Computer Science and Information Systems 19(2):935–955 https://doi.org/10.2298/CSIS211220019Z

A Consortium Blockchain-Based Information Management System For Unmanned Vehicle Logistics

Manjie Zhai[1], Dezhi Han[1,*], Chin-Chen Chang[2], and Zhijie Sun[1]

[1] College of Information Engineering, Shanghai Maritime University, Shanghai, 201306, China
dzhan@shmtu.edu.cn

[2] The Department of Information Engineering and Computer Science, Feng Chia University, Taichung, Taiwan, 000400, China

Abstract. Unmanned vehicle (UDV) delivery technology can meet the special needs of users and realize efficient and flexible distribution of logistics orders. However, there are risks of order data leakage and tampering in the intelligent logistics distribution environment. To solve this problem, this paper designs and implements a system based on the Hyperledger Fabric blockchain platform. Based on the blockchain technology, the system adopts a distributed architecture to establish a secure and trustworthy logistics data management platform to achieve the integrity and traceability of data in the logistics process. The data dual-chain storage strategy is used to ensure the efficiency of data queries. Furthermore, four smart contracts including order management contract (OMC), access control management contract (ACC), access control policy management contract (ACPC), and environmental data management contract (EDC) are designed in combination with the attribute-based access control strategy. By triggering the smart contract, the controllable access of order data can be realized. Finally, two groups of experiments are designed to test the performance of the system. Experimental results show that the proposed system can maintain high throughput in a large-scale request environment under the premise of ensuring data security.

Keywords: Blockchain, Logistics, Attribute-based access control, Hyperledger Fabric, Smart contract.

1. Introduction

With the rapid development of the Internet of things (IoT), the market scale of the logistics industry is gradually expanding. Compared to traditional delivery methods, unmanned vehicle (UDV) delivery can meet the delivery needs of many specific scenarios[26]. For example, during the epidemic period, UDV can realize the non-contact distribution of living materials and medicines, reducing the risk of contact infection. In addition, UDV can also be used for street mail distribution, extreme weather distribution, and other scenarios. However, the existing intelligent logistics platform lacks a complete and reliable credit guarantee system, and problems such as product counterfeiting, loss, and package loss continue to occur[39],[22],[18]. The logistics business is composed of many participants, involving a wide area and a long time span. It is difficult for core institutions to meet the information management and controllable access in the IoT environment. Logistics information is facing the risk of leakage and tampering, and information security is becoming more and more prominent[27],[7],[15],[19].

Blockchain is a distributed data storage and management technology based on a public-key encryption mechanism. The blocks are linked by a hash algorithm as a chain to ensure the integrity and traceability of data, which provides conditions for optimizing logistics management and information traceability[1]. Each distributed node of the blockchain realizes data communication through p2p network and consensus algorithm to ensure data consistency between nodes and secure mutual trust sharing of information[11],[24]. As an open-source consortium chain platform, Hyperledger Fabric inherits the characteristics of the blockchain and also provides a more efficient consensus mechanism, higher throughput, and support for multiple channels[13],[29],[17]. Access control technology is an important means to protect resources and has been widely used in various industries. The traditional access control technology belongs to centralized access control, which has problems such as single-point failure and poor scalability. Traditional access control technology does not meet the access requirements of logistics platform information distribution and mobility[30],[16],[23]. Attribute-based access control (ABAC) is an extension of role-based access control(RBAC). The algorithm extracts the attributes of the user, resource, permission, and environment respectively and flexibly combines these attributes. Finally, the management of permissions is transformed into the management of attributes, which can solve the problem of fine-grained access control that is difficult to be solved by traditional access control and the problem of agent dynamic authorization access in a large-scale environment[8],[31],[20]. ABAC can effectively solve the controllable access of the information in the logistics process, and has a wide range of application scenarios.

To solve the above-mentioned drawbacks, this paper designs a blockchain-based UDV logistics information management system, which mainly focuses on the security and privacy protection of users' data, access control, in the logistics platform. The main contributions of this paper are summarized as follows:

1. We combine Hyperledger Fabric with a distributed architecture to establish a secure and trustworthy logistics data management system to implement the integrity and traceability of data in the logistics.
2. We propose a data storage strategy with dual chains that divides data into order data and real-time environment data, improving the efficiency of data queries.
3. Combined with the ABAC access control algorithm, this paper designs four smart contracts, including the order management contract (OMC), access control management contract (ACC), access control policy management contract (ACPC), and environmental data management contract (EDC). Among them, the OMC manages order information, the ACC manages user access requests, the ACPC manages access policies set by administrators, and the EDC manages environmental data. The ABAC access control policy is deployed on blockchain using a smart contract, and the effective management of logistics data is achieved by triggering a smart contract to ensure controlled access to logistics data.

The rest of this paper is organized as follows. Section 2 introduces the related work. Section 3 introduces the overall design of the system in detail. Section 4 provides the detailed experiment process. Section 5 provides the safety analysis. Finally, Section 6 summarizes the paper.

2. Related Work

2.1. Hyperledger Fabric

Hyperledger Fabric is an open-source distributed ledger platform for enterprise applications [3],[2], using modular structures to provide extensible components. The Fabric includes four types of nodes, namely CA node, Client node, Peer node, and Order node. Client nodes are used to interact with Peers and implement operations such as adding, deleting, and modifying blockchain networks. CA nodes can generate or cancel member identity certificates, providing unified management for the digital certificates of member nodes. Peer nodes are used to store blockchain ledger and chaincode and the application program updates the ledger and checks the chaincode by connecting Peer nodes. The Order node will receive the transactions sent by the Peer node and sort them according to certain rules, and finally package the transactions in a certain order into blocks.

Chaincode: Chaincode is the code deployed on Fabric network nodes to operate and manage the data in the distributed ledger, and is called to implement the smart contract[6].

Channel: Channel in Fabric isolate blockchain data from different organizations, each channel has a proprietary account, and organizational nodes in different channels cannot access directly. Each Peer node in the network needs to be identified by the administrator to join the channel and each communication party must be authenticated and authorized to trade on the channel. This mechanism effectively ensures the security of transaction and improves the utilization of data storage space and parallel processing efficiency[37].

2.2. Attribute-based access control model

Access control technology implements authorized access to resources according to predefined access policies and prevents unauthorized information disclosure by controlling the access rights of the subject to the object[38]. Access strategy is the set of attributes required for specific operations on data resources. The binary group can be expressed as Eq. (1).

$$ABACPolicy \leftarrow \langle AttrSet, Rule \rangle. \tag{1}$$

where *AttrSet* represents the attribute set of the strategy and can be represented as a set of quaternions, as shown in Eq. (2).

$$AttrSet = \langle AS, AO, AE, AP \rangle. \tag{2}$$

AS represents the attribute of the subject, including the identity, role, location, certificate, etc. *AO* represents the resource attributes, including the identity, location, department, type, etc. *AE* represents the attribute of the environment, which is used to judge whether the policy is satisfied the request. *AP* represents the attribute of permission, which means the operation of the subject on the object, such as write, modify, delete, etc. The *Rule* represents a set of rules that can be expressed as $Rule = \{rule_1, rule_2, \ldots, rule_n\}$, $n \geq 1$, $rule_n$ denotes the nth rule. *Rule* can be expressed as a set of quaternions as Eq. (3) and Eq. (4).

$$Rule = Result = F(.). \tag{3}$$

$$F(Attr(S_i), Attr(O_i), Attr(E_i), Attr(P_i)) \rightarrow \{Permit, Deny\}. \tag{4}$$

Eq. (4) indicates that the subject with the authorization attribute S_i performs an access action with the attribute value P_i to the object O_i in the context of the environment attribute E_i.

2.3. Elliptic curve digital signature method

The elliptic curve digital signature algorithm is based on the elliptic curve algorithm and signature algorithm, which is mainly used to create digital signatures for data. It has the characteristics of identifiability and unforgeability, ensuring that the authenticity of the data is verified without destroying the security of the data [34],[28]. The elliptic curve digital signature method is constructed based on Eq. (5).

$$y^2 = (x^3 + a \times x + b) \; mod \; p. \tag{5}$$

The parameters of the elliptic curve are (a, b, p, n, G), where a and b are the parameters of the curve equation, p is the base of the modular operation, n is the number of points on the curve, the parameter G represents the selected reference starting point that can be any point on the curve, and the key pair is (SK, PK), where SK is the private key and PK is the public key.

Signature phase:

1. Generate a random number k, which satisfies the condition: $1 \leq k \leq n-1$.
2. Compute $p = k \times G$, the abscissa of p is R.
3. Compute $r = R$ **mod** n. if $r = 0$, return to Step 1.
4. Calculate the hash $H(m)$ of message m and convert the obtained value into a large integer z.
5. Calculate s using $s = k^{-1}(z + SK \times r)$ **mod** p . if $s = 0$, return to Step 1.
6. (r, s) is the signature of the message m.

Verification phase:

1. Calculate $H(m)$ and convert it to integer Z.
2. Calculate w, where $w = s^{-1}$ **mod** n.
3. Calculate $u_1 = (z \times w)$ **mod** n, $u_2 = (r \times w) \; mod \; n$.
4. Calculate $X = (x_1, \; y_1) = u_1 G + u_2 PK$. If $X = 0$, the verification is wrong; otherwise, convert the abscissa of X to R and calculate $v = R$ **mod** n.
5. If $v = r$, the verification passes.

3. System overall design

This paper proposed a UDV logistics information management system based on the alliance chain, solving the security and privacy protection issues of data in the logistics platform and ensuring the controllable access of logistics data. This section introduces the overall design of the system. Tab. 1 lists the key symbols used in this paper. Fig. 1 is a framework diagram of the system model. The infrastructure layer is the Fabric underlying module. The data storage layer includes order data storage module and real-time environmental data storage module. The smart contract layer includes four smart contracts and the corresponding state database key-value pair storage. The user interaction layer includes the UDV registration and management module, order allocation module, real-time navigation, and ABAC module.

Table 1. Key symbol description.

Notations	Description
$Cert_i$	Certificate of entity
PK_i	Public key of $user_i$
SK_i	Private key of $user_i$
$Info_i$	Identity information of $entity_i$
$TimeStamp_i$	Timestamp of $entity_i$
Sign(.)	Signature algorithm
ABACPolicy	Attribute-based access control policy
OMC	Order management contract
ACC	Access control management contract
ACPC	Access control policy management contract
EDC	Environmental data management contract
UDV	Unmanned vehicle
CA	Authority
DockerImage	Docker image
Config(.)	Config file of the node
Start(.)	Start of the node
Install(.)	Install of the chaincode

3.1. Entity description

1. **UDV**: It is the transportation mode for delivering goods. UDV should register its identity information with CA and obtain order distribution qualification before distribution.
2. **User**: There are two types of users including data requester and order owner.
3. **CA**: As the system administrator, CA is responsible for system initialization and entity authentication[4],[35]. Any entity that wants to join the blockchain should register its identity information and obtain PK and SK, and only entities certified by CA can perform operations such as uploading and querying data.
4. LC: The LC reasonably allocates logistics orders to UDV according to *OrderDistribute()* in OMC.
5. Sensor: The Sensor is an important configuration on the UDV, which can sense vehicle position change, navigate the route, improve communication reliability, and ensure low latency interaction of measurement information.

3.2. Data storage

This section introduces the double-chain storage strategy in detail. Logistics data is divided into order data and real-time environment data. Order data refers to the information of receiving and delivering users and commodity-related information in the logistics, and the fields included are shown in Tab. 2. On the blockchain, the order number of the order data is taken as the key value, the order creation time, the sender's and receiver's information, the order information, and other fields as the value of the map after JSON serialization.

Environmental data refers to real-time data such as road conditions collected by sensor equipment. When the UDV encounters a failure, the surrounding road condition status and the information of the nearby UDV can be obtained by querying the environmental

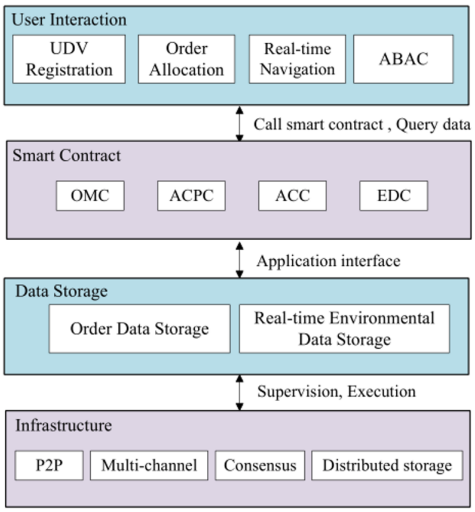


Fig. 1. System architecture diagram.

Table 2. Order data field description.

Notations	Description
OrderId	Order number
OrderTime	Order create time
SenderAddress	Sender address
SenderPhone	Sender telephone
ReceiverName	Receiver name
ReceiverAddress	Receiver Address
ReceiverPhone	Receiver telephone
ReceiverReput	Receiver reputation
OrderState	Order state
Private	Order private
Urgent	Order urgency

data information to prepare for the order handover[12]. The data contained in the environmental information is shown in Tab. 3. Environmental data includes the IOT module and the message queue module. The IoT module mainly through sensor devices to collect environmental data such as vehicle speed, road condition status, and nearby UDV information. To reduce the error of the data, the original data collected 30 times per second are taken as a group, and its average value is stored in the environmental data link. The smart contract saves the data by calling the relevant interface. If the data can be successfully written into the ledger, the corresponding message will be dequeued. otherwise, the data will be temporarily stored in the queue for the next data uplink. The environmental data uplink process is shown in Fig. 2.

Table 3. Environment data field description.

Notations	Description
IotId	Internet of things device number
VehDataTime	Vehicle data acquisition time
VehDataAddress	Vehicle data acquisition address
VehSpeed	Vehicle speed
RoadState	Road condition status
NeighUdvNum	Number of nearby UDV
NeighUdvState	Status of nearby UDV
NeighUdvSpeed	Speed of nearby UDV

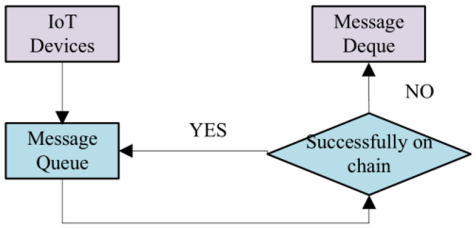


Fig. 2. Environment data uplink process.

3.3. Smart Contract

OMC. OMC is to manage the order data according to the user's request, including the request situation of order addition, order cancellation, order information change, and order handover. The methods included in the OMC are as follows.

AddOrder(): When a user request to add an order, *CheckOrder()* in the OMC is triggered to check the rationality of the order request. If the order request is reasonable, the order will be released and the order information will be added to the status database, and the order operation record will be written to the blockchain. The pseudocode of *AddOrder()* is shown in Algorithm 1, where the *OrderId* is stored as the key, and *OrderTime, SenderAddress, SenderPhone* and other attributes are stored as value in the blockchain.

QueryOrder(): It realizes the function of querying the details of order data according to *OrderId*.

UpdateOrder(): In some special cases, the information needs to be modified after the user places an order, which can be processed according to the order status. If the order is not issued, *UpdateOrder()* can be called to complete the information modification operation; otherwise, it cannot be modified.

DeleteOrder(): when the user sends a request to delete an order, *CheckOrder()* is first triggered to check the rationality of the order information, and then *QueryOrder()* is called to check the order status. If the order is not issued, *DeleteOrder()* is called to delete the relevant information in the order data chain. Otherwise, the wireless sensor device should send the withdrawal or cancellation command to the UDV in time, and change the order status information to undelivered.

CheckOrder(): It is used to check the reasonableness of key field information when the users submit order requests.

Algorithm 1: OMC.*AddOrder()*

Input: *Request*(*AddOrder*)
Output: *Success* or *Error*

$APIstubChaincodeStub \leftarrow Invoke()$;
if $CheckOrder(.) == False$ **then**
  return $Error(Illeagleorder)$;
end
$Id \leftarrow sha256(UserId, OrderId)$;
$ans \leftarrow APIstub.PutState(Id)$;
if $ans \neq null$ **then**
  return $Success$;
end
return $Error$;

Algorithm 2: OMC.*DeleteOrder()*

Input: *Request*(*DeleteOrder*)
Output: *DeleteResult* or *Error*

$APIstubChaincodeStub \leftarrow Invoke()$;
if $CheckOrder(.) == False$ **then**
  return $Error$;
end
$ans == QueryOrder(OrderId)$;
if $ans == NoIssue$ **then**
  $DeleteOrder(OrderId)$;
end
$Revoke(OrderId)$;

Checkudvinfo(): It is used to verify the identity and status information of UDV applying for distribution qualification.

OrderDistribute(): After the UDV is qualified for order distribution, the LC will distribute the order reasonably, then update the order status in time and store it in the order chain.

OrderCharge(): Before the order is delivered, the shipper and the user will negotiate the charging criteria. If the goods are lost, damaged, or the order is delivered over time during the delivery process, the compensation should be paid by the agreed compensation criteria. After the user successfully obtains the goods, *OrderCharge()* will be called to complete the payment operation.

ACPC. The ACPC is to provide management functions for the established attribute-based access control strategy. Combining the characteristics of logistics data and the attribute-based access control model, this paper defines the attribute characteristics as follows:

P = {*AS, AO, AP, AE*},
AS = {*UserId, Role, PK, UserGroup*},
AO = {*OrderId, Singer, SignOrderData*},
AP = {*OrderPermission*},
AE = {*CreateTime, EndTime, Address, CurrentAccess*}.

AS mainly refers to the attribute of users who access data resources, including *UserId, Role, PK*, and *UserGroup*, where *UserId* is the unique identification of user information.

AO mainly refers to the order data stored in the order chain, including *OrderId, Singer*, and the signature of order data *SignOrderData*.

AP includes the order data access permission attribute *OrderPermission*.

AE refers to the environmental attribute of the order requester, including *CreateTime, EndTime, Address*, and *CurrentAccess*.

Administrator formulates access policies based on user, resource, operation, and environment attributes, as shown in Eq. (6).

$$f(AS,\ AO,\ AP,\ AE) \rightarrow ABACPolicy \tag{6}$$

Add the policy to the SDB by calling *AddPolicy()* in the ACPC. *VerifyPolicy()* is used to verify whether the access policy formulated by the administrator is a legal policy, *QueryPolicy()* supports querying policies through *AS* or *AO* features, and *UpdatePolicy()* is invoked to update the operation records on the blockchain. When the specified access policy exceeds the specified period, *DeletePolicy()* is called to delete.

Algorithm 3: ACC.*CheckAccess()*

Input: *ABACRequest*
Output: *Success* or *Error*

$\langle A_uS,\ A_uO,\ A_uE\rangle \leftarrow GetAttrs(ABACRequest)$;
$P = \langle P_1,\ P_2,\ ...,\ P_n\rangle \leftarrow ACPC.QueryPolicy(A_uS,\ A_uO)$;
if $P == Null$ **then**
  return *Error("This is an illeagle policy")*;
end
for P *in* $\langle P_1,\ P_2,\ ...,\ P_n\rangle$ **do**
  $\langle ...,\ A_PP,\ A_PE\rangle \leftarrow P$;
  if $Value(A_PP) == deny$ **then**
    $continue$;
  end
  $ans \leftarrow QueryOrder()$;
end
if $ans \neq Null$ **then**
  return *Success("OK")*;
end
return *Error("UnAccess")*;

ACC. ACC is used to verify whether the user's data access request meets the access control policy formulated by the administrator, including the following methods.

Auth(): The main function is to use the user public key to verify the authenticity of user identity. When the user sends data access requests, the user will use the ACC's public key to encrypt the data in the request, and then sign the request with his own private key. After receiving the request, ACC calls the *Auth()* method and uses public key to verify user's identity, and then uses its own private key to decrypt. The successful decryption means that the user's access request is reasonable and the data can be successfully accessed.

GetAttrs(): After verifying the user's identity, parse the attribute fields contained in the user request by calling *GetAttrs()*. The request contains subject attributes and objects attributes $\{AS, AO\}$.

CheckAccess(): First, get the attribute $\{AS, AO\}$ through *GetAttrs()*, and then call *QueryPolicy()* of the ACPC to query the corresponding attribute access control policy according to *AS* and *AO*. If the query result is null, there is no policy to support the request. Otherwise, one or more access policies will be obtained, and then judge whether the attribute value *AE* in the request matches the *AE* in the access policy and whether the value of *OrderPermission* in the *AP* is 1. If all attributes match the policy, the verification passes. Finally, the related functions in the OMC are invoked to complete the access to the order resources, update, and delete operations. Otherwise, return the request fails. Algorithm 3 is the pseudocode of *CheckAccess()*.

EDC. The EDC includes the environmental data uplink *AddEnvData()* and *GetEnvData()* to obtain environmental data.

AddEnvData() is similar to *AddOrder()* in Algorithm 1. The *AddEnvData()* method implements the interface for inserting data, adds environmental data to the state database, and updates the operation record to the environmental data chain.

The main function of *GetEnvData()* is to obtain environment data and query the corresponding environment data from the database based on the IoT ID.

3.4. Workflow

This section details the system workflow, as shown in Fig. 3, including five stages, entity registration, the user placing an order, granting UDV order delivery qualifications, order allocation, delivery, and user request.

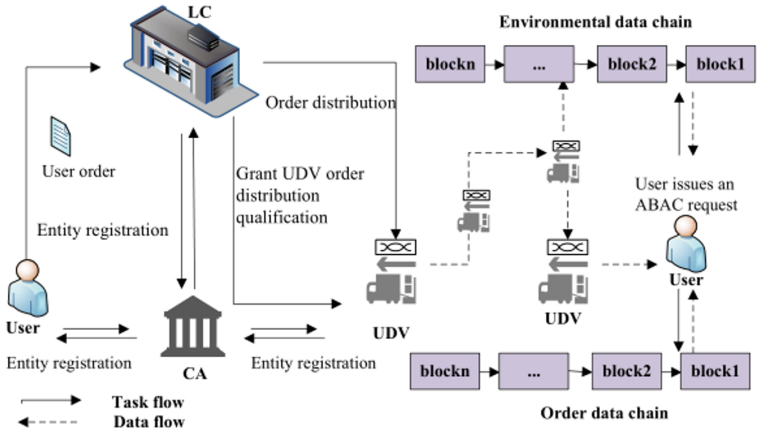


Fig. 3. Environment data uplink process.

Entity registration. The entity requiring authentication sends identity information to CA through the secure channel. CA encrypts the received entity identity information with PK

and then issues a certificate to the entity. As shown in Eq. (7), $Info_i$ is the entity identity, $Sign_{CA}$ is the CA's signature for the entity, and $TimeStamp_i$ is the corresponding timestamp.

$$CA \rightarrow Certi\{PK_i,\ Info_i,\ Sign_{CA},\ TimeStamp_i\} \tag{7}$$

User Order. Users order according to their own needs, and the system will create a transaction order according to the order time, recipient address, and contact information.

$$OMC(AddOrder()) \rightarrow \{Ledger,\ SDB\} \tag{8}$$

Grant UDV order distribution qualification. After registration, UDV needs to obtain the order delivery qualification granted by CA. Firstly, verify the identity of the UDV through *Auth()* in the ACC, and then the UDV sends a distribution request to CA. Once receiving the request, CA checks *UdvCert, UdvState, Distance* by calling *CheckUdvInfo()* in the OMC, the UDV distribution qualification can be granted after passing the verification. The process is shown in Algorithm 4.

Algorithm 4: Grant UDV order delivery qualification

Input: *Request(Delivery)*
Output: *Success* or *Error*

```
if Auth(Udv) == False then
    return Error("UnAuthorized");
end
ans ← CheckUdvInfo(UdvId, UdvCert, UdvState, Distance);
if ans ≠ Null then
    return Error("UnAuthDelivery");
end
return Success("OK");
```

Order Distribution. LC calls *OrderDistribute()* in the OMC, and refer to *OrderTime*, *UdvCert*, and *UdvState* to allocate orders to UDV and prepare for the delivery task reasonably. After the UDV finishes loading, it starts to distribute the goods.

$$OrderDistribute(OrderId,\ OrderTime,\ UdvCert,\ UdvState) \rightarrow Udv \tag{9}$$

During the delivery process, the sensor hardware device configured on the UDV will collect environmental data in real-time, and upload the data to the environmental data chain by calling *AddEnvData()* in the EDC.

$$EDC(AddEnvData()) \rightarrow \{Ledger,\ SDB\} \tag{10}$$

User request. Users can query order information in real-time according to their own needs, which ensures that they have a dynamic understanding of the entire order distribution process. When the goods arrive at the agreed place, the sensor will send the current location information to the blockchain node nearest to the UDV and notify the user that

the order has been delivered[21]. When the pick-up user arrives at the destination, the order can be successfully obtained after being verified as a legitimate user.

$$ACC(Request\{AS,\ AO,\ AP\}) \rightarrow \begin{cases} 1, \textit{Permit} \\ 0, \textit{Deny} \end{cases} \tag{11}$$

The user initiates a request to access the order data. Once receiving the request, the blockchain triggers the smart contract and calls *Auth()* in the ACC to verify the authenticity of the user's identity.

$$Request\{AS,\ AO,\ AP\} \rightarrow BlockChain \tag{12}$$

$$ACPC(Auth(AS,\ AO,\ AP,\ PK_i)) \rightarrow Entrpted\_ABACPolicy \tag{13}$$

If the verification passes, *QueryPolicy()* in the ACPC will be invoked to query the ACC based on the subject and object attributes. If the result is not empty, one or more access policies are obtained. Otherwise, there is no policy to support the request.

$$Decrypte(Entrpted\_ABACPolicy, SK_i) \rightarrow \langle ABACPolicy, OK \rangle \tag{14}$$

$$ACPC(QueryPolicy(AS,\ AO)) \rightarrow \begin{cases} one\ or\ more\ policy, & OK = 1 \\ null, & OK = 0 \end{cases} \tag{15}$$

If all attributes match the policy, the relevant methods in the OMC or EDC are invoked to complete the operation of accessing, updating and deleting logistics data.

$$OMC \xrightarrow{add/delete/query/update} OrderData \tag{16}$$

$$EDC \xrightarrow{add/query} EnvData \tag{17}$$

4. Experiment

The experiment was completed on a single computer. The CPU model of the computer is i7-6700 and the memory size is 8G. The software environment required for the experiment is shown in Tab. 4. The experimental steps include configuration fabric network, system initialization and startup steps, chaincode installation, system function test, and system throughput test. The functional test mainly tests a series of operations on the order data chain, such as user querying order information, modifying order information, etc. For the environment data, *AddEnvData()* and *QueryEnvData()* is only tested because the amount of the environment data is large and the query is only used when the order is handed over in the case of the failure of the UDV.

Two groups of experiments were conducted. The first group tested the throughput of four smart contracts when the numbers of concurrent requests were 10, 50, 100, 200, 300, 400, 500, 600, 700, 800, 900, and 1000. The second group of experiments tested the time spent on four smart contracts with 10-1000 concurrent requests, respectively.

Table 4. Environment data field description.

Software and System	Version
OS	Ubuntu 16.04
Hyperledger Fabric	v1.4.3
Docker	v18.09.7
Docker-compose	v1.8.0
Node	v12.18.3
Golang	v1.14.6
Git	v2.7.4

Table 5. Environment data field description.

Environmental Parameters	Value
Couchdb	4
CA	2
Orderer	1
Peer	4
Fabric-tools	1
Fabric-iot/chaincode	16

4.1. System Construction

Tab. 5 is the composition of experimental environment nodes. The experimental steps include system environment initialization, system startup, chaincode install, and update.

System initialize: Firstly, the binary tools provided by the Hyperledger Fabric are used to generate certificates and key pairs for Order and Peer nodes of different organizations. Secondly, move the node's certificate and key pair to the file directory mounted on the docker image of the CA node. When the CA container is started, other nodes can authenticate their identities with the signature.

$$CA \rightarrow \{Certpeer,\ Certpeer,\ Certchannel,\ PK_i,\ SK_i\} \tag{18}$$

$$Build(Certi,\ PK_i,\ SKi) \rightarrow DockerImage \tag{19}$$

System startup: Use the configtxgen tool provided by Hyperledger Fabric to generate the genesis block [25] and configure the channel. Write the configuration information of nodes and channels into transactions to ensure that the identity information of each section in the system can be traced and tampered with. Blockchain starts nodes based on the docker container and Hyperledger. Each node has an independent environment, communicates with each other through port calls, and executes the blockchain to the startup script, to quickly realize a series of processes such as creating channels and joining channels.

$$Config(Cert_i,\ PK_i,\ SK_i) \overset{configtxgen}{\longrightarrow} Transaction \tag{20}$$

$$Start(Docker,\ Fabric) \overset{join}{\longrightarrow} Channel \tag{21}$$

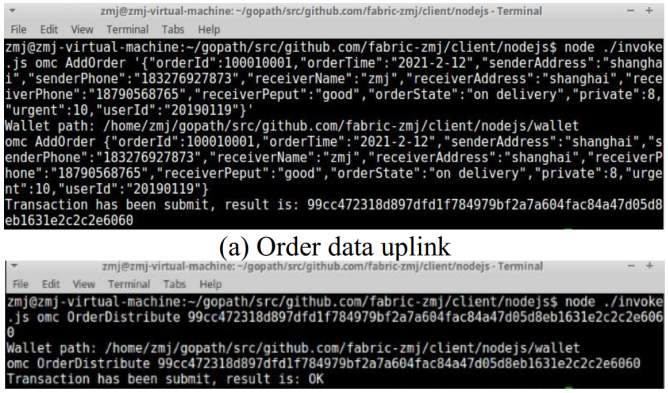
Chaincode install and update: After the system starts successfully, chaincode starts to install. Since the client's image has attached the directory of chaincode, chaincode can be compiled directly in the client's image. The quick installation of chaincode is

realized by executing the script install.sh, and the quick update of chaincode is realized by executing the script upgrade.sh. Once the chaincode is successfully installed, the order storage, user access and strategy formulation process are tested.

$$Install(Chaincode) \overset{Client}{\longrightarrow} Transaction \tag{22}$$

4.2. Function testing

Order stored procedure. First, add the order data to the blockchain by calling *AddOrder()*, and then call *OrderDistribute()* to allocate the order stored in the blockchain reasonably. Fig. 4 is the order stored procedure.



(a) Order data uplink

(b) Order allocation

Fig. 4. Order storage process.

Fig. 5. Order storage process.

User access process. Fig. 5 shows calling CheckAccess() to access order data.

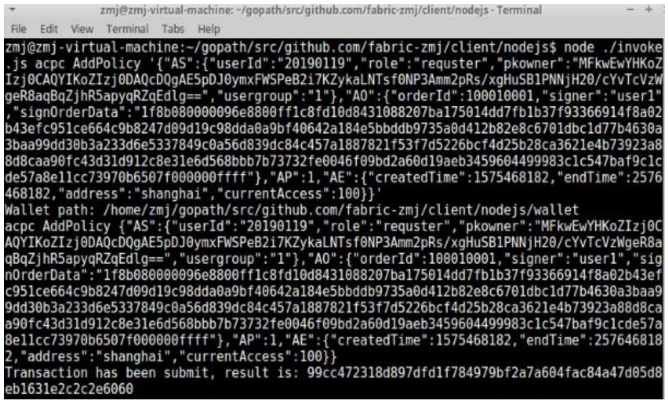
Access control policy development process. The access control policy is shown in Fig. 6. The administrator formulates relevant access policies based on *UserId* and *OrderId* and adds the access policies to the blockchain network through *AddPolicy()*.

4.3. Throughput testing

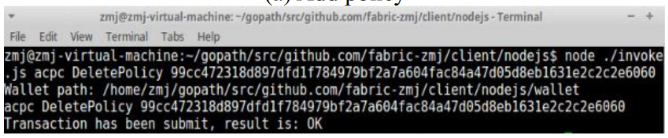
Throughput is an important index to evaluate system performance that is affected by software and hardware device, block size, and consensus algorithm. In the blockchain, throughput refers to the number of transactions that can be processed per unit time.

$$TPS = \frac{Transactions\ concurrency}{Average\ response\ time} \tag{23}$$

To test the system performance, the throughput of four smart contracts with 10, 50, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000 concurrent requests are tested in the first group of experiments. To avoid the impact of uncertain factors such as network downtime, under different concurrent requests, ten groups of throughput data with relatively stable network status are selected respectively and the average value is taken as the final throughput. The test results are shown in Fig. 7.



(a) Add policy



(b) Delete policy

Fig. 6. Access control policy development process.

In the second group of experiments, the same method was used to test the average time spent by the four smart contracts in the case of 10-1000 requests. The test results are shown in Fig. 8.

Fig. 7 shows that when the OMC is executed, the throughput of *DeleteOrder()* is the highest and that of *UpdateOrder()* is the lowest as the number of transactions increases. The throughput of *OrderDistribute()* finally stabilized at 175-180. Since both *OrderDistribute()* and *AddOrder()* contain a read operation and a write operation, the throughput

is almost equal. While *QueryOrder()* contains only one read operation, its throughput is higher than *OrderDistribute()* and *AddOrder()* is stable in the range of 200-205.

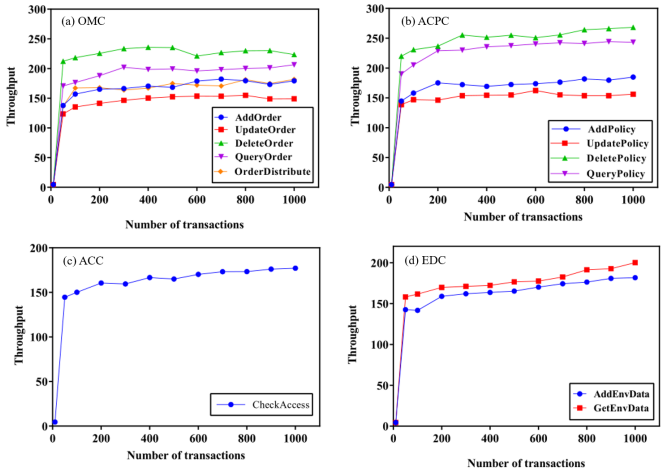


Fig. 7. Access control policy development process.

When the ACPC is executed, the throughput of *DeletePolicy()* is the highest and *UpdatePolicy()* is the lowest as the number of transactions increases. Since *DeletePolicy()* does not generate transactions and can directly delete existing transactions in the database, the throughput is high and finally stabilizes at 260-265 as the number of transactions increases. *QueryPolicy()* contains only one read operation, its throughput is high and finally stabilizes at 240-245. *AddPolicy()* contains one read operation and one write operation and its throughput is finally stable at 180-185. Compared with *AddPolicy()*, *UpdatePolicy()* contains one read operation and two write operations, its throughput is lower than 180 and stable at 150-155.

When the ACC is executed, the throughput of *CheckAccess()* is finally stabilized at 175-180. Since *CheckAccess()* calls *QueryPolicy()* in the ACPC and *CheckAccess()* includes a read operation and a write operation, so its throughput is lower than *QueryPolicy()* in the ACPC.

When the EDC is executed, the throughput of *AddEnvData()* is finally stabilized at 170-175. *GetEnvData()* calls *QueryPolicy()* in the ACPC, and it only includes one read operation with its throughput finally stabilized at 185-190, which is higher than *AddEnvData()* but lower than *QueryPolicy()*.

Fig. 8 shows the average time spent by ACPC, OMC and EDC in the case of 10-1000 requests. As shown in Fig. 8(a), as the number of transactions increases, the time spent continues to increase. In general, the time spent by each method is as follows:

$DeletePolicy() < QueryPolicy() < AddPolicy() < UpdatePolicy()$

The time spent by each method in Fig. 8(b) is as follows:

$DeleteOrder() < QueryOrder() < AddOrder() \approx OrderDistribute() < UpdateOrder()$

In Fig. 8(c), *AddEnvData()* takes more time than the *GetEnvData()*. After analysis, software and hardware devices, the number of nodes, block size, and consensus algorithm will affect the reading and writing speed of the blockchain. Writing operation usually

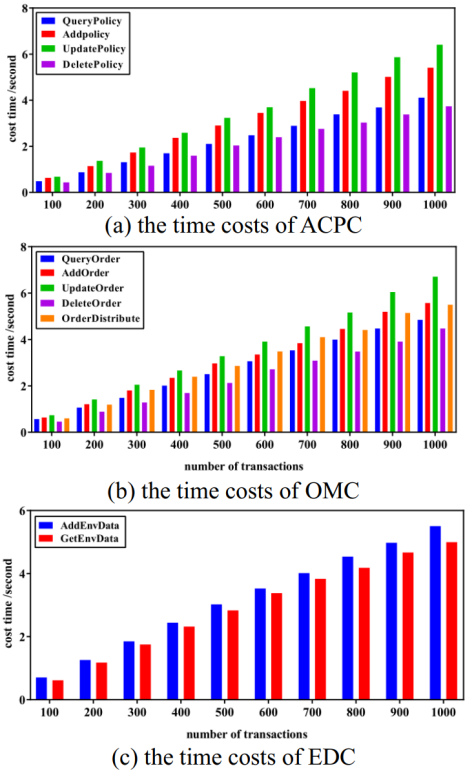


Fig. 8. Access control policy development process.

involves creating new blocks, generating new transactions, etc., which takes more time. Experiments show that the read operation takes less time than the write operation. The system throughput increases as the number of requests increases, and the throughput tends to be stable when the throughput reaches a certain value. With the further increase in the number of customer requests, there is no significant decline in throughput.

The experimental results are in line with expectations. The system can maintain high throughput in a large-scale request environment, realize dynamic fine-grained access control of logistics information, and meet the operation requirements of the actual logistics platform for data information.

5. Safety analysis

Entity authentication: Assuming that a malicious entity attempts to impersonate a legitimate user since each legitimate user obtains the unique certificate issued by CA, the system can verify the user's identity according to the user certificate [32],[33]. If the user certificate is not disclosed, the malicious entity cannot obtain any valid data information through counterfeiting. Therefore, entity authentication can effectively confirm the identity of the entity and ensure that the order data cannot be leaked.

Data integrity: All order records in the system are stored after being signed by using the principle of asymmetric cryptography. The data requester verifies the correctness of the data source after obtaining the order data. Since all records are stored on the blockchain, the decentralized environment provided by the blockchain can ensure that they will not be tampered with by any malicious entity[10],[5]. Therefore, data integrity can be guaranteed.

Data access security: When the data requester sends a data access request, the data can be accessed only when its attributes meet the access policy[36], [9]. Therefore, it can ensure that only authorized users can obtain order data, realizing controllable access to order data and the security of order data.

Information traceability: Suppose the blockchain is $BC = bc_1, bc_2, bc3, \ldots, bc_n$, where $bc_i(1 < i < n)$ is the ith block, tx_{ij} represents the jth record in the ith block, and tx_{ijk} represents the information of order k corresponding to the jth transaction in the ith block. According to the order information k, the order record can be obtained through a query on the order data chain to further obtain specific information such as order number[14]. Since the block header contains timestamp proof, the corresponding order record information can be queried through the order number, and the earliest order creation record can be traced according to the timestamp order. To sum up, the order information in this system can be tracked and queried.

6. Conclusions and future work

Based on the Hyperledger Fabric platform, this paper designs and implements the UDV logistics information management system based on the alliance chain, which effectively solves the problems of lack of trust, data leakage, and controllable access. Using the advantages of blockchain technology, such as decentralization, traceability, and non-tampering, ABAC is deployed on the blockchain by designing smart contracts to ensure data integrity, traceability, and controllability. At the same time, the dual-chain storage strategy is designed to alleviate the pressure of the main chain and ensure the efficiency of data queries. Finally, the experimental results prove that the scheme can meet the operation requirements of data information in the actual logistics platform, realize the dynamic fine-grained access control of logistics information, and ensure the security of data. In summary, the system is effective and feasible for the storage of logistics data and access control management. Future work will try to improve the following two aspects:

1. Consider using more physical equipment to test the reliability and throughput of the system.
2. Consider the reputation of UDV and the LC in the process of logistics distribution.

Acknowledgments. This work was supported by the National Natural Science Foundation of China under Grant 61672338 and Grant 61873160.

References

1. Ahmad, R.W., Hasan, H., Jayaraman, R., Salah, K., Omar, M.: Blockchain applications and architectures for port operations and logistics management. Research in Transportation Business & Management p. 100620 (2021)

2. Baliga, A., Solanki, N., Verekar, S., Pednekar, A., Kamat, P., Chatterjee, S.: Performance characterization of hyperledger fabric. In: 2018 Crypto Valley conference on blockchain technology (CVCBT). pp. 65–74. IEEE (2018)
3. Benhamouda, F., Halevi, S., Halevi, T.: Supporting private data on hyperledger fabric with secure multiparty computation. IBM Journal of Research and Development 63(2/3), 1–8 (2019)
4. Cui, M., Han, D., Wang, J.: An efficient and safe road condition monitoring authentication scheme based on fog computing. IEEE Internet of Things Journal 6(5), 9076–9084 (2019)
5. Cui, M., Han, D., Wang, J., Li, K.C., Chang, C.C.: Arfv: an efficient shared data auditing scheme supporting revocation for fog-assisted vehicular ad-hoc networks. IEEE Transactions on Vehicular Technology 69(12), 15815–15827 (2020)
6. Dai, W., Wang, Q., Wang, Z., Lin, X., Zou, D., Jin, H.: Trustzone-based secure lightweight wallet for hyperledger fabric. Journal of Parallel and Distributed Computing 149, 66–75 (2021)
7. Gu, Q., Fan, T., Pan, F., Zhang, C.: A vehicle-uav operation scheme for instant delivery. Computers and Industrial Engineering 149, 106809 (2020)
8. Han, D., Pan, N., Li, K.C.: A traceable and revocable ciphertext-policy attribute-based encryption scheme based on privacy protection. IEEE Transactions on Dependable and Secure Computing PP(99), 1–14 (2020)
9. Han, D., Zhu, Y., Li, D., Liang, W., Souri, A., Li, K.C.: A blockchain-based auditable access control system for private data in service-centric iot environments. IEEE Transactions on Industrial Informatics 18(5), 3530–3540 (2022)
10. Hang, L., Kim, D.H.: Design and implementation of an integrated iot blockchain platform for sensing data integrity. Sensors 19(10), 2228 (2019)
11. Huang, J., Kong, L., Chen, G., Wu, M.Y., Liu, X., Zeng, P.: Towards secure industrial iot: Blockchain system with credit-based consensus mechanism. IEEE Transactions on Industrial Informatics 15(6), 3680–3689 (2019)
12. Huang, K., Wen, M., Park, J., Sung, Y., Cho, K.: Enhanced image preprocessing method for an autonomous vehicle agent system. Computer Science and Information Systems 18(2), 461–479 (2021)
13. Islam, M.A., Madria, S.: A permissioned blockchain based access control system for iot. In: 2019 IEEE International Conference on Blockchain (Blockchain). pp. 469–476. IEEE (2019)
14. Kamble, S.S., Gunasekaran, A., Sharma, R.: Modeling the blockchain enabled traceability in agriculture supply chain. International Journal of Information Management 52, 101967–101978 (2020)
15. Kuru, K., Ansell, D., Khan, W., Yetgin, H.: Analysis and optimization of unmanned aerial vehicle swarms in logistics: An intelligent delivery platform. Ieee Access 7, 15804–15831 (2019)
16. Li, D., Han, D., Crespi, N., Minerva, R., Sun, Z.: Fabric-scf: A blockchain-based secure storage and access control scheme for supply chain finance (2021)
17. Li, D., Han, D., Liu, H.: Fabric-chain chain: A blockchain-based electronic document system for supply chain finance. In: Zheng, Z., Dai, H.N., Fu, X., Chen, B. (eds.) Blockchain and Trustworthy Systems. pp. 601–608. Springer Singapore, Singapore (2020)
18. Li, D., Han, D., Zheng, Z., Weng, T.H., Li, H., Liu, H., Arcangelo: Moocschain: A blockchain-based secure storage and sharing scheme for moocs learning. Computer Standards Interfaces 81, 103597 (2022)
19. Li, H., Han, D., Tang, M.: A privacy-preserving storage scheme for logistics data with assistance of blockchain. IEEE Internet of Things Journal (2021)
20. Li, J., Chen, X., Chow, S.S., Huang, Q., Wong, D.S., Liu, Z.: Multi-authority fine-grained access control with accountability and its application in cloud. Journal of Network and Computer Applications 112, 89–96 (2018)
21. Liang, W., Xie, S., Cai, J., Xu, J., Hu, Y., Xu, Y., Qiu, M.: Deep neural network security collaborative filtering scheme for service recommendation in intelligent cyber-physical systems. IEEE Internet of Things Journal pp. 1–1 (2021)

22. Liu, H., Han, D., Li, D.: Behavior analysis and blockchain based trust management in vanets. Journal of Parallel and Distributed Computing (2) (2021)
23. Liu, H., Han, D., Li, D.: Fabric-iot: A blockchain-based access control system in iot. IEEE Access 8, 18207–18218 (2020)
24. Mohanty, S.N., Ramya, K., Rani, S.S., Gupta, D., Shankar, K., Lakshmanaprabu, S., Khanna, A.: An efficient lightweight integrated blockchain (elib) model for iot security and privacy. Future Generation Computer Systems 102, 1027–1037 (2020)
25. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. Decentralized Business Review pp. 260–268 (2008)
26. Neghabadi, P.D., Samuel, K.E., Espinouse, M.L.: Systematic literature review on city logistics: overview, classification and analysis. International Journal of Production Research 57(3-4), 865–887 (2018)
27. Ni, H., Deng, X., Gong, B., Wang, P.: Design of regional logistics system based on unmanned aerial vehicle. In: 2018 IEEE 7th Data Driven Control and Learning Systems Conference (DDCLS). pp. 1045–1051. IEEE (2018)
28. Nyame, G., Qin, Z., Obour Agyekum, K.O.B., Sifah, E.B.: An ecdsa approach to access control in knowledge management systems using blockchain. Information 11(2), 111 (2020)
29. Outchakoucht, A., Hamza, E., Leroy, J.P.: Dynamic access control policy based on blockchain and machine learning for the internet of things. Int. J. Adv. Comput. Sci. Appl 8(7), 417–424 (2017)
30. Qiu, J., Tian, Z., Du, C., Zuo, Q., Su, S., Fang, B.: A survey on access control in the age of internet of things. IEEE Internet of Things Journal 7(6), 4682–4696 (2020)
31. Rahman, M.U., Guidi, B., Baiardi, F.: Blockchain-based access control management for decentralized online social networks. Journal of Parallel and Distributed Computing 144, 41–54 (2020)
32. Shen, M., Liu, H., Zhu, L., Xu, K., Yu, H., Du, X., Guizani, M.: Blockchain-assisted secure device authentication for cross-domain industrial iot. IEEE Journal on Selected Areas in Communications 38(5), 942–954 (2020)
33. Tian, Q., Han, D., Li, K.C., Liu, X., Duan, L., Castiglione, A.: An intrusion detection approach based on improved deep belief network. Applied Intelligence 50(10), 3162–3178 (oct 2020)
34. Xiao, T., Han, D., He, J., Li, K.C., de Mello, R.F.: Multi-keyword ranked search based on mapping set matching in cloud ciphertext storage system. Connection Science 33(1), 95–112 (2021)
35. Xu, Z., Liang, W., Li, K.C., Xu, J., Zomaya, A.Y., Zhang, J.: A time-sensitive token-based anonymous authentication and dynamic group key agreement scheme for industry 5.0. IEEE Transactions on Industrial Informatics pp. 1–1 (2021)
36. Yu, Y., Li, Y., Tian, J., Liu, J.: Blockchain-based solutions to security and privacy issues in the internet of things. IEEE Wireless Communications 25(6), 12–18 (2018)
37. Zhang, Y., Sun, W., Xie, C.: Blockchain in smart city development—the knowledge governance framework in dynamic alliance. In: International Conference on Smart City and Intelligent Building. pp. 137–152. Springer (2018)
38. Zhang, Y., Kasahara, S., Shen, Y., Jiang, X., Wan, J.: Smart contract-based access control for the internet of things. IEEE Internet of Things Journal 6(2), 1594–1605 (2018)
39. Zraková, D., Demjanoviová, M., Kubina, M.: Online reputation in the transport and logistics field. Transportation Research Procedia 40, 1231–1237 (2019)

Manjie Zhai is currently pursuing the M.S.degree with the School of Information Engineering, Shanghai Maritime University, Pudong, China. Her current research interests include blockchain and internet of things security.

Dezhi Han received the B.S. degree in applied physics from the Hefei University of Technology, Hefei, China, in 1990, and the M.S. and Ph.D. degrees in computing science from the Huazhong University of Science and Technology, Wuhan, China, in 2001 and 2005, respectively. He is currently a Professor with the Department of Computer, Shanghai Maritime University, Pudong, China, in 2010. His current research interests include cloud and outsourcing security, blockchain, wireless communication security, network, and information security.

Chin-Chen Chang received the Ph.D. degree in computer engineering from National Chiao Tung University, Hsinchu, Taiwan, in 1982, and the B.E. and M.E. degrees in applied mathematics, computer and decision sciences from National Tsinghua University, Hsinchu, Taiwan, in 1977 and 1979, respectively. He was with National Chung Cheng University, Minxiong, Taiwan. Currently, he is a Chair Professor with the Department of Information Engineering and Computer Science, Feng Chia University, Taichung, Taiwan, since 2005. His current research interests include database design, computer cryptography, image compression, and data structures.

Zhijie Sun is currently pursuing the M.S.degree with the School of Information Engineering, Shanghai Maritime University, Pudong, China. His current research interests include blockchain and internet of things security.

Received: December 20, 2021; Accepted: March 15, 2022.