Blockchain-Based Multisignature Lock for UAC in Metaverse

Keke Gai, *Senior Member, IEEE*, Shuo Wang, Hui Zhao, Yufeng She, Zijian Zhang, *Member, IEEE*, and Liehuang Zhu, *Senior Member, IEEE*

***Abstract*— As an emerging digital concept offering interconnections across multiple platforms, the metaverse provides digital transformations for various aspects of the physical world, facilitated by a few novel technologies, for example, cloud computing offers data support for the digital world. Humans immersed in the metaverse are digital entities who communicate with others or objects, such that ubiquitous access controls (UACs) are indispensable sectors for multiple platforms. However, in the metaverse, UACs have opened a wide scope of bridges for individuals to shuttle the virtual world, which implies that numerous threats exist at the access layer due to a great pool of entries. In this paper, to solve security issues in the UAC setting of the metaverse, we propose a novel blockchain-based multisignature lock for UAC (BMSL-UAC) scheme. All data institutions reconstruct a consortium blockchain system. In addition, our proposed scheme ensures that only authorized users can access an institution's data. Finally, we abstract the user's data access behaviors into the transaction information of the consortium blockchain system to realize full life-cycle data management and traceability. To verify the performance of our scheme, a series of experiments are carried out on the Hyperledger, and evaluation results have demonstrated that the resource consumption, delay, and throughput of this scheme are all within a reasonable range.**

***Index Terms*— Blockchain, metaverse, multisignature lock, ubiquitous access control (UAC).**

I. Introduction

THE prosperous development of the internet has dramatically enriched the network-related applications that are connecting humans with the virtual world. As an emerging concept of future network service representations, the metaverse [1], [2], [3] is expeditiously attracting attention from both academia and the industry. The metaverse builds an interconnected digital world that allows people to enter anytime, anywhere, so there is ubiquitous access control (UAC). UAC refers to the metaverse's ability to be widely accessed. Different levels of access are often required to be tailored to the specific needs of different metaverse users.

In the metaverse, UACs allow individuals to move in and out freely. However, challenges still exist, for example, data breaches. Considering cloud computing [4], for instance, the technology is used for storing data so that people can shape another life in the virtual world, differing from that of the physical world [5]. A data center has all access rights to each institutional dataset, and it is difficult for institutional data owners to obtain UAC rights and data usage traceability [6]. Therefore, insecure access control has become a challenging problem in the metaverse.

Recent research suggests that blockchain [7] is one of the solutions to UAC and data traceability [8], [9]. For example, Yang et al. [10] point out that blockchain and artificial intelligence (AI) provide the technical infrastructure for the metaverse, especially emphasizing that the blockchain can empower the economic circulation and data traceability of virtual users in the metaverse. Tang et al. [11] argue data traceability can be achieved by the historical timestamp in the blockchain. Despite access control in the metaverse has been explored, intercross access control for multiple organizations or platforms has been rarely addressed by prior studies.

In this paper, we have investigated the UAC and data lifecycle traceability issues in the metaverse and proposed a blockchain-based multisignature lock for UAC (BMSL-UAC) system for facilitating intercross platform access controls. First, we use the consortium blockchain architecture to build institutional systems in the metaverse, that is, making each institution an organization and the data center a peer node. We also separate users and peers of the system. Users cannot directly access peer nodes and can only communicate with the blockchain system by registering a client, so that the constructed blockchain system has a better privacy performance. Second, we propose a consortium blockchain-based multisignature locking mechanism (MSLM). This mechanism creates locks on the ledger to manage access to datasets. MLSM specifies default permission rules from the dataset

Manuscript received 15 July 2022; revised 19 September 2022 and 15 November 2022; accepted 25 November 2022. This work was supported in part by the National Key Research and Development Program of China under Grant 2021YFB2701300, in part by the Natural Science Foundation of Shandong Province under Grant ZR2020ZD01, and in part by the Defense Industrial Technology Development Program under Grant JCKY2020206C058. *(Corresponding author: Hui Zhao.)*

Keke Gai is with the School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing 100081, China, and also with the Yangtze Delta Region Academy, Beijing Institute of Technology, Jiaxing, Zhejiang 314019, China (e-mail: gaikeke@bit.edu.cn).

Shuo Wang and Liehuang Zhu are with the School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing 100081, China (e-mail: 3220215214@bit.edu.cn; liehuangz@bit.edu.cn).

Hui Zhao is with the Educational Information Technology Laboratory, Henan University, Kaifeng 475001, China (e-mail: zhh@henu.edu.cn).

Yufeng She is with the School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China (e-mail: 3220190860@bit.edu.cn).

Zijian Zhang is with the School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing 100081, China, and also with the Southeast Institute of Information Technology, Beijing Institute of Technology, Fujian 351100, China (e-mail: zhangzijian@bit.edu.cn).

Digital Object Identifier 10.1109/TCSS.2022.3226717

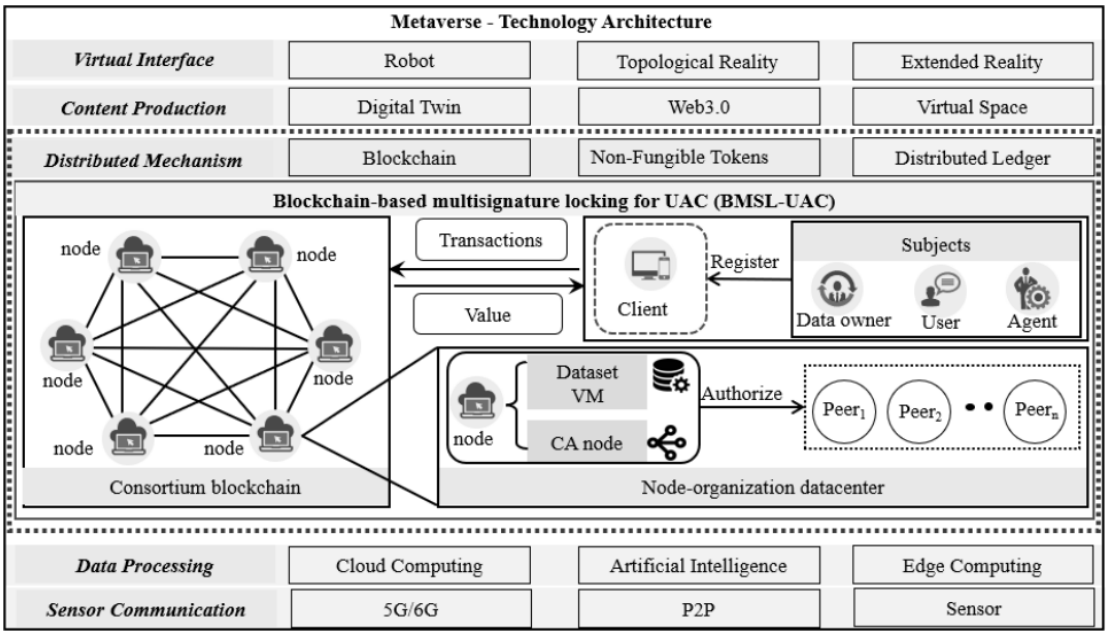


Fig. 1. High level architecture of the BSML-UAC system.

owners and cascading agents involved. The smart contract technology of the consortium blockchain is used to query the default access permission rules or collect signatures to verify unlocking conditions. Finally, we abstract the user's data access behavior into the transaction information of the consortium blockchain system. The cross-agency data access information is stored in the distributed ledger of blockchain in the form of transaction information to realize the full life-cycle management and traceability of data.

The main contributions of this work are summarized in the following.

1) We propose a novel UAC technique in the metaverse scenario. We realize the integration of various data institutions in the metaverse with the consortium blockchain system, and each data institution center is designed as an organization in the consortium blockchain. Unlock conditions are validated in MSLM using the default access rules defined in multisignature lockout from the user, agent, and owner. The proposed MSLM effectively solves the problem of UAC across institutions in the metaverse scenario.
2) We redefine the transaction structure of the data source, separate system users from peer nodes, and solve the privacy leakage problem of source data. We abstract the user's data access behavior into the transaction information of the consortium blockchain system to realize the full life cycle management and traceability of the data.

The rest of this paper is organized in the following order. Section II describes the background of the proposed work. Section III introduces the problem definition and threat model. The operating mechanism of the proposed model and crucial algorithms are presented in Sections IV and V, respectively. Moreover, we provide evaluation results in Section VI. Finally, we conclude the work in Section VII.

TABLE I
SYMBOLS TABLE

Symbols	Description
BMSL-UAC	**B**lockchain-based **M**ulti-**S**ignature **L**ock **U**biquitous **A**ccess **C**ontrol
MSL	**M**ulti-**S**ignature **L**ock
MSLM	**M**ulti-**S**ignature **L**ocking **M**echanism
IPFS	**I**nterplanetary **F**ile **S**ystem
ABE	**A**ttribute-**B**ased **E**ncryption
PoP	**P**ublic-**P**rivate
VMs	**V**irtual **M**achines
DAP	**D**efault **A**ccess **P**olicy
AT	**A**gent **T**able
LCA	**L**ock **C**ontrol **A**lgorithm

II. BACKGROUND AND RELATED WORKS

In this section, we present the background knowledge and related work. In addition, Table I presents some symbols frequently used throughout this paper.

A. Background

The vigorous development of the internet has greatly enriched the network-related applications that connect human beings with the virtual world. As an emerging concept of future network service representations, the metaverse is expeditiously attracting attention from both academia and the industry. The metaverse [1] is a network of social, immersive environments interconnected in a persistent multiuser platform. It is based on the fusion of various technologies including internet of thing (IoT), blockchain [12], digital twin [13], [14], [15], AI [16], and Big Data [17]. These technologies enable users to digitally interact in real time and dynamically with virtual environments, digital objects, and people through their digital avatars. The metaverse has great benefits in the

future. However, a natural question in the metaverse is how institutional data is managed and controlled throughout its lifecycle.

Blockchain technology [18] is a new technology combining various technologies of modern cryptography, peer-to-peer network, distributed storage, consensus mechanism, and smart contract. Furthermore, blockchain is a technical solution that does not rely on the participation of trusted third parties and uses its own distributed nodes to store, verify, transmit and communicate network data. Blockchain essentially refers to a distributed ledger in which data or transactions are stored that are immutable, traceable, and permanent. According to the different participants, blockchain can be divided into the public chain, private chain, and consortium blockchain. A consortium blockchain [19] is a blockchain that several organizations work together to maintain and must have certain permissions to access and use the blockchain. In this paper, we use the consortium blockchain to realize the information storage and the full life-cycle management of data in the process of cross-institutional data sharing in the metaverse.

A smart contract is a computer protocol that propagates, verifies, and executes contracts in an information-based manner [20], [21]. There are two types of smart contract accounts, externally owned accounts, and contract accounts. Each account node has a 160-bit address and its own state. A contract can be viewed as a machine based on the state of things and consists of accounts that change the global state by relying on transactions [22]. The rules of smart contracts are defined by machine language and complex interactions between various entities and system management can be programed into smart contracts as long as the rules are met. In this paper, we use smart contracts to enforce UAC.

B. Related Work

The metaverse [23] has recently attracted attention to the potential of the future Internet. As one of the key basic technologies of the metaverse, blockchain provides a transparent, stable, and credible trust mechanism for the metaverse. Researchers [10], [16], [17], [23], [24] conducted research on the metaverse and blockchain. For example, to support large-scale transactions, huge resource consumption, and interoperability issues in the metaverse, Nguyen et al. [24] conducted research on the scalability of blockchains supporting the metaverse. They design a new sharding scheme to improve the scalability of the underlying blockchain. However, Nguyen et al. [24] did not comprehensively analyze the role of blockchain in the metaverse from a technical point of view.

From a technical point of view, a blockchain-based metaverse approaches have been extensively discussed by Gadekallu et al. [17] and Mozumder et al. [16]. Especially, Mozumder et al. [16] constructed a blockchain-based metaverse data trust framework and pointed out that blockchain can realize data traceability with unique identification tags in the metaverse. Gadekallu et al. [17] analyzed the role of blockchain in the metaverse, including ensuring data privacy and security, ensuring data quality, enabling seamless and secure data sharing, data interoperability, and ensuring data integrity. Although the existing research shows that the blockchain can realize the sharing and traceability of data in the metaverse. However, the traceability and access control of data centers in the metaverse have not implemented a complete solution.

To solve the problems of data security traceability and access control in the metaverse, a lot of research has been done on blockchain-based data traceability technology. Çolak et al. [25] pointed out that blockchain technology can solve the problems of traceability and transparency of traditional supply chains. For example, Amazon-managed blockchain could provide the end-to-end visibility needed in today's supply chain to automate the tracking and traceability of its entire production process. However, the original data stored in the blockchain in this scheme has the risk of privacy leakage and adversary attack. To improve the privacy and security of traceability data, Hasan et al. [26] proposed an efficient method for storing provenance data using blockchain and the Interplanetary File System (IPFS). This approach distributes the provenance data into the IPFS system and stores the digest value on the blockchain system, making it impossible for adversaries to obtain it. To improve the trustworthiness and traceability of data, Nyaletey et al. [27] proposed a new blockchain-based approach to IPFS access. In addition, people have also conducted related research on data traceability in specific scenarios. For example, in the field of food safety, researchers [28], [29], [30] proposed a blockchain-based food and drug traceability system to realize the full life-cycle management of food and drug production. In addition, in the aviation field, Ho et al. [31] proposed a blockchain-based aircraft spare parts inventory management system to achieve the quality of traceable data within the spare parts supply chain. However, few studies considered blockchain-based UAC in the metaverse.

Some research [32], [33], [34], [35] helps to solve the problem of UAC in blockchain-based metaverse. For example, Laurent et al. [33] proposed a blockchain-based access control strategy to regulate access to valuable resources. However, their solution is deployed in bitcoin and is not suitable for the existing wide range of scenarios. Therefore, Li and Qin [34] proposed a blockchain-based access control model for decentralized systems to enhance access control in IoT. However, the access control model of this scheme is based on the public chain, so there is a privacy problem in accessing data. To solve the access data privacy problem, attribute-based access control is an effective solution. For example, Huang et al. [35] proposed a blockchain and attribute-based access control model. Arasi et al. [36] proposed a new auditable attribute-based encryption scheme for data-sharing systems. Moreover, to achieve fine-grained access control of data, Wang et al. [37] proposed an access control framework based on blockchain and property encryption. Wang et al. [38] proposed a dynamic and lightweight attribute-based access control to achieve secure and fine-grained authorization and provided secure data sharing and integrity auditing using blockchain technology. However, the properties of this scheme exist in a single node and therefore have a single point of failure problem. To solve this problem, Qin et al. [39] proposed a blockchain-based multiauthority access control

strategy, where each attribute is jointly managed by multiple authorities to avoid the single point of failure problem.

In summary, a lot of related work has been done on the application of blockchain in the metaverse, the data source and access control management of blockchain, and other technical dimensions. Most of the existing work focuses on handling trusted traceability using the tamper-resistant properties of blockchain, which solves the problem of data traceability in the metaverse. However, there is currently few research on access control for cross-agency data sharing in the emerging metaverse setting. In contrast to existing work, our research focuses on the lack of control over data sharing in the metaverse scenario. In our work, we decouple peer nodes from users of the blockchain system, thus solving the problem of data leakage. In addition, we propose a BMSL-UAC scheme to address the lack of control, managing access to datasets through attribute-based access policies and multisignatures.

III. Problem Definition and Threat Model

A. Problem Definition

The UAC of the metaverse allows broad access to the metaverse. However, UAC brings certain data security risks for cross-institutional data access in the metaverse. Although cloud computing provides data storage and sharing technology for the virtual world. In addition, centralized cloud computing data centers have all access control rights to data institutions, which makes it difficult for institution owners to obtain UAC and full life-cycle management of data access.

To realize the UAC of the metaverse and the full life-cycle management of data, our proposed approach is to use the consortium blockchain system to build a distributed system of institutions in the metaverse. In addition, we propose a multisignature locking mechanism based on the consortium blockchain to manage the access rights of data institutions. Finally, we store the information of cross-institutional data access in the distributed ledger of the blockchain in the form of transaction information. This realizes the full life-cycle management of control data information.

In addition to addressing the challenges of full life-cycle management of metaverse data, we also need to formulate different levels of access rights for the specific needs of different metaverse users. This ensures that only users who meet the access control requirements can access the metaverse's data organization. In other words, whether users are allowed to access metaverse data institutions is the result of UAC. Definition 1 presents the blockchain-based UAC problem in the metaverse.

Definition 1 (Ubiquitous Access Control Problem): Input: user information user = (name,pubkey), VMID, there exists agents of the data institutions, Agent_i, where $i \in [1, n]$, $\text{AttrSet}(x)$, where $x \in \{s, r, a, e\}$, Sig_i, where $i \in [1, n]$, ledgerStub. Output: an access control tag rst. The purpose is to determine whether the user can access the data organization of the metaverse.

In Definition 1, the user refers to the requester who accesses the metadata organization, which includes the user name and the public key pubkey. VMID refers to the virtual machine (VM) number mapped by the metaverse data organization. The user's agent group agent_i refers to the agent table (AT) describing the VMID of the data VM, where agent_1 is owned by the data owner, and $\text{agent}_2, \ldots, \text{agent}_n$ is owned by the agent users separately. $\text{AttrSet}(x)$ defines the set of attributes for access control. sig_i represents the agent's signature for the access control request. We determine whether the user can access the metaverse's data VM VMID based on the attribute set and the agent's sig_i.

B. Threat Model

In our threat model, we assume that an adversary acts as a registered legitimate user in the system and obtains a key pair. An adversary obtains access to resources in the metaverse by pretending to own a character he does not own. Furthermore, since the adversary has a legitimate identity in the system, we assume that the adversary can obtain the previous access records stored on the consortium blockchain and the public keys of other users in the system. However, due to the tamper-proof characteristics of the consortium blockchain, the adversary cannot modify the previous access record information stored in the consortium blockchain.

Since the security of this scheme is based on the security of encryption technology, the adversary cannot use the encryption technology (hash function and signature algorithm) applied in the scheme. Finally, since the security of this system is based on the practical Byzantine fault tolerance (PBFT) protocol of the consortium blockchain, the adversary cannot obtain more than $(n-1)/3$ (where n represents the number of all nodes in the consortium blockchain) consensus nodes and destroy the consortium blockchain.

IV. Proposed Model

A. Model Design

The BMSL-UAC mechanism proposed in this paper is shown in Fig. 1, which belongs to the distributed mechanism layer in the metaverse technology architecture. The model includes three entities, namely consortium blockchain, subject, and data center. The consortium blockchain includes storage ledgers and access control. The storage ledger is responsible for storing default access policies and data access information. Access control includes lock creation smart contracts and lock control smart contracts. The lock creation smart contract is responsible for creating a lock mechanism for the dataset when the system is initialized; the lock control smart contract is responsible for processing the client's request for access and feeding back the access control result to the client. The user module includes actual access data users and ubiquitous access agents. Data users are responsible for issuing requests for access to the data institutes and providing signature information. The data broker is responsible for accepting data requests provided by users through the client and sending them to the consortium blockchain. The data center represents the data information of each institution. The composition of the three entities is described in detail below.

The organization of the consortium blockchain is composed of all the institutions that own the dataset. There is a one-to-one relationship between an organization and a data center.

In order to authorize peer nodes and users in the consortium blockchain, the system configures a certificate authority CA certificate for each organization. Each peer in the consortium blockchain belongs to a specific organization and is issued a digital certificate by the organization's CA node. Peer nodes are the participants used to maintain the ledger and consensus mechanism and the data center cannot tamper with transaction data alone. However, the system admission of peer nodes is restricted by the source system. And when the system has been successfully authenticated, peer nodes can join the system.

To support the access control of storage ledgers and data at the same time, the design of distributed ledgers adopts an account-based model. The model consists of two parts: one is the on-chain transaction ledger and the other is the world state ledger. The on-chain transaction ledger holds immutable records of transactions and is therefore used to store default proxy access control policies and transaction information for users accessing datasets. The world state ledger holds a mutable record of application state and is therefore used to store smart contracts that manage UAC. In Section IV-B, we describe the process of the BMSL-UAC scheme.

In this model, users cannot directly access peer nodes and can only communicate with the blockchain system by registering a client. Clients are tools for subjects to communicate with the consortium blockchain, where subjects include dataset owners, proxies, and dataset users. In order to facilitate the communication between the client and the consortium blockchain, we define a transaction $T = \{\text{txid}, \text{srtcontraName}, \text{args}\}$, where txid stands for the unique identifier of the transaction, srtcontraName is the name of the smart contract, and args contains the requisite argument list for the smart contract by the smart contract. Users register clients with their own digital certificates and send transaction messages to the source system through their clients.

Data centers contain dataset VMs, which are external transparent peers to datasets. Users perform data analysis and statistical work on the dataset VM, but they cannot obtain or change the dataset encapsulated in the VM. In addition, dataset VMs are managed using multiple signature locks in the system and users are only allowed access to dataset VMs when the locks are opened.

We build a cross-institutional UAC system in the metaverse based on the multisignature consortium blockchain. An access control layer is added to the consortium blockchain system to build an access mechanism for control and identity management. Organizations issue digital certificates to users and peer nodes, and only those authenticated users/nodes can be added to the system. In order to realize the whole life-cycle management of data, we have added transaction information of user access data to the consortium blockchain. The core part of this paper is introduced below, based on the multisignature locking access control strategy.

B. BMSL-UAC

We use the BMSL-UAC mechanism to control the access of users in the metaverse to the data VM. The BMSL-UAC includes two models: the attribute-based access control model and the MSL model. In order to understand this scheme, first, we define the concepts of default access policy (DAP), AT, and MSL, respectively. Second, we introduce the access.

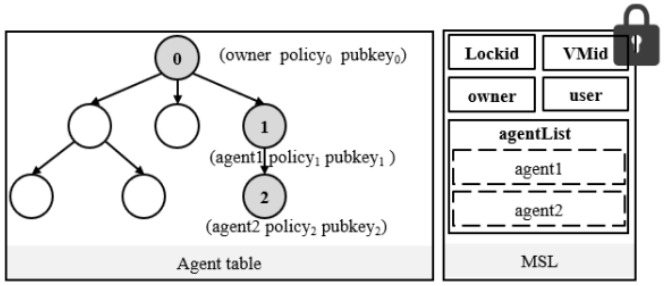


Fig. 2. AT and the structure of MSL.

Definition 2 (DAP): $P(\Theta\text{AttrSet}(x), \quad x \in \{s, r, a, e\})$ defines the rule of the attribute-based access control. $\Theta\text{AttrSet}(x)$ denotes a logical expression that is formed by conjuncting and extracting attributes from AttrSet(x). When $\Theta\text{AttrSet}(x)$ is true for the attributes from the Request[AttrSet(s), AttrSet(r), AttrSet(a), AttrSet(e)], it means that the request meets the default access policy.

Definition 3 (AT): AT is a state table in a tree structure. Nodes in the AT tree are denoted by (subjectName, subjectPolicy, subjectPubkey), among which subjectName, subjectPolicy, and subjectPubkey represent the subject, the corresponding DAP, and the subject's public key, respectively. The node covers the subject's agent state to a VM and access control rule.

Definition 4 (MSL): MSL is an access controller for dataset VM that consists of three attribute fields, denoted by {Lockid, VMId, $textowner$, $textagentList$, and user}. VMId is an identity of VM managed by the MSL; owner is the owner of VM; agentList$[1, \ldots, n]$ is an agent list; user is the subject of the access request to VM, which consists of attributes {subjectName, subjectPubkey}.

1) Attribute-Based Access Control Model: The attribute-based access control model proposed in this paper uses the concept of "agent" to describe the attributes of the dataset VM. The AT (as shown in Fig. 2) represents the corresponding table of the dataset VM in the data center. ATs are stored on the blockchain's state ledger and represent the state of dataset VMs in multiple data centers. More specifically, the proxy table is set up as a tree structure. Nodes in the tree are represented by (subject name, subject policy, subject Pubkey), where subject name, subject policy, and subject Pubkey represent the subject, the corresponding DAP, and the subject's public key, respectively. The relationship between parent and child nodes in the proxy table tree describes the proxy relationship of VMs between two subjects. It means that the VM is proxied by the subject parent node to the subject child node, and the subject parent node is the direct parent proxy of the subject child node.

In our system, the VM owner needs to have a higher-level access control priority than agents. It means that policy$_1$ is a subset of policy$_0$. That is to say, a request matching policy$_1$ will certainly match policy$_0$, but a request matching

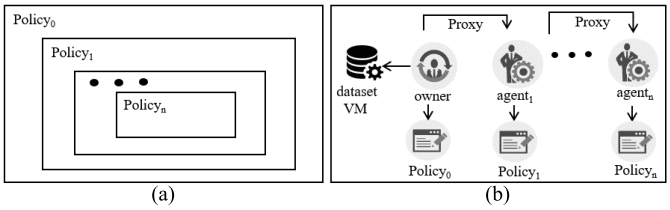


Fig. 3. Proxy status and policy coverage mechanism. (a) Policy coverage. (b) Proxy status.

$policy_0$ is not necessarily matched $policy_1$. When considering multilayer agent settings, the dataset owner authenticates the access of VM_i to $agent_1$ who authenticates access to other agents, for example, $agent_2, \ldots, agent_n$. We construct a multilayer mechanism for ruling access authentications for data sharing facilitated by agents. A basic rule of our mechanism is that a higher-level agent has a wider coverage of the policy than lower-level agents, as shown in Fig. 3(a). A route of data sharing is called an *Agent Chain*. Besides the basic rule, there are two operating rules formulating our access control.

1) Rule 1: Assume that a user sends a request R to access VM_i via an $agent_j$. The request will be permitted when the attribute of R matches the DAP in the agent chain (from the owner to the end agent).
2) Rule 2: Assume that the data owner and agents at different layers have preconfigured DAPs. The system will distribute a message containing R to those agents that R cannot meet the policy, as well as require a signature. The request R can be validated only when these agents' signatures are collected with approved validations. The system also will request a signature from the owner when R does not match owner's policy.

2) MSL Model: We use MSL to represent the access management of dataset VMs in a certain agent state. The dataset VM uses the relevant protocol to create an MSL for the user and then stores the MSL in the distributed ledger.

For an MSL object lock = {Lockid, VMId, owner, agentList, user}, pointer variables are stored in the arrays of the owner and agentList, which points at nodes in the AT tree. Hence, nodes in AT as well as the policy of the corresponding subject can be searched by using pointers. Our approach also configures that each MSL governs a usage state of VM. For example, a VM is agented to $agent_1$ by owner and an agent chain is constructed after a series of agents join in (from $agent_1$ to $agent_n$ in a level-by-level manner). Once the user and $agent_n$ reach a usage agreement to VM, each time when the user access to the VM needs to be governed by MSL. Therefore, for a user, $agent_n$ is a direct agent of VM, and $agent_{1,\ldots,n}$ are indirect agents.

In the model of MSL (as shown in Definition 4), we assume that there exist multilayer agents for governing dataset VMs. Each layer agent is authorized to make its attribute-based DAP. An AT is used to describe the agent state of the VM in the cloud data center. When the third party matches the access policy to a VM via an agent, the system will create an MSL on the state ledger by implementing a smart contract. Using MSL allows governing the access request.

a) MSL creation: The creation of the MSL will be achieved by searching the AT of VM. This operation is implemented by a direct search on agent nodes ($agent_n$, $Policy_n$, $pubkey_n$) in the AT. Tracing back forward on the tree structure can eventually reach the original node, so that a complete set covering all participant nodes can be found, for example, $\{(agent_i, policy_i, pubkey_i), i \in (0, 1, 2, \ldots, n-1)\}$. On the basis of the searched results, the system creates smart-contract-lock, $Lock_i$ = {VMiId, $owner_i$, agentList, user}, agentList = $\{agent_1, agent_2, \ldots, agent_n\}$, $(n \geqslant 1)$; thereafter, $Lock_i$ will govern the user's access to the VM_i.

b) MSL unlock: A user sends the system an access request R, [AttrSet(s), AttrSet(r), AttrSet(a), AttrSet(e)]. Once the request is received, the system triggers the smart contract to determine whether attributes carried by the request R match the condition of the unlock. Searching both VM owner's DAP and agents on the AT can be completed by indexing agentList and owner fields in $Lock_i$. A bottom-top indexing order is implemented for tracing agents from a lower layer to a higher layer. It implies that the indexing operation starts with $agent_n$ and traces toward $agent_0$.

By using this method, assume that the first indexed DAP approving the request R is defined to be an $agent_i$, $i \in (\text{nil}, 0, 1, 2, \ldots, n)$. Therefore, $\{agent_1, \ldots, agent_{i-1}\}$ is the higher-agent set of $agent_i$; the set $\{agenti + 1, \ldots, agent_n\}$ is $agent_i$'s lower-agent set. The owner is deemed the highest-layer agent in our system, denoted by an $agent_0$. We use i = nil to represent the nonsubject of DAP matching the request R is found by implementing the mechanism. The rule-based MSL unlock mechanism is given in the following.

1) Rule 3: Under the condition of $i = n (n \geqslant 1)$, when there is only one time indexing happened to the AT and the request R matches the direct agent $agent_n$'s DAP $policy_n$, unlock implementation is approved.
2) Rule 4: Under condition of $i \in \{0, 1, \ldots, n\}$, when signatures in a message set S all match validation requirements, unlock implementation is approved. A message set $S = \{sig_{i+1}, sig_{i+2}, \ldots, sig_n\}$ is a set that collects all returned signed messages from agents who are lower agents of $agent_i$, responding to the signature request of the message M. The system needs to validate signatures.
3) **Rule 5:** Under the condition of i = nil, similar to Rule 3, when signatures in a message set S all match validation requirements, unlock implementation is approved. The system sends out the validation request to the agent set $\{agent_0, agent_1, \ldots, agent_n\}$ in order to validate the access message M from the user.

C. Main Process

1) Phase 1: System Initialization: This phase mainly includes steps (1.1–1.4) as shown in Fig. 4.

Step 1.1: Start the consortium blockchain network. The organization parameters are configured in the consortium blockchain network, and the organizations in the consortium blockchain store distributed ledgers and ensure the consensus mechanism, and issue identity certificates for nodes. The system verifies the identity of the node before it joins the consortium blockchain network.

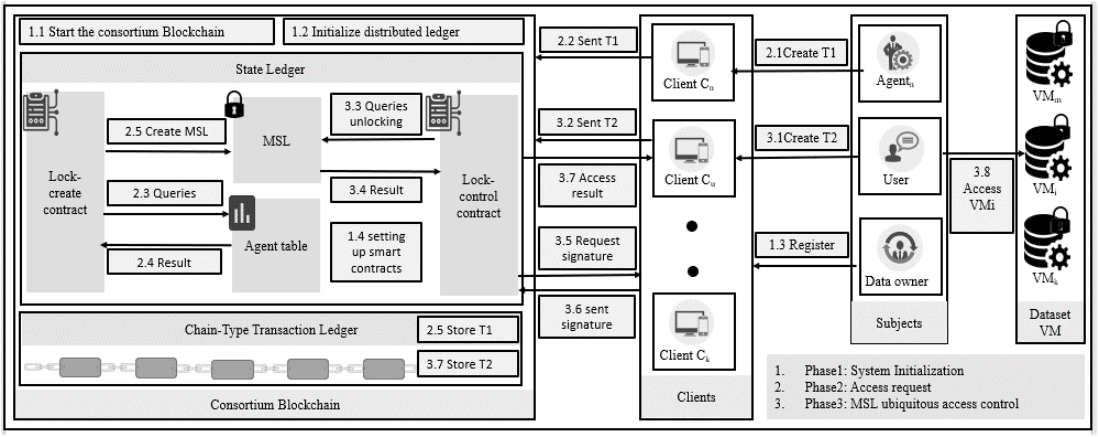


Fig. 4. Proposed system's running process.

Step 1.2: Initialize the distributed ledger. Initialize the distributed ledger of the consortium blockchain nodes, including initializing the chain transaction ledger and the state ledger. The configuration information of the consortium blockchain is written in the genesis block of the chain transaction ledger. Other block stores are used for packaged transactions at the origin of the data. At the same time, a distributed ledger based on key-value pairs is created to manage the proxy table and MSL. The initial state of the state ledger is empty.

Step 1.3: Registrate the user/client. The system issues identity certificates to subjects (e.g., dataset VM owners, agents, and users). Subject creates clients with its own credentials.

Step 1.4: Set up smart contracts. Set up lock creation smart contracts and lock control smart contracts on the consortium blockchain nodes. The lock creation smart contract is used to create MSL. Lock control smart contracts manage dataset access control via MSL.

2) Phase 2: Access Request: The user makes a data access request to the dataset VM. We assume that a user requests access to the dataset VM through a proxy. This process corresponds to (2.1–2.6) steps in Fig. 4.

Steps (2.1–2.2): The direct agent agent$_n$ creates transaction T_1 = {txid, lockCreateContract, VMId, agentnName, user} through the client C_n and sends it to the blockchain.

Steps (2.3–2.6): The consortium blockchain automatically creates a (lockCreateContract) after receiving the transaction T_1. Thus, lockCreateContract queries the AT to acquire the agent state of VM and creates a MSL (Lock$_i$ = {lockid, VMId, owner, agentList, user}) to govern access requests. Meanwhile, the system uploads the transaction T_1 to the chain-type transaction ledger for the traceability function.

3) Phase 3: Multisignature-Based UAC: This phase corresponds to (3.1–3.8) steps in Fig. 4.

Steps (3.1–3.2): The user creates a transaction T_2 (T_2 = {txid, lockControlContract, args}) by the client C_u and sends it to the consortium blockchain system. The parameter list args include MSL's id, lockid, and attributes of the access request [AttrSet(s), AttrSet(r), AttrSet(a), AttrSet(e)].

Steps (3.3–3.4): The transaction triggers the smart contract (lockControlContract), which queries VM owners and agents' access policies through MSL. Relying on MSL unlock mechanism, when the access request satisfies agent$_n$'s access policy, the system approves the access request to the VM.

Steps (3.5–3.6): If there exists an unsatisfying policy set [{policy$_r$, policy$_{r+1}$, ..., policy$_n$}, $r \in (0, 1, \ldots, n)$] with a corresponding subject set [{agent$_r$, agent$_{r+1}$, ..., agent$_n$}, $r \in (0, 1, \ldots n)$], the system inquires and validates those subjects' signatures via client nodes ($\{C_r, C_{r+1}, \ldots, C_n\}$). When the signature is validated, the system approves the user's access request to the VM; otherwise, the request is denied. Similarly, the system uploads the transaction T_2 to the chain-type transaction ledger for the traceability function.

Step (3.7): The lock control smart contract returns the verification result to the client. If the client obtains the unlocking information, it performs step (3.8) to access the data VM VM$_i$ and finally stores the transaction information T_2 in the blockchain ledger. Otherwise, this access request will be invalid.

In summary, we propose an MSL-based access control strategy, which realizes the function of trusted traceability based on data agent management. The system governs the proxy state of each dataset VM in a multilevel proxy environment through an AT and governs access requests to the dataset VM using the MSL method. Through the MSL unlock mechanism, each access request needs consent from the owner and relevant agents. Our approach reasonably allocates control powers for participant parties (e.g., the owner, agents, and users). Moreover, our approach implements the consortium blockchain to separate blockchain nodes from system users (subject). Interactions between users and consortium blockchain systems can only be processed by having transactions via clients. It means that users cannot directly obtain the blockchain ledger, so existing data leakage issues caused by blockchain-based data tracing can be solved.

V. Algorithms

To support the UAC of the metaverse and protect the privacy of users, we design two algorithms based on attribute-based access control, namely *multisignature lock (MSL) creation* and *lock control algorithm (LCA)*. First, in order to implement the access control policy, we propose MSL to manage the user's access control request. Second, in order to further improve the

Algorithm 1 MSL Creation Algorithm

Require: $user$={$Name$, $Pubkey$}, $VMId$, $agent_n$, $ledgerStub$
Ensure: $lockid$

Obtain VMId's $AgentTable$; AT= ledgerStub.GetState(VMId); pointer=AT.root;
Create an empty stack s[] for storing nodes' pointers
Create a variable *elem* for storing temporary stack elements
while Stack s[] is not empty $vee\ pointer$ is not an empty pointer **do**
  while $pointer$ is not an empty pointer **do**
    elem.pointer=pointer; elem.visitedNum=0; s.push(elem);
    pointer=pointer→child[0];
  end while
  Stack top element out of stack: elem=s.pop(); pointer = elem→pointer;
  if elem.visited < len(pointer→child) **then**
    elem.visited++; s.push(elem);
    pointer=elem.child[visited];
  else
    if pointer→subjectName == $agent_n$ **then**
      Break;
    end if
  end if
end while
Smart contract automatically creates an id=UUIDGen() to create pointer array alist[], alist[i]=s[i].pointer;
Create MSL ($Lock_i$) and value $Lock_i$'s attributes; Locki.Lockid=id; Locki.VMId=VMId;
Locki.owner=elem.pointer; Locki.agentList=alist; Locki.user=user;
Store SML object on state ledger ledgerStub.put(Lockid, Locki);

Algorithm 2 Lock Control Algorithm

Require: Lockid, AttrSet(s), AttrSet(r), AttrSet(a), AttrSet(e), Sig, ledgerStub
Ensure: rst

Obtain MSL objects from the state ledger, Locki=ledgerStub.GetState(Lockid);
Validate user identity v=validate(Sig, Lockid.user)
if v==false **then**
  return false;
else
  Construct access request, R=AttrSet(s)∧AttrSet(r)∧AttrSet(a)∧AttrSet(e)
  for i=len(Locki.agentList)-1; i ⩾ 0; i–; **do**
    Validate if access request satisfy agent's policy: PolicyDecide(agentList[i].policy, R);
    if PolicyDecideRst==true **then**
      The index of 1st agent matching the request: index=i;
      Break
    else
      Continue
    end if
  end for
  if index==len(Locki.agentList)-1 **then**
    return True;
  else
    Ask agent set for signatures;
  end if
  Validate if access request matches owner's policy: PolicyDecide(owner. policy, R)
  if PolicyDecideRst==false **then**
    Ask VM owners for signature;
  end if
  Collect signature set, SigSet
  if SigSet is valid **then**
    return True;
  else
    return False;
  end if
end if

security and privacy of access control, we design LCA based on multisignature to protect and control access to resources.

A. MSL Creation Algorithm

The *MSL creation* is designed to create MSL on the state of the consortium blockchain to manage user access requests. The input parameters of the MSL creation algorithm are the parameters (user, VMId, $agent_n$, ledgerStub) carried by the smart contract receiving transaction parameter field; user is the subject of the dataset VM that sends the access request; VMId is the unique identifier of the dataset VM; $agent_n$ represents the direct agent of the dataset VM responding to the request; ledgerStub represents the state of the distributed ledger. The output parameter of the MSL creation algorithm is lockid, which is used to create the identity of MSL.

Pseudo-codes of the MSL creation algorithm are given in Algorithm 1. We assume that a smart contract can read or update data objects on the blockchain state ledger through a handle variable ledgerStub. The main phases of the algorithm are described in the following.

1) Smart contract obtains VMId's AT from the state ledger through ledgerStub. Then, AT.root points to the root node of the AgentTable tree. The smart contract creates a stack S to assist the traversal of AT tree. For each node of the AT tree, the node is allowed to be traversed only after its child node has been traversed. The execution of the algorithm is based on this traversal rule and will end when AT.Node$_n$.subjectName is equal to $agent_n$.
2) When the traversal ends, the stack S constructs a pointer collection, where the pointer variable $s[i]$.pointer points to the parent nodes (AT.Node$_n$). Here, the AT.Node$_0$ is the root node of AT tree, and these pointer variables are used to create the MSL.
3) The smart contract creates a MSL object by Lock$_i$ = {Lockid, VMId, owner, agentList, user}. We have introduced the concept in the prior sections.

4) MSL object Lock_i is stored on the consortium blockchain state ledger through the smart contract. Transactions are uploaded to the chain-type transaction ledger. The returned identity of the MSL is Lockid.

1) Time Complexity Analysis of Algorithm 1: We give a brief analysis of the time complexity of the MSL creation algorithm. Suppose the number of the AgentTable is N. To find whether there are qualified agents, we need to search in the AgentTable. This algorithm uses the stack method to traverse the AgentTable. If the agent being queried is not in the AgentTable, then we need to traverse all the AgentTable. Therefore, the time complexity of Algorithm 1 is $O(N)$.

B. Lock Control Algorithm

To support the UAC of the metaverse and protect the privacy of users, we propose an *LCA*. When this mechanism is created, users can only obtain access rights to resources through LockControlContract, thus improving the security of access control. The user creates and sends a transaction through the client to request access to the dataset VM (VMId). A set of attributes of the access request are carried by the transaction. The lockControlContract returns a value of the Boolean value, representing the response to the access request (keep locking or unlock). A phase-by-phase explanation of LCA is also provided below.

1) Lockid user requests an MSL identity for the purpose of the unlock. Sig_{user} is the user's signature and the smart contract obtains the state ledger handle variable, ledgerStub.
2) Obtain an MSL object from the state ledger via ledgerStub. Validate the user's signature and identity.
3) Query the DAP for the dataset VM agent set and the owner to the AT through owner and agentList. Determine whether user access requests can meet DAP. If satisfied, return an True; otherwise, sign a proxy request to an agent whose DAP is not satisfied. Similarly, when the request does not match the DAP, the system asks for owners' signatures.
4) The smart contract collects signatures collected from the prior steps and validates all signatures. If all signatures are validated, return *True*; otherwise, the return value is a *False*.

Time Complexity Analysis of Algorithm 2: We briefly analyze the time complexity of the lock control algorithm. We assume that the length of the VM's locking AgentTable is N, and the number of all agents is M. To query whether the current agent is in the lock AgentTable, we need to traverse all the elements in the lock AgentTable, so the number of times to execute the for loop is N. If the current agent is not in the lock AgentTable, we collect the signatures of all agents and verify them. The time complexity of Algorithm 2 is $O(N + M)$.

VI. Security Analysis and Experiment Evaluations

A. Security Analysis

In this section, we analyze the security of the proposed BMSL-UAC scheme based on the threat model proposed by SectionIII-B-B.

1) Spoofing Attacks: In the threat model, although the adversary obtains the public keys of other users participating in the access control system, the adversary cannot pretend to be an authorized user in the system to obtain access control rights. If the adversary pretends to be a legitimate user in the system with the public key pk to access some VMs. First, the adversary should construct an access request through the proxy and sign it with the private key sk. Second, the smart contract deployed to the consortium blockchain uses pk to verify the signature of the request, and the VM will be unlocked only after the signature is passed. However, the adversary cannot obtain the private key of the authorized user and the adversary cannot pretend to be a legitimate user in the system to obtain the unlocking information of the VM and access the VM. In addition, the transport layer security (TLS) protocol ensures the communication security between the consortium chain node and the client, so this scheme can resist spoofing attacks.

2) Replay Attacks: The access request of the BMSL-UAC scheme proposed in this paper is sent in the form of a transaction, which is finally stored on the consortium blockchain. The txid of the transaction is the unique identifier of the access request. Therefore, the consortium blockchain nodes can verify whether the transaction is a replay attack based on the txid of the transaction. If this transaction is a replay attack, the consortium blockchain node marks it as invalid for rejecting the user request.

3) Tampering Attacks and DoS Attacks: The PBFT consensus mechanism and digital signature technology in the consortium blockchain ensure the authenticity and consistency of the access control data on the blockchain and can be guaranteed when the malicious node does not exceed $(n-1)/3$. Therefore, this scheme can effectively prevent tampering attacks and denial of service (DoS) attacks.

In addition, we also analyze the security of the unlocking mechanism of the BMSL-UAC scheme. Assume that there exists a dataset VM_i, and a state ledger of blockchain stores VM_i's AT and MSL (Lock = {VMId, owner, agentList, user}). When a user sends system the request R while assuming that R does not match $\text{policy}_i, i \in (0, 1, \ldots, n)$. Assume that there exists a request R does not satisfy policy_i, such that we can deduce that there also exists a R not matching the policy set $\{\text{policy}_r, \ldots, \text{policy}_i, \ldots, \text{policy}_n\}$, $(r \leqslant i)$.

The system assesses the request R following the unlock mechanism. Find out the first agent (agent_j, $j = r - 1$) that lets R satisfy its DAP. There are two situations for j, namely $j \geqslant 0$ and $j =$ nil. When $j \geqslant 0$, the request R satisfy the policy set $\{\text{policy}_0, \text{policy}_1, \ldots, \text{policy}_j\}$. That is to say, the set $\{\text{owner}, \text{agent}_1, \ldots, \text{agent}_j\}$ approves the request R and the system does not need to inquire their responses. This situation fits in the Rule 2 of the unlock mechanism. The system only needs signatures from those agents who do not approve R, $\{\text{agent}_r, \text{agent}_{r+1}, \ldots, \text{agent}_n\}$. When $j =$ nil, the set approving R is an $\emptyset$ and this situation fits in the Rule 4 of the unlock mechanism. The system will ask and validate signatures from $\{\text{owner}, \text{agent}_1, \ldots, \text{agent}_n\}$.

Therefore, when a user's request R cannot satisfy (some) agent(s) or DAP (made by the owner), the system will notice

TABLE II
RESOURCE OCCUPANCY AT A TRANSACTION SENDING RATE OF 50/100 TPS

TYPE	NAME	Memory(max) (MB)	Memory(avg) (%)	CPU(max) (%)	CPU(avg) (%)	Traffic In (MB)	Traffic Out (MB)
Process	node local-client.js(avg)	100.5 / 106.1	93.3 / 103.0	55.79 / 30.23	19.41 / 18.10	- / -	- / -
Docker	dev-peer0.org2.exa...le-v0	6.5 / 6.6	5.8 / 6.5	12.40 / 11.56	8.49/6.13	1.8 / 0.858	1.017/0.459
Docker	dev-peer1.org1.exa...le-v0	6.3 / 6.4	5.6 / 6.4	8.96 / 10.68	6.19 / 5.7	1.3 / 0.855	0.758 / 0.458
Docker	dev-peer1.org2.exa...le-v0	6.0 / 6.5	5.0 / 6.3	3.84 / 11.62	2.47 / 6.49	0. 496 / 1.1	0.281 / 0.577
Docker	dev-peer0.org1.exa...le-v0	6.2 / 6.5	5.4 / 6.4	6.70 / 13.87	4.81 / 7.62	0. 967 / 1.2	0. 548 / 0.584
Docker	peer1.org2.example.com	91.1 / 96.0	88.0 / 93.1	45.77 / 103.01	32.07 / 77.33	4.3 / 5.1	0.820 / 4.9
Docker	peer0.org2.example.com	79.5 / 97.4	76.3 / 85.8	80.01 / 91.85	56.61 / 67.44	6.0 / 4.7	17.5 / 8.7
Docker	peer0.org1.example.com	93.9 / 99.8	90.7 / 96.1	68.58 / 109.87	43.14 / 75.38	4.8 / 5.1	5.3 / 8.8
Docker	peer1.org1.example.com	92.2 / 97.3	88.8 / 94.5	64.73 / 97.08	46.60 / 65.92	5.3 / 4.7	2.3 / 1.5
Docker	ca_peerOrg1	6.5 / 6.5	6.5 / 6.5	0.01 / 0.00	0.00 / 0.00	0 / 0	0 / 0
Docker	ca_peerOrg2	6.5 / 6.5	6.5 / 6.5	0.01 / 0.00	0.00 / 0.00	0 / 0	0 / 0
Docker	orderer.example.com	32.6 / 50.4	24.0 / 42.8	22.87 / 41.43	16.91 / 24.32	4.5 / 4.3	15.5 / 15.3
Docker	simplenetwork_ca_1	6.7 / 6.7	6.7 / 6.7	0.00 / 0.00	0.00 / 0.00	0 / 0	0 / 0
Docker	simplenetwork_peer_1	0 / 0	0 / 0	0.00 / 0.00	0.00 / 0.00	0 / 0	- / -

relevant subjects and ask for signatures. The final decision on the access request will be made based on a consensus.

B. Experimental Evaluation

In this section, we present experimental evaluations mainly related to system performance. We simulate a system of heterogeneous data institutions in the proposed metaverse through a consortium blockchain network of Hyperledger. The specific details of this experiment are described below.

The configuration of the simulation was presented in the following. Hardware configuration included operating system Ubuntu 16.04.1; system kernel version was 4.15.0-50-generic; a central processing unit (CPU) Intel(R) Core(TM) i5-4210U CPU @1.70 GHz; a memory of 4 GB. We used Docker containers to simulate physical nodes and constructed Hyperledger Fabric for consortium blockchain. To be specific, we configured two organizations for the consortium blockchain system. Each organization configured two blockchain nodes and one CA node.

In the setup phase of the experiment, we use Docker containers to simulate physical blockchain nodes. Organizations were set up in the experimental species, where each organization represented a data institution. For each organization, we launched two peer containers of the superclass structure as endorsing nodes. Peer nodes provided by multiple organizations together build a trusted execution environment. We also set up a Hyperledger Fabric CA container to simulate each organization's certificate authority, issuing identity certificates and key pairs for users and nodes of the blockchain system. Additionally, we started a client container as a user in order to send transactions to the consortium blockchain nodes.

We chose the solo consensus module in Hyperledge Fabric to implement the consensus in consortium blockchain. Smart contracts are based on the Fabric chaincode mechanism, written in Golang and deployed on consortium blockchain nodes. When the system launches, Fabric creates a Docker container that provides a separate operating environment for chaincode. Users interact with the consortium blockchain by registering the client and sending transaction requests to the system to trigger the chain code or receive the return message. We mainly evaluate the performance of this scheme on the Caliper from two aspects: resource occupancy and system throughput.

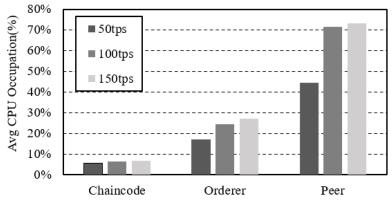


Fig. 5. Average CPU occupation at transaction send rate of 50/100/150 tps.

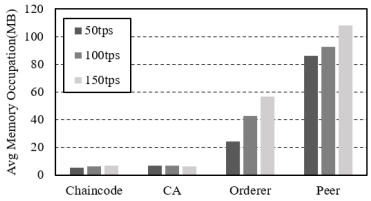


Fig. 6. Average memory occupation at transaction send rate of 50/100/150 tps.

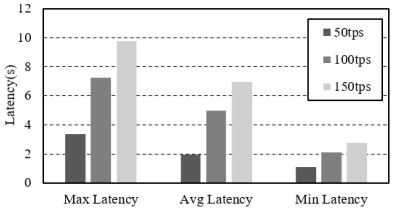


Fig. 7. Transaction processing latency at transaction deliver rates of 50/100/150 tps.

In the resource occupancy experiment, we send 1000 transaction requests to the consortium blockchain, including 50, 100, and 150 tps. Due to page limitations, we only present some experimental results and results in this section. Table II are the experimental results obtained from the transaction sending efficiency at 50 and 100 tps, respectively. From Table II, we get that the resource usage and traffic of this scheme are within the acceptable range.

To clearly describe the resource occupancy of this scheme, we analyze the average occupancy of CPU and memory. From Fig. 5, we can get the average CPU occupation of chaincode, orderer, and peer Docker when the transaction sending rate is 50, 100, and 150 tps, respectively. When the transaction

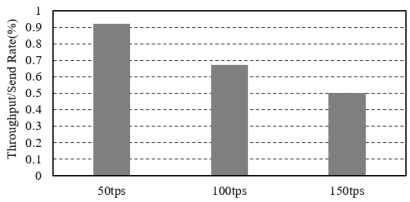


Fig. 8. Throughput/send rate at transaction send rates of 50/100/150 tps.

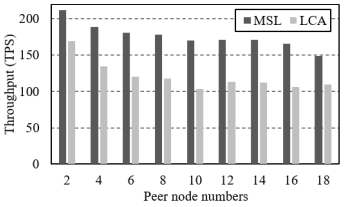


Fig. 9. Throughput at peer node numbers.

sending rate is 150 tps, the average CPU occupation of chaincode is only 6.57%. When the transaction sending rate is from 100 to 150 tps, the average CPU usage of the orderer only increases by 10% and the average CPU occupation of the peer node only increases by 2%. The results show that the CPU resource occupancy rate is within an appropriate range.

From Fig. 6, we obtain the average memory of chaincode, CA, order, and peer Docker when the transaction sending rate is 50, 100, and 150 tps. When the transaction sending rate is 150 tps, the average memory occupation of chaincode and CA is about 6 MB. The peer node occupies the most memory. When the transaction sending rate is 150 tps, the memory of peer Docker is only 108 MB. The results show that the memory resource occupancy rate is within an appropriate range.

Then we measured the impact of the send rate of transactions on the performance of latencies and throughput of smart contracts, where we fixed the network scale of the experiment with three organizations and two endorsement nodes for each organization. Figs. 7 and 8 described the evaluation results. The results showed that the latencies of the system to process transactions were increasing as the send rate of transactions increased and the throughput was on a downward trend. This phenomenon could be caused by Fabric's nonparallel sorting consensus mechanism. This analysis was supported by the fact that the resource occupancy of the Docker container was associated with the rate of transaction delivery.

Furthermore, we experimentally evaluate the impact of different blockchain nodes and network scales on throughput performance and latency. We set up the configuration for two organizations, and we started two endorsing nodes and one CA node for each organization. Additionally, we configured each organization with a different number of peer nodes. In the simulation experiments, we set the fixed transaction sending rate to 100 tps. The experimental results are shown in Figs. 9 and 10. The results show that although the throughput decreases slowly and the latency increases slowly with the increase of the number of peer nodes, they are all within a

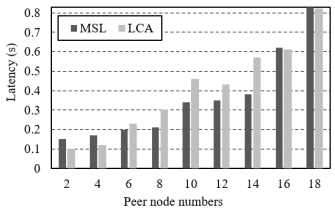


Fig. 10. Latency at peer node numbers.

reasonable range. Therefore, this scheme can provide UAC for the metaverse.

In summary, the traditional centralized attribute-based access control has a single point of failure problem and is not suitable for distributed metaverse scenarios. This paper proposes a multisignature lock access control scheme based on distributed consortium blockchain to provide UAC for the metaverse. This scheme proposes a distributed access control mechanism based on traditional attribute access control and consortium blockchain, which realizes the full life-cycle management and traceability of access information. Furthermore, in order to protect the privacy of users, we propose a multisignature lock mechanism to protect the privacy of information. Security analysis and simulation results show that this scheme can resist spoofing attacks, replay attacks, tampering attacks, and DoS attacks. In addition, the resource consumption, throughput, and latency of this scheme are all within a reasonable range. Therefore, compared with previous work, we not only achieve full-lifecycle traceable and secure distributed multisignature lock access control, but also provide UAC for the metaverse, promoting the realization of the metaverse.

VII. Conclusion

This BMSL-UAC scheme mainly solved the problem of lack of control in the process of data sharing among agencies in the metaverse environment. To solve this problem, the datasets of each institution were mapped to the VM of the dataset in the data center and the multisignature lock mechanism is used to realize the access control of the data of each institution. In addition, this scheme of our proposed realized the tracking and governance of the whole life cycle of data based on the consortium blockchain. The experimental results showed that the UAC scheme proposed in this paper has a better performance.

This scheme belonged to the distributed architecture layer in the metaverse technology architecture and provides an effective solution for UAC in the metaverse. In the future, we can further combine this scheme with other technical layers of the metaverse to realize a privacy-safe UAC metaverse system.

References

[1] M. Stylianos, "Metaverse," *Encyclopedia*, vol. 2, no. 1, pp. 486–497, 2022.

[2] S.-M. Park and Y.-G. Kim, "A metaverse: Taxonomy, components, applications, and open challenges," *IEEE Access*, vol. 10, pp. 4209–4251, 2022.

[3] H. Xu, Z. Li, Z. Li, X. Zhang, Y. Sun, and L. Zhang, "Metaverse native communication: A blockchain and spectrum prospective," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, May 2022, pp. 7–12.

[4] M. Armbrust et al., "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, 2010, doi: 10.1145/1721654.1721672.

[5] L. Chang et al., "6G-enabled edge AI for metaverse: Challenges, methods, and future research directions," *J. Commun. Inf. Netw.*, vol. 7, no. 2, pp. 107–121, Jun. 2022.

[6] M. F. Bari et al., "Data center network virtualization: A survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 2, pp. 909–928, 2nd Quart., 2013.

[7] M. Nofer, P. Gomber, O. Hinz, and D. Schiereck, "Blockchain," *Bus. Inf. Syst. Eng.*, vol. 59, no. 3, pp. 183–187, Mar. 2017.

[8] J. Sunny, N. Undralla, and V. M. Pillai, "Supply chain transparency through blockchain-based traceability: An overview with demonstration," *Comput. Ind. Eng.*, vol. 150, pp. 106–895, Dec. 2020.

[9] X. Xu, Q. Lu, Y. Liu, L. Zhu, H. Yao, and A. V. Vasilakos, "Designing blockchain-based applications a case study for imported product traceability," *Future Gener. Comput. Syst.*, vol. 92, pp. 399–406, Mar. 2019.

[10] Q. Yang, Y. Zhao, H. Huang, Z. Xiong, J. Kang, and Z. Zheng, "Fusing blockchain and AI with metaverse: A survey," *IEEE Open J. Comput. Soc.*, vol. 3, pp. 122–136, 2022.

[11] F. Tang, X. Chen, M. Zhao, and N. Kato, "The roadmap of communication and networking in 6G for the metaverse," *IEEE Wireless Commun.*, early access, Jun. 24, 2022, doi: 10.1109/MWC.019.2100721.

[12] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: A survey," *Int. J. Web Grid Services*, vol. 14, no. 4, pp. 352–375, 2018.

[13] A. E. Saddik, "Digital twins: The convergence of multimedia technologies," *IEEE Multimed.*, vol. 25, no. 2, pp. 87–92, Apr./Jun. 2018.

[14] K. Gai et al., "Digital twin-enabled AI enhancement in smart critical infrastructures for 5G," *ACM Trans. Sensor Netw.*, vol. 18, no. 3, pp. 1–20, Aug. 2022.

[15] K. Gai, Y. Zhang, M. Qiu, and B. Thuraisingham, "Blockchain-enabled service optimizations in supply chain digital twin," *IEEE Trans. Services Comput.*, early access, Jul. 19, 2022, doi: 10.1109/TSC.2022.3192166.

[16] M. A. I. Mozumder, M. M. Sheeraz, A. Athar, S. Aich, and H.-C. Kim, "Overview: Technology roadmap of the future trend of metaverse based on IoT, blockchain, AI technique, and medical domain metaverse activity," in *Proc. 24th Int. Conf. Adv. Commun. Technol. (ICACT)*, Feb. 2022, pp. 256–261.

[17] T. R. Gadekallu et al., "Blockchain for the metaverse: A review," 2022, *arXiv:2203.09738*.

[18] K. Gai, J. Guo, L. Zhu, and S. Yu, "Blockchain meets cloud computing: A survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 2009–2030, 3rd Quart., 2020.

[19] P. Zheng, Q. Xu, Z. Zheng, Z. Zhou, Y. Yan, and H. Zhang, "Meepo: Sharded consortium blockchain," in *Proc. IEEE 37th Int. Conf. Data Eng. (ICDE)*, Apr. 2021, pp. 1847–1852.

[20] W. Zou et al., "Smart contract development: Challenges and opportunities," *IEEE Trans. Softw. Eng.*, vol. 47, no. 10, pp. 2084–2106, Oct. 2021.

[21] X. Yan, S. Wang, and K. Gai, "A semantic analysis-based method for smart contract vulnerability," in *Proc. IEEE IEEE 8th Int. Conf. Big Data Secur. Cloud (BigDataSecurity) Int. Conf. High Perform. Smart Comput., (HPSC) IEEE Int. Conf. Intell. Data Secur. (IDS)*, May 2022, pp. 23–28.

[22] C. Schneidewind, I. Grishchenko, M. Scherer, and M. Maffei, "EThor: Practical and provably sound static analysis of Ethereum smart contracts," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2020, pp. 621–640.

[23] Y. Wang et al., "A survey on metaverse: Fundamentals, security, and privacy," *IEEE Commun. Surveys Tuts.*, early access, Sep. 7, 2022, doi: 10.1109/COMST.2022.3202047.

[24] C. T. Nguyen, D. T. Hoang, D. N. Nguyen, and E. Dutkiewicz, "MetaChain: A novel blockchain-based framework for metaverse applications," in *Proc. IEEE 95th Veh. Technol. Conf. (VTC-Spring)*, Jun. 2022, pp. 1–5.

[25] M. Çolak, I. Kaya, B. Özkan, A. Budak, and A. Karaşan, "A multicriteria evaluation model based on hesitant fuzzy sets for blockchain technology in supply chain management," *J. Intell. Fuzzy Syst.*, vol. 38, no. 1, pp. 935–946, Jan. 2020.

[26] S. S. Hasan, N. H. Sultan, and F. A. Barbhuiya, "Cloud data provenance using IPFS and blockchain technology," in *Proc. 7th Int. Workshop Secur. Cloud Comput. (SCC)*, 2019, pp. 5–12.

[27] E. Nyaletey, R. M. Parizi, Q. Zhang, and K.-K.-R. Choo, "BlockIPFS—Blockchain-enabled interplanetary file system for forensic and trusted data traceability," in *Proc. IEEE Int. Conf. Blockchain (Blockchain)*, Jul. 2019, pp. 18–25.

[28] G. Baralla, A. Pinna, and G. Corrias, "Ensure traceability in European food supply chain by using a blockchain system," in *Proc. IEEE/ACM 2nd Int. Workshop Emerg. Trends Softw. Eng. Blockchain (WETSEB)*, May 2019, pp. 40–47.

[29] J. Lin, Z. Shen, A. Zhang, and Y. Chai, "Blockchain and IoT based food traceability for smart agriculture," in *Proc. 3rd Int. Conf. Crowd Sci. Eng. (ICCSE)*, 2018, pp. 1–3.

[30] L. Cocco and K. Mannaro, "Blockchain in Agri-food traceability systems: A model proposal for a typical Italian food product," in *Proc. IEEE Int. Conf. Softw. Anal., Evol. Reeng. (SANER)*, Mar. 2021, pp. 669–678.

[31] G. T. S. Ho, Y. Tang, K. Y. Tsang, V. Tang, and K. Y. Chau, "A blockchain-based system to enhance aircraft parts traceability and trackability for inventory management," *Expert Syst. Appl.*, vol. 179, pp. 101–115, Oct. 2021.

[32] K. Gai, Y. She, L. Zhu, K. R. Choo, and Z. Wan, "A blockchain-based access control scheme for zero trust cross-organizational data sharing," *ACM Trans. Internet Technol.*, 2022, doi: 10.1145/3511899.

[33] M. Laurent, N. Kaaniche, C. Le, and M. V. Plaetse, "A blockchain based access control scheme," in *Proc. 15th Int. Joint Conf. e-Bus. Telecommun.*, 2018, pp. 206–220.

[34] M. Li and Y. Qin, "Scaling the blockchain-based access control framework for IoT via sharding," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2021, pp. 1–6.

[35] D. Huang, C.-J. Chung, Q. Dong, J. Luo, and M. Kang, "Building private blockchains over public blockchains (PoP): An attribute-based access control approach," in *Proc. 34th ACM/SIGAPP Symp. Appl. Comput.*, Apr. 2019, pp. 355–363.

[36] V. E. Arasi, K. I. Gandhi, and K. Kulothungan, "Auditable attribute-based data access control using blockchain in cloud storage," *J. Supercomput.*, vol. 78, no. 8, pp. 10772–10798, May 2022.

[37] S. Wang, Y. Zhang, and Y. Zhang, "A blockchain-based framework for data sharing with fine-grained access control in decentralized storage systems," *IEEE Access*, vol. 6, pp. 38437–38450, 2018.

[38] P. Wang, N. Xu, H. Zhang, W. Sun, and A. Benslimane, "Dynamic access control and trust management for blockchain-empowered IoT," *IEEE Internet Things J.*, vol. 9, no. 15, pp. 12997–13009, Aug. 2022.

[39] X. Qin, X. Huang, Z. Yang, and X. Li, "A blockchain-based access control scheme with multiple attribute authorities for secure cloud data sharing," *J. Syst. Archit.*, vol. 112, pp. 1–11, Jan. 2021.

Keke Gai (Senior Member, IEEE) received the Ph.D. degree in computer science from Pace University, New York, NY, USA.

He is currently a Professor at the School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing, China. He has published four books and more than 150 peer-reviewed journal/conference papers. His research interests include cyber security, blockchain, privacy-preserving computation, decentralized identity.

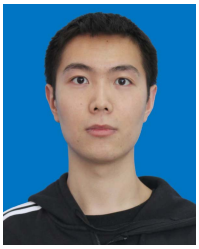
Dr. Gai has been granted ten Best Paper awards in recent years. He serves as an Associate Editor for a few journals, including IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING (TDSC), *Journal of Parallel and Distributed Computing* (JPDC), Future Generation Computer Systems (FGCS), and so on. He also serves as a Co-Chair of IEEE Technology and Engineering Management Society's (TEMS's) Technical Committee (TC) on Blockchain and Distributed Ledger Technologies (DLT), a Standing Committee Member at China Computer Federation (CCF) Blockchain Committee, a Secretary-General at IEEE Special Technical Community in Smart Computing (STCSC).

Shuo Wang is currently pursuing the Ph.D. degree in electronic information with the School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing, China.

Her current research interests include quantum cryptography, blockchain, and cloud computing.

Hui Zhao received the B.E. degree from Xi'an Technology University, Xi'an, Shanxi, China, in 2000, the M.S. degree from Henan University, Kaifeng, Henan, China, in 2008, and the Ph.D. degree from Pace University, New York, NY, USA, in 2018.

He is currently an Associate Professor with the Software School, Henan University, and the Director of the Educational Information Technology Laboratory. His research interests include cybersecurity and artificial intelligence (AI).



Yufeng She received the master's degree in computer technology from the School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China, in 2022.

His research interests include blockchain-based access control and multisignature.

Zijian Zhang (Member, IEEE) received the Ph.D. degree from the Beijing Institute of Technology, Beijing, China, in 2012.

He is an Associate Professor with the School of Computer Science and Technology, Beijing Institute of Technology. His research interests include authentication and key agreement, behavior recognition, and privacy preserving.

Liehuang Zhu (Senior Member, IEEE) received the B.E. and M.E. degrees from Wuhan University, Wuhan, China, in 1998 and 2001, respectively, and the Ph.D. degree in computer science from the Beijing Institute of Technology (BIT), Beijing, China, in 2004.

He is currently a Professor with the School of Cyberspace Science and Technology, BIT. He has published more than 100 peer-reviewed articles, including 10 + IEEE/ACM TRANSACTIONS articles (IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE TRANSACTIONS ON SMART GRID, Information Sciences (INS), and *IEEE Network*). His research interests include cybersecurity and cloud computing.

Dr. Zhu has been granted a number of IEEE best paper awards, for example, IWQoS 17' and TrustCom 18'.