requirement of distributed resource sharing in large-scale environment. In response to this problem, a trusted and secure mechanism for WoT resources sharing based on context and blockchain (CWoT-Share) was proposed. Firstly, the mechanism can respond quickly to the changes of the application environment by dynamically determining resource access control rules according to the context. Then, the flexible resource charging strategies, which reduced the fees paid by the users who shared more resources and increased the fees paid by users who frequently used resources maliciously, were used to fulfill efficient sharing of WoT resources. Meanwhile, the charging strategies also achieve load balancing by dynamic selection of WoT resources. Finally, the open source blockchain platform Ethereum was used for the simulation and the simulation results show that CWoT-Share can flexibly adapt to the application environment and dynamically adjust strategies of resource access control and resource charging.

Keywords: Web of things; resource sharing; smart contract; context; billing strategy

1 Introduction

The physical environment data and services provided by Internet of Things (IoT) for applications through heterogeneous sensing devices are called resources [1,2]. This heterogeneity makes IoT application development difficult [3,4]. In order to simplify the IoT applications development, the Web of Things (WoT) adopts standardized Web technology to connect and control various physical and virtual things [5] to realize the Web sharing of various resources of things. Therefore, WoT technology can be utilized to build an open and shared WoT business ecosystem by integrating heterogeneous

IoT technologies on a unified, open and standardized Web platform, thus reducing the cost of IoT application development and deployment.

In the WoT open environment, the secure sharing and efficient billing of resources have become important issues which are described by following smart tourism application scenarios. Before visiting a scenic spot, tourist Alice wants to know the current information of the scenic spot, such as the crowdedness, comfortableness, air quality, etc. Bob, a tourist in the scenic spot, can obtain such information through his own perception device such as mobile phone and share it with Alice. On the one hand, Bob needs to share such resources to Alice safely and quickly without obtaining Alice's information and establishing a trust relationship with Alice in advance. On the other hand, Bob decides whether to share resources according to his actual situation, such as whether he is still in the scenic spot or whether the remaining power of the mobile device is sufficient. However, the manual operation will inevitably lead to cumbersome operations, thereby hindering resource sharing. Therefore, a mechanism that does not require an authority center and can dynamically adapt to changes in the environment has become an important requirement for WoT resource sharing.

The key factors for the success of WoT applications include such technologies as data security, privacy protection [6], and policy-based access control [7,8]. The smart tourism application scenario fully illustrates that in the open and complex WoT ecosystem, how to share resources securely and efficiently is also very important for WoT applications. At present, resource access control technology is a means to implement the secure sharing of resources. However, WoT resources are distributed in a wide range and applied in different domains and areas. The limitations of the sensing device in terms of memory, CPU and battery make its availability change quickly. Therefore, the traditional centralized access control mechanism can no longer meet the security requirements of the WoT open ecosystem [9–11].

In order to solve this problem, we proposed a trusted and secure sharing mechanism of WoT resources based on contextual information by using smart contract in blockchain environment. The main works are as follows.

1) A framework for WoT resource access control based on smart contracts in blockchain environment was proposed to implement fine-grained access control of WoT resources.
2) Resource access control rules were dynamically generated by using the context information of the requester and provider, so that WoT applications can respond quickly and flexibly to the changes.
3) Reward and punishment mechanism based on flexible billing strategies was adopted to meet the demand for efficient sharing of resources. The flexible billing strategies reduced the fees paid by users who shared more resources and increased the fees paid by users who frequently used resources maliciously, which implement the dynamic selection of WoT resources, thus achieving load balancing.
4) The blockchain platform Ethereum was used for the simulation and the simulation results show that the system can flexibly adapt to the application environment, user behavior and dynamically adjust strategies of resource access control and resource billing.

2 Related Research

The traditional centralized resource access control mechanism lacks flexibility and scalability. In response to these shortcomings, the IoT resource access control framework [12–14] based on the OAuth2.0 [15] standard provides flexibility in authorization for applications. However, these methods still authenticate users by centralized authorization center, which cannot meet the WoT resource access control requirements in distributed environment.

Blockchain technology has anonymity and distribution characteristic [16–19], and its distributed ledger technology makes it suitable for various applications that rely on real and reliable transaction data storage and data sharing between transaction parties [20,21], such as blockchain-based power monitoring and trading [22]. Meanwhile, the smart contract can be deployed on the blockchain to implement transactions based on the contract rules [23,24]. Therefore, in a distributed environment, blockchain and smart contracts have gradually been applied in IoT resource access control. For example, compared with the authorization mechanism in [12–14], the access control model of FairAccess [7] uses Token to represent access right, and transfers Tokens between the provider and the requester in a distributed manner through blockchain network. However, this mechanism does not consider how the resource is charged and how the control strategy changes in a dynamic context environment. Blockchain for edge of things (BEoT) architecture is used to provide safe surveillance data sharing in smart city by combining edge computing and blockchain technology. The smart contracts are deployed in the BeoT edge node for secured and trusted authentication [25]. Literature [26,27] propose a distributed access control mechanism for IoT resources management by using blockchain and smart contracts. Literature [27] also uses smart contracts to judge the legitimacy of user behaviors for punishments. This framework emphasizes the punishment of malicious behavior and lacks rewards for positive behaviors such as resources sharing. At the same time, it does not consider the context of the resource provider. The EdgeChain [28] framework implements the access control of the cloud resources based on the reputation of the edge IoT device by blockchain. Although EdgeChain bills centralized cloud resources based on the available resources, the request priority, and the number of resources requested, it cannot realize the billing of IoT resources of different owners. Therefore, EdgeChain cannot meet the requirements of the WoT open environment for the flexibility and scalability of the billing mechanism. In terms of resource sharing, literature [29] uses smart contract to implement access control of big data to meet the needs of big data sharing within the enterprise. This method lacks resource billing strategy. IoTShare [30] proposed a blockchain-based IoT resource sharing protocol which can realize distributed management of IoT devices, discovery of IoT resources, and billing operations for resource usage. However, IoTShare failed to realize the dynamic management of resource access control, and did not consider the reward and punishment mechanism.

A blockchain-based framework CWoT-Share is proposed here to realize trusted and secure sharing of WoT resources based on context in a distributed environment. Tab. 1 clarifies the characteristics of CWoT-Share by comparing it with other mechanisms. In Tab. 1, solution [8] uses centralized method to implement role-based access control of WoT resources, which can only be implemented in smaller-scale applications. Compared with the literature [27], the dynamic adaptation mechanism proposed here also involves the resources sharing. Moreover, CWoT-Share achieves fine-grained control in two aspects. On the one hand, the control granularity for WoT resource operations is similar to that in t [31]. On the other hand, CWoT-Share can determine its access control rules based on the context of the resource provider and requester. For example, the requester's access can be dynamically controlled based on such information as the resource provider's CPU usage, available memory, network bandwidth, and current requests number.

Table 1: Comparison of CWoT-share with existing scheme

Scheme	Distribution	Flexibility	Adaptability	Flexible billing	Fine-grained	Complexity
ASAM4IoT [26]	√	√	×	×	√	Average
BAC_SC4IoT [27]	√	√	Medium	×	×	Average
EdgeChain [28]	√	√	×	Average	√	High
FairAccess [7]	√	√	×	×	√	High
IoTShare [30]	√	√	×	×	×	High
RBAC-SC [32]	√	×	×	×	×	High
RBAC4WoT [8]	×	×	×	×	×	High
CWoT-Share	√	√	√	√	√	Low

3 Trusted and Secure Sharing of WoT Resources

This section firstly describes the resource sharing abstract model, then gives the mechanism of CwoT-Share, and finally describes the implementation of the efficient sharing of resources and load balancing between WoT resources based on the reward and punishment mechanism.

3.1 Resource Sharing Abstract Model

Fig. 1 shows the resource sharing abstract model. Resource owners share WoT resources with the outside world. The resource is used by other users with the support of the infrastructure. When resources are shared, users should access them in accordance with access control rules. At the same time, a fee should be paid for the usage. Efficient provision and use of resources are constrained by the context of stakeholders. Each element in Fig. 1 is described in detail as follows.

1) Resource owner: It shares resources and obtains benefits from the sharing.
2) Resource user: It requests the shared resources for its requirements.
3) Fee: The cost paid for using the shared resource.
4) Permission: Constraint rules for resource operations.
5) Infrastructure: Common functions required for the resources sharing.
6) Context information [33]: The status of shared resources, owners, users, and infrastructure in the WoT open ecosystem, which affects whether resources can be shared securely and efficiently.

It can be seen from the model that the requirements for implementing end-to-end security of WoT resources sharing mainly includes (SR, Security Requirement):

SR1: The authenticity of resources shared by resource providers.

SR2: Resource users should use resources in a secure, efficient, and controllable manner.

SR3: The security of the infrastructure on which the resource depends.

A solution for SR2 is proposed here and the detailed description is as follows.

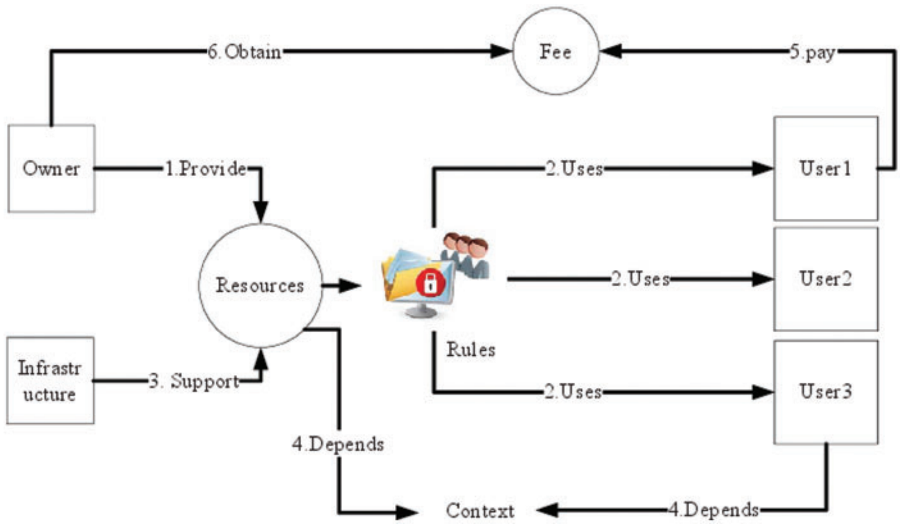


Figure 1: The abstract model of resource sharing

3.2 *Trusted and Secure Resource Access Control*

3.2.1 *Trusted and Secure Resource Access Control Mechanism*

Fig. 2 shows the implementation of CWoT-Share. The elements include:

(1) WoT environment. A variety of sensors and gateways responsible for managing these sensor devices constitute the WoT environment. The gateway encapsulates the data and services provided by the sensing device as WoT resources to implement the Web interconnection, and is also responsible for the interaction between WoT resources. Access control rules of shared WoT resources are updated in the smart contract by their owner and then queried by the requesters through the gateway.

(2) Local certificate management module. The resource stakeholder obtains public and private key corresponding to its identity through this module for data encryption and signing during resource sharing. Meanwhile, there is no need to know the identity information of the resource stakeholder.

(3) Blockchain network and smart contracts. The blockchain network stores the transaction information of the shared resource, and automatically implements the consistency of this information among nodes. Smart contracts are used for the management of resource access control strategies. When WoT resources update their own access control rules, the smart contract will be verified by miners in the blockchain and stores them throughout the network.

(4) Dynamic context processing. Fig. 3 shows the dynamic context model used in CWoT-Share. It includes two roles: requester and provider. The context of the requester includes behavioral state, geographic location, access time, and its own resource sharing situation, which changes dynamically as the resource is accessed in the process of resource sharing. The requestor's behavior may be malicious, abnormal, and normal. The context of the provider includes status

of CPU, memory, and battery, as well as active self-management capability. When the CPU is overloaded, the memory or the battery is exhausted, it will stop providing service. Active self-management capability means the owner also can decide when to provide or stop service. Based on the context model, the provider can adjust the shared resource access rules according to its own context combined with the behavior of the requester. For example, when the request received by the WoT resource exceeds its capacity, the module will adjust the access control rules of the resource. When the requester's behavior is abnormal, the mechanism updates the access control rules to restrict user behavior. The function of this module is handled by the WoT environment deployed in various places.

5) Resource access control rules. According to the dynamic context model defined in Fig. 3, the rule describes who can access the resource and how to access it. As shown in Tab. 2 the rules consist of three parts: provider context, requester context and corresponding actions. For example, When the provider's available capabilities of CPU, memory, and battery are 35%, 65% and 80% respectively, and the behavior of the requester is normal, the humidity resource will accept the request. When the battery is only 20% left and the current request number is 10, although the behavior is normal, the resource refuses the request and allows it to try again after a period of time.

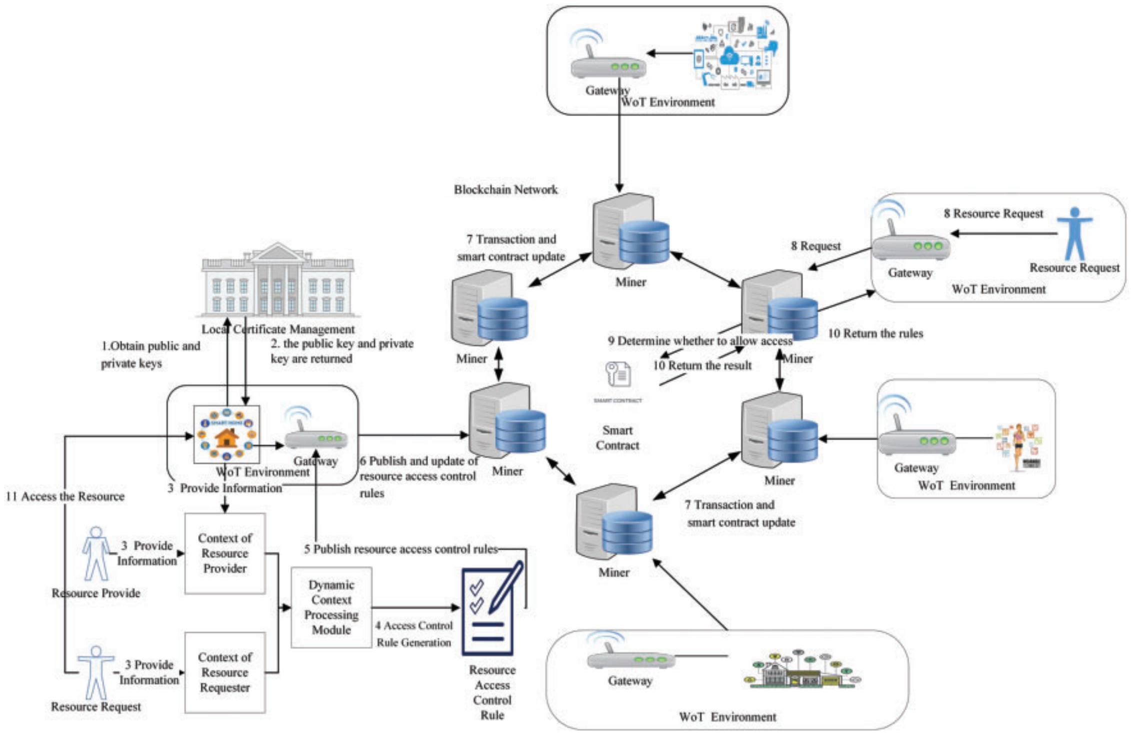

Gateway
WoT Environment
Blockchain Network
7 Transaction and smart contract update
Miner
8 Request
8 Resource Request
Resource Request
Gateway
WoT Environment
Local Certificate Management
1.Obtain public and private keys
2. the public key and private key are returned
9 Determine whether to allow access
10 Return the rules
10 Return the result
Smart Contract
Gateway
WoT Environment
6 Publish and update of resource access control rules
11 Access the Resource
3 Provide Information
3 Provide Information
Context of Resource Provider
Resource Provide
5 Publish resource access control rules
Dynamic Context Processing Module
4 Access Control Rule Generation
3 Provide Information
Context of Resource Requester
Resource Request
Resource Access Control Rule
7 Transaction and smart contract update
Gateway
WoT Environment

Figure 2: WoT resource sharing based on dynamic context information

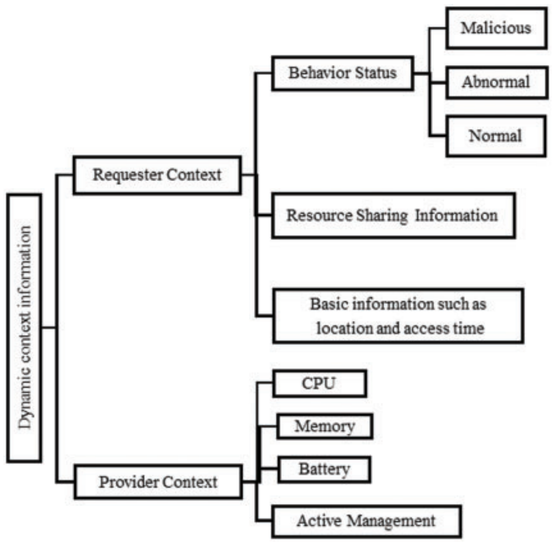

Dynamic context information
Requester Context
Behavior Status
Malicious
Abnormal
Normal
Resource Sharing Information
Basic information such as location and access time
Provider Context
CPU
Memory
Battery
Active Management

Figure 3: Dynamic context model

Table 2: WoT resource access control based on dynamic context information

Provider context			Requester context		Access control rule			
Resource	Operation	Available resource	Public key	Behavior	Rights	Fee	Valid period	Frequency
Humidity sensor	GET	cpu:35% memory:65% concurrent request:5 battery:80%	Provided	Normal	ALLOW	1	2021-8-1	100
Humidity sensor	GET	cpu:80% memory:50% concurrent request:10 battery:20%	Provided	Normal	DENY		Try again in 30 min	
LED	PUT	Normal	None		DENY			
LED	PUT	cpu:35% memory:75% concurrent request:5 battery:60%	Provided	abnormal	DENY		Try again in 10 h	
Temperature sensor	SUBSCRIBE	Normal	Provided	Normal	ALLOW	1	1 min	20

The implementation of CWoT-Share is shown in Fig. 2, and the details are as follows:

Step 1: Shared WoT resources are registered by the owner to the certificate management center to obtain public and private key which are used for signing the transaction.

Step 2: Certificate Authority returns the public/private key.

Step 3: The context processing module obtains the context of the stakeholders to judge user behavior and resource status.

Step 4: According to the result of Step 3, resource access control rules are generated.

Step 5: The gateway in the WoT environment submits the generated rules to the smart contract.

Step 6: The access control rules are stored and updated in the smart contract.

Step 7: The blockchain network verifies the transaction to store the rules in the smart contract distributedly and consistently.

Step 8: The resource requester accesses the blockchain network through the local gateway and the request is forwarded by the gateway to the smart contract.

Step 9: The smart contract reads the access control rules and determines whether the request is allowed to access the resource.

Step 10: The judgment result is sent to the requester.

Step 11: If the request is allowed, the requester sends request to the resource. During the request process, the dynamic context information of both parties is processed by the context module to update the access control rules in time. Resource access records are stored for subsequent context processing.

Fig. 4 shows the dynamic update process of WoT resource access control rules based on context information. The update process includes two ways. The first way is that when the access is successful, the provider updates the access control rules according to the access result. The process is as follows: Steps 1 to 4 show the process of accessing resources according to the current access control rules. In step 5, when the access is successful, the provider publishes the requester's access status and the provider's resource sharing information on the blockchain network. The information includes: the frequency of requesting the resource, the duration of the request, whether the requester's behavior is legal, and the times of the requester's abnormal behavior and so on. When the provider is also a requester of other resources, its resource sharing information can be used to calculate the fees it should pay. In step 6, the provider updates its own state information to the context processing module. In step 7 and step 8, the dynamic context processing module obtains the requester's access history and resource sharing information respectively for context processing. Step 9 generates new access control rules based on the information from step 6 to step 8, and publishes it on the blockchain network for synchronization (steps 10–11). The second way is implemented by step 12 to step 14. The provider regularly updates the resource access control rules to the blockchain network through the context processing module according to its own status or active management requirements.

3.2.2 Trusted and Secure Access Control Implementation

CWoT-Share generates rules to determine whether the resource can be accessed according to the context and historical behavior of the resource requester. There are three types of requester behavior: normal, abnormal and malicious. If the requester's behavior is normal, it is allowed to access shared resources. If it is an abnormal behavior caused by operations such as excessive frequent request, the request will be forbidden for a certain period of time. If it is malicious, the request will be rejected. Algorithm 1 presents the process of behavior judging based on the requester's context and access behavior.

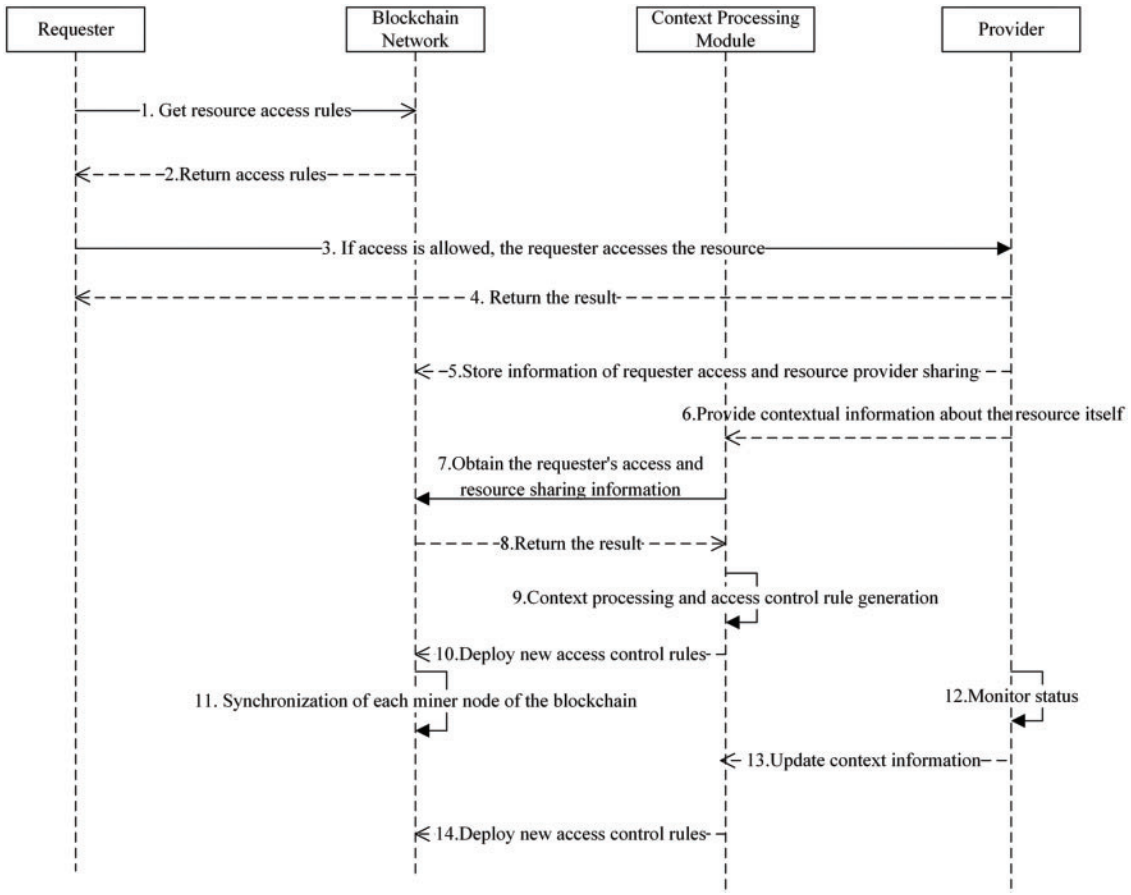

Requester
Blockchain Network
Context Processing Module
Provider
1. Get resource access rules
2.Return access rules
3. If access is allowed, the requester accesses the resource
4. Return the result
5.Store information of requester access and resource provider sharing
6.Provide contextual information about the resource itself
7.Obtain the requester's access and resource sharing information
8.Return the result
9.Context processing and access control rule generation
10.Deploy new access control rules
11. Synchronization of each miner node of the blockchain
12.Monitor status
13.Update context information
14.Deploy new access control rules

Figure 4: Blockchain-oriented dynamic context processing mechanism

Algorithm 1: User behavior judgment algorithm based on context information. BeHJudge (REQ, RES_{avail})

```
Input: REQ (Resource request), $RES_{avail}$ (number of available resource).
Output: User behavior: normal, abnormal, malicious.
(1) $REQ.PBK = (URL_{resq})_{prikey}||PBK_{req}$
(2) $PBK_{key}$ = GetPublicKey(REQ.PBK);
//Step (1) uses the requester's private key to encrypt the requested resource URL, and then send the information together with the requester's public key to the resource provider. In step (2), the provider uses the public key to obtain the URL address, and then compares it with the local resource URL. If it matches, the public key is considered valid, and if it does not match, the public key is considered invalid.
(3) if ( $PBK_{key}$ == NULL || $PBK_{key}$ == FAIL )
(4)     return MALICIOUS;
(5) end if
```

(Continued)

Algorithm 1: Continued

```
(6) f = Frequence(REQ, t); //Get the number of times of requests for the resource during the time t.
(7) $S_{oper}$ = GetOperation(REQ); /Get the operation performed by the request on the resource
(8) if (!allowed($S_{oper}$)&&(f > threshold))
//If the operation is not allowed and the access frequency exceeds the threshold, the request is determined to be malicious.
(9)     return MALICIOUS;
(10) end if
(11) if (!allowed($S_{oper}$) || f > threshold)//When the operation is not allowed or the frequency exceeds the threshold, the behavior is determined to be abnormal
(12)     return ABNORMAL;
(13) end if
(14) if (!isAuthedbyContractor(REQ)) //If the request is not allowed by the access control rules in the smart contract, it will be considered abnormal.
(15) return ABNORMAL;
(16) end if
(17) ...... //Execute other behavior judgment rules
(18) return NORMAL;
```

Based on the Algorithm 1, Algorithm 2 implements resource access control based on dynamic context.

Algorithm 2: Resource access control method ACByContext() based on dynamic context information.

```
Input: resource request REQ, resource available number $RES_{avail}$.
Output: Resource request response $R_{esp}$ and update of access control list.
(1)  Receive external WoT request;
(2) acl = getACLbySC(); //Get list of access control rules from smart contract.
(3) permitted = validate(REQ, $RES_{avail}$); // Call the validate method of the smart contract to determine whether the request is allowed.
(4) if (permitted )
(5)     if (BeHJudge(Req,$RES_{avail}$) == NORMAL)
(6)         Resp = invokeResource(REQ, $RES_{avail}$);
(7)         return Resp;
(8)     end if
(9)  end if
(10) UpdateACL(SC, Req, $RES_{avail}$); (11) goto (1);
```

3.3 *Resource Sharing Billing*

3.3.1 Resource Sharing Billing Mechanism

The billing mechanism uses the reward and punishment mechanism to encourage more resources to be shared. If the requester uses shared resources, but shares very few resources, the system will increase the requester's fee of using shared resources accordingly. The implementation of the reward and punishment mechanism is represented by formula (1), Wherein, X_t refers to the number of times the requester successfully accessed the shared resource in t minutes, S_t refers to the number of times the requester's shared resource was used by other users in t minutes, R_τ represents the number of resources accessed by the requester at time τ, C_τ refers to the number of available system resources of the shared resource at time τ. The l in formula (1) refers to the service level of the resource requester. κ indicates the importance of resource sharing. When $\kappa = 1$, it means that the fee of resource is inversely proportional to the number of resources successfully accessed by the requester within a certain period of time.

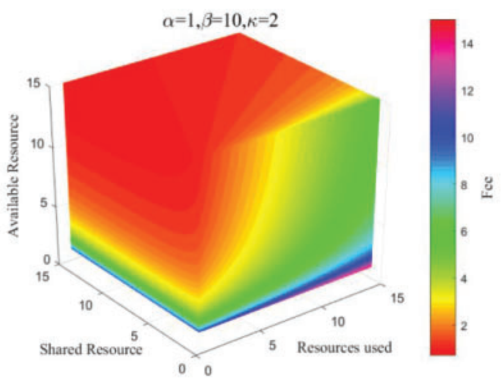
$$f(t, \tau, l) = \alpha * \frac{X_t}{1 + \kappa S_t} + \beta * \frac{R_\tau}{C_\tau} + \eta^l \tag{1}$$

In formula (1), α, β, η respectively represent the weight coefficient of different billing items. α indicates the importance of the proportion of the number of requested resources to the number of shared resources. β indicates the importance of the proportion of the number of consumed system resources to the currently available system resources. η indicates the priority of the resource requester (>1).

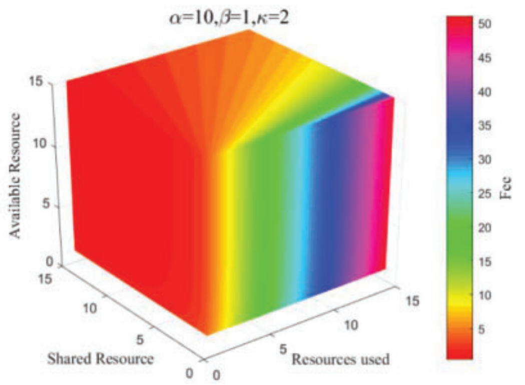
Fig. 5 shows the fee change considering various context, resource sharing, and resource access times, that is, the fee of the first two billing items in formula (1). The requester's priority and service level are determined under certain circumstances. Therefore, the simulation does not consider the third term of formula (1). Fig. 5a shows the change of resource fee when $\alpha = 1$, $\beta = 10$, $\kappa = 2$. Fig. 5b shows the change of resource fee when $\alpha = 10$, $\beta = 1$, $\kappa = 2$. Comparing Fig. 5a with Fig. 5b, it can be seen that in Fig. 5a, the number of currently available resources has a greater impact on resource fees. In Fig. 5b, the number of shared resources and the number of times the resource has been accessed have a greater impact on the fees. It can also be seen from Figs. 5a, 5c and 5d that with the continuous increase of the value, the correlation between resource fee and the number of currently available resources continues to increase.

In the WoT ecosystem, different providers may provide resources with the same function at the same time, and the fee of each resource will constantly change according to context. According to this, the billing mechanism can be used to achieve load balancing between resources. For example, different providers share the same functional resources S1 and S2. When S1 has less available resources than S2, its fee is higher. Therefore, the requester will use S2. As S2's requests number continue to increase, the available resources become less and less. Hence, the fee of S2 will increase. Meanwhile, S1's resource requests continue to be completed, its available resources gradually increase, and hence its fees will also be reduced accordingly. Therefore, new requests will choose S1 again, and the number of requests for S1 continues to increase again. Hence, this billing mechanism efficiently implemented the dynamic load balance between S1 and S2. This mechanism does not require complex protocols and can also be adjusted on demand. Hence it is easy to implement and has good flexibility.

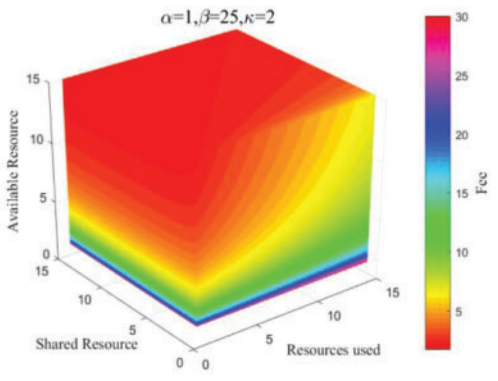
Using the Matlab R12 platform, the mechanism is simulated and analyzed from the number of available resources and the number of concurrent requests. Figs. 6–8 shows the simulation results. The number of resources consumed by each request is a random number between 1–13. The rising part in the figure indicates that the consumed resources are released, and the falling part indicates that part of resources are consumed.



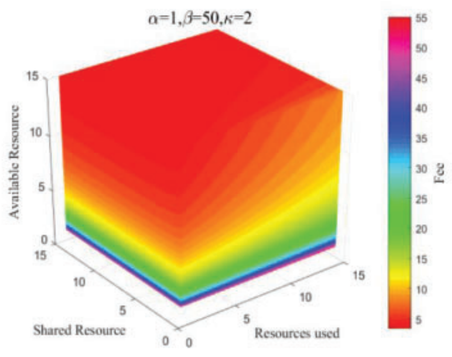
(a) α =1, β =10, κ =2



(b) α =10, β =1, κ =2

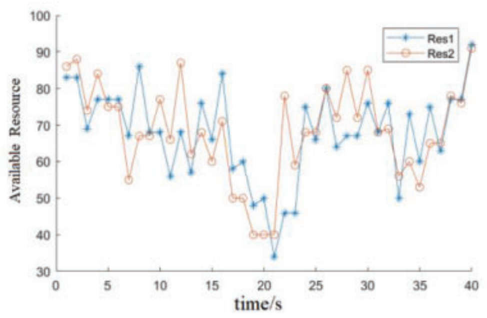


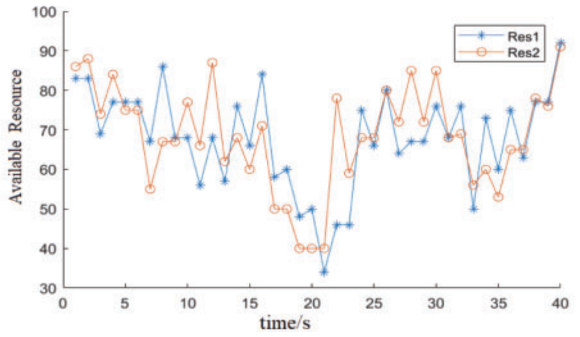
(c) α =1, β =25, κ =2



(d) α =1, β =50, κ =2

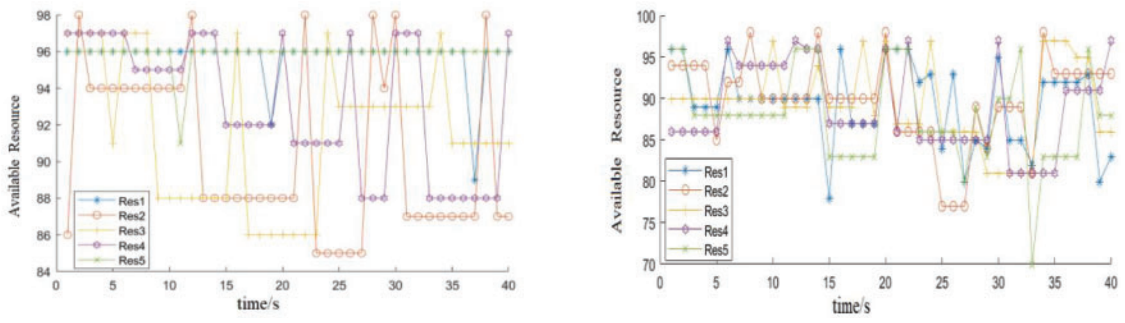
Figure 5: WoT resource billing mechanism





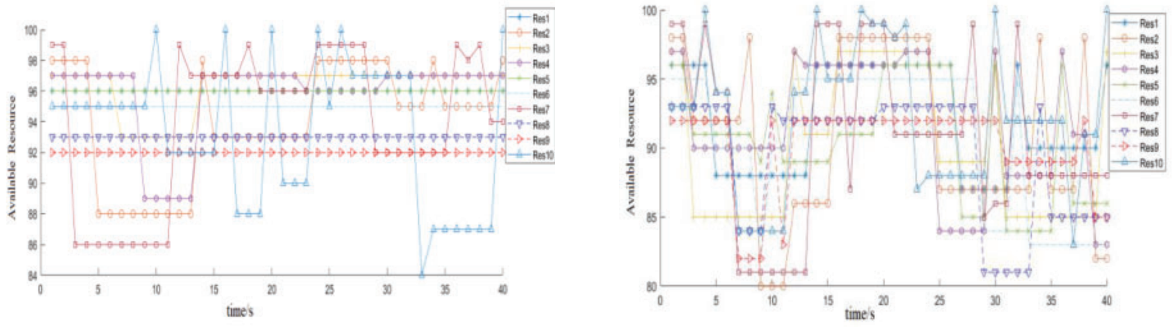
(a) Concurrent number is 1 (b) Concurrent number is a random number between 1-5

Figure 6: WoT resource selection, when the number of the same resource is 2



(a) Concurrent number is 1 (b) Concurrent number is a random number between 1-5

Figure 7: WoT resource selection, when the number of the same resource is 5



(a) Concurrent number is 1 (b) Concurrent number is a random number between 1-5

Figure 8: WoT resource selection, when the number of the same resource is 10

Fig. 6 shows the simulation result when the number of the same resources is 2. Figs. 6a and 6b respectively show the case where the number of concurrent requests is 1 and a random number between 1–5. Fig. 6a shows that the user requests resources at time 2. At this time, the number of available resources of Res2 is more than that of Res1. When only considering the number of available resources, the fee of Res2 is lower than that of Res1. Therefore, the requester will send request to Res2, and hence the available resource number of Res2 will decrease accordingly. At time 12, Res1 has more available resources than Res2, so it will provide services to users. It can also be seen in the figure that as the resources are consumed and released, the number of available resources of Res1 and Res2 is also dynamically changing, and the fee will also change between high and low. Therefore, they will provide services to users in turn and achieve load balancing. Fig. 6b shows the situation when there are multiple concurrent requests. At time 3, 13, and 16, multiple requests were generated concurrently. Res2 has more available resources than Res1. Therefore, Res2 firstly serves some concurrent requests. In simulation, concurrent requests are placed in a global queue, and services are provided by polling the available resources. When the number of available resources of Res2 is less than that of Res1, Res1 will provide services for the request.

Figs. 7a and 7b show the case where the number of the same resource is 5, and the number of requests is 1 and a random number between 1–5, respectively. Figs. 7a and 7b illustrate the situation where resources Res1-Res5 take turns to serve the request. For example, in Fig. 7b, at time 3, Res1 and Res5 have the same number of available resources. At this time, there are two concurrent requests applying for different numbers of resources. One of the two resources is randomly selected to serve one of the requests. Another resource serves the remaining requests. At time 14, there are 4 concurrent requests. Res2 has the largest number of available resources 98, which serves one of concurrent requests (set as request 1). Request 1 applied for 8 resources. Both Res4 and Res5 have 95 available resources, which serve request 2 and request 3 respectively. The number of resources consumed by these two requests is 9 and 12 respectively. The number of available resources of Res3 is 93. It serves request 4 which consumes 4 resources. At this moment, both Res2 and Res1 have the largest number of available resources, 90, and Res1 is randomly selected to serve Request 5 which consumes 12 resources. Request 1–5 occupy 5, 5, 5, 3, and 1 second(s) respectively. At time 16, the consumed resources of Res1 are released. The resources released at this time are the resources consumed by the request at time 6 and the request at time 14. The Res1 resource requested at time 6 is occupied by 9 s. The Res1 resource requested at time 14 is occupied by 1 s. Therefore, it can be seen from the figure that the number of Res1 available resources at time 16 is more than that at time 14.

Figs. 8a and 8b show the case where the number of the same resource is 10, and the number of requests is 1 and a random number between 1–5, respectively. Fig. 8a shows how the resources provide services for a single request. In Fig. 8a, there are enough resources to choose and a few requests, hence the Res5, Res8, and Res9 did not yet provide services during the simulation. But this does not affect the results. As the number of concurrent requests increases, these resources will provide services in succession. The analysis in Fig. 8b is similar to that in Fig. 7b. For example, at time 3, 10, and 30, the number of requests is 5, 3, and 4, respectively. From the above analysis, it can be seen that the resource billing mechanism can achieve load balancing among resources, avoiding the situation where the performance of the provider is rapidly degraded due to excessive service provision.

3.3.2 Resource Sharing Billing Implementation

Algorithm 3 describes the implementation of the context-based billing mechanism for shared resources.

Algorithm 3: Algorithm ResouceBillandSelect() for context-based shared resource billing and load balancing.

```
Input: REQ (Resource request), RS (The collection of same resources shared by the open business ecosystem), SC (smart contract interface), $\alpha, \beta, \eta, \kappa$ (weight coefficient).
Output: The selected shared resource and the current fee of the resource.
(1) Receive resource access request REQ;
(2) The record of the requester's access to the shared resource is read from the smart contract to calculate the number of successful access $X_t$ within the time t;
(3) Read the successfully accessed times $S_t$ of the resource shared by the requester during the time t from the smart contract;
(4) Calculate the value of $X_t/S_t$;
(5) for i in {RS}
(6)     Calculate the ratio of the resources applied by the request to the available resources of $RS_i$ at the current moment $\tau$;
(7)     Calculate the fee corresponding to the requested service level;
(8)     Calculate the $RS_i$ fee f($RS_i$, REQ) according to formula (1);
```

(Continued)

Algorithm 3: Continued

```
(9) end for
(10) Sort resources according to their fees from low to high;
(11) The request uses the ACByContext() method to access the shared resource with the lowest cost;
(12) goto (1);
```

4 Simulation Analysis

4.1 Simulation Environment Settings

The simulation environment is constructed by using the Ethereum platform. The blockchain network is simulated by 8 Ethereum clients running on the host. The host is ThinkPad T470 and the running environment is: Windows 10 64-bit operating system, 8G memory, Intel Core i5-7200U 2.71 GHz CPU.

4.2 Simulation Results and Analysis

4.2.1 The Impact of the Number of Miners on the Performance of the Blockchain Network

Fig. 9 analyzes the impact of the number of miners on the response time of WoT resource requests in the simulation environment. The number of miners affects the response time from two aspects. Firstly, the more miners there are, the faster the block generation and verification will be accordingly. Secondly, as the number of miners increases, miners will use more local system resources, thus affecting the performance of the blockchain network. Fig. 9 shows the response time of the blockchain when the number of miners is 3–7. It can be seen from Fig. 9 that in terms of different numbers of miners, as the number of requests increases, the response time of the blockchain changes accordingly. Meanwhile, the response time did not decrease with the increase in the number of miners. In the simulation environment, when the number of miners is 4, the blockchain network has the best response performance.

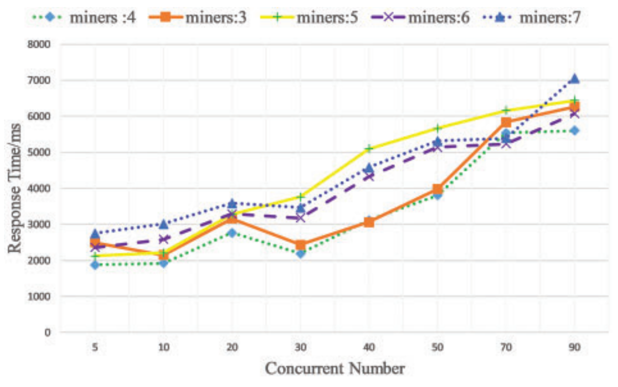


Figure 9: The impact of the number of miners

4.2.2 Performance Analysis of Access Control Rules Updating

In CWoT-Share, the update time of access control rules includes context processing time, block construction time, and data synchronization time of blockchain nodes. Fig. 10 shows the performance analysis of dynamic access control rule update when the number of miners is 4. It can be seen from Fig. 10, that as the number of access control rules increases, the system response time also increases.

When the number of rules is 45 and 80 respectively, the required update time increases faster. It can also be seen from the figure that in the simulation environment, when the number of updated rules is less than 40, the update time is about 2–3 s, which is relatively stable. The rule update should be as fast as possible, so that the access control rules read from the smart contract are the latest.

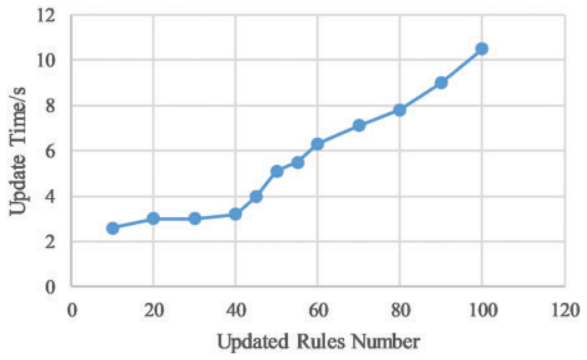
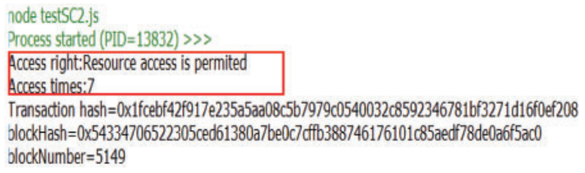


Figure 10: Performance analysis of dynamic rules update

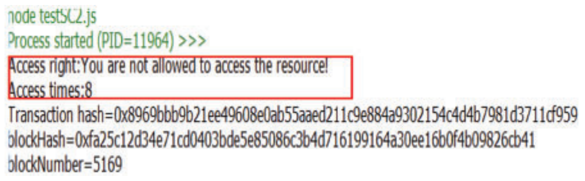
4.2.3 Access Control Capability Analysis

Figs. 11–13 describes the simulation implementation of the CWoT-Share mechanism in the Ethereum. In Fig. 11, when user requests a resource, the number of times she/he has used the resource is 7, but it does not exceed the prescribed threshold. According to the rules, the user will be allowed to use the resource. In Fig. 12, when the user makes a new request, the number of times he accesses the resource is 8, which will exceed the threshold. Therefore, the request is rejected. Fig. 13 shows the resource access control event AccessControlled activated by the smart contract.



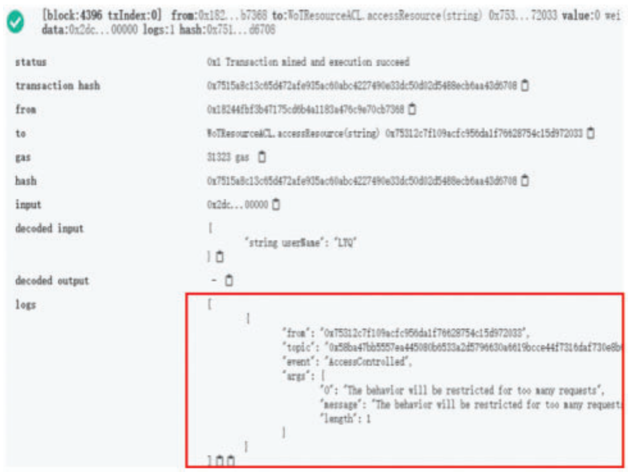
```
node testSC2.js
Process started (PID=13832) >>>
Access right:Resource access is permited
Access times:7
Transaction hash=0x1fcebf42f917e235a5aa08c5b7979c0540032c8592346781bf3271d16f0ef208
blockHash=0x54334706522305ced61380a7be0c7cffb388746176101c85aedf78de0a6f5ac0
blockNumber=5149
```

Figure 11: Allowed to access WoT resources



```
node testSC2.js
Process started (PID=11964) >>>
Access right:You are not allowed to access the resource!
Access times:8
Transaction hash=0x8969bbb9b21ee49608e0ab55aaed211c9e884a9302154c4d4b7981d3711cf959
blockHash=0xfa25c12d34e71cd0403bde5e85086c3b4d716199164a30ee16b0f4b09826cb41
blockNumber=5169
```

Figure 12: Denied to access WoT resources



```
[block:4396 txIndex:0] from:0x182...b7368 to:WoTResourceACL.accessResource(string) 0x753...72033 value:0 wei
data:0x2dc...00000 logs:1 hash:0x751...d6708

status              0x1 Transaction mined and execution succeed
transaction hash    0x7515a8c13c65d472afe935ac60abc4227490e33dc50d02d5488ecb6aa43d6708
from                0x18244fbf3b47175cd6b4a1183a476c9e70cb7368
to                  WoTResourceACL.accessResource(string) 0x75312c7f109acfc956da1f76628754c15d972033
gas                 31323 gas
hash                0x7515a8c13c65d472afe935ac60abc4227490e33dc50d02d5488ecb6aa43d6708
input               0x2dc...00000
decoded input       {
                          "string userName": "LTQ"
                    }
decoded output      -
logs                [
                          {
                                "from": "0x75312c7f109acfc956da1f76628754c15d972033",
                                "topic": "0x58ba47bb5557ea445080b6533a2d5796630a6619bcce44f7316daf730e8b
                                "event": "AccessControlled",
                                "args": {
                                      "0": "The behavior will be restricted for too many requests",
                                      "message": "The behavior will be restricted for too many request
                                      "length": 1
                                }
                          }
                    ]
```

Figure 13: Trigger of WoT resource access control event

4.2.4 *Analysis of the Cost of Trusted and Secure Sharing*

In the blockchain, the execution of the smart contract method requires corresponding fee, which is described by the gas consumed. Tab. 3 shows the method implemented in the smart contract and the corresponding gas consumption. Among them, the context update method requestContextUpdate and the rule update method ruleUpdate will cost more gas, because they implement complex functions. However, the methods of user register userRegister, resource register resourceRegister, and rule acquisition getRule consume less gas.

Table 3: Cost analysis of smart contract methods

Smart contract method	Function	Consumed gas
userRegister()	User register	25735
resourceRegister()	Resource register	34102
accessResource()	Access resource	31323
userContextUpdate()	Update user context	127641
requestContextUpdate()	Update requester context	138536
ruleUpdate()	Update access control rules	150383
getRule()	Get access control rules	68742
costCalculate()	Calculate the fee	160237

5 Conclusion

A trusted and secure sharing mechanism (CWoT-Share) for WoT resources based on contextual information in a blockchain environment was proposed. The main features of CWoT-Share include:

1) WoT resource access control strategies are dynamically generated to realize trusted and safe sharing of resources by utilizing the context of resource stakeholders. `

2) Based on the context of the resource provider and the resource sharing situation of the resource requester, the reward and punishment mechanism for WoT resource sharing is implemented. This mechanism reduces the fees paid by users who share more resources, increases the fees paid by users who share fewer resources, and punishes malicious users in terms of billing and access control to establish a good WoT resource sharing ecosystem.
3) The resource billing strategies can also implement the dynamic selection of WoT resources to achieve their systems load balancing.

CWoT-Share still has some shortcomings. For example, it stores part of the information in the blockchain, which will bring additional performance overload for blockchain network. In the future, this information will be stored off-chain, and resource sharing will be implemented by combining on-chain and off-chain technology. At the same time, future work will also combine lightweight blockchain technology for the IoT environment [34] to conduct large-scale resource sharing research and applications such as IoV (Internet of Vehicle) [35] and UAV-based traffic monitoring [36].

Funding Statement: This study is funded by "The National Natural Science Foundation of China (No. 61972211, No. 61771258)".

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] ITU-T, "Y.2060: Overview of the Internet of things," 2012. [Online]. Available: https://www.itu.int/rec/T-REC-Y.2060-201206-I (accessed on June 15th 2021).
[2] ITU-T, "Y.2063: Framework of the Web of things," 2012. [Online]. Available: https://www.itu.int/rec/T-REC-Y.2063/en. (accessed on June 15th 2021).
[3] Z. Y. Wu, "Research on Web of Things service environment architecture and key technologies," Ph.D. dissertation, College of Information and communication, Beijing University of Posts and Telecommunications, Beijing, China, 2013.
[4] M. A. Razzaque, M. Milojevic-Jevric, A. Palade and S. Clarke, "Middleware for internet of things: A survey," *IEEE Internet of Things Journal*, vol. 3, no. 1, pp. 70–95, Feb. 2016.
[5] W3C, "Web of Things (WoT) architecture," 2020. [Online]. Available: https://www.w3.org/TR/2019/CR-wot-architecture-20190516/ (accessed on July 5th 2021).
[6] X. R. Zhang, X. Sun, X. M. Sun, W. Sun and S. K. Jha, "Robust reversible audio watermarking scheme for telemedicine and privacy protection," *Computers, Materials & Continua*, vol. 71, no. 2, pp. 3035–3050, 2022.
[7] A. Ouaddah, A. A. Elkalam and A. A. Ouahman, "FairAccess: A new blockchain-based access control framework for the internet of things," *Security and Communication Networks*, vol. 9, no. 18, pp. 5943–5964, 2016.
[8] E. Barka, S. S. Mathew and Y. Atif, "Securing the Web of Things with role-based access control," in *Lecture Notes in Computer Science*, vol. 9084, Springer, Cham, pp. 14–26, 2015.
[9] Y. Zhang, B. Li, B. Liu, J. Wu, Y. Wang *et al.,* "An attribute-based collaborative access control scheme using blockchain for IoT devices," *Electronics*, vol. 9, no. 2, pp. 285, 2020.
[10] X. R. Zhang, W. F. Zhang, W. Sun, X. M. Sun and S. K. Jha, "A robust 3-D medical watermarking based on wavelet transform for data protection," *Computer Systems Science & Engineering*, vol. 41, no. 3, pp. 1043–1056, 2022.
[11] C. Xu, K. Wang, P. Li, S. Guo, J. Luo *et al.,* "Making big data open in edges: A resource-efficient blockchain-based approach," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 4, pp. 870–882, 2019.

[12] S. Cirani, M. Picone, P. Gonizzi, L. Veltri and G. Ferrari, "IoT-OAS: An OAuth-based authorization service architecture for secure services in IoT scenarios," *IEEE Sensors Journal*, vol. 15, no. 2, pp. 1224–1234, Feb. 2015.
[13] S. Sciancalepore, G. Piro, D. Caldarola, G. Boggia and G. Bianchi, "OAuth-IoT: An access control framework for the internet of things based on open standards," in *Proc. IEEE Symp. on Computers and Communications (ISCC)*, Heraklion, Greece, pp. 676–681, 2017.
[14] S. -R. Oh, Y. -G. Kim and S. Cho, "An interoperable access control framework for diverse IoT platforms based on OAuth and role," *Sensors*, vol. 19, no. 8, pp. 1884, 2019.
[15] OAuth Working Group, "The OAuth 2.0 authorization framework," 2012. [Online]. Available: https://tools.ietf.org/id/draft-ietf-oauth-v2-27.html (accessed on June 19th 2021).
[16] M. Wu, K. Wang, X. Cai, S. Guo, M. Guo *et al.,* "A comprehensive survey of blockchain: From theory to IoT applications and beyond," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8114–8154, 2019.
[17] T. M. Fernández-Caramés and P. Fraga-Lamas, "A review on the use of blockchain for the internet of things," *IEEE Access*, vol. 6, pp. 32979–33001, 2018.
[18] A. O. Almagrabi and A. K. Bashir, "Service-aware access control procedure for blockchain assisted real-time applications," *Computers, Materials & Continua*, vol. 67, no. 3, pp. 3649–3667, 2021.
[19] S. Hakak, W. Z. Khan, G. A. Gilkar, B. Assiri, M. Alazab *et al.,* "Recent advances in blockchain technology: A survey on applications and challenges," *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 38, pp. 82–100, 2021.
[20] X. Peng, J. Zhang, S. Zhang, W. Wan, H. Chen *et al.,* "A secure signcryption scheme for electronic health records sharing in blockchain," *Computer Systems Science and Engineering*, vol. 37, no. 2, pp. 265–281, 2021.
[21] J. Wang, C. Han, X. Yu, Y. Ren and R. S. Sherratt, "Distributed secure storage scheme based on sharding blockchain," *Computers, Materials & Continua*, vol. 70, no. 3, pp. 4485–4502, 2022.
[22] M. M. Ahmed, M. K. Hasan, M. Shafiq, M. O. Qays, T. R. Gadekallu *et al.,* "A peer-to-peer blockchain based interconnected power system," *Energy Reports*, vol. 7, pp. 7890–7905, 2021.
[23] S. Wu, Y. Chen, Q. Wang, M. Li, C. Wang *et al.,* "CReam: A smart contract enabled collusion-resistant e-auction," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 7, pp. 1687–1701, 2019.
[24] P. Wang, J. Meng, J. Chen, T. Liu, Y. Zhan *et al.,* "Smart contract-based negotiation for adaptive QoS-aware service composition," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 6, pp. 1403–1420, 2019.
[25] T. R. Gadekallu, Q. V. Pham, D. C. Nguyen, P. K. R. Maddikunta, N. Deepa *et al.,* "Blockchain for edge of things: Applications, opportunities, and challenges," *IEEE Internet of Things Journal*, vol. 9, no. 2, pp. 964–988, 2021.
[26] O. Novo, "Blockchain meets IoT: An architecture for scalable access management in IoT," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1184–1195, 2018.
[27] Y. Zhang, S. Kasahara, Y. Shen, X. Jiang and J. Wan, "Smart contract-based access control for the internet of things," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1594–1605, 2019.
[28] J. Pan, J. Wang, A. Hester, I. Alqerm, Y. Liu *et al.,* "EdgeChain: An edge-IoT framework and prototype based on blockchain and smart contracts," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4719–4732, 2019.
[29] A. D. Liu, X. H. Du, N. Wang and S. Z. Li, "Blockchain-based access control mechanism for big data," *Journal of Software*, vol. 30, no. 3, pp. 2636–2654, 2019.
[30] B. Hamdaoui, M. Alkalbani, A. Rayes and N. Zorba, "IoTShare: A blockchain-enabled IoT resource sharing on-demand protocol for smart city situation-awareness applications," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 10548–10561, 2020.
[31] S. Zhao, L. Yu and B. Cheng, "A real-time web of things framework with customizable openness considering legacy devices," *Sensors*, vol. 16, no. 10, pp. 1596, 2016.
[32] J. P. Cruz, Y. Kaji and N. Yanai, "RBAC-SC: Role-based access control using smart contract," *IEEE Access*, vol. 6, pp. 12240–12251, 2018.

[33] P. S. Gandodhar and S. M. Chaware, "Context aware computing systems: A survey," in *Proc. 2nd Int. Conf. on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)*, Palladam, India, pp. 605–608, 2018.
[34] Y. Liu, K. Wang, Y. Lin and W. Xu, "LightChain: A lightweight blockchain system for industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3571–3581, 2019.
[35] W. Sun, G. Z. Dai, X. R. Zhang, X. Z. He and X. Chen, "TBE-Net: A three-branch embedding network with part-aware ability and feature complementary learning for vehicle re-identification," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–13, 2021. [Online]. Available: https://doi.org/10.1109/TITS.2021.3130403.
[36] W. Sun, L. Dai, X. R. Zhang, P. S. Chang and X. Z. He, "RSOD: Real-time small object detection algorithm in UAV-based traffic monitoring," in *Applied Intelligence*, pp. 1–16, 2021. [Online]. Available: https://doi.org/10.1007/s10489-021-02893-3.