Secure and Distributed IoT Data Storage in Clouds Based on Secret Sharing and Collaborative Blockchain

Na Wang, Junsong Fu, Shancheng Zhang, Zheng Zhang, Jiawen Qiao, Jianwei Liu, *Member, IEEE*, and Bharat K. Bhargava, *Life Fellow, IEEE*

***Abstract*— With the rapid development of 5G/6G, most Internet of Things (IoT) devices will embrace wireless connection in the near future. A public concern is how to securely organize, store and retrieve data generated from IoT devices. Many cloud-based IoT data storage schemes have been proposed recently. However, for an untrusted or vulnerable cloud server, the stored IoT data can be easily accessed, modified and even destroyed given that the IoT data are stored in total centralization. Moreover, the servers in a cloud are generally homogeneous and thus vulnerable to attacks. For improvements, we design a novel framework for secure and efficient IoT data storage based on secret sharing and a collaborative blockchain. First, an ultra-lightweight secret sharing algorithm is designed to map original messages generated by IoT devices to a set of shorter message shares. Second, all the shares of IoT messages are separately delivered to different clouds for storage. To guarantee the security of shares, the delivery is notarized on a proposed blockchain. Specifically, both hash values of the shares and their information of location are embedded in blocks which are then chained to form a blockchain. Third, we create a balanced index structure about the shares for each cloud storage node based on the information in the blockchain, and we also propose a depth-first data search algorithm to improve IoT data retrieval efficiency. Theoretical analysis and simulation results illustrate that our scheme can store and retrieve the IoT data securely and efficiently.**

***Index Terms*— Cloud computing, Internet of Things, secure data storage, secret sharing, blockchain.**

I. Introduction

WIDE applications of the Internet of Things (IoT) and subsequent explosive data have raised some serious issues in data storage and security in various industries. For data storage, existing solutions [1], [2] choose to store the data on remote devices such as cloud servers to avoid strictly limited localization. Besides this unlimited feature, cloud computing [3] is popular as well because of its valuable properties such as on-demand service, scalability and stability, and thus it becomes a promising tool to better data storage and move IoT forward [4]. They together make a new paradigm, termed as Cloud of Things (CoT) [5]. In between, high-speed telecommunication like 5G/6G is necessary so that it is possible for IoT data to be continuously transmitted to a cloud storage system in time. Quite a few IoT data storage schemes have been proposed based on cloud platforms [6], [7], [8]. Present cloud-based storage approaches can be roughly classified into four categories: relational IoT database management system [6], NoSQL IoT data storage system [7], Hadoop-based IoT data storage system [9], and IoT data storage scheme based on resource description framework [8]. Several related cloud-based commercial IoT data storage platforms are also developed, such as Apache IoTDB [10], AWS IoT platform [11], Ali Cloud IoT platform [12] and Azure IoT platform [13]. It is worth noting that cloud storage platforms have an "honest but curious" feature, meaning data on these platforms are possibly exposed to certain attacks [14], so it is crucial to have a storage solution that can secure outsourced data.

However, these platforms above are not capable of achieving completely distributed storage, and the management system of such platforms is centralized, so data security cannot be guaranteed when its main server is breached. To properly secure IoT data in terms of confidentiality, integrity and availability, a storage system should be of the following properties:

- Each IoT message is supposed to be distributively stored on multiple storage entities rather than just one server. Overall confidentiality of the messages can be protected, even though some of the entities are compromised by an adversary.
- All the uploaded information by IoT devices is unmodifiable by a storage system or adversaries without authorization. Data receiving operations are not deniable by cloud nodes.
- Data storage with some redundancy are favored so that original IoT messages can be accurately recovered, even though some nodes fail in a storage system.
- Authorized data users can efficiently retrieve the IoT messages of interest from a storage system and examine correctness of related data.

In order to understand if a storage system meet the four properties above, we investigate security of data transmission and storage process separately. First, the secret sharing

Manuscript received 15 February 2022; revised 21 July 2022 and 24 September 2022; accepted 28 October 2022; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor J. S. Sun. Date of publication 14 November 2022; date of current version 18 August 2023. This work was supported in part by the National Natural Science Foundation of China under Grant 62001055, Grant 62102017, Grant 61932014, and Grant 61972018; and in part by the Fundamental Research Funds for the Central Universities under Grant YWF-22-L-1273. *(Corresponding author: Junsong Fu.)*

Na Wang, Shancheng Zhang, Zheng Zhang, Jiawen Qiao, and Jianwei Liu are with the School of Cyber Science and Technology, Beihang University, Beijing 100191, China (e-mail: nawang@buaa.edu.cn; zscbuaa@buaa.edu.cn; zz_shiwo@buaa.edu.cn; selina@buaa.edu.cn; liujianwei@buaa.edu.cn).

Junsong Fu is with the School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: fujs@bupt.edu.cn).

Bharat K. Bhargava is with the Department of Computer Science, Purdue University, West Lafayette, IN 47906 USA (e-mail: bbshail@purdue.edu).

Digital Object Identifier 10.1109/TNET.2022.3218933

scheme [15] splits a secret into multiple shares, and a specified number of multiple shares can recover the original secret. Such a scheme also achieves the purpose of risk dispersion and intrusion tolerance by dividing secret and increasing redundancy, and thus suits to enhance security of data transmission. Existing schemes with secret sharing, such as (t, n)-threshold secret sharing scheme [16] and improved Shamir secret sharing scheme [17], are optimized in terms of scalability and space efficiency, but there is still room for improvement in computing overhead. In addition, after completion of secure transmission of data, it is necessary to consider safe storage and efficient use of data. Distributed storage effectively protects data security even when cloud nodes are damaged, and verification of outsourced data's integrity defends well against malicious tampering and storage errors. The two protective measures are well equipped with the technology of blockchain [18], which refers to a kind of decentralized distributed ledger with its own tamper-proof property. A blockchain system [19] is treated as a distributed ledger with the property of tamper-proof and each transaction is completely stored in the ledger. Each node in the blockchain system maintains a duplicate of the ledger and hence the system is not under control of any centralized entity. However, existing blockchain schemes that combine IoT and blockchain [20] need to be improved for secure storage when dealing with large resource consumption of consensus mechanism and small transaction throughput.

Aiming at the problems, computing overhead from secret sharing, and large resource consumption and low throughput from blockchain technologies, we creatively design a secure and distributed IoT data storage scheme. In our system, a set of heterogeneous cloud nodes rather than one cloud is employed. Each node is of triple roles, i.e., an IoT data storage node, a blockchain node and a data retrieval server. By cooperating with one another, these nodes constitute a whole system that can provide a set of secure services including IoT data collection, storage and retrieval. Specifically, we first design an ultra-lightweight (t, n)-secret sharing scheme based on congruence equations, where a message is mapped to n shorter length shares. The original message is recoverable with any t $(t \leq n)$ shares of them or more, while it is impossible to be recovered with less than t shares. Then, n shares are sent to n cloud nodes and stored by these nodes. We adopt the $(4, 7)$-secret sharing scheme with the best performance in the experiment, and introduce 10 distributed cloud nodes with a quantity greater than n in order to store these shares in a distributed manner. Since each node stores at most one share of the message, if an adversary cannot tamper with the shares in more than $n - t$ cloud nodes, users can still recover the original information according to the secret sharing scheme. Therefore, the security of IoT data can be guaranteed even if the adversary breaks into some cloud nodes. Second, our scheme deploys a blockchain system on distributed cloud nodes to record location and hash value of shares, so that each node is undeniable to the received messages and capable of verifying the integrity of stored messages. When part of the nodes are corrupted, we can recover the message by finding the other shares that is lost based on the storage location on a blockchain. Moreover, we create a new blockchain consensus mechanism, in which nodes cooperate to mine new blocks instead of competing with one another seen in traditional blockchains, which significantly improves overall storage efficiency. Finally, we construct a search structure for cloud nodes that balances retrieval efficiency and storage space, so as to meet the needs of users to retrieve data quickly.

A. Novelty

The novelties of the work against existing schemes are as follows:

- We creatively propose a distributed IoT data cloud storage scheme based on secret sharing and blockchain. This innovative scheme improves security and storage efficiency relative to traditional centralized storage schemes and yields a balanced load on each storage entity.
- In the data storage system, a new type of private blockchain system is designed to ensure reliability and immutability of stored data. An introduction of a flow token mechanism enables cloud nodes to generate new blocks through cooperation rather than competition, further improving data storage efficiency.
- For massive distributed storage data, an index tree structure matching blockchain storage and a corresponding depth-first algorithm are constructed to greatly improve retrieval efficiency and thus enhance availability of the scheme.

B. Contributions

The main contributions of this paper are summarized as follows:

- An ultra-lightweight secret sharing scheme is particularly designed for resource-limited IoT applications, where both computational complexity and average length of shares decrease.
- We design a new blockchain system for the distributively stored shares. Any modification of a share can be easily detected and located while information of shares' location gets disclosed.
- An IoT data retrieval scheme is proposed. Users can efficiently query the data, generated within a certain period, from an IoT terminal device through this scheme.
- Theoretical analysis and a set of simulations are conducted to illustrate security and efficiency of our scheme.

C. Article Structure

The rest of this paper is organized as follows. We summarize the related work in Section II. The overall system model is presented in Section III. The modules of the system including ultra-lightweight secret sharing, the blockchain of IoT data shares and efficient IoT data retrieval are discussed in Section IV, V and VI, respectively. We analyze the security of the system in Section VII and evaluate its efficiency in Section VIII. At last, Section IX concludes this paper.

II. Related Work

Cloud storage provides low-cost mass data outsourcing storage services. A large number of cloud platform based IoT data storage solutions have been proposed [21], [22], [23], [24]. Wang et al. [21] studied a secure cloud-assisted IoT data management method that protects data confidentiality when collecting, storing and accessing IoT data. However, the data in this scheme is vulnerable to tampering attacks. In order to ensure integrity of stored data, Rashmi et al. [22] proposed their cloud storage scheme, resistible to replacement attack initiated by malicious server, based on homomorphic hash

algorithm, using Merkle hash trees to help locate each dynamic operation on IoT data. With access cost in consideration, Wu et al. [24] presented an IoT terminal node with the load balancing cloud storage data distribution optimization system. Their approach improves availability and data processing speed. For expansive applications, Jiang et al. [25] designed a data storage framework of cloud computing platform oriented to IoT, which not only efficiently stores massive IoT data, but also integrates structured and unstructured data. Further, Lin et al. [26] proposed an efficient tree-like storage scheme for IoT data in cloud storage, which stores unstructured IoT data in a structured way.

Again, a centralized system does not guarantee data security given a main server is breached. In order to improve security of data, many solutions [19], [27], [28] have adopted the idea of blockchain. Uthayashangar et al. [1] combined cloud and blockchain technologies to create a secure file storage system. This system is resistible to a brute force attack, but a more complex blockchain system is introduced. Kim et al. [19] studied the blockchain compression consensus algorithm in an IoT environment. They realized distributed storage of a large amount of information on lightweight devices by compressing blockchain in each IoT device to enlarge storage capacity. However, these schemes have insufficient protection for system security. Mohammed et al. [27] designed a hybrid framework between blockchain and IoT devices, aiming to use blockchain technology to protect security of data transmission in IoT and prevent denial of service (DoS) attacks. At the same time, Xu et al. [28] proposed a non-trusted storage IoT data protection framework based on blockchain to address data security.

In order to simultaneously meet requirements of security and efficiency of IoT data cloud storage, Wiraatmaja et al. [29] designed a layered architecture that combines blockchain with tamper-proof decentralized storage. By migrating metadata from the blockchain to a distributed database for storage while maintaining the tamper-proof functionality of the blockchain, they achieved cost-effective access control. Zhang et al. [30] proposed a solution for blockchain-based mobile edge computing with a secure and efficient data storage and sharing scheme that enables low-latency message response to end users. Huang et al. [31] also introduced blockchain technology for the auditing and data transaction scheme of cloud storage in IoT. Their versatile auditing scheme supports efficient data transaction through blockchain on the premise of ensuring data security. In addition to theoretical researches, there are now quite a few blockchain-based data storage technologies, including AWS Blockchain [32], IBM blockchain [33] and Microsoft Azure Blockchain [13]. These technologies have been applied in business. For instances, FoodTrust [34], a farm-to-supermarket tracking system backed by Walmart, and TradeLens [35], a maritime container logistics blockchain backed by Maersk, are built on these commercial blockchain platforms. And BurstIQ, a big data blockchain platform, provides a mature solution to the problem of outsourcing cloud storage and sharing of IoT data [36].

Despite all the applications and progresses, if a small number of nodes in the solutions above are breached, the compromised IoT data, unfortunately, cannot be recovered. Therefore, it is critical for the data upload process to be robust against such breaches. Using the (t, n)-threshold secret sharing scheme in cloud storage, Farhadi et al. [37] proposed a new method to store aggregated data in IoT, which significantly

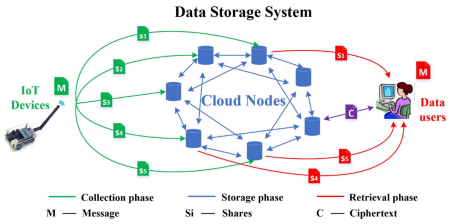

Fig. 1. Framework of IoT data storage system.

improves robustness of data. However, they took no consideration of a "curious" cloud that may attempt to obtain sensitive information from uploaded data. To this end, Galletta et al. [38] evaluated two most common secret sharing algorithms to address privacy and security issues of stored data in remote services and determine their applicability to different environments. However, the scheme in [38] may face tampering attacks that deprive availability of their system. Tan et al. [39] designed an IoT data protection scheme based on blockchain and Shamir secret sharing. This scheme effectively prevents attackers from stealing data and protects encryption keys.

Blockchain plays an important role securing data in these schemes above. However, the majority of them are lack of efficiency in storage and retrieval due to their direct storage on a blockchain. Li et al. [40] first proposed a blockchain-based IoT data cloud storage scheme. By using blockchain miners to perform transaction verification and record auditing with help of certificateless cryptography, IoT data storage and transaction information can be efficiently and effectively implemented. It is well-recognized that a blockchain provides outstanding security and data storage off-chain and transaction information on-chain ensures better operational efficiency. Lin et al. [26] presented a bottom-up tree storage scheme for the combined placement of IoT data after uploaded to a cloud server, which noticeably improves efficiency of data storage and retrieval. In addition, Wan et al. [41] proposed an energy-saving and time-saving multidimensional data index scheme, which utilizes hierarchical index structure and binary space partitioning technology to significantly reduce data query delay.

Although the proposed schemes address security and efficiency in IoT data storage from different dimensions, no overall scheme simultaneously takes into consideration of retrieval and storage efficiency of massive data in a large-scale IoT system with a strong security model. Aiming at the issues of efficiency and security in data uploading and cloud storage, we respectively design a lightweight (t, n)-threshold secret sharing transmission scheme and a blockchain storage structure with cooperation rather than competition. In order to improve data retrieval efficiency, we also design a time-stamped search tree structure and a depth-first search algorithm.

III. Problem Statement

A. System Model

As shown in Fig. 1, there are mainly three types of entities in the whole system including *IoT devices, heterogeneous cloud nodes* and *data users*. They collectively make up the data storage system. Each entity is of its own responsibilities and the related data processing phases of different entities are presented in different colors. For simplicity, we take $(3, 5)$-secret sharing scheme as an example.

As illustrated in the green part of Fig. 1, the *IoT devices* periodically generate information about the monitored

environments, including numerical data, audios and even videos. Each IoT device is assigned with a unique identifier denoted as ID and it is bound with a data user in deployment. They share a symmetric secret key to encrypt and decrypt the delivered information. Once a message M is generated, the IoT device first encrypts it by the secret key and then maps it to a set of shares, i.e., s_1, s_2, s_3, s_4, s_5, which are shorter in length. The shares rather than M are stored in the data storage system.

The blue blocks in the blue part of Fig. 1 represent a set of *heterogeneous cloud nodes*, each with three roles: data storage node, blockchain node, and data retrieval server. First, as a data storage node, the shares of a message M are dispersedly stored in different nodes, where, ideally, the amount of data should be approximately equal to one another. Secondly, to make the stored data tamper-resistant, the IoT data shares and corresponding information are hashed to a hash value with a fixed length. The hash values are then employed to construct the blocks which are appended to the blockchain. Finally, the generated blockchain is shared by all blockchain nodes. It should be noted that the exact values of the shares are not reflected in the blockchain and they are different from the transactions in existing blockchains. The exact structure of the blockchain and update process will be introduced in Section V. As a data retrieval server, its function is embodied in the process of IoT data retrieval, and the details are as follows. It is important to note that even if different roles play as the sub-scheme subjects in the following sections, the entity corresponding to data storage node, blockchain node, and data retrieval server is the cloud node.

In IoT data query process, as shown in the red part of Fig. 1, the *data users* first request the shares' locations of interest, i.e., l_1, l_2, l_3, l_4, l_5, to any cloud node in the system, considering that a blockchain is shared by all the nodes in the system. Then, they request the shares from the nodes where the shares are stored. The original messages are recoverable, given 3 shares or more, i.e., s_1, s_4, s_5, are received. The details of IoT data query will be discussed in Section VI.

B. Threat Model

Cloud Server Threat Model: In our scheme, several cloud platforms are employed, and they are of different owners. There is no centralized manager within cloud nodes. We assume that there is no collusion among them. Each platform is of some vulnerabilities and thus is possible to be mined and exploited by an adversary. Moreover, we assume that different clouds are of different and independent vulnerabilities.

Besides security vulnerabilities, the clouds themselves also tend to collect and infer the IoT data. Similar to the threat model in [3] and [2], each cloud platform is considered as being "honest but curious", which has been widely employed in the field of cloud-based encrypted data storage. Namely, the cloud server can honestly execute the preset instructions, but due to its curiosity, it tends to access original IoT messages and to analyze and infer private information about the data users.

Adversary Threat Model: We assume that an adversary is interested in all the IoT data stored in our system and attempt to steal, falsify and destroy these data. They attack the system by mining and exploiting any vulnerability. As mentioned in the system model, a cloud server node has three identities, from which the threat posed by the adversary's breach of a cloud server is developed. When a cloud node plays as a data storage node, the adversary is able to obtain and modify the data stored in it directly. As for a blockchain node, the adversary can forge transaction information and construct a fake block. As a retrieval server is destroyed, the adversary is capable of capturing the shares requested by a user. However, in this paper, all the cloud nodes in the data storage system are assumed to be heterogeneous and their vulnerabilities of them are different from one another. In other words, the adversary cannot attack a set of nodes in a similar mechanism.

Another way for the adversaries to access IoT data is to capture the shares which can be used to recover original messages. Moreover, the adversaries can also prevent the system from providing data query service by denial of service (DoS) attacks. Once a set of cloud nodes are attacked, all the data users lose connection with them.

Data Users Threat Model: Certain malicious data users may also attempt to access the data without authorization. Specifically, they pretend as valid users to retrieve IoT data from cloud nodes.

IoT Device Threat Model: Prestored coefficients yield higher efficiency and less computation. This prestorage approach is particularly suitable for IoT. Correlation coefficients of a system prestored in IoT devices include selected threshold values (t, n) and a symmetric encryption key. However, it is common for IoT device manufacturers, for cost-effectiveness, to store coefficients that are the same or from a smaller group lacking overall randomness, posing a threat to message recovery. In addition, reverse analysis of IoT devices makes it possible to crack the correlation coefficient information stored in them.

In Section VII, we discuss impact of these threat models on solutions' security in detail. We will start from a CIA triplet, analyze security performance of the scheme under various threat models, and prove that our scheme maintains good security properties under these threat models.

C. Design Goals

The design goals of our IoT data storage framework are summarized as follows:

IoT Data Confidentiality: Confidentiality refers to limiting IoT information access and disclosure to only authorized data users, as well as preventing access by, or disclosure to unauthorized ones.

IoT Data Integrity: Integrity refers to trustworthiness and veracity of IoT information.

IoT Data Availability: Availability means that the system can run normally and provide services to users. For example, in a centralized data storage system, data may totally be lost once a server is destroyed by an adversary.

IoT Data Retrieval Efficiency: Our system allows data users to access stored IoT data in an efficient way.

IV. Distributed IoT Data Storage in the Clouds

A. Mapping IoT Messages to Message Shares Based on Lightweight Secret Sharing

A (t, n)-secret sharing scheme maps a message M to a set of n shares $\{s_1, s_2, \cdots, s_n\}$ in which any subset of $t(t \leq n)$ shares can recover secret M, or otherwise M cannot be recovered. Many classical secret sharing schemes have been proposed in the literatures [42], [43], [44].

Shamir [42] first designed a secret sharing scheme based on polynomial interpolation. Assume that the secret M is a secret

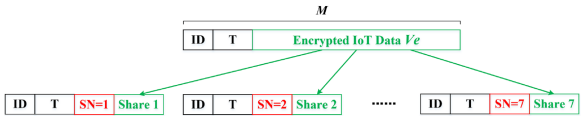

Fig. 2. Secret sharing framework.

number. To divide it into a set of shares $\{s_1, s_2, \cdots, s_n\}$, we pick a random $t-1$ degree polynomial $q(x) = a_0 + a_1 x + \cdots + a_{k-1}x^{k-1}$ in which $a_0 = M$ and $s_1 = q(1), \cdots, s_i = q(i), \cdots, s_n = q(n)$. Then, given any t shares, we first construct the polynomial $q(x)$ and then get M by calculating $q(0)$.

In our system, we denote the original data generated by an IoT device as V, encrypted as follows:

$$V_e = E_k(V), \tag{1}$$

where V_e is the ciphertext of V, E is a proper encryption algorithm, and k is the secret key between the IoT device and its data user. Each message M comprises three parts including the identifier of the IoT device, the timestamp of data generation and the encrypted data V_e. For simplicity, we denote the three parts as ID, T and V_e, respectively.

To improve data security and storage efficiency, we store the message shares rather than M in the system. A $(4, 7)$-secret sharing scheme is employed to construct the secret shares of V_e. As shown in Fig. 2, message M is mapped to 7 message shares and any 4 of them can recover M. Each share comprises four parts, i.e., the identifier of the IoT device, timestamp of birth, serial number of the share, and the secret share of V_e. The identifier ID and timestamp T are directly inherited from M. The serial numbers SN of the shares are selected from 1 to 7 in order.

We treat V_e as a bit stream with length L and it is first divided into four parts, i.e., $\{x_1, x_2, x_3, x_4\}$, with an equal length of $L/4$. If L is indivisible by 4, we append a set of 0 to the end of V_e. Based on x_1, x_2, x_3 and x_4, we construct 7 secret shares $\{s_1, s_2, s_3, s_4, s_5, s_6, s_7\}$ as follows:

$$s_i = x_1 + a^{i-1} \cdot x_2 + b^{i-1} \cdot x_3 + c^{i-1} \cdot x_4 \bmod p, \tag{2}$$

where a, b, c are three different integers larger than 1 and p is a large prime number larger than $2^{L/4}$.

In IoT, terminal devices are strictly limited in resources, such as computation, communication and energy. For convenience, the coefficients in equation (2) can be precalculated and stored in the devices. To construct the shares, we need to execute multiplication operation in total for 21 times, addition operation for 21 times and modulus operation for 7 times. Compared with most existing schemes [42], [45], our scheme is quite lightweight in terms of secret share construction.

The length of the prime number p decides the average length of the shares. The smaller p becomes, the shorter the shares are, and vice versa. To improve data transmission efficiency, we select p as the smallest prime number larger than $2^{L/4}$. In this case, the average length of the shares is approximately $L/4$ which is much shorter than the original messages.

B. Correctness of the Secret Sharing Scheme

Theorem 1: Given a set of at least four shares, we obtain $\{x_1, x_2, x_3, x_4\}$ by solving an equation set and recover V_e.

Proof: Without loss of generality, we first randomly choose four shares s_i, s_j, s_k, s_l where $1 \le i < j < k < l \le 7$. Then, we construct four equations with variables x_1, x_2, x_3, x_4 and demonstrate them in the form of matrix as follows:

$$\begin{pmatrix} s_i \\ \vdots \\ s_l \end{pmatrix} = \begin{pmatrix} 1^{i-1} & \cdots & c^{i-1} \\ \vdots & \ddots & \vdots \\ 1^{l-1} & \cdots & c^{l-1} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_4 \end{pmatrix} = B \begin{pmatrix} x_1 \\ \vdots \\ x_4 \end{pmatrix} \tag{3}$$

Mathematically speaking, we have a unique solution from the equation set if and only if the determinant of B, i.e., $|B|$, is nonzero. We assume that a, b, c are three different positive integers and we need to prove that:

$$\begin{vmatrix} 1 & 1 & 1 & 1 \\ a^i & a^j & a^k & a^l \\ b^i & b^j & b^k & b^l \\ c^i & c^j & c^k & c^l \end{vmatrix} \neq 0. \tag{4}$$

We calculate the value of the determinant as follows:

$$\begin{aligned} \mathrm{D} &= \begin{vmatrix} 1 & 1 & 1 & 1 \\ a^i & a^j & a^k & a^l \\ b^i & b^j & b^k & b^l \\ c^i & c^j & c^k & c^l \end{vmatrix} \\ &= \begin{vmatrix} \frac{b^k - b^i}{a^k - a^i} - \frac{b^j - b^i}{a^j - a^i} & \frac{b^l - b^i}{a^l - a^i} - \frac{b^j - b^i}{a^j - a^i} \\ \frac{c^k - c^i}{a^k - a^i} - \frac{c^j - c^i}{a^j - a^i} & \frac{c^l - c^i}{a^l - a^i} - \frac{c^j - c^i}{a^j - a^i} \end{vmatrix} \\ &= \left(\frac{b^k - b^i}{a^k - a^i} - \frac{b^j - b^i}{a^j - a^i}\right)\left(\frac{c^k - c^i}{a^k - a^i} - \frac{c^j - c^i}{a^j - a^i}\right) \\ &\quad \times \left(\frac{\frac{c^l - c^i}{a^l - a^i} - \frac{c^j - c^i}{a^j - a^i}}{\frac{c^k - c^i}{a^k - a^i} - \frac{c^j - c^i}{a^j - a^i}} - \frac{\frac{b^l - b^i}{a^l - a^i} - \frac{b^j - b^i}{a^j - a^i}}{\frac{b^k - b^i}{a^k - a^i} - \frac{b^j - b^i}{a^j - a^i}}\right) \\ &= \frac{b^k - b^i}{a^j - a^i}\left(\frac{a^j - a^i}{a^k - a^i} - \frac{b^j - b^i}{b^k - b^i}\right)\frac{c^k - c^i}{a^j - a^i}\left(\frac{a^j - a^i}{a^k - a^i} - \frac{c^j - c^i}{c^k - c^i}\right) \\ &\quad \times \left(\frac{\frac{c^{l-i} - 1}{a^{l-i} - 1} - \frac{c^{j-i} - 1}{a^{j-i} - 1}}{\frac{c^{k-i} - 1}{a^{k-i} - 1} - \frac{c^{j-i} - 1}{a^{j-i} - 1}} - \frac{\frac{b^{l-i} - 1}{a^{l-i} - 1} - \frac{b^{j-i} - 1}{a^{j-i} - 1}}{\frac{b^{k-i} - 1}{a^{k-i} - 1} - \frac{b^{j-i} - 1}{a^{j-i} - 1}}\right) \end{aligned} \tag{5}$$

Let $A = \frac{b^k - b^i}{a^j - a^i} \cdot \left(\frac{a^j - a^i}{a^k - a^i} - \frac{b^j - b^i}{b^k - b^i}\right) \cdot \frac{c^k - c^i}{a^j - a^i} \cdot \left(\frac{a^j - a^i}{a^k - a^i} - \frac{c^j - c^i}{c^k - c^i}\right)$, then we have

$$\begin{aligned} D &= A\left(\frac{c^{j-i} - 1}{a^{l-i} - 1}\right)\left(\frac{\frac{c^{l-i} - 1}{c^{j-i} - 1} - \frac{a^{l-i} - 1}{a^{j-i} - 1}}{\frac{c^{k-i} - 1}{c^{j-i} - 1} - \frac{a^{k-i} - 1}{a^{j-i} - 1}} - \frac{\frac{b^{l-i} - 1}{b^{j-i} - 1} - \frac{a^{l-i} - 1}{a^{j-i} - 1}}{\frac{b^{k-i} - 1}{b^{j-i} - 1} - \frac{a^{k-i} - 1}{a^{j-i} - 1}}\right) \\ &= A\left(\frac{a^{k-i} - 1}{a^{l-i} - 1}\right)\left(\frac{\frac{c^{l-i} - 1}{c^{j-i} - 1} - \frac{a^{l-i} - 1}{a^{j-i} - 1}}{\frac{c^{k-i} - 1}{c^{j-i} - 1} - \frac{a^{k-i} - 1}{a^{j-i} - 1}} - \frac{\frac{b^{l-i} - 1}{b^{j-i} - 1} - \frac{a^{l-i} - 1}{a^{j-i} - 1}}{\frac{b^{k-i} - 1}{b^{j-i} - 1} - \frac{a^{k-i} - 1}{a^{j-i} - 1}}\right) \end{aligned} \tag{6}$$

We set

$$\begin{aligned} a^{j-i} &= \tau, b^{j-i} = t, c^{j-i} = s, \\ a^{k-i} &= \tau^\beta, b^{k-i} = t^\beta, c^{k-i} = s^\beta, \\ a^{l-i} &= \tau^\alpha, b^{l-i} = t^\alpha, c^{l-i} = s^\alpha, \end{aligned} \tag{7}$$

where $\alpha = \frac{l-i}{j-i} > \beta = \frac{k-i}{j-i} > 1$ and $s > t > \tau$. Then, we have

$$D = A\left(\frac{\tau^\beta - 1}{\tau^\alpha - 1}\right)\left(\frac{\frac{s^\alpha - 1}{s - 1} - \frac{\tau^\alpha - 1}{\tau - 1}}{\frac{s^\beta - 1}{s - 1} - \frac{\tau^\beta - 1}{\tau - 1}} - \frac{\frac{t^\alpha - 1}{t - 1} - \frac{\tau^\alpha - 1}{\tau - 1}}{\frac{t^\beta - 1}{t - 1} - \frac{\tau^\beta - 1}{\tau - 1}}\right). \tag{8}$$

Therefore, we only need to prove that $f(t) = \frac{\psi_\alpha(t)}{\psi_\beta(t)}$ is a monotonically increasing function, where $\psi_\alpha(t) = \frac{t^\alpha - 1}{t-1} - \frac{\tau^\alpha - 1}{\tau - 1}$ and $\psi_\beta(t) = \frac{t^\beta - 1}{t-1} - \frac{\tau^\beta - 1}{\tau - 1}$.

We take the derivative of $f(t)$ and obtain the following form.

$$f'(t) = \frac{1}{\psi_\beta^2(t)} \left(\psi'_\alpha(t) \int_\tau^t \psi'_\beta(s) ds - \psi'_\beta(t) \int_\tau^t \psi'_\alpha(s) ds \right) = \frac{1}{\psi_\beta^2(t)} \int_\tau^t \left(\psi'_\alpha(t)\psi'_\beta(s) - \psi'_\beta(t)\psi'_\alpha(s) \right) ds. \quad (9)$$

We need to assert that $f'(t) > 0$. In other words, we need to prove that

$$\psi'_\alpha(t)\psi'_\beta(s) > \psi'_\beta(t)\psi'_\alpha(s), \quad \text{where} \quad s > t > \tau, \quad (10)$$

i.e.,

$$\frac{\psi'_\alpha(t)}{\psi'_\beta(t)} > \frac{\psi'_\alpha(s)}{\psi'_\beta(s)}, \quad \text{where} \quad s > t > \tau. \quad (11)$$

Considering that:

$$\frac{\psi'_\alpha(t)}{\psi'_\beta(t)} = \frac{(\alpha-1)t^\alpha - \alpha t^{\alpha-1} + 1}{(\beta-1)t^\beta - \beta t^{\beta-1} + 1} = \frac{\varphi_\alpha(t)}{\varphi_\beta(t)}, \quad (12)$$

We take $\varphi'_\alpha(t) = (\alpha-1)\alpha t^{\alpha-1} - \alpha(\alpha-1)t^{\alpha-2} = \alpha(\alpha-1)t^{\alpha-2}(t-1) > 0$ and $\varphi'_\beta(t) = \beta(\beta-1)t^{\beta-2}(t-1) > 0$, hence $\frac{\varphi'_\alpha(t)}{\varphi'_\beta(t)} = \frac{\alpha(\alpha-1)t^{\alpha-2}(t-1)}{\beta(\beta-1)t^{\beta-2}(t-1)} = \frac{\alpha}{\beta}\frac{\alpha-1}{\beta-1}t^{\alpha-\beta} > 0$, and we have $\frac{\psi'_\alpha(t)}{\psi'\beta(t)} > 0$.

As a consequence, we prove that any 4 shares can be used to reconstruct the original message. □

Based on *Theorem 1*, we infer that even if some shares are lost or destroyed by the adversary, the original messages are still recoverable and hence data availability improves.

C. Message Recovery Based on the Shares

In message recovery process, at least 4 shares need to be collected and then four equations with x_1, x_2, x_3, x_4 as variables are constructed based on equation (2). In Section IV.B, we have proved that we can always get x_1, x_2, x_3, x_4 by solving the equation set and at last recover V_e.

Consider the following case: let us assume that x_1, x_2, x_3, x_4 are $7, 3, 2, 5$ and p equals to 13. Then, the seven shares $s_1, s_2, s_3, s_4, s_5, s_6, s_7$ are calculated based on equation (2) and they are $4, 0, 0, 2, 2, 2$ and 11, respectively. When the data user receives at least 4 shares (e.g., the first 4 shares $4, 0, 0, 2$), 4 equations can be listed as follows:

$$\begin{cases} 4 = x_1 + x_2 + x_3 + x_4 \bmod 13 \\ 0 = x_1 + 2x_2 + 3x_3 + 4x_4 \bmod 13 \\ 0 = x_1 + 4x_2 + 9x_3 + 16x_4 \bmod 13 \\ 2 = x_1 + 8x_2 + 27x_3 + 64x_4 \bmod 13 \end{cases} \quad (13)$$

By Gauss-Jordan elimination algorithm, we eliminate variables x_1, x_2, x_3 and simplify them as

$$6\ x_4 \bmod 13 = 4 \quad (14)$$

Considering that x_4 is smaller than 13, we can accurately recover it as 5. By substituting x_4 into equation (2), we have x_1, x_2, x_3 are $7, 3, 2$, respectively. In this way, the original encrypted message is recovered. And a combination of x_1, x_2, x_3, x_4 in order yields V_e.

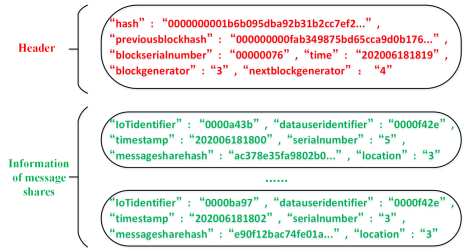

Fig. 3. Structure of a block.

D. Balanced IoT Data Storage Among the Storage Nodes

As presented in Fig. 1, each cloud node is treated as data storage node in the system when it can locally store a large amount of IoT data. Once a set of 7 message shares are constructed for a message, the IoT device separately sends them to a set of different data storage nodes. Specifically, 7 different serial numbers are first randomly selected from 1 to 10 and then each share is delivered to the storage node with corresponding serial number. In this way, the workloads of the nodes in the storage system are balanced on average and thus the overall storage is more efficient. After the transmission of shares is completed, the IoT device delivers data transaction information to all 10 cloud storage nodes. Specifically, there are 7 pieces of transaction information corresponding to 7 shares.

For each data storage node, the message shares are organized in order based on two dimensions, i.e., generation time and IoT device identifier. We first sort the shares based on birth time extracted from parameter T. Then, if a set of shares have the same birth time, they are sorted based on the identifier of the IoT device which can be extracted from parameter ID. In fact, the above process is very efficient. This can be explained by the fact that, in data collection process, the IoT shares are approximately received in order according to their generation time. As a consequence, the newly received shares can be easily inserted into existing entries by accessing a small set of IoT shares. This mechanism greatly improves the IoT data retrieval efficiency and is to be revisited in Section VI.

V. Data Integrity Protecting and Location Information Disclosure by Blockchain

To protect integrity of IoT data shares and disclose location information among all the nodes, a new blockchain is specifically designed for the data storage system. In our blockchain system, the cloud nodes play as the blockchain node, which take turns to form blocks once they receive enough shares. The newly generated blocks are appended to the blockchain and then updated for all the blockchain nodes. The nodes in our system collaborate with one another to generate new blocks. This idea is totally different from that of existing blockchains in which the nodes compete with each other to mine new blocks and update the chain.

In this section, we first discuss the structure of a block in Section V.A. The update process of the blockchain is presented in Section V.B. How to check the integrity of the shares based on the blockchain is illustrated in Section V.C. In Section V.D, we introduce members' management of our blockchain system.

A. Construction of New Blocks

The structure of a block is presented in Fig. 3. Each block is consisted of two modules, i.e., the block header and the

block body which stores the information of message shares. In the header of a block, a hash value of the block is calculated by the SHA-256 hash algorithm. The next entry is the hash value of the previous block. After iterations, all the blocks are chained together and any former block cannot be tampered. To control the order of block generation, the identifier of a block and next block generator are both stored in the header. As such, the nodes take turns to form new blocks. Some other necessary information including serial number and generation time of the block is also stored in the header.

The information of IoT message shares is stored behind the header. For each message share, six entries are employed to form a transaction in the block. The first entry is the IoT identifier indicating the IoT device that generates the message share. The data user identifier indicates which user groups can access the message share. Timestamp is the birth time of the share. Serial number of a share ranges from 1 to 7 in our scheme. The hash value of a message share is also integrated into the block. Therefore, based on the blockchain properties, all the IoT data are safe in terms of integrity. The last entry, location, indicates where the share is stored. Then all shares in a block are organized into a Merkle tree structure so that other nodes can examine if the block is constructed in a correct way. We give an example in Fig. 3, exhibiting the specific transaction information on the blockchain for the two shares collected within 2 minutes. From the IoT identifier it can be seen that they are collected by two different IoT devices, designated to be retrieved by the same user with the common user identifier, and are stored on the storage node numbered 3. In this paper, 10 cloud nodes are employed and hence the location of a share ranges from 1 to 10. In a block, the locations of all the shares are the same with one another, because they are all stored in one cloud node.

B. Update of the Blockchain

In a blockchain, each node forms a block and then appends it to the chain. To synchronously update all the blockchains in different nodes, several consensus algorithms have been proposed in the existing blockchain systems such as Proof of Work (PoW), Proof of Stake (PoS) and Practical Byzantine Fault Tolerance (PBFT). It consumes a large amount of resources for these blockchains to reach a consensus among all the members because of the competition of the members and imbalance of the workloads. Apparently, these mechanisms are not compatible with IoT data storage, given the computing resource of IoT devices is insufficient and the amount of IoT data is extremely large. Therefore, a more efficient mechanism is necessary.

Different from the present blockchains, all the nodes in our system receive similar number of messages shares from the IoT devices and they are of similar workloads. At the same time, our scheme adopts a structure similar to a private chain. Each node needs to go through a strict review. Member nodes are to be specified before their deployment by the system, and part of the information is to be written into the hardware of IoT devices. Therefore, each node is fairly reliable. In PoS, all blockchain nodes can be chosen to construct new blocks based on their contribution to a blockchain system. We simplify this further by assuming that each node is trusted and has the same equity. With these features in consideration, we design a token-based mechanism to efficiently update the blockchain as shown in Fig. 4. The blue lines between the nodes represent the connection between the nodes in the blockchain network,

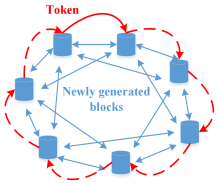

Fig. 4. Update process of the blockchain system.

Algorithm 1 Consensus Building Algorithm

Input: All the nodes in the blockchain $N = \{N_1, \cdots, N_m\}$, the token $Token$, the parameter of period P

Output: Consensus building among all the nodes

1: **for** each node $N_i \in N$ receiving $Token$ **do**
2: **for** each IoT data message share out of the chain **do**
3: Constructing the transaction entry based on the share's information;
4: **end for**
5: Calculating a hash value of the block based on all the transaction entries and forming a new block NB;
6: Appending block NB to the blockchain;
7: Broadcasting the update information to all the node members;
8: Delivering $Token$ to the next node in N at the end of period P;
9: **end for**

and each node will update their blocks synchronously when a new block is generated. In our system, only one token exists and it is passed among the orderly nodes as shown by the red arrow in Fig. 4. A node constructs a new block and appends it to the blockchain if and only if the node holds the token. When a new block is under construction, all those received message shares but not yet appended to the chain in the node are employed.

The specific update process is presented in Algorithm 1. Parameter P is used to control the speed of block generation. With a preset size of a block, i.e., the number of shares in a block, we enlarge P if the generation speed of IoT shares is small; we decrease P if otherwise. When receiving the token, the node begins to construct a new block. Each received IoT share is mapped to a transaction information in the block. Once all the information are constructed, the node finally calculates a hash value of the block and a new block is generated.

Node holding token updates the blockchain by trying to append the new block to the end of the chain and needs to deliver the updated information to all the members in the system. When a new block is received, each storage node compares the transaction information stored locally with the data in the new block. If the information is the same, the block is added to the end of the chain and the corresponding data transaction information for the block is deleted locally. This authentication mechanism ensures that malicious nodes cannot tamper data transaction information. At last, the token is delivered to the next node at the end of period P. By iterating the above process, the blockchain expands along with the IoT data generation.

With this token-based mechanism, each blockchain node takes its turn to generates new blocks. There is no need for the

nodes to compete for the construction power of new blocks through workload like PoW or prove their rights and interests like PoS. Compared with other traditional consensus schemes, the token-based consensus mechanism greatly improves updating speed of blocks. It meets the requirements of updating efficiency for high-speed data generation in IoT. We also carry out a detailed simulation experiment on updating efficiency of blockchain. The specific experimental data are introduced in detail in Section VIII.D.

C. Integrity Check of the Shares

Our scheme has a dual mechanisms for shares integrity protection. On the one hand, every block containing transaction information will be verified before appending to the chain. On the other hand, the shares recieved by data users can also be verified for integrity.

First, the integrity of the new block needs to be verified. If an adversary breaks into the token holding node, a new block is forged where an error message is stored. Under the block update mechanism, each node is required to broadcast the received shared data transaction information over the network. In this way, after a new block is created, each node verifies the integrity of the block by determining whether the hash values of the same transaction broadcast over the network are the same as those stored in the block. If other nodes discover that a newly generated block is wrong, they refuse to synchronize the block. Otherwise, the next node continues to generate new blocks on the original chain based on the broadcast information.

Then, in data retrieval process, hash values of the shares are returned to data users along with their locations, as discussed in Section VI. Therefore, when receiving a message share, data users can easily check its integrity by calculating its hash value and then comparing it with the received hash value. Specifically, A block records all message shares received by the node over a period of time. We take a share S of message M as an example. After receiving the share S, the blockchain node collects the block information as shown in Fig. 3, where the information corresponding to the share S includes hash value, storage location, timestamp, and serial number. When the user obtains the share S from the storage node, he or she needs to find the block that stores the information related to the share S publicly on the blockchain and calculate the $hash'$ corresponding to the share S using a specified hash algorithm. If the $hash'$ is the same as the corresponding hash value stored on the blockchain, then the share has not been modified since the storage node takes it.

If the data user finds that some message shares are tampered, one can request some other shares to accurately recover the original message. Moreover, if a share's integrity is destroyed, the data user can upload the abnormal cases to the regulatory agency of the system. The regulators can easily locate the storage node of the share and then evaluate the reliability of the storage node.

D. Members Management

In reality, as the system data aggregate over time, new cloud nodes may need to be inserted to balance the storage overhead of other nodes, and old nodes may be deleted due to failure or dishonesty. Next, we take the operation of adding new nodes in the blockchain system as an example to introduce the process of new nodes' insertion.

First of all, the owner of the system needs to assess performance of security and other indicators of a cloud server, then review and authenticate identities of the new nodes. Only with the authorization of the manager can the nodes be allowed to discover one another and mark them as newly added cloud nodes according to their unique identification.

Then, in the private chain network, once the original nodes discover that a new member has joined the network and receive the broadcast information of the new node, they establish a trusted connection with the new node. At this point, the new node normally synchronizes blockchain data and uploads transaction information.

Finally, according to the consensus mechanism, the next node that constructs a new block is identified by the "nextBlockgenerator" in the current block header, which indicates the constructor of the next new block. When the constructor node of the previous block receives a node update broadcast, the newly added node is included in the value of "nextBlockgenerator". And when a token is passed to a new node, the new node is responsible for building the new block, and the blockchain system formally accepts the new node with writing access to the data.

As mentioned earlier, some old nodes may be deleted due to reasons such as failures. In fact, the process of deleting nodes is similar to adding nodes to the system. It is necessary to broadcast node deletion information in the network according to operative specification of a private chain.

VI. Secure and Efficient IoT Data Retrieval

In a secure and distributed IoT data storage system, a basic operation of the data users is searching IoT data of interest. As shown in Fig. 3, for a share, the information of IoT identifier, serial number, hash value and location are all disclosed by the blockchain. Therefore, each cloud node knows the location of any share in the system and thus is helpful for the data users to retrieve the IoT data.

A. The Framework of IoT Data Retrieval

In this paper, the IoT users retrieve the data of an IoT terminal device by providing two parameters, i.e., the identifier, ID, of the IoT device and a time period (T_i, T_j). Given a data request with parameters ID and (T_i, T_j), the system needs to return all the IoT data generated by the IoT device with ID as an identifier in time period (T_i, T_j). In this process, the cloud node where the user submits search parameters performs the function of a retrieval server and retrieves the IoT data stored in the cloud nodes whose roles are storage nodes. Note that, though each storage node has the locations of shares with ID and T as search parameters, it has no access to the shares in other storage nodes according to the threat model.

The framework of the whole IoT data retrieval process comprises two sub-processes, i.e., locating the nodes in the system where the interested shares are stored and locating the share by a storage node among a large number of shares. The first one is of great difficulty. But fortunately, all the retrieval servers can infer the location of a share based on the information in the blockchain. Then, to improve search efficiency, an index tree structure is constructed in which the share vectors with similar generation time are stored in nearby leaf nodes. The leaf nodes aggregate into different clusters based on their similarities until all the vectors belong to a huge cluster. The index tree needs to be maintained in all the storage nodes and hence each node is capable of responding to query

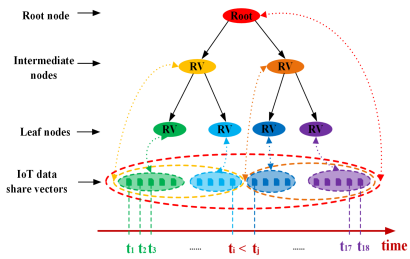

Fig. 5. Index structure of share vectors.

requests. Specifically, once a query request is generated, the data users send it to any retrieval server in the system. Then, the retrieval server sends both the locations and hash values of the searched shares to the data users. Based on the received locations, the users can communicate with the corresponding storage nodes to request the shares. For each searched IoT message, it is recoverable once minimum t shares are received and the data retrieval process is completed. In the following paragraphs, we first exhibit the structure of index tree for the share vectors and then propose a depth-first IoT data retrieval algorithm to efficiently locate the shares of interest in Section VI.B. Then, we describe how to organize and search IoT data in a storage node in Section VI.C.

B. Locating the Storage Nodes of the Interested Shares

1) Structure of the Balanced Index Tree: To return the locations and hash values of the searched shares, each storage node needs to construct and maintain an index structure for the shares. We first map each share to a share vector, SV, with five elements, i.e., the IoT device's identifier ID, generation time T, serial number SN, its storage location and hash value. For each share, all the above information can be found in the blockchain and hence the vectors of all the shares in the storage nodes can be easily constructed.

Then, the vectors are organized by a balanced index tree as shown in Fig. 5. Two branch parameters, i.e., B_1 and B_2, are employed to control shape of the tree. Specifically, when the number of share vectors in a leaf node is larger than B_1, we generate a new leaf node to store the newly arrived vectors. Similarly, for a non-leaf node, we need to construct a new node when the number of its child nodes is larger than B_2. With the increase of B_1 and B_2, the height of the tree decreases and the width of the tree increases, and vice versa. The two branch parameters need to be carefully assigned based on the share arrival speed and data search frequency.

Another important property of the tree is that the vectors in a leaf node are similar with each other. The similarity between a pair of vectors can be flexibly defined based on data retrieval patterns and it may include a set of independent dimensions. Considering the basic search pattern in our scheme, each node in the tree maintains only one time period entry, (T_m, T_n), indicating that all the shares under the node are generated in period (T_m, T_n). Moreover, as shown in Fig. 5, all the share vectors in the leaf nodes are sorted in the ascending order. As a consequence, the time period entries of the nodes with the same depth in the tree are mutually disjoint and they are sorted in order. This property greatly improves search efficiency and are further discussed next.

2) Construction of the Index Tree: Considering that the IoT data are continuously generated, we need to construct the index structure in an incremental manner. The distance between a share vector SV, with T as generation time, and a node N_s, with (T_m, T_n) as time period entry, is defined as follows:

$$\text{dist}(SV, N_s) = \begin{cases} 0, & \text{if } T \text{ locates in } (T_m, T_n), \\ \min(|T - T_m|, |T - T_n|), & \text{otherwise.} \end{cases} \tag{15}$$

where $|*|$ is the absolute value of $*$. Based on its definition, the index tree can be dynamically constructed in five steps and they are presented as follows.

- *Locating the leaf node.* When a new share vector arrives, we descend the vector in the tree by continuously selecting the child node whose time period contains the generation time, T, of the share. If T is not covered by any child node, we select the child node closest to T. Finally, the leaf node closest to T will be found.
- *Checking the status of the leaf node.* Once the closest leaf node is located, we need to first insert the vector to the leaf node. If the located leaf node is the newest leaf node, i.e., the purple leaf node in Fig. 5, and the number of the leaf node's share vectors is smaller than B_1, we insert SV to the leaf node. Otherwise, we split the node. Moreover, if the located leaf node is not the newest leaf node, and the number of share vectors is larger than $\rho * B_1 (\rho \geq 1)$, we split the node as well.
- *Splitting the leaf node.* For the newest leaf node with a set of share vectors, we split it by putting the share vector with the largest generation time to a newly constructed leaf node. In the split process of the other leaf nodes, the two share vectors with the largest distance are selected as seed nodes and then the other vectors are assigned to the nearer seed node. Finally, the two vector clusters form two new leaf nodes.
- *Updating the entry in the leaf node.* Once the leaf nodes absorb a share vector or they are split, the entry in the leaf node needs to be updated. Based on the definition of time period entry, the update process is quite straightforward.
- *Updating the intermediate nodes on the path to the root node.* Once a leaf node is updated, all the intermediate nodes on the path to the root node need to be updated in a bottom-up manner. Similarly, if an intermediate node contains more than B_2 child nodes, it also needs to be split and the process is similar to that of leaf nodes. The split process is iterated until a new root node is generated and, in this case, the height of the index tree is increased by one.

In our scheme, the index structure dynamically expands with more and more share vectors inserted into the tree. Besides, in principle, putting similar vectors into nearby clusters, we attempt to keep the mature leaf nodes stable, which is another rule to be complied. The rationale is based on the mainstream situation in which most of the newly received share vectors are inserted into the latest leaf node. Though a portion of vectors need to be inserted into some other leaf nodes, the number of share vectors is still approximately balanced and the search efficiency of the tree is approximately optimal. Therefore, the latest leaf node is directly split once the number of its share vectors is beyond B_1, while the other nodes are split until the number of their share vectors are larger than $\rho * B_1$.

To make the index tree compact, we need to carefully preset parameter ρ based on the random delay of share vectors. In fact, if all the time delays of the share vectors are

Algorithm 2 Depth-First Share Vector Retrieval

Input: An index structure of the share vectors with $Root$ as the root node and a request query with parameters ID and (T_i, T_j)

Output: The locations and hash values of all the interested shares

$S_{node} = \{Root\}$, $S_{result} = \{\emptyset\}$;
while S_{node} contains at least a non-leaf node **do**
  for each non-leaf node γ in S_{node} **do**
    Compare γ's time period entry (T_m, T_n) with (T_i, T_j);
    if $(T_m, T_n) \cap (T_i, T_j) \neq \emptyset$ **then**
      Delete γ from S_{node};
      for each child node γ' of γ **do**
        if $(T_{m'}, T_{n'}) \cap (T_i, T_j) \neq \emptyset$ **then**
          Insert γ' into the head of S_{node};
        end if
      end for
    else
      Delete γ from S_{node};
    end if
  end for
end while
for each leaf node, γ_f, in S_{node} **do**
  Search all the share vectors in γ_f whose generation time locate in (T_i, T_j) by the binary search algorithm;
  Insert the share vectors whose identifiers are ID into S_{result};
end for

exactly the same as one another, ρ can be set as 1. With an incremental delays' randomness, we should gradually increase ρ to reserve some places for the delayed share vectors in advance. There is an interesting positive correlation between ρ and the randomness of share vectors' time delay.

3) Depth-First IoT Data Share Vector Retrieval Based on the Index Tree: Based on the index tree, we can efficiently return the share vectors of interest based on the depth-first search algorithm as shown in Algorithm 2. Given an index tree and a query request with parameters ID and (T_i, T_j), the exact locations and hash values of the shares desired are returned. Initially, the set of nodes, S_{node}, needed to be searched in the tree, contains only the root node. Then, we continuously update S_{node} by descending the index tree until S_{node} is empty or all the nodes in S_{node} are leaf nodes. By scanning the share vectors in the leaf nodes, we at last obtain the searched share vectors. Consider that we first descend the index tree to the leaf nodes and then enlarge the search area in horizontal direction, the proposed algorithm is thus named as the depth-first search algorithm.

As shown in Algorithm 2, in the initial phase, we check whether the time period entry of the root node intersects with (T_i, T_j). If they are not intersectant, the search result is an empty set, or otherwise, we delete the root node from S_{node} and put all the lawful child nodes into S_{node}. A node is lawful if and only if its time period entry is intersectant with (T_i, T_j). Then, we descend to the child nodes in the next level and check the relations between their entries with (T_i, T_j). The lawful nodes in the next level are also inserted into S_{node} and they replace the parent node. By iterating the above process, we finally have a node set S_{node} which is either empty or all the nodes in S_{node} are leaf nodes.

After having a set of leaf nodes, we need to first filter out the share vectors that are located in (T_i, T_j). Then, we obtain the final search results by comparing the identifiers of the candidate share vectors with ID. At last, both the locations and hash values of the interested message shares are returned to the data users.

C. Locating a Specific Share in a Storage Node

Once a data user knows where the interested shares are stored in the system, one can request a share to a particular storage node. As discussed in Section IV.D, the IoT shares in the storage nodes are organized in order based on the generation time. We employ the binary search algorithm to locate a share in a node and apparently the complexity is $O(logN_u)$ where N_u is the number of shares in the node. Once a set of at least t shares are collected for an IoT message, the data user can finally recover the original message based on the secret sharing scheme discussed in Section IV.

VII. System Security Analysis

The IoT data security mainly comprises three aspects: Confidentiality, Integrity and Availability (CIA), and we analyze them respectively under the threat model presented in Section III.B. Moreover, we also briefly discuss how to recover a totally destroyed cloud node and the overall security performance of our scheme.

A. IoT Data Confidentiality

Secret sharing scheme plays a key role in protecting the security of IoT data. we summarize an important property as the following theorem.

Theorem 2: Given any set of less than t shares, the adversary cannot recover V_e.

Proof: Without loss of generality, we assume that the adversary has $t-1$ shares. Obviously, if the adversary cannot recover V_e with $t-1$ shares, he or she cannot recover V_e with less than $t-1$ shares. Let F be a field and $H_{n\times t} = (H_1, H_2, \cdots, H_n)^T$ be a matrix over F. Given t variables $x_1, x_2, \cdots, x_t$ and n numbers $s_1, s_2, \cdots, s_n$ from field F, consider the following equation.

$$H_{n\times t}\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \end{pmatrix} \Leftrightarrow H_{n\times t}X_{t\times 1} = S_{n\times 1} \qquad (16)$$

where $X_{t\times 1} = (x_1, x_2, \cdots, x_t)^T$ and $S_{n\times 1} = (s_1, s_2, \cdots, s_n)^T$. We prove that if matrix $H_{n\times t}$ is of a rank $t-1$, equation (16) has no solution or has $|F|$ solutions, where $|F|$ is the number of elements in field F.

Consider matrix $\overline{H}$ defined as follows:

$$\overline{H} = \begin{pmatrix} H_1 s_1 \\ H_2 s_2 \\ \vdots \\ H_n s_n \end{pmatrix} \qquad (17)$$

If the rank of H is not equal to the rank of $\overline{H}$, equation (16) has no solution and the adversary cannot recover V_e. On the other hand, if the rank of H is equal to the rank of $\overline{H}$, equation (16)

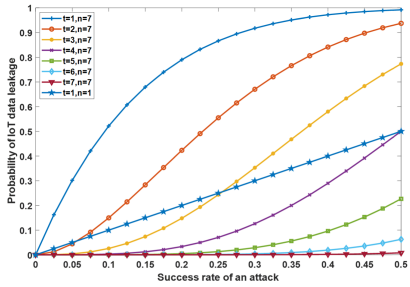

Fig. 6. IoT data leakage probability with different success rate of attacks.

has a special solution $G = (a_1, a_2, \cdots, a_t)^T$ satisfying:

$$HG = S \tag{18}$$

Meanwhile, matrix equation $HX = 0$ has a solution vector space of 1-dimension over field F generated by vector $Y_{t\times 1}$. Then, we infer that $HX = S$ has a set of solutions: $\lambda Y + G$, where λ is the element in F. By combining the above two cases, *Theorem 2* is proved. □

Based on *Theorem 2*, we know that the original message can be recovered if and only if at least t shares are received. In our framework, each cloud server stores only one share of a message and hence they cannot recover the original IoT data without the help of another at least $t-1$ cloud servers. Therefore, the IoT data are secure under the cloud server threat model.

To seize an original message, the adversary needs to break through the following challenges.

- One needs to acquire at least t shares of a message.
- Based on the shares, one can recover the encrypted message. At last, one needs to obtain the secret key to decrypt the original message.

As mentioned in the threat model of Section III, when the roles of storage node and retrieval server are destroyed, the adversary can obtain part of the shares, but the specific number of the shares depends on the adversary's capability. Compared with the existing centralized data storage schemes, our scheme assumes that the adversary can destroy multiple cloud nodes to capture enough shares which are useful to recover the original messages. Considering that the cloud nodes are heterogeneous with one another, the adversary faces the first enormous challenge to obtain the original data. Assume that probability of compromising a cloud node ranges from 0 to 0.5 and the simulation result in terms of IoT data leakage probability is shown in Fig. 6. It is observed that with the increase in attack success rate, IoT data leakage probability linearly increases for the centralized scheme, i.e., the scheme with parameters $t = 1, n = 1$. With different parameters, data leakage probabilities of our scheme varies greatly from one another. For a constant n, with the increase of t, data leakage probability greatly decreases and data confidentiality of our scheme increases. Apparently, our scheme greatly outperforms the centralized scheme, especially for a small success rate of an attack. When we set the attack success rate as 0.1, data leakage probability of the centralized scheme is also 0.1 and that of our scheme with $t = 4, n = 7$ is only about 0.001.

Furthermore, given a larger t, which results in a shorter message share, our scheme yields even higher data storage efficiency. However, the robustness of our scheme decreases with the increase of t. There is an interesting balance between data confidentiality, storage efficiency and robustness, which will be discussed in Section VII.C.

In addition, even if the adversary recovers the original encrypted message, one still needs to obtain the symmetric encryption key of the IoT device to access the original IoT data, which is the second challenge.

Finally, security threats to data confidentiality from IoT devices also need to be addressed. As mentioned in the threat model of IoT devices in Section III, the pre-stored coefficients in IoT devices face certain leakage risks. In order to improve security in this regard, the coefficients can be obfuscated and hidden to increase the difficulty of reverse analysis, and can be stored in the underlying hardware chip to further enhance security. According to the security proof of *Theorem 2*, even if the pre-stored coefficients in individual devices are leaked, the security of data is not compromised if sufficient shares cannot be obtained.

B. IoT Data Integrity

In our scheme, we create a blockchain for IoT data shares. The hash values of the shares are also added into the blocks. As discussed in Section VI.D, in an IoT data retrieval process, both the message shares and their hash values are returned to the data users. Therefore, it is straightforward for the users to assess the integrity of shares, even if shares were slightly modified. The integrity theorem and proof of our scheme are given below.

Definition 1 (Weak Collision Resistance): For a hash function h, set the hash value of a message M to $h(M)$. For any polynomial time adversary $\mathcal{A}$, the corresponding $M' \neq M$ cannot be found by advantage ε ($\varepsilon > 0$ is a function with negligible value) so that $h(M) = h(M')$. That is, the hash function is said to be weak collision resistance when the following formula holds.

$$\Pr\left(h(M) = h\left(M' := \mathcal{A}(M)\right)\right) < \varepsilon \tag{19}$$

Definition 2 (Data Integrity Protection): We claim that a cloud storage scheme is of IoT data integrity protection when the following condition is meet. Given any shared information S of any IoT message M, the adversary finds a different $S' \neq S$ in any polynomial time, so that the advantage of $h(S) = h(S')$ is δ ($\delta > 0$ is a function with negligible value). In probability terms, the following formula holds:

$$\Pr\left(h(S) = h\left(S' := \mathcal{A}(S)\right)\right) < \delta \tag{20}$$

Theorem 3: Assuming that the hash function h used in this scheme has weak crash resistance, the scheme has capability of data integrity protection.

Proof: A block records all message shares received by the node over a certain period of time, considering a share S of message M. After receiving share S, the blockchain node constructs the block information shown in Fig. 3, where the information corresponding to share S includes the hash value $h(S)$, storage node, time stamp, serial number. When a user takes the share S' from the storage node, one needs to find the publicly available block storing the share S-related information on the blockchain and follow the prescribed hash algorithm to calculate the corresponding hash value $h(S')$. If $h(S) = h(S')$, the user considers the received share to be complete. If the adversary successfully destroys the integrity of the scheme, which indicates that there is a non-zero advantage $\sigma > 0$ making $h(S) = h(S')$ when $S' \neq S$. That is, the

following formula holds:

$$\Pr\left(h(S) = h\left(S' := \mathcal{A}(S)\right)\right) > \sigma \tag{21}$$

Then we have the advantage σ of finding the message pair $M \neq M'$(take $M = S, M' = S'$), which leads to

$$\Pr\left(h(M) = h\left(M' := \mathcal{A}(M)\right)\right) > \sigma \tag{22}$$

Then the adversary successfully breaks through the weak collision resistance of the hash function, which is contradictory to the assumption. Therefore, this scheme effectively ensures the integrity of stored data. □

However, if the node is broken by the adversary before accepting the share S, the adversary may modify the share S and store the hash value $h(S)$ of the modified S on the newly constructed block. Then the above integrity verification becomes ineffective. However, the $(4, 7)$-secret sharing scheme designed in this study can effectively resist tampering attacks on the stored shares. As shown in the analysis of secret sharing scheme in IV.C, as long as the enemy tampers with no more than three shares at the same time, a user can still recover the original complete message through comparisons. Specifically, considering that the user takes in all seven shares $s_1, \cdots, s_7$ of V_e in a message M, the user has thus seven linear equations:

$$\begin{cases} s_1 = x_1 + a^0 \cdot x_2 + b^0 \cdot x_3 + c^0 \cdot x_4 \bmod p \\ \quad \vdots \\ s_7 = x_1 + a^6 \cdot x_2 + b^6 \cdot x_3 + c^6 \cdot x_4 \bmod p \end{cases} \tag{23}$$

C_7^4 solutions are obtainable by selecting four equations to form an equation group. At least one of the C_7^4 solutions is the available original message. Considering that the adversary does not know a, b, c, so tampering of share s_i is random, and the solution of the equation system containing the tampered share is of garbled code that does not conform to the data specification. Therefore, a user can distinguish available solutions from tampered ones. Further, one can find which equations are tampered and thus locate the malicious nodes. If the data user finds that some message shares are tampered, one can request some other shares to accurately recover the original message. Moreover, if the integrity of a share is destroyed, he or she can upload the abnormal cases to the regulatory agency of the system. We then easily locate the cloud node of the share and assess reliability of the cloud node.

In other existing schemes, integrity of stored data totally depends on reliability of the cloud platform. As a consequence, if the cloud is untrusted or attacked by an adversary, the integrity is not guaranteed.

C. IoT Data Availability

As discussed in the adversary threat model, an adversary can decrease availability of a data storage system by breaking the system through vulnerabilities or executing Denial of Service (DoS) attack. Fortunately, even some cloud nodes are attacked or some stored shares are lost, we can still recover the original IoT data based on the property of secret sharing. However, manufacturers of IoT devices prefer smaller coefficients to save cost, which is detrimental to message reconstruction. Therefore, we need to explore influences of different parameters on the probability of message reconstruction to demonstrate that our scheme has good security properties and can successfully recover the original information under different parameters. Recalling in Section III.B, the failures of the cloud nodes are assumed independent with one another. For different average node failure probabilities, the success rate of recovering the original message changes as shown in Fig.7.

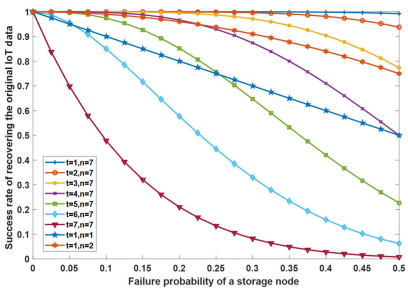

Fig. 7. Success rate of recovering the original IoT data with different cloud node failure probability.

It appears that the success rate of recovering the IoT data gradually decreases with the incremental node failure probability. This reflects the fact that a larger failure probability leads to more failed nodes. We denote the failure probability of the cloud nodes as p_r and then the success probability of recovering the message is mathematically calculated as follows.

$$P(\text{successful recovery}) = \Sigma_{m=t}^{n} C_n^m (1 - p_r)^m p_r^{n-m} \tag{24}$$

where C_n^m is the number of different results when we select m cloud node from n nodes. Based on the above equation, it is easy to infer that a larger p_r leads to a smaller success rate of message recovery.

We compare our scheme with the existing centralized ones in Fig. 7. As it is well known that an original message is stored in only one node, and if the node fails, the message is lost. In theory, the centralized schemes are equal to our scheme when we set $t = 1, n = 1$. It is observed from Fig. 7 that, when node failure probability is smaller than 50%, our scheme with $t = 1, 2, 3, 4$ performs much better than the centralized schemes. When the failure probability is smaller than 25%, our scheme with $t = 1, 2, 3, 4, 5$ performs much better than the centralized schemes. If we further decrease the failure probability to 5%, then our scheme always performs better unless $t = 7$. Therefore, the security of our solution is better than that of those centralized ones, even if the IoT device manufacturers reduce the gap between t and n in (t, n) to some extent factoring in cost.

Reliability of the centralized schemes is improvable with independently storing the IoT message in two storage nodes, which provides a backup for the schemes. Theoretically speaking, this improved approach is equivalent to our scheme with $t = 1, n = 2$ and corresponding simulation results are also illustrated in Fig.7. Though such a scheme greatly improves its reliability, it performs worse than our scheme with $t = 1, 2, 3, 4$ when the node failure probability is smaller than 20%.

In our scheme, t is set as 4. Considering that cloud node failures are events with small probability, we assume that the failure probability is 5%. Then, the failure probability of our scheme is 0.02% and meanwhile the failure probability of the existing centralized schemes is 5%. Apparently, our scheme greatly outperforms the centralized schemes in terms of data availability.

D. Recovery of a Totally Destroyed Cloud Node

If a cloud node is compromised or totally destroyed in some extreme cases, it's three roles are compromised at the same

time. Apparently, the blockchain can be recovered by making a duplicate of the blockchain in other nodes, and the retrieval server can be restored to normal after removing the malicious nodes. Therefore, our main focus of work is to recover all the stored shares in the storage node. The recovery process cannot be implemented locally but needs help from the whole system. Specifically, the lost shares can be reconstructed based on the property of secret sharing, i.e., some redundant information is distributively stored in the shares. Therefore, we can recover the original message and then calculate a new share.

E. Overall Security Discussion

As discussed in Section III, the most important goal of our scheme is improving security of IoT data. Through theoretical analyses, we prove that our scheme can provide a much secure data storage service to IoT. In summary, the distributed data storage manner and heterogeneity of cloud nodes raise the bar of attacks and hence yield high confidentiality. The employment of blockchain keeps the data from illegal modification and thus protects integrity of IoT data. In the data distribution process, an ultra-lightweight secret sharing scheme is designed and employed to map the original messages to a set of shares. Based on the properties of secret sharing mentioned in Section VI.C, some redundant information is hidden in the shares, which leads to better availability of IoT data.

VIII. Efficiency Evaluation

A. Simulation Setup

We implement the IoT data storage system in Python and the whole framework in Fig. 1 is decomposed into 7 sub-modules including IoT data collection module, secret sharing module, data transmission module, data storage module, Blockchain module, data retrieval module and data user module. The simulation is conducted on a DELL tower server with two intel CPUs and 128G memory. In order to achieve best virtual reality of IoT, we deploy a distributed P2P network and use Python to simulate the steps such as communication between nodes. We develop the blockchain system based on the Hyperledger Fabric 1.1 framework, rewriting its consensus and block structure as well as specifying API interfaces for nodes to interact with the system. Nodes in the network are divided into IoT device nodes and cloud server nodes, among which cloud server nodes are assumed the identity of data storage nodes, blockchain nodes and data retrieval servers. In this distributed network, the communication between IoT devices and cloud servers adopts OPC UA protocol [46] with an average delay of 300 ms, while the implementation of blockchain P2P networks relies on the Gossip protocol in the framework. For consistency, the communication delay between nodes is also set to 300 ms. In total, 10 cloud nodes are employed in the system and we set (t, n) as $(4, 7)$ in the secret sharing scheme. A temperature IoT terminal device is employed in our system and it generates a numerical temperature every second. We encode a temperature as a float number with 64 bits length and each group of 16 temperature numbers are encoded as a message M which is, in total, 1024 bits in length. The head of a message and a share is set as 80 bits. We collect the temperature data for a day and hence overall 5400 original messages are generated. Then these messages are mapped into 37800 message shares which are distributively stored in 10 nodes. When each share is delivered from the IoT device to its cloud node which collects the share's location information and generates a block with the structure shown in Fig. 3 every 480 seconds. Each block records the information of about 21 message shares with average size $4k$.

TABLE I
SIMULATION PARAMETERS

Parameter	Value
Simulation last time	86400 seconds
Number of IoT device	1
Message generation period	Every 16 seconds
Number of the original messages in total	5400
Length of a message	1024 bits
Length of the head	80 bits
(t, n)-secret sharing	$(4, 7)$
Number of storage nodes	10
Block generation period	Every 480 seconds
Size of a block	About 21 shares
Parameter B_1 in the index tree	20
Parameter B_2 in the index tree	8
Number of data users	1
$\lvert T_i - T_j \rvert$ of a query	600 seconds
Data query interval	Every 2 hours
Number of data query requests	In total 120 times

In the index tree, parameter B_1 is set as 20, i.e., each leaf node contains at most 20 share vectors and parameter B_2 is set as 8, i.e., each non-leaf node contains at most 8 child nodes. While collecting the IoT data, we also evaluate the data retrieval efficiency dynamically. In addition, we test data query efficiency with different sizes of datasets. In data retrieval process, a data user sends a set of 10 query requests to the system every 2 hours, while the query process lasts for a day. In each query, the parameter (T_i, T_j) is randomly selected and $|T_i - T_j|$ is always set at 600 seconds. The average results of 100 simulations are presented and analyzed.

For convenience, all the simulation parameters are summarized in Table I. Under these assumptions, we mainly evaluate the performance of our scheme in terms of efficiencies in share construction, data storage, blockchain and data retrieval.

B. Efficiency of Message Share Construction

The overall efficiency of secret sharing comprises two parts, i.e., share construction efficiency and message recovery efficiency. In real life, IoT terminal devices are often resource-limited, but data users are often rich in this regard. So we mainly focus on evaluating efficiency of message share construction considering that the calculate consumption for shares' construction and recovery are totally same.

Computational complexity of constructing IoT shares highly affects resource consumption of IoT terminal devices. We employ the metrics of time efficiency to compare efficiency of our scheme with that of the classic Shamir secret sharing scheme. Each collected message is mapped to 7 shares and simulation results are presented in Fig. 8. It appears that the time consumption of both our scheme and Shamir's secret sharing scheme linearly increases with the increasing number of messages. Moreover, our scheme immensely outperforms Shamir's secret sharing scheme.

The time consumption of our scheme with different parameter t is also illustrated in Fig. 8. Specifically, we set n as 7 and then select t from $\{1, 2, 3, 4, 5, 6, 7\}$. It is clear that with an incremental t, the time consumption gradually increases. Obviously, when $t = 1$, the scheme consumes the least time, because the original message is directly duplicated for 7 times. With t increasing, the time consumption of our scheme also

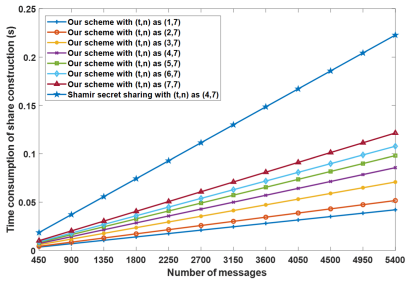

Fig. 8. Time efficiency of message share construction.

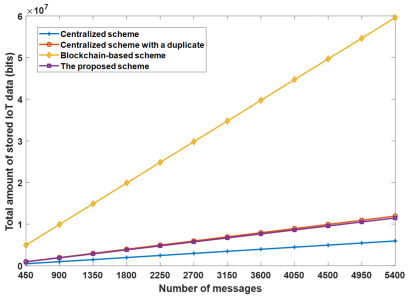

Fig. 9. IoT data storage efficiency.

increases. This reflects the fact that a larger t leads to more mathematical operations.

C. IoT Data Storage Efficiency

In this section, we compare our approach with existing schemes in terms of data storage efficiency. The three specific schemes, described below, have been employed to store IoT data based on cloud platforms.

- Centralized schemes: similar to the existing IoT platforms, the encrypted information M is stored in a centralized cloud server database and occupies the same storage space as the size of the encrypted information M.
- Centralized schemes with duplicates: to improve robustness of the centralized scheme, some duplicates are also stored on the cloud server. In our simulation, only one duplicate is considered. That is, the same server stores the encrypted information M and its backup data, occupying a total of $2M$ storage space.
- Blockchain-based schemes: we directly insert the encrypted message M into the transactions in the existing blockchains. Here, the blockchain scheme used for comparison is in a block structure of an Bitcoin [47], one kind of blockchain which will be discussed below. Note that the compared blockchain scheme with 10 nodes needs to store the encrypted message M synchronously, occupying a total of $10M$ storage space.

For a different number of messages generated from IoT devices, simulation results of different schemes are presented in Fig. 9. It is observed that as the number of messages increases from 450 to 5400, the storage space required for all scenarios gradually increases, an evident correlation. In comparison, the centralized scheme achieves the greatest storage efficiency. For $n \times 1024$ bits of plaintext, the storage space is about $n \times 10^6$ bits. This is reasonable considering that each bit of the stored data is indispensable and no redundant information is imported to the system. The centralized scheme with a duplicate consumes another space to store the duplicate

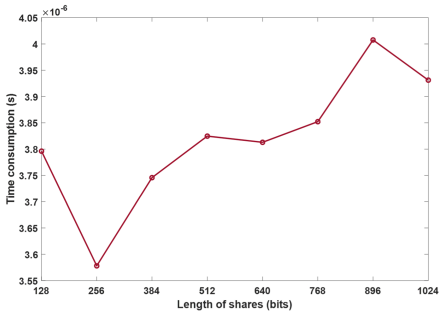

Fig. 10. Efficiency of block construction.

and hence the storage space is about twice of that of the centralized scheme itself. The blockchain-based scheme needs much more storage space than the centralized one. Schemes based on blockchains such as Bitcoin, require far more storage overhead than the other three schemes above, up to 6×10^7 bits, on average, more than seven times that of the other schemes. This makes sense considering that each message needs to be stored in all the blockchain nodes and quite a large amount of redundant data is stored in the system. A total of 10 nodes are used in the blockchain scheme, and the block header is set to 80 bits. It appears that, when storing the same number of messages, the blockchain scheme requires about 11 times as much storage space as that for the centralized ones.

Our solution reduces storage overhead by at least 80% compared to traditional blockchain methods, while the necessary storage space of our scheme is slightly less than that of the centralized scheme with a duplicate. Nonetheless, with some redundant information generated by the secret sharing scheme, our data security is greatly enhanced with tolerable storage overhead. Overall, in terms of data storage efficiency, the performance of our scheme is similar to that of the centralized ones, but largely outperforms the existing blockchain-based systems.

D. Blockchain Efficiency

In our scheme, blockchain nodes take turns to generate blocks which are continuously appended to the end of chain. The time consumption of constructing a block has a great impact on the performance of our scheme in terms of efficiency and security. For a low update speed, the IoT data are appended to the chain slowly and thus it leaves more time data tampering for the adversary.

As shown in Fig. 10, the length of shares in a block increases, and the time consumption slightly increases, with a few exceptions. This is due to the fact that a share's hash value calculation is weakly related to its length. However, it is clearly seen from the figure that the time to construct the block is basically controlled within 4×10^{-6} s, a very small number. Therefore, with this computational setup, some random factors have an enormous impact on the experimental results, and the test results under limited datasets demonstrate the apparent volatility, as shown in Fig. 10. The block construction efficiency appears naturally higher than that of existing blockchains such as Bitcoin, because we do not need to mine coins which is extremely difficult.

Consensus building among all the blockchain nodes is another challenge for the blockchain system. The blockchain scheme adopted in our solution is a private one, where the number of member nodes is strictly controlled, and the transaction is very fast. In our chain, the consensus building

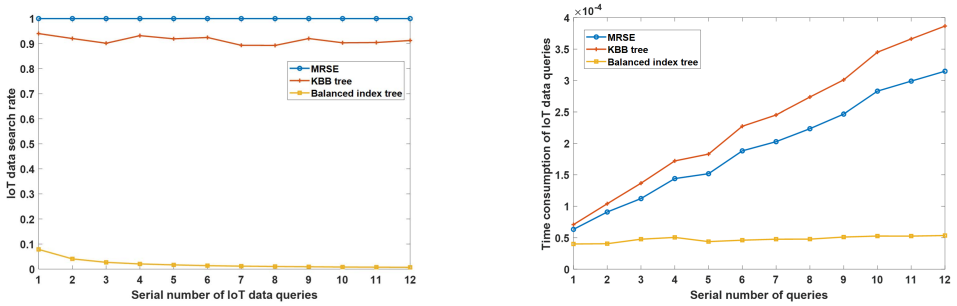

Fig. 11. IoT data retrieval efficiency in terms of (a) search rate of vectors; and (b) search time.

process is simple, i.e., delivering the newly generated block to all the other nodes. In theory, the speed of building blocks is within 4×10^{-6} s, a tiny number relative to the communication delay. Therefore, the time it takes for each node to consistently update new blocks actually depends on the communication delay. As mentioned above we set the communication delay to 300 ms. We test the time of new block construction process extensively and the results are all within 1 s. Even the nodes are far away from others, the time is controlled in 1 second. It is observed that the consumed time of our blockchain remains stable more frequently and it is more efficient than existing blockchains. This matches with the fact that the token-based consensus building is simpler than existing algorithms. Considering that the blockchain in this paper is particularly designed and simplified for IoT data storage system, and thus its efficiency increases remarkably.

E. IoT Data Retrieval Efficiency

In an IoT data retrieval process, the greatest challenge is to accurately attain the locations and hash values of the shares of interest in a gigantic data storage system. In this section, we compare search efficiency of our index structure with that of the structure in MRSE [3] and the keyword balanced binary (KBB) tree in [2].

In simulation, two metrics, i.e., search rate of the vectors and time consumption, are employed to evaluate data retrieval efficiency. The search rate of vectors is measured by the number of searched vectors over that of all the vectors. Apparently, the smaller of a search rate, the more efficient of a scheme. Time consumption of data search is the average search time of data query requests. The simulation results are presented in Fig. 11(a) and Fig. 11(b), respectively.

It appears from Fig. 11(a) that the MRSE scheme is of the highest data search rate at 100%. This is reasonable considering that all the share vectors need to be scanned so that the search results are received without errors. The data search rate of KBB tree is lower than that of MRSE. The search rate remains stable, at about 90%, with an incremental share vectors. This reflects the fact that all the vectors are randomly organized as clusters in KBB tree, and in data retrieval, some clusters are ignored if all the vectors in the cluster are not the resulting candidates.

The balanced index tree proposed in this paper is of the lowest data search rate and it outperforms the other two schemes in a great deal. Based on Algorithm 2, most irrelevant branches in the tree are wiped out and hence the retrieval efficiency is greatly improved. Different from the KBB tree, in the index tree proposed in this paper, all the vectors in a cluster are similar and the time entries of two nodes in the same level are mutually exclusive with each other. As a consequence, the IoT data retrieval efficiency is further improved. Moreover, with share vectors increasing, the search rate gradually decreases, because all the share vectors are organized in order and parameter $|T_i - T_j|$ remains stable in our simulation.

It is observed in Fig.11(b), the time consumption of MRSE and KBB tree gradually increases with the increase of share vectors. This is quite straightforward considering that searching more paths and share vectors needs to consume more time. An interesting observation is that KBB tree performs worse than the MRSE scheme. This reflects the fact that the share vectors in the leaf nodes are chaotic and the index tree cannot efficiently specify the candidates. Apparently, the balanced tree proposed in this paper largely outperforms the other two schemes with reasons similar to those for data search rate.

IX. Conclusion

In this paper, we propose a novel secure and distributed IoT data storage framework in the presence of untrusted or vulnerable cloud servers. In our system, the smart devices first map original messages to a set of shorter message shares. The shares rather than the original messages are randomly and distributively stored in the cloud nodes. In each node, the stored shares are employed to construct the blocks which are chained together. Based on the properties of blockchains, the integrity of the shares is protected and meanwhile the location information of the shares is disclosed among the whole system. To improve IoT data search efficiency, a balanced index tree and a depth-first data retrieval algorithm are proposed. Once a minimum t shares are received, data users can recover the original IoT message. Simulation experiment shows that our scheme reduces the storage overhead by 80% compared with the traditional blockchain cloud storage scheme. At the same time, the time-consuming of the retrieval algorithm we designed is not related to the number of queries, which has efficiency advantage of more than 70% compared with ciphertext retrieval schemes such as MRSE. Finally, the formal analysis of the scheme proves that our scheme can effectively protect data security.

Our future work is discussed as follows. First, for simplicity, the system employs only one IoT device and one data user in simulation. However, a large number of IoT devices and data users practically coexist in the framework, and we will thoroughly evaluate the scalability of our approach in the future. Second, in our scheme, only one IoT data retrieval pattern, which refers to searching the data of interest on an IoT device in a period, is provided to a data user. In reality, IoT data users may want to have a maximal, minimal or middle value within a period, which is not supported by our approach and we will update our scheme accordingly.

REFERENCES

[1] S. Uthayashangar, T. Dhanya, S. Dharshini, and R. Gayathri, "Decentralized blockchain based system for secure data storage in cloud," in *Proc. Int. Conf. Syst., Comput., Autom. Netw. (ICSCAN)*, Jul. 2021, pp. 1–5.

[2] Z. Xia, X. Wang, X. Sun, and Q. Wang, "A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 2, pp. 340–352, Jan. 2015.

[3] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 1, pp. 222–233, Jan. 2013.

[4] J. S. Fu, Y. Liu, H. C. Chao, B. K. Bhargava, and Z.-J. Zhang, "Secure data storage and searching for industrial IoT by integrating fog computing and cloud computing," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4519–4528, Jan. 2018.

[5] A. S. Alrawahi, K. Lee, and A. Lotfi, "A multiobjective QoS model for trading cloud of things resources," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 9447–9463, Dec. 2019.

[6] H. Yaish, M. Goyal, and G. Feuerlicht, "Multi-tenant elastic extension tables data management," *Proc. Comput. Sci.*, vol. 29, pp. 2168–2181, Jan. 2014.

[7] O. Curé, F. Kerdjoudj, D. Faye, C. Le Duc, and M. Lamolle, "On the potential integration of an ontology-based data access approach in NoSQL stores," *Int. J. Distrib. Syst. Technol.*, vol. 4, no. 3, pp. 17–30, Jul. 2013.

[8] Z. Kaoudi and I. Manolescu, "Cloud-based RDF data management," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, Jun. 2014, pp. 725–729.

[9] M. Li, Z. Zhu, and G. Chen, "A scalable and high-efficiency discovery service using a new storage," in *Proc. IEEE 37th Annu. Comput. Softw. Appl. Conf.*, Jul. 2013, pp. 754–759.

[10] C. Wang et al., "Apache IoTDB: Time-series database for Internet of Things," *Proc. VLDB Endowment*, vol. 13, no. 12, pp. 2901–2904, Aug. 2020.

[11] W. Tärneberg, V. Chandrasekaran, and M. Humphrey, "Experiences creating a framework for smart traffic control using AWS IOT," in *Proc. IEEE/ACM 9th Int. Conf. Utility Cloud Comput. (UCC)*, 2016, pp. 63–69.

[12] T. Zhou and J. Zhang, "Design and implementation of agricultural Internet of Things system based on Aliyun IoT platform and STM32," *J. Phys., Conf. Ser.*, vol. 1574, no. 1, 2020, doi: 10.1088/1742-6596/1574/1/012159.

[13] (2017). *Microsoft Azure Blockchain Solutions*. [Online]. Available: https://azure.microsoft.com/en-in/solutions/blockchain/

[14] K. Xue, N. Gai, J. Hong, D. S. L. Wei, P. Hong, and N. Yu, "Efficient and secure attribute-based access control with identical sub-policies frequently used in cloud storage," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 1, pp. 635–646, Jan. 2022.

[15] G. Asharov and Y. Lindell, "A full proof of the BGW protocol for perfectly secure multiparty computation," *J. Cryptol.*, vol. 30, no. 1, pp. 58–151, Aug. 2017.

[16] H. Chen and C.-C. Chang, "A novel (t, n) secret sharing scheme based upon Euler's theorem," *Secur. Commun. Netw.*, vol. 2019, pp. 1–7, Apr. 2019, doi: 10.1155/2019/2387358.

[17] L. Wu, F. Miao, K. Meng, and X. Wang, "A simple construction of CRT-based ideal secret sharing scheme and its security extension based on common factor," *Frontiers Comput. Sci.*, vol. 16, no. 1, pp. 1–9, Feb. 2022.

[18] J. Xie et al., "A survey of blockchain technology applied to smart cities: Research issues and challenges," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2794–2830, 3rd Quart., 2019.

[19] T. Kim, J. Noh, and S. Cho, "SCC: Storage compression consensus for blockchain in lightweight IoT network," in *Proc. IEEE Int. Conf. Consum. Electron. (ICCE)*, Jan. 2019, pp. 1–4.

[20] P. Ratta, A. Kaur, S. Sharma, M. Shabaz, and G. Dhiman, "Application of blockchain and Internet of Things in healthcare and medical sector: Applications, challenges, and future perspectives," *J. Food Qual.*, vol. 2021, pp. 1–20, May 2021, doi: 10.1155/2021/7608296.

[21] W. Wang, P. Xu, and L. T. Yang, "Secure data collection, storage and access in cloud-assisted IoT," *IEEE Cloud Comput.*, vol. 5, no. 4, pp. 77–88, Jul. 2018.

[22] R. P. Rashmi, Y. Gandhi, V. Sarmalkar, P. Pund, and V. Khetani, "RDPC: Secure cloud storage with deduplication technique," in *Proc. 4th Int. Conf. I-SMAC (IoT Social, Mobile, Analytics Cloud) (I-SMAC)*, Oct. 2020, pp. 1280–1283.

[23] R. Cao, Z. Tang, C. Liu, and B. Veeravalli, "A scalable multicloud storage architecture for cloud-supported medical Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 3, pp. 1641–1654, Mar. 2019.

[24] J. Wu, W. Xu, and J. Xia, "Load balancing cloud storage data distribution strategy of Internet of Things terminal nodes considering access cost," *Comput. Intell. Neurosci.*, vol. 2022, pp. 1–11, Jan. 2022, doi: 10.1155/2022/7849726.

[25] L. Jiang, L. D. Xu, H. Cai, Z. Jiang, F. Bu, and B. Xu, "An IoT-oriented data storage framework in cloud computing platform," *IEEE Trans. Ind. Informat.*, vol. 10, no. 2, pp. 1443–1451, May 2014.

[26] J.-W. Lin, J. M. Arul, and J.-T. Kao, "A bottom-up tree based storage approach for efficient IoT data analytics in cloud systems," *J. Grid Comput.*, vol. 19, no. 1, pp. 1–19, Mar. 2021.

[27] M. H. Salih Mohammed, "A hybrid framework for securing data transmission in Internet of Things (IoTs) environment using blockchain approach," in *Proc. IEEE Int. IoT, Electron. Mechatronics Conf. (IEMTRONICS)*, Apr. 2021, pp. 1–10.

[28] T. Xu, Z. Fu, M. Yu, J. Wang, H. Liu, and T. Qiu, "Blockchain based data protection framework for IoT in untrusted storage," in *Proc. IEEE 24th Int. Conf. Comput. Supported Cooperat. Work Design (CSCWD)*, May 2021, pp. 813–818.

[29] C. Wiraatmaja, Y. Zhang, M. Sasabe, and S. Kasahara, "Cost-efficient blockchain-based access control for the Internet of Things," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2021, pp. 1–6.

[30] L. Zhang, M. Peng, W. Wang, Z. Jin, Y. Su, and H. Chen, "Secure and efficient data storage and sharing scheme for blockchain-based mobile-edge computing," *Trans. Emerg. Telecommun. Technol.*, vol. 32, no. 10, Oct. 2021, doi: 10.1002/ett.4315.

[31] K. Huang et al., "EVA: Efficient versatile auditing scheme for IoT-based datamarket in jointcloud," *IEEE Internet Things J.*, vol. 7, no. 2, pp. 882–892, Feb. 2020.

[32] B. Huang et al., "BoR: Toward high-performance permissioned blockchain in RDMA-enabled network," *IEEE Trans. Services Comput.*, vol. 13, no. 2, pp. 301–313, Mar./Apr. 2020, doi: 10.1109/TSC.2019.2948009.

[33] (2016). *IBM Blockchain*. [Online]. Available: https://www.ibm.com/blockchain

[34] B. Tan, J. Yan, S. Chen, and X. Liu, "The impact of blockchain on food supply chain: The case of walmart," in *Proc. Int. Conf. Smart Blockchain*. Cham, Switzerland: Springer, 2018, pp. 167–177.

[35] T. Jensen, J. Hedman, and S. Henningsson, "How TradeLens delivers business value with blockchain technology," *MIS Quart. Executive*, vol. 18, no. 4, pp. 221–243, Dec. 2019.

[36] G. Srivastava, R. M. Parizi, A. Dehghantanha, and K. K. R. Choo, "Data sharing and privacy for patient IoT devices using blockchain," in *Proc. Int. Conf. Smart City Inf.*, vol. 1122. Springer, 2019, pp. 334–348, doi: 10.1007/978-981-15-1301-5_27.

[37] M. Farhadi, H. Bypour, and R. Mortazavi, "An efficient secret sharing-based storage system for cloud-based IoTs," in *Proc. 16th Int. Iranian Soc. Cryptol. Conf. Inf. Secur. Cryptol. (ISCISC)*, Aug. 2019, pp. 122–127.

[38] A. Galletta, J. Taheri, and M. Villari, "On the applicability of secret share algorithms for saving data on IoT, edge and cloud devices," in *Proc. Int. Conf. Internet Things (iThings) IEEE Green Comput. Commun. (GreenCom) IEEE Cyber, Phys. Social Comput. (CPSCom) IEEE Smart Data (SmartData)*, Jul. 2019, pp. 14–21.

[39] L. Tan, K. Yu, C. Yang, and A. K. Bashir, "A blockchain-based Shamir's threshold cryptography for data protection in industrial Internet of Things of smart city," in *Proc. 1st Workshop Artif. Intell. Blockchain Technol. Smart Cities 6G*, Oct. 2021, pp. 13–18.

[40] R. Li, T. Song, B. Mei, H. Li, X. Cheng, and L. Sun, "Blockchain for large-scale Internet of Things data storage and protection," *IEEE Trans. Services Comput.*, vol. 12, no. 5, pp. 762–771, Sep./Oct. 2018.

[41] S. Wan, Y. Zhao, T. Wang, Z. Gu, Q. H. Abbasi, and K.-K. R. Choo, "Multi-dimensional data indexing and range query processing via Voronoi diagram for Internet of Things," *Future Gener. Comput. Syst.*, vol. 91, pp. 382–391, Feb. 2019.

[42] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979.

[43] N. Wang, J. Fu, J. Li, and B. K. Bhargava, "Source-location privacy protection based on anonymity cloud in wireless sensor networks," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 100–114, 2019.

[44] A. Beimel, "Secret-sharing schemes: A survey," in *Proc. Int. Conf. Coding Cryptol.*, vol. 6639. Springer, 2011, pp. 11–46, doi: 10.1007/978-3-642-20901-7_2.

[45] Z. Eslami and J. Zarepour Ahmadabadi, "A verifiable multi-secret sharing scheme based on cellular automata," *Inf. Sci.*, vol. 180, no. 15, pp. 2889–2894, Aug. 2010.

[46] P. Ferrari et al., "Evaluation of communication delay in IoT applications based on OPC UA," in *Proc. Workshop Metrol. Ind. 4.0 IoT*, 2018, pp. 224–229.

[47] S. Nakamoto. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. [Online]. Available: https://bitcoin.org/bitcoin.pdf