Future Generation Computer Systems 151 (2024) 32–44

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

MMDS: A secure and verifiable multimedia data search scheme for cloud-assisted edge computing

Shiwen Zhang [a,b], Jiayi He [a], Wei Liang [a], Keqin Li [c,*]

[a] School of Computer Science and Engineering, Hunan University of Science and Technology, XiangTan, Hunan, China
[b] Advanced Cryptography and System Security Key Laboratory of Sichuan Province, China
[c] Department of Computer Science, State University of New York, New Paltz, NY 12561, USA

ARTICLE INFO

Keywords:
Keyword search
Fine-grained verification
Independent protection
Internet of Things (IoT)
Blockchain
Cloud-assisted edge computing

ABSTRACT

With the rapid development and popularity of Internet of Things (IoT) technology, IoT devices can generate MultiMedia Big Data (MMBD) as multimedia devices. MMBD is often encrypted and stored on an untrusted cloud server. Searchable encryption is an effective way which can finish a controllable keyword search of ciphertext. However, there are two limitations to these existing works. On the one hand, it is hard to independently protect the data privacy of each IoT device. On the other hand, when faced with an untrusted cloud server, it is difficult to implement more flexible fine-grained verification for search results. To overcome these limitations, we propose a secure and verifiable MultiMedia Data Search (MMDS) scheme for cloud-assisted edge computing. To protect the data of IoT devices independently, we designed a secure, flexible, and efficient keyword search mechanism based on bilinear pairings. To achieve a more flexible and practical search result verification mechanism, we design a fine-grained verification algorithm combining blockchain and hashing techniques. We have conducted performance evaluations and security proof and analysis for MMDS scheme. Finally, we applied MMDS scheme to solve real problems in intelligent multimedia systems. Security proof and analysis prove the security of MMDS. Performance analyses and evaluations further confirm that MMDS is efficient and feasible for practical applications.

1. Introduction

As the supply of multimedia devices on the Internet of Things (IoT) grows exponentially, enormous amounts of MultiMedia Big Data (MMBD) are being generated [1]. New forecasts from the International Data Corporation (IDC) predict that in 2025, there will be 41.65 billion connected IoT devices generating 79.4 ZB of data. MMBD greatly enhances current multimedia applications such as multimedia searches, healthcare services, advertisements, recommendations, and smart cities [2,3].

Meanwhile, IoT also brings new challenges to MMBD. Due to the limited storage of IoT devices, MMBD are often outsourced and stored on a cloud server. Other users can search for stored MMBD using specific keywords to obtain the desired data. If MMBD is collected from IoT devices and stored directly on a cloud server, it will lead to potential security issues. Hence, IoT devices usually encrypt data before it is outsourced to a cloud server for increased security. However, IoT devices have limited computing power. They are usually unable to perform complex encryption algorithms effectively. The emergence of cloud-assisted edge computing has fortunately compensated for the lack of IoT devices. Edge servers approach IoT devices more closely than the cloud server. They can provide data computation, data transmission, and other services to IoT devices [4].

As shown in Fig. 1, edge servers can collect massive amounts of MMBD from various IoT devices. This article uses file data in MMBD as an example. To ensure the security of the data, edge servers encrypt the data and outsource it to the cloud server. However, data encryption makes the data availability much lower. How to search the encrypted data is a pressing issue. To search for encrypted data, scholars have extensively explored Searchable Encryption (SE) technology. Unfortunately, existing schemes [5–10] are not able to independently protect data from IoT devices, i.e. the edge server uses the same key to encrypt outsourced data for nearby IoT devices. Such schemes may not be suitable for deployment in IoT environments. Once an edge server accidentally leaks the encryption key, the data privacy of all IoT devices may be violated. One solution is for edge servers to generate independent encryption keys for each IoT device in its coverage. But IoT devices are movable. The IoT devices in the edge server's coverage are different and uncertain at different times and spaces. In this case, edge servers

* Corresponding author.
E-mail address: lik@newpaltz.edu (K. Li).

https://doi.org/10.1016/j.future.2023.09.023
Received 3 July 2023; Received in revised form 14 September 2023; Accepted 16 September 2023
Available online 27 September 2023

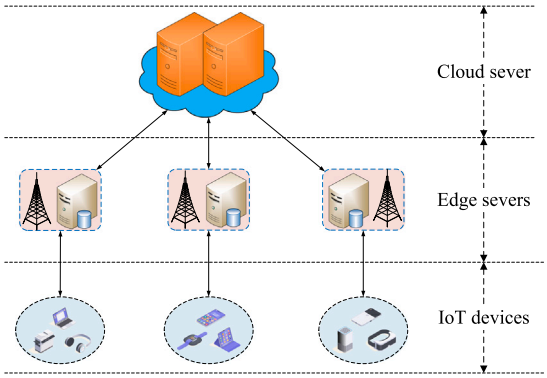

Fig. 1. An overview of MMBD sharing for cloud-assisted edge computing.

are required to maintain numerous encryption keys. It will inevitably result in a heavy key management burden. In addition, data users would need to submit multiple trapdoors using different encryption keys to retrieve data. This will directly lead to high communication and search costs. Therefore, they need a secure, flexible, and efficient keyword search mechanism to independently protect the data privacy of IoT devices.

In practice, deletion and tampering of search results by an untrusted cloud server may be motivated by a variety of reasons. Thus verification schemes are essential in a keyword search. Most existing verification schemes [5–8,11–16] only provide coarse-grained verification, i.e. they can only answer "yes" or "no" to a set of search results returned by a cloud server. Once they output "no", the set of search results will be discarded, even though only one of the search results may be incorrect. Such schemes may not be suitable for practical applications. Because data users prefer to find out the unqualified ciphertexts from a set of unqualified search results and use the qualified ciphertexts as the final search results. Secondly, if the cloud server ignores some search results, they also want to check how much or which ciphertexts are not returned. In addition, the cloud server has the same behavior of tampering with or falsifying the verification information itself. This will result in ineffective verification. Verification overhead is one of the main reasons for the existing performance limitations of verification schemes. Therefore, it is urgent and challenging to implement a more flexible and practical fine-grained verification scheme.

In this paper, in contrast to existing work, aiming at the challenges mentioned above, we propose a secure and verifiable multimedia data search scheme for cloud-assisted edge computing, called MMDS. We have implemented a secure, flexible, and efficient keyword search mechanism. Specifically, we use bilinear pairings to encrypt keywords to build searchable encrypted indexes. The edge servers are able to build encrypted indexes for IoT devices using random keys, which protects the data privacy of each IoT device independently. When an IoT device searches for encrypted data, the edge server sends a trapdoor to the cloud server without knowing the encryption key. The cloud server performs a keyword ciphertext search. We also design a fine-grained verification algorithm that combines blockchain and hashing techniques. Specifically, we use the hashing function to construct the verification information. Using the unidirectional and collision resistance of hashing function to protect and verify the search results. The verification information is recorded on the blockchain. The tamper-proof function of the blockchain is used to ensure the truthfulness of the verification information. Moreover, the verification information is returned by using the consensus mechanism of the blockchain, thus enabling fine-grained verification. In addition, to minimize the computational pressure on IoT devices, we delegate a large number of computational tasks (e.g., encryption, constructing trapdoors, verifying search results and decryption, etc.) from IoT devices to edge servers. Finally, high efficiency and safety are well demonstrated in our scheme through extensive experiments and security proof and analysis. Summing up, the key contributions of this paper are as follows.

1. We proposed a secure and verifiable multimedia data search scheme for cloud-assisted edge computing. It independently protects the data privacy of IoT devices and achieves fine-grained verification. At the same time, it reduces the overhead of IoT devices. To the best of our knowledge, it is the first scheme to implement fine-grained verification for cloud-assisted edge computing.
2. We designed a secure, flexible, and efficient keyword search mechanism based on bilinear pairs for cloud-assisted edge computing. It allows edge servers to encrypt data for IoT devices using a random key. Edge servers can generate a trapdoor for an IoT device without knowing of the encryption key. The purpose is to search for encrypted data in the cloud server.
3. We have created a fine-grained search results verification algorithm combining blockchain and hashing technology for cloud-assisted edge computing. Firstly, the algorithm is able to verify the correctness and completeness of the search results. Secondly, for a failed search result, it is able to find the correct ciphertext as the final search result. It is also able to check how many failed ciphertexts and which ciphertexts were ignored by the cloud server. Finally, it ensures the validity of the verification results and delegates all the verification overhead to the edge servers.
4. We have conducted performance evaluations and security proof and analysis for the proposed MMDS scheme. In addition, we discuss proposed schemes for engineering application problems related to intelligent multimedia systems (i.e., electronic libraries). The extensive experiment evaluations prove that MMDS is efficient and feasible.

The remainder of the paper is as follows. Section 2 reviews related work. In Section 3, the system and threat models and design goals are presented. Section 4 is a description of the associated preparation work. Then, Section 5 introduces the details of the proposed MMDS scheme. Furthermore, Section 6 and Section 7 give the security proof and analysis and the performance evaluation. Finally, Section 8 concludes the paper and looks at future work.

2. Related work

An untrusted cloud server may have illegitimate behaviors in the search process. Therefore, we will focus on keyword search and search result verification.

2.1. Keyword search in cloud computing

SE is a technique that enables efficient search of encrypted data. The academic who first proposed the SE concept was Song et al. [17]. In their scheme, the cloud server scans each encrypted keyword in turn, and this approach makes the search cost overloaded. To raise the search efficiency, Goh et al. [18] introduced the Bloom Filter technique to design the index. To address the problem of key distribution in SE, Boneh et al. [19] proposed Public key Encryption with Keyword Search (PEKS), in which the sender uses the receiver's public key to encrypt emails and constructs a secure index. The receiver uses his private key to generate a query trapdoor, solving the key distribution. The secure inverted indexes structure was proposed by [20]. Their starting point was to enable further improvements in search performance. Zhang et al. [21] discuss and analyze the security model for multiple data owners, and the first scheme is proposed. They introduced an additional management server. To address this difficulty, Yin et al. [22] removed the administration server and further implemented a sorted search scheme.

These schemes do not consider the possibility of the untrusted cloud server, it will return unqualified search results. As a result, various verification schemes have emerged. Chai et al. [23] first used tree-based indexing and hash chaining techniques, making verifiable SE

schemes a reality. Wang et al. [24] combined the techniques of the Bloom Filter and the Merkel Hash Trees and implemented a verifiable data search scheme. The implementation of a multi-user searchable encryption scheme based on a combination of RSA accumulator and broadcast encryption is proposed by [25]. Jiang et al. [26] introduced the theory of relevance score. On the basis of this, they proposed a verifiable search scheme for ranked multi-keyword. To reduce the overhead of verification, Zhang et al. [27] introduced the concept of deterrence, making the cloud server avoid illegal operations. To improve the practicality of the scheme, Zhu et al. [28] proposed a multi-user search scheme that enables data updating and verifiability. Liu et al. [29] implement a sorted search of dynamic document collections with guaranteed verifiability, but this scheme imposes some expenses. Tong et al. [30] proposed a verifiable data search scheme, this scheme can prevent key leakage and further improve the security of verification. Yin et al. [31] first proposed a fine-grained search result verification scheme using the Counting Bloom Filter. But it exists the common problem of the Counting Bloom Filter, bringing a large storage overhead and false positives with a certain probability.

In the context of cloud computing, many keyword search schemes have been proposed. Their algorithms for encryption, decryption, and verification have huge overheads. They cannot satisfy the needs of today's IoT environment anymore. Hence, they lack a certain practicality.

2.2. Keyword search in blockchain

Blockchain technology has features such as decentralization, tamper-proof, and a complete history of all transactions. Smart contracts in blockchain are self-executing contracts whose terms are written directly in the lines of code. Once incidents trigger the terms in the contract, the code will be executed automatically. As a result, blockchain and smart contracts are widely used to perform verification operations in SE schemes. Hu et al. [32] enables decentralized and fair keyword search of encrypted databases via smart contracts in Ethernet. [33] have similarities to [32]. In this type of scheme, the search index is stored in a smart contract, and the search is executed through the smart contract to ensure the validity of the search results. However, executing smart contracts can lead to huge gas consumption. Miners in the blockchain may skip verification and acknowledge the validity of the search results directly to save computational overhead. This phenomenon is known as the verifier's dilemma [34], meaning the search results are claimed to be verified, but in fact, are not verified.

Therefore, some scholars combined traditional verifiable search schemes with blockchain. Li et al. [12] proposed a searchable encryption scheme verification mechanism based on Bitcoin. In this scheme, blockchain nodes compare the Message Authentication Code (MAC) of search results with the reserved MAC in the prediction list to verify the correctness of the search results. Li et al. [13] proposed a verification using tags stored on the blockchain to construct the Merkle Hash Tree for public auditing of data integrity. Zhao et al. [14] used blockchain technology to construct a scheme for checking data integrity using bilinear pairings, the Lifted EC-ElGamal cryptosystem, and aggregated signatures for bulk verification. Cai et al. [15] proposed a two-level verification mechanism, the users and the blockchain nodes perform verification sequentially to reach a consensus on the search results. In the [16] scheme, blockchain nodes use intermediate evidence generated during the matching process to verify the correctness of the search results. However, none of them implemented fine-grained search results verification.

2.3. Keyword search for cloud-assisted edge computing

Next, we briefly review recent advances in keyword search schemes for cloud-assisted edge computing. Combined SSE and public key encryption techniques, Mollah et al. [5] proposed the first secure data-sharing scheme for cloud-assisted edge computing. But it requires edge servers to share indexed encryption keys. It cannot independently protect the data security of IoT devices. To solve this problem, Ye et al. [11] introduced PEKS. It generates public and private keys for each user. Edge servers use the public key to build encrypted indexes. Like traditional PEKS, it generates a large number of ciphertext copies. Moreover, they do not completely offload the computational burden from IoT devices. Wang et al. [6] delegate expensive operations to the edge servers and speed up the process of ciphertext generation by the edge servers. Li et al. [7] absorbed the features of the chinese remainder theorem and latent dirichlet allocation, and proposed a semantic-based verification keyword search scheme. With the objective of improving the search efficiency of edge and the cloud server, a lightweight SE scheme based on untrusted cloud/trusted edge architecture was proposed [8]. In order to achieve lightweight access control in the IoT environment, Zhang et al. [9] proposed a lightweight attribute encryption scheme. A multi-keyword SE scheme was constructed by Liu et al. [10] and implemented fine-grained access control based on attribute encryption.

Cloud-assisted edge computing combined with the advantages of cloud and edge computing. It can serve today's IoT environment better and provide more feasibility for keyword search schemes [35]. Probably because of the different research focuses of the schemes, none of them achieve fine-grained validation. But MMDS can achieve fine-grained verification and protect the data privacy of each IoT device independently at the same time. Based on this, MMDS also minimizes the overhead of IoT devices. Thus, MMDS is highly feasible for real-world IoT environments.

3. Problem formulation

In this section, the system model and threat model of MMDS are established, and the design goal of our MMDS is stated.

3.1. System model

First, we present the system model of MMDS. In Fig. 2, our system model consists of five entities: data owners, data users, Cloud Server (CS), Edge Severs (ESs), and BlockChain (BC), respectively. In the following, the functions of each entity will be described in detail.

Data owners: Data owners upload a large number of raw data flies to nearby ESs.

Data users: Data users send search keywords to nearby ESs.

CS: CS is in charge of storing ciphertexts as well as performing search operations.

ESs: When data owners upload data files, ESs establish encrypted indexes and encrypt the file sets, as well as generate verification information for each file; when data users send search requests, ESs generates trapdoors, verify the search results, and decrypts the correct ciphertexts.

BC: BC is responsible for storing the verification information and performing the search operation.

In MMDS, the storage and computing capacity of most IoT devices is limited, so data owners upload their raw data flies to nearby ESs (Step (1)). ESs outsource encrypted indexes and ciphertexts to CS (Step (2)), and store verification information in BC (Step (3)). It is worth noting that to improve the security and flexibility of our MMDS system, ESs build encrypted indexes using random keys each time. The encrypted indexes are built differently for different IoT devices, even for the same keywords. Data users submit search keywords (Step (4)) via their nearby ES. Once an ES receives the search request, it generates a trapdoor and delivers it to the CS and BC (Step (5)). The search result is returned by CS to ES (Step (6)). The BS returns the corresponding verification information to the ES (Step (7)). Finally, the ES completes the fine-grained verification of the search results, decrypts qualified ciphertext, and returns it to the data users (Step (8)).

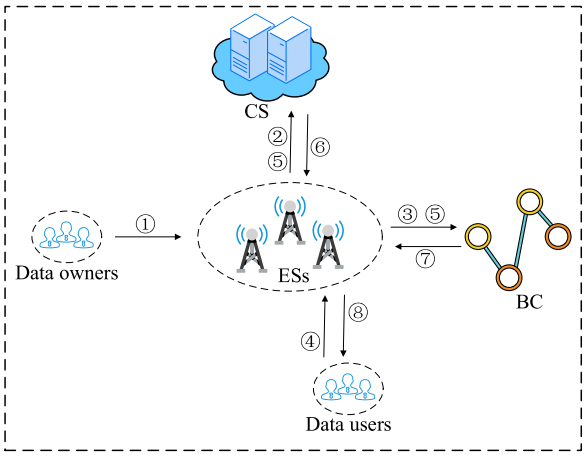

Fig. 2. System model of MMDS.

3.2. Threat model

In the MMDS scheme, data owners and data users are trusted entities. ESs are deployed for their IoT devices, so IoT devices trust ESs in their vicinity. BC are open, transparent, decentralized, non-falsifiable entities, which are trusted in terms of correctness and availability of execution. Unlike other works, CS in this scheme is malicious. Due to profit or other motives, it may perform only a small part of the search operation, falsifying or tampering with the search results. If CS knows that a search result verification algorithm is deployed, it will remove or tamper with the verification information in order to avoid responsibility. Besides the above description, our scheme may be attacked by CS or external attackers such as Chosen-Plaintext Attack (CPA), which is the goal to be achieved.

3.3. Design goals

In this work, our MMDS should satisfy the following three goals:

1. Independent privacy-protection: MMDS should independently protect the privacy of data files and keywords of every IoT device. The unlinkability of trapdoors should also be implemented.

2. Fine-grained verifiability: MMDS should provide a fine-grained search results verification mechanism for data users.

3. Search effectiveness and efficiency: MMDS should provide accurate search results for data users. MMDS should efficiently implement keyword search and verification mechanisms.

4. Preliminaries

4.1. Bilinear pairing map

Bilinear pairing is a binary mapping, which is used as a construction tool for cryptographic algorithms and is used in the SE mechanism to encrypt keywords or data files. Their security is all based on different security assumptions, such as DLP (Section 4.2), and DDHP (Section 4.3).

Suppose G_1, G_2 represents the group of two multiplicative cycles of prime order p, generating element g. A bilinear map e: $G_1 \times G_1 \rightarrow G_2$ following nature is guaranteed:

1. Computable: For any g, $h \in G_1$, there is a time algorithm for the polynomial to compute $e(g, h) \in G_2$.

2. Bilinear: For all x, $y \in Z_p^*$ and g, $h \in G_1$, the equality $e(g^x, h^y) = e(g, h)^{xy}$ holds.

3. Non-degenerate: If g, h are generators of G_1, then $e(g, h)$ is a generator of G_2.

4.2. Discrete logarithm problem (DLP)

The DLP problem is as follows: g represent a generator of the group G_1 with order p. If the values of g and g^a are given, ask for the calculation of a $\in Z_p^*$. The non-existence of polynomial-time algorithms that have non-negligible advantages in solving DLP.

4.3. Decisional diffie–hellman problem (DDHP)

The DDHP problem is as follows: g represent a generator of the group G with order p, Randomly generate a, b, $c \in Z_p^*$ given (g, g^a, g^b), effectively distinguishing between g^{ab} and g^c (i.e., determining whether $ab = c$ holds). The non-existence of polynomial-time algorithms that have non-negligible advantages in solving DDHP.

4.4. Hashing function

In a hash algorithm, a binary value string of any length is mapped to a fixed length binary value string. It is mapped through the original data using the hash algorithm. The result of the mapping is the hash value string obtained from the original data. The hash function has the following properties.

1. Unidirectional: It is not feasible to calculate message m according to $Hash_{(m)} = h$ for a given hash value for any hash function.

2. Collision resistance: It is infeasible to find so that $Hash_{(m_1)} \neq Hash_{(m_2)}$ for two messages $m_1 \neq m_2$ for any Hash function.

4.5. Consensus mechanism in blockchain

Blocks are sequentially linked into a chain, known as a blockchain, in the order in which they were formed, where each block holds specific information. The blockchain can be understood as a shared, tamper-evident ledger. It does not rely on third-party regulation and stores, verifies, and transmits data through its own distributed nodes.

Storing data as chains in the blockchain is necessary to safeguard the data from tampering. Each participating node in the blockchain has equal rights to record data, which is drastically different from centralized architectures. For consistency and correctness of the data, the nodes need a consensus mechanism to make the data agreeable.

Consensus mechanisms determine the bookkeeping rights of data through special rules known to each node. The aim is to keep the data consistent across the distributed nodes in the blockchain. Some of the more common blockchain consensus mechanisms [36–39] are Proof Of Work (POW), Proof Of Stake (POS), etc.

5. Our proposed MMDS scheme

In this section, we firstly introduced the main idea of MMDS. Then we formally define MMDS, and finally, demonstrate its concrete structure. Before that, the main notations and descriptions are given in Table 1.

5.1. Main idea

In the previous scheme, the ES used an index key to build a searchable encrypted index and upload it to the CS. When a data user wants to access the ciphertext, in order to be able to retrieve the encrypted index efficiently, it needs to communicate with the ES to obtain the corresponding key to build a trapdoor. However, there is a significant communication overhead and security risk associated with this approach. To solve this problem, we have cleverly constructed an encryption algorithm and a trapdoor generation algorithm using bilinear pairings. The encryption algorithm allows ESs to construct a searchable encryption index using a different random key. The trapdoor generation algorithm allows ESs to generate trapdoors for data users without having to communicate with other ESs.

Table 1
Notations and Descriptions.

Notations	Descriptions
$F = \{f_1, f_2, \ldots, f_n\}$	File set
K	Symmetric key of file
$C = \{c_1, c_2, \ldots, c_n\}$	Ciphertext set
$W = \{w_1, w_2, \ldots, w_m\}$	Keyword set
$CI = \{ci_{w_1}, ci_{w_2}, \ldots, ci_{w_m}\}$	Encryption index in ciphertext of keyword w
$V = \{v_1, v_2, \ldots, v_n\}$	Verification information of file set F
$VI = \{vi_{w_1}, vi_{w_2}, \ldots, vi_{w_m}\}$	Encryption index in verification information of keyword w
T_w	Trapdoor of keyword w
$CR_w = \{cr_1, cr_2, \ldots, cr_t\}$	Search result set in ciphertext of keyword w
$VR_w = \{vr_1, vr_2, \ldots, vr_t\}$	Search result set in verification information of keyword w
R	Result of verification
$\|\alpha\|$	If α is a string, $\|\alpha\|$ denotes the bit length of α;
	If α is a set, $\|\alpha\|$ denotes cardinality of α.

In addition, previous schemes did not allow for fine-grained verification of search results. To solve this problem, we have designed a verification algorithm. We use the unidirectional nature of the hash function to protect the search results, and the tamper-proof nature of BC to ensure the validity of the verification information. The collision resistance of the hash function and the consensus mechanism of BC achieve fine-grained verification. During the encryption phase, the ESs use a hash function to compute the verification information and record it on the BC. In order to be able to obtain the corresponding verification information from the BC, the ESs build a searchable encryption index for the verification information. This is recorded on the blockchain along with the verification information in the form of a backward index. If the encrypted indexes of the verification information and the ciphertext are the same, the attackers may find the connection and the privacy of the data may be leaked. Taking this into account, we require that a different indexing key is used each time an encrypted index is constructed. In other words, the same keyword is encrypted and indexed differently. Finally, we cleverly design the structure of the encrypted index using the nature of bilinear pairings. With no extra communication, only one trapdoor is needed to be able to retrieve the encrypted indexes in both CS and BC.

5.2. Scheme definition

In the proposed MMDS scheme, system setup, encrypt, generate trapdoor, search, verify and decrypt are the six algorithms. The following formal definitions exist.

- **System Setup** $(1^{\lambda}) \rightarrow (CP, SP)$: Input security parameters λ to the system, output common parameters CP and secret parameters SP, and distribute the SP secretly to ESs in the system.
- **Encrypt** $(F, W, K, CP, SP) \rightarrow (C, CI, V, VI)$: ESs input file set F, keyword set W, symmetric key K, random key k_c, $k_v \in Z_p^*$, common parameters CP and secret parameters SP, output ciphertext set C and its encryption indexes CI, and verification information set V and its encryption indexes VI.
- **Generate Trapdoor** $(w, CP, SP) \rightarrow (T_w)$: ESs input a data user's search keyword w, random numbers r_1, $r_2 \in Z_p^*$, common parameters CP and secret parameters SP, and it generates the corresponding trapdoor T_w.
- **Search** $(T_w, CI, VI) \rightarrow (CR_w, VR_w)$: CS inputs trapdoor T_w and CI, outputs the corresponding set of ciphertext search results CR_w. BC inputs T_w and VI, outputs the corresponding set of verification information results VR_w.
- **Verify** $(CR_w, VR_w) \rightarrow (R)$: ESs inputs the set of ciphertext search results CR_w, the set of verification information search results VR_w, and outputs the verification results R.
- **Decrypt** $(CR_w, R, K) \rightarrow (F_w)$: ESs inputs the search result set of ciphertext CR_w, the verification result R, symmetric key K, and outputs the set of correct files F_w.

Keywords	Files
w_1	$f_1, f_2, f_4, f_8 \ldots$
w_2	$f_1, f_3, f_5, f_{10} \ldots$
...	...
w_m	$f_2, f_4, f_6 \ldots$

Fig. 3. An inverted index.

5.3. Construction of MMDS

5.3.1. System setup

In the system setup phase, our scheme generates the working environment. We first choose two multiplicative cyclic groups of meromorphic order p, G_1, G_2. g is the generating element of G_1. Define the bilinear mapping $G_1 \times G_1 \rightarrow G_2$. Choose two one-way conflict-proof hash functions: H_1: $\{0,1\}^* \rightarrow Z_p^*$ and H_2: $\{0,1\}^* \rightarrow \{0,1\}^{\lambda}$. In addition, the system generates four large prime numbers and $p_1, p_2, p_3, p_4 \in Z_p^*$, and $p_1 \times p_2 = p$, $p_3 \times p_4 = p$. Finally the system issues common parameter $CP = \{e, g, p, G_1, G_2, H_1, H_2\}$ and distributes secret parameter $SP = \{p_1, p_2, p_3, p_4\}$ to ESs.

We assume that before new ESs and data users join the system, strict system authorization needs to be imposed. Once the authorization is passed, ESs obtains the SP, and data users received the file symmetric key K via a secure communication channel.

5.3.2. Encrypt

For the purpose of reducing the computing costs of IoT devices, data owners will send data files to their near ESs via security channels, and abstract keywords send to their near ESs.

An ES receives a data owner file set F and a keyword set W within its coverage area. To enable data users to securely and efficiently obtain the data files they need, ESs first construct searchable and secure indexes. For the purpose of improving search efficiency, an inverted index structure is used in the scheme. An inverted index is the mapping of a set of data files (including keywords) that store keywords. As shown in Fig. 3, an inverted index, where the first row indicates that the files containing the keyword w_1 are f_1, f_2, f_4, f_8. ESs uses the following way to encrypt the keyword set W of a data owner, as shown below: $CI_{w_i} = g^{H_1(w_i)+p_1 \cdot k_c}$, where $1 \le i \le |W|$. H_1 is a hash function in the system public parameters. p_1 is an important parameter shared between the system and ESs. $k_c \in Z_p^*$ is a random key for the encryption indexes. ESs uses different random keys every time for data owners. It

can increase the security of each data owner's privacy precisely. ESs uses the different random key for data owners, ensuring protect the data security of each data owner independently.

To protect the privacy of the files, the ES uses the following formula to encrypt each file $f_i \in F$: $C_i = \mathrm{Enc}(f_i, K)$, where $1 \le \mathrm{i} \le |F|$. $Enc(,)$ is a systematic encryption method (e.g., AES, DES). K is a secret Symmetric key.

To curb the untrusted cloud server and achieve fine-grained verification, the ES uses the following formula to calculate the verification information for each ciphertext $c_i \in C$: $v_i = H_2(c_i)$, where $1 \le \mathrm{i} \le |F|$. H_2 is a hash function in the system public parameter.

In order to efficiently search for verification information, the ES also builds searchable encrypted indexes for the verification information. Each keyword $w_i \in W$ is encrypted by the following method, as shown below: $VI_{w_i} = g^{H_1(w_i)+p_3 \cdot k_v}$, where $1 \le \mathrm{i} \le |W|$. H_1 is a hash function in the system public parameters. p_3 is an important parameter shared between the system and ESs. $k_v \in Z_p^*$ is a random key for the encryption indexes. Similar to the encrypted indexes used in the construction of CS. ESs uses a different random key. It independently safeguards the privacy of each data owner and strengthens privacy security. It is worth noting that since different keys are used in the scheme, CS and BC correspond to different ciphertexts, even if they store the same keyword.

Finally, the ES stores the ciphertext set C and its encrypted indexes CI in CS to enjoy its powerful computing and storage capabilities. To prevent malicious CS from deleting or tampering with the verification information, the ES records the verification information V and its encrypted index VI on the BC.

5.3.3. Generate trapdoor

To enable efficient, low-overhead search for data users, data users first log on to their nearby ES and send the search keywords over a secure channel to their nearby ES. To protect the search keywords, ESs encrypts the keywords to generate trapdoors before submitting the search request to the CS.

In order to allow ESs to securely generate trapdoors, keyword encryption should satisfy two main conditions. First, ESs can generate a different trapdoor for the same keyword each time. Second, ESs do not need to ask other ESs to generate keys to construct trapdoors. That is, ESs are able to generate trapdoors independently. ESs encrypt searched keyword w is as follows: $T_w = (g^{r_1 \cdot H_1(w)/r_2}, g^{r_1 \cdot r_2 \cdot p_2 \cdot p_4}, g^{{r_1}^2 \cdot p_2 \cdot p_4})$, where $r_1, r_2 \in Z_p^*$ are two variable and random numbers. Since the value of r_1, r_2 are variable, the security and the randomness of trapdoors have significant improvement.

Finally, ESs sends it to CS and broadcasts it to the full BC.

5.3.4. Search

The search process is conducted in two steps.

In the first step, after receiving the trapdoor $T_w = (T_1, T_2, T_3)$ from an ES, CS searches encrypted indexes CI in turn. For each sub-index ci_{w_i}, CS will test for a match between T_w and ci_{w_i} by using the following equation:

$$
\begin{aligned}
&e\ (ci_{w_i}, T_3)\\
=&e\ (g^{H_1(w_i)+p_1 \cdot k_c}, g^{{r_1}^2 \cdot p_2 \cdot p_4})\\
=&e\ (g^{H_1(w_i)}, g^{{r_1}^2 \cdot p_2 \cdot p_4})\ e\ (g^{p_1 \cdot k_c}, g^{{r_1}^2 \cdot p_2 \cdot p_4})\\
=&e\ (g^{H_1(w_i)}, g^{{r_1}^2 \cdot p_2 \cdot p_4})\ e\ (g,g)^{{r_1}^2 \cdot k_c \cdot p_1 \cdot p_2 \cdot p_4}\\
=&e\ (g^{H_1(w_i)}, g^{{r_1}^2 \cdot p_2 \cdot p_4})\ e\ (g,g)^{{r_1}^2 \cdot k_c \cdot p \cdot p_4}\\
=&e\ (g^{H_1(w_i)}, g^{{r_1}^2 \cdot p_2 \cdot p_4}) \times 1\\
=&e\ (g^{H_1(w_i)}, g^{{r_1}^2 \cdot p_2 \cdot p_4}),\\
&e\ (T_1, T_2)\\
=&e\ (g^{r_1 \cdot H_1(w)/r_2}, g^{r_1 \cdot r_2 \cdot p_2 \cdot p_4})\\
=&e\ (g^{H_1(w)}, g^{{r_1}^2 \cdot p_2 \cdot p_4}).
\end{aligned}
$$

If the above equation holds, it shows the ciphertext corresponding to the inverted index ci_{w_i} is the correct search result. Because $e(g,g)$ is a group element in G_2 with the order $p = p_1 \times p_2$. Therefore $(g,g)^{{r_1}^2 \cdot k_c \cdot p_1 \cdot p_2 \cdot p_4} = e\ (g,g)^{{r_1}^2 \cdot k_c \cdot p \cdot p_4} = e\ (g,g)^{({r_1}^2 \cdot k_c \cdot p \cdot p_4) mod p} = e\ (g,g)^0 = 1$. Obviously, $e\ (ci_{w_i}, T_3) = e\ (T_1, T_2)$ holds if $w = w_i$.

In the second step, after the BC receives the transaction T_w from a data user, the nodes of the BC, motivated by the consensus mechanism (e.g., POW, POS, etc.), retrieve the encrypted indexes $vi_{w_i} \in VI$ recorded in the blockchain in turn. The blockchain will use the following equation to test whether T_w and vi_{w_i} match:

$$
\begin{aligned}
&e\ (vi_{w_i}, T_3)\\
=&e\ (g^{H_1(w_i)+p_3 \cdot k_v}, g^{{r_1}^2 \cdot p_2 \cdot p_4})\\
=&e\ (g^{H_1(w_i)}, g^{{r_1}^2 \cdot p_2 \cdot p_4})\ e\ (g^{p_3 \cdot k_v}, g^{{r_1}^2 \cdot p_2 \cdot p_4})\\
=&e\ (g^{H_1(w_i)}, g^{{r_1}^2 \cdot p_2 \cdot p_4})\ e\ (g,g)^{{r_1}^2 \cdot k_v \cdot p_2 \cdot p_3 \cdot p_4}\\
=&e\ (g^{H_1(w_i)}, g^{{r_1}^2 \cdot p_2 \cdot p_4})\ e\ (g,g)^{{r_1}^2 \cdot k_v \cdot p \cdot p_2}\\
=&e\ (g^{H_1(w_i)}, g^{{r_1}^2 \cdot p_2 \cdot p_4}) \times 1\\
=&e\ (g^{H_1(w_i)}, g^{{r_1}^2 \cdot p_2 \cdot p_4}),\\
&e\ (T_1, T_2)\\
=&e\ (g^{r_1 \cdot H_1(w)/r_2}, g^{r_1 \cdot r_2 \cdot p_2 \cdot p_4})\\
=&e\ (g^{H_1(w)}, g^{{r_1}^2 \cdot p_2 \cdot p_4}).
\end{aligned}
$$

If the above equation holds, the consensus mechanism of the BC will return the corresponding verification information to the ESs. Obviously $e\ (vi_{w_i}, T_3) = e\ (T_1, T_2)$ holds if $w = w_i$.

5.3.5. Verify

Algorithm 1 Verify

Input: CR_w, VR_w
Output: R
for $i \leftarrow 1$ **to** $\{CR_w\}$ **do**
  Compute $H_2(cr_i)$;
end
if $\prod_{i=1}^{i=|CR_w|} H_2(cr_i) == \prod_{j=1}^{j=|VR_w|} vr_i$ **then**
  Return true;
else
  for $i \leftarrow 1$ **to** $\{VR_w\}$ **do**
    if $vr_i == H_2(cr_i)$ **then**
      Return i;
    else
      Return vr_i;
    end
  end
end

After the search results are received from the CS, in order to prevent the untrusted cloud server returned incorrect or incomplete search results, the ES implements a verification algorithm based on the verification information returned by the BC. The ES takes the parameters (CR_w, VR_w) as input to verify the validity of the search results. This is shown in Algorithm 1. It first verifies the overall correctness and completeness of the search results. It first gets the product of the hash values of each ciphertext. If it is equal to the value of the verification message returned by BC, then the search result is correct and complete. Otherwise, it further finds out the correct ciphertext in the search result. It will match the ciphertext with the same hash value as the verification information, i.e., the correct ciphertext. Finally, for unmatched ciphertexts, it returns the corresponding file flags $\{vr_i\}$.

5.3.6. Decrypt

The data users send the file symmetric key K to the ES via a secure channel. If R = true, the ES decrypts all search results and sends them to the data users. Otherwise, the ES decrypts the ciphertext of file number {i} based on the verification result R. The ES sends it to the data users together with $\{vr_i\}$. In the following progress, the data users are able to interact with the data owners or the cloud server to find the missing files based on $\{vr_i\}$, eventually to get the correct and complete search results.

6. Security proof and analysis

We first formally prove the semantic security of MMDS satisfying CPA by challenger and adversary game. Challenger and adversary game is a common model for security proofs. The model evaluates the security of the scheme by simulating a challenger's attack on the encryption scheme. Then, we provide a comprehensive security analysis for MMDS.

6.1. Security proof

In the encryption phase of MMDS, we construct two encryption indexes, the encryption index of the ciphertext and the encryption index of the verification information. Their basic encryption blocks can be expressed as $E(m) = g^{m+qr}$, of which r is the random key, q represent the p_1 or p_3 of secret parameter. Before proving that the encryption index in MMDS satisfies semantical security against CPA. We define a polynomial challenger X has a non-negligible advantage, namely the ability to break the E with the traditional challenger and adversary game, as we will demonstrate.

Setup: The challenger setup the algorithm through the system. The opponent has access to the common parameters announced by the challenger.

Phase 1: The adversary adaptively accesses oracle E multiple times, then outputs never accessed for oracle E with two equal length messages m_0, m_1.

Challenge: The challenger sends $E(m_b)$ to the opponent, where $b \in \{0,1\}$ is chosen randomly by the challenger.

Phase 2: Same as Phase 1. The adversary enters any message other than m_0 and m_1 and continues to access oracle E.

Guess: The opponent makes a guess on m_0 or m_1, and outputs the result b'. If $b' = b$, it means that the opponent wins. The opponent's advantage is:

$Adv^{CPA}_{Adversary} = |Pr[b' = b]\text{-}\frac{1}{2}|$.

Definition 1. If the adversary X without any polynomial has a non-negligible advantage. Then the encryption E of MMDS satisfies the indistinguishable security under CPA.

Theorem 1. If the DDHP assumption holds, the encryption E of satisfies indistinguishable security under CPA in random oracle model.

Proof: If the polynomial adversary X can win the game by breaking oracle E with a non-negligible advantage ε, then we can construct a simulator S, S can solve the DDHP with a non-negligible advantage.

The challenger Z first flips a binary coin μ, if $\mu = 0$, Z sets $t_0 = (g, A = g^a, B = g^b, C = g^{ab})$. If $\mu = 1$, Z sets $t_1 = (g, A = g^a, B = g^b, C=g^c)$, where a, b, c are chosen from Z_q^* at random uniformly. Z will sent t_μ to S, and S plays the role of challenger Z in the following game.

Setup: S sends common parameters $\{g, p\}$ to X.

Phase 1: X accesses oracle E several times by using any information in Z_q^*, and asks for the corresponding ciphertext each time. Finally, he outputs two equal-length and unaccessed oracle E informations m_0 and m_1, and sends them to S.

Challenge: S tosses the coin $b \in \{0,1\}$, and encrypts message m_b as $E' = g^{m_b} \times C$. If $\mu = 0$, $C = g^{ab}$, We let $ab = qr$, $E' = g^{m_b} \times C = g^{ab} \times C = g^{m_b+qr}$. Since r is a randomly chosen element of oracle E, qr is also a random element. Therefore, E' is a valid encryption for oracle E. If $\mu = 1$, $C = g^c$. Then we have $E' = g^{m_b+c}$, Since c is a random element, E' is a random element in G_1 from the point of view of X and contains no message about m_b.

Phase 2: X continues to access any message in oracle E other than m_0 and m_1.

Guess: X outputs a guess b' of b. Then S outputs a guess $\mu' = 0$ of μ. This means that Z sends a valid encrypted tuple $t_0 = (g, A = g^a, B = g^b, C = g^{ab})$ to S. Since X has the advantage of breaking E, X outputs a guess b' of b that satisfies $b' = b$ with probability $\frac{1}{2}+\varepsilon$. Correspondingly, X outputs a guess $\mu' = \mu = 0$ with probability $\frac{1}{2}+\varepsilon$. If $b' \neq b$, then Z outputs a guess $\mu' = 1$ of μ. This means that a random tuple t_1 is sent to S. Thus, X outputs a guess $b' \neq b$ with probability $\frac{1}{2}$. Correspondingly, S outputs a guess μ' that satisfies $\mu' = \mu = 1$ with probability $\frac{1}{2}$.

Thus, the overall advantage of S solving the DDHP is:

Adv^{CPA}_S

$=|\frac{1}{2}Pr[\mu = \mu'|\mu = 0] + \frac{1}{2}Pr[\mu = \mu'|\mu = 1] - \frac{1}{2}|$

$=|]\frac{1}{2}(\frac{1}{2}+\varepsilon) + \frac{1}{2} \times \frac{1}{2}] - \frac{1}{2}|$

$=\frac{\varepsilon}{2}$.

Clearly, if X can break E with a non-negligible advantage ε, the S able to solve the DDHP with the non-negligible advantage $\frac{\varepsilon}{2}$. All of these contradict the DDHP assumption.

6.2. Security analysis

1. Security of data files: Based on the security of symmetric encryption (e.g., AES, DES), data files also possess indistinguishability. As long as the symmetric key is not leaked, the cloud server cannot obtain the information in the data file from the ciphertext.

2. Security of trapdoor: If the DLP assumption holds, it is impossible to recover the search keyword w from T_w. At the same time, ESs introduce random numbers r_1, and r_2 each time when a trapdoor is constructed, ensuring the unlinkability of the trapdoor.

3. Security of encrypted indexes: In Section 6.1, we demonstrate that our encrypted indexes satisfy the indistinguishable security in the random oracle model under CPA based on the DDHP. Our encrypted indexes are more compatible with the security requirements of keyword search in cloud-assisted edge environments. ESs uses different keys to build encrypted indexes for different IoT devices each time. Even if an edge server accidentally leaks an encryption key, it will not endanger the data security of other IoT devices. It independently protects the privacy of each IoT device. Secondly, each encryption index is built using a different random key, even if the same keyword generates a different encryption index, ensuring the flexibility and security of the system.

4. Security of verification information: In MMDS, each verification information is encrypted by a hash function. The unidirectional hash function ensures the security of the verification information. The security is demonstrated in [40]. In addition, the tamper-evident nature of the BC and the consensus mechanism [36–39] ensures the authenticity of the verification information.

7. Performance analysis and evaluations

In this section, in order to simulate a real IoT environment, we chose time cost and storage costs as the evaluation indicators to compare the performance of the algorithms. We implement all the necessary processes such as constructing encryption indexes, generating trapdoors, searching, generating verification information, and verifying. In the experimental tests, for each scheme, we averaged the results of multiple experiments to ensure accuracy.

We compare some representative schemes with similar research directions as in this paper, such as CECS [11] and QRVS [31]. CECS achieves secure and flexible data sharing for cloud–edge collaborative storage. It uses traditional PEKS to avoid the burden of key management. It uses each data user's public key to construct an encrypted index. But a large number of data users would place a huge burden on CECS. It still requires the IoT device itself to generate a trapdoor. It is not friendly to resource-constrained IoT devices. Our scheme cleverly uses the nature of bilinear pairings to design the structure of the encrypted index, effectively avoiding these problems. CECS, like most other schemes, does not implement fine-grained verification of search results. QRVS implements fine-grained verification in the cloud environment with the Counting Bloom Filter. The Counting bloom filter take up a lot of space to ensure their accuracy, and complex deletion operations can be computationally burdensome. Our scheme uses the small size and speed of hash functions to improve efficiency while ensuring security and accuracy.

Table 2
Summary of theoretical analysis.

Schemes	CECS [11]		QRVS [31]		MMDS	
	Computation costs	Storage costs	Computation costs	Storage costs	Computation costs	Storage costs
Encryption index	d(2E+P+H_1+H_2)	$d(\|Z_p^*\|+\|G_1\|)$	/	/	E+H_1	$\|G_1\|$
Trapdoor	E+H_1	$\|G_1\|$	/	/	3E+H_1	3$\|G_1\|$
Search	dm(P+H_2)	/	/	/	2mP	/
Verification information	nH_2	$n\|H_2\|$	$m\{f_{w_{max}}(H_2+lH_3)\}$	$m\|B\|$	m(E+H_1)+nH_2	$m\|G_1\|+n\|H_2\|$
Verify	tH_2	/	$t(H_2+lH_3)$	/	tH_2	/

Notes. d: Number of data users; m: Number of keywords; n: Number of files; t: Number of search results; $|B|$: length of a counting bloom filter B; l: number of hash function in a counting bloom filter; /: Without consideration.

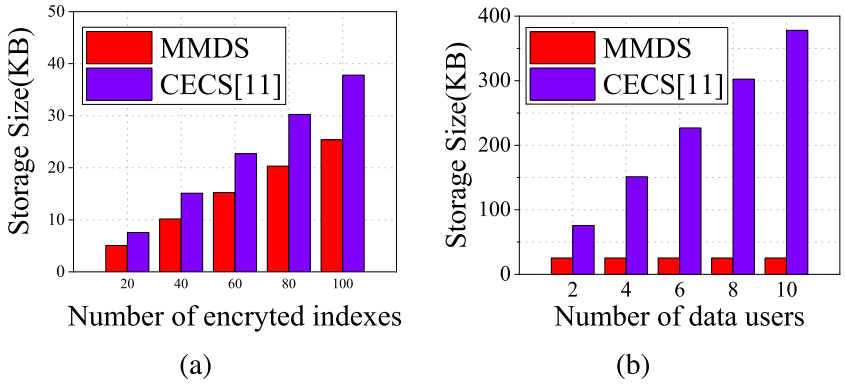

Fig. 4. Storage costs of encrypted indexes.

7.1. Theoretical analysis

Table 2 presents an analysis of the theoretical overheads around the computational and storage costs of the above schemes. Before the analysis, we introduced some time-consuming operations. For example, E: modular exponentiation operation in group G; H_1: a hash operation which maps any string with arbitrary length into a group element in Z_q^*; H_2: a hash operation which maps any string with arbitrary length into a group element in 256 bits (e.g., Sha-256); H_3: a hash operation which maps any string with arbitrary length into a group element in counting bloom filters.

The outcomes in Table 2 reveal the MMDS performance in constructing encrypted indexes, searching, generating verification information, and verifying is significantly better than the other schemes. Compared to CECS, the computational and storage overhead of MMDS does not increase as the increase of data users in the system. A professional, powerful system may include hundreds or thousands of data users, and MMDS is far more efficient and responsive. MMDS has a slightly higher computational and storage overhead for trapdoor than CECS. In MMDS, ESs bear the cost of generating trapdoors for data users in the coverage area. It is acceptable for ESs with some computational and storage capacity. However, in CECS data users bear the cost of trapdoor generation, which is unaffordable for most resource-constrained IoT devices. MMDS has a slightly higher computational cost for verification information than CECS, but CECS is only able to perform verification of the correctness of search results. Compared to MMDS, QRVS introduces the Counting Bloom Filter, bringing with it significant computational and storage costs and false positives with a certain. MMDS performs fine-grained verification of search results for data users via ESs, easing the burden on resource-constrained data users.

7.2. Experiment setup

We build a subset of our experiment by selecting 2000 random files on the real Request For Comments Database data set and then extracting 200 keywords from them. The maximum number of files contained in a keyword is 600. Our code uses Java Pairing-Based Cryptography (JPBC) library.

Edge servers are Windows 10 desktop systems with an i7-7700HQ and 8 GB RAM, which are responsible for the execution of encrypted indexes, trapdoors, and the generation of verification information. The cloud server is also Windows 10 desktop system with an i7-7700 CPU and 8 GB RAM, which is responsible for performing search operations.

7.3. Storage costs

We invoke the API getLengthInBytes() to acquire the size. $|z_q^*| = 128$ bytes, $|G_1| = |G_2| = 260$ bytes of group elements in elliptic curves of type A1. Fig. 4(a) shows that the storage size of MMDS and CECS is linearly related to the number of encrypted indexes. When the number of encrypted indexes is 100 for a single data user, the storage size of their indexes is 25.39 KB, and 37.81 KB respectively. Fig. 4(b) shows that when m = 10, the index storage size varies with the number of data users d. We can observe that the index storage size in CECS increases as d increases, whereas the storage size in MMDS is not affected by d. This is because the number of encrypted indexes in CECS grows exponentially in line with the increasing quantity of data users. This denotes that the more data users there are, the more obvious the advantage of MMDS.

According to QRVS, when we set up the number of hash functions at 7, the size of a counting bloom filter is set to approximately 4 KB. Fig. 5(a) shows the cost of storing verification information for different numbers of data files, and it can be observed that the storage overhead of MMDS increases linearly when the amount of files changes from 100 to 500 for m = 5, while the amount of data files on the verification of QRVS information has little effect on the storage overhead of QRVS, but the storage overhead of QRVS is much higher than that of MMDS. Fig. 5(b) shows that given 1000 data files, the storage overhead of MMDS grows independently of the keyword number, as the number of keywords changes from 10 to 50, while the storage costs of QRVS grow linearly. This is because CECS constructs a counting bloom filter for

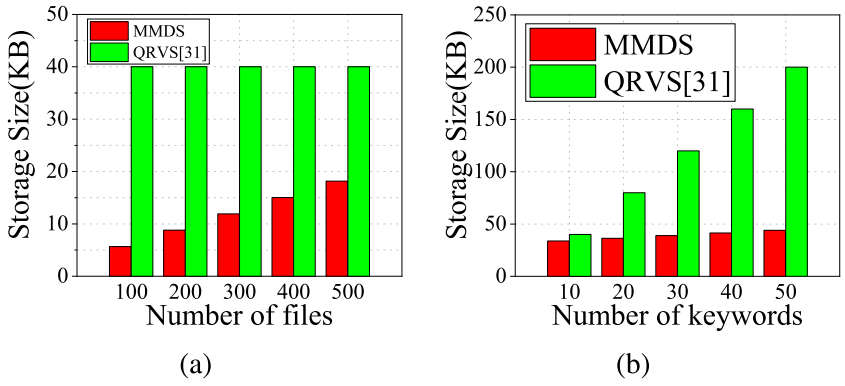

Fig. 5. Storage costs of verification information.

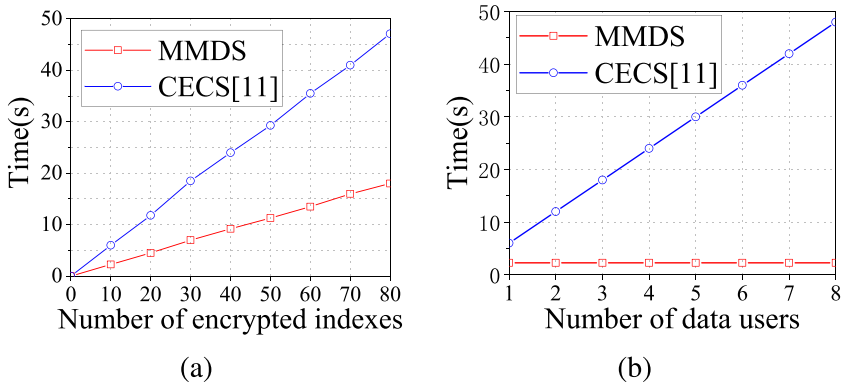

Fig. 6. Time cost of constructing encryption indexes.

each keyword and therefore imposes heavy storage overhead. MMDS only needs to store a small number of encrypted indexes and hashes. It shows that MMDS has a greater advantage when faced with large-scale data with a large number of keywords.

7.4. Time cost

In Fig. 6(a) and 6(b), we take into account both the quantity of encrypted indexes and those of data users. In Fig. 6(a), it is known that the linear increase in time overhead is accompanied by an increase in the number of encrypted indexes for a single data user. The time overhead of CECS is higher than that of MMDS because CECS requires more power and pairing operations. In Fig. 6(b), when the number of keywords is set to m = 5, the time cost of CECS increases linearly for an increasing number of data users d, while MMDS is not affected by d. This is because CECS data owners are required to build an encrypted index for each data user using their public key, resulting in a large time overhead, whereas the data owner of MMDS needs only one encrypted index. A system should have a large number of data users, which means that MMDS is more practical in real-life situations.

Fig. 7 shows as the number of keywords varies, the time cost of generating trapdoors. As the number of trapdoors changes from 1 to 8, the time overhead for MMDS and CECS construction increases linearly, and it is obvious that MMDS requires more time overhead than CECS. This is because MMDS requires one additional pairing when constructing a trapdoor. However, data users in CECS need to generate trapdoors on IoT devices using private keys. MMDS generates trapdoors for data users by ESs, and the overhead incurred is borne by the ES. It means MMDS is more friendly and attractive to data users, especially to resource-constrained data users.

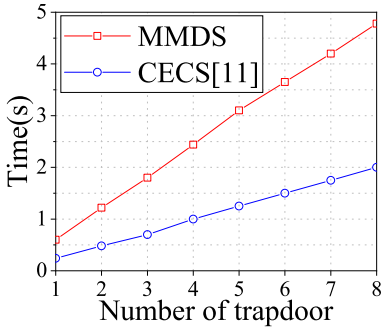

Fig. 7. Time cost of generating trapdoors.

In Fig. 8(a) and 8(b), the two factors we need to consider, include the impact on search performance of the number of encrypted indexes and the number of data users. Fig. 8(a) shows that the time overhead of MMDS and CECS increases linearly with the number of encrypted indexes for a single data user. The time overhead of MMDS is slightly higher than that of CECS. This is because MMDS requires one more pairing. And in Fig. 8(b), we set the number of keywords to 10 and it shows the change in time overhead with the increase in the number of data users. We can see that the time overhead of CECS increases linearly with the increase of data users, while MMDS is hardly affected. This is because the increase in the number of data users leads to an increase in the number of encrypted indexes, and thus requires more pairings. This further suggests that MMDS is the more efficient and practical scheme.

Fig. 9(a) shows the time cost of verification information when the number of data files varies. It can be observed that MMDS time cost

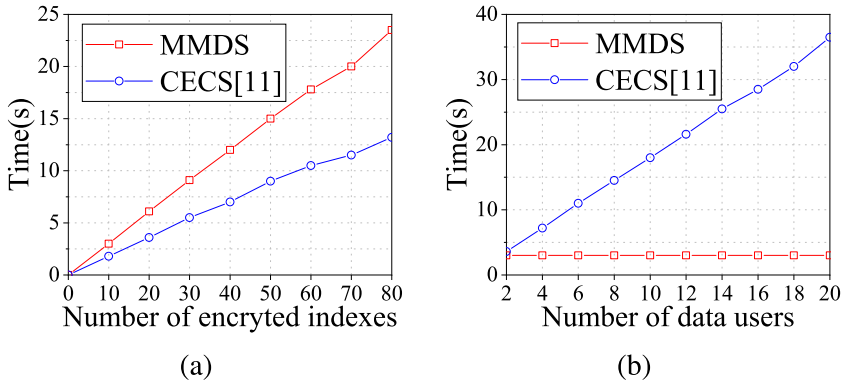

Fig. 8. Time cost of searching.

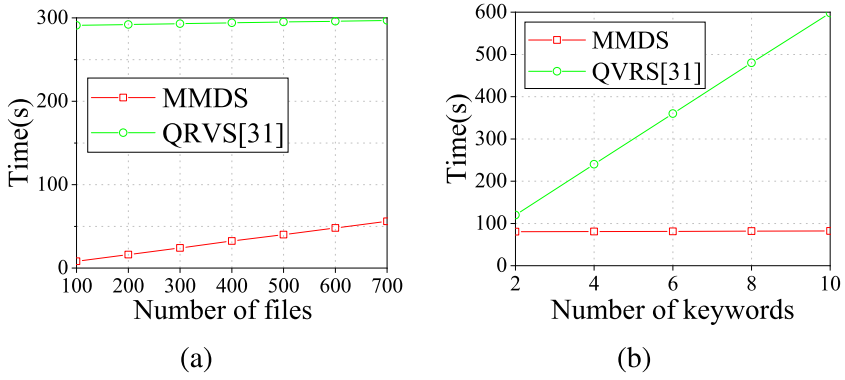

Fig. 9. Time cost of generating verification information.

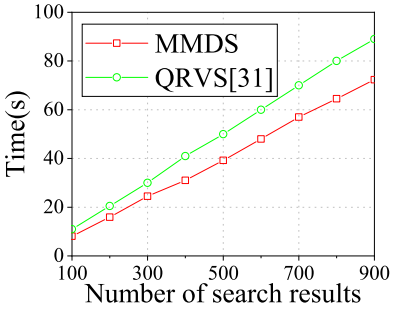

Fig. 10. Time cost of verifying.

can be seen to increase linearly at m = 5 when the number of files is increased from 100 to 700. While the number of data files has almost no effect on QRVS. This is due to the fact that in our scheme, we require a hash operation for each data file. But the time overhead of QRVS is much. Fig. 9(b) shows that with 1000 data files, MMDS time cost has almost no effect when the number of keywords changes from 2 to 10. While the time overhead of QRVS grows linearly. This is because, in QRVS, a counting bloom filter is generated for each keyword, regardless of the number of data files, even if some keywords include only a small number of files. And each counting bloom filter has an almost constant time overhead. That means that MMDS is more advantageous in large data sets.

The content of Fig. 10 shows the time cost of search result verification. Both MMDS and QRVS time overheads increase linearly for the change in the number of data files from 100 to 900. QRVS has a higher time overhead than MMDS. MMDS only needs one hash operation per file. QRVS requires finishing multiple hash functions, and need to delete element in the Counting Bloom Filter, incurring additional overhead.

7.5. Engineering applications

With the application of technologies such as 5G and IoT bringing massive growth in data, in order to enjoy the vast benefits of cloud-assisted edge computing, multimedia devices in an increasing number outsource their MMBD to a cloud server via edge servers. Intelligent multimedia systems are already used in large numbers of industrial applications such as healthcare, vehicles, and education. An electronic library is a typical application. As shown in Fig. 11, considering the following example: library staff collect ebooks in various locations and they use IoT devices to store the ebooks on the cloud server via nearby edge servers. Readers are also able to use IoT devices to access the ebooks they want via the edge servers. However the untrusted cloud server can control or maliciously leak these ebooks. Currently, most research into this problem involves encrypting ebooks before they are uploaded to the cloud server. In this case, the implementation of searching for encrypted ebooks in the cloud server is one of the key issues for privacy protection.

To protect the library's privacy, edge servers need to encrypt ebooks before they can be uploaded cloud server, and at the same time build searchable encrypted indexes. However, staff members and readers are on the move. The edge servers near them are uncertain. If all edge servers use the same indexing key, this will reduce the security of the system. If the edge servers use different index keys, when a user wants to search for an ebook, the edge server will need to obtain authorization from other edge servers and build multiple trapdoors to be able to perform a useful search. This would result in significant communication overhead and security risks. In this case, the data privacy of the library may be compromised. We can use the proposed MMDS scheme to protect the library's privacy for a cloud-assisted edge computing environment, and at the same time accomplish a secure and efficient search. Specifically, after a staff member of the library collects ebooks in various locations, he/she can upload them to nearby edge servers

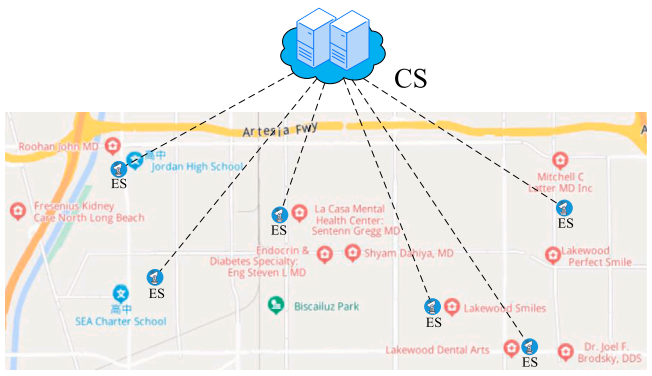

Fig. 11. SE example of electronic library.

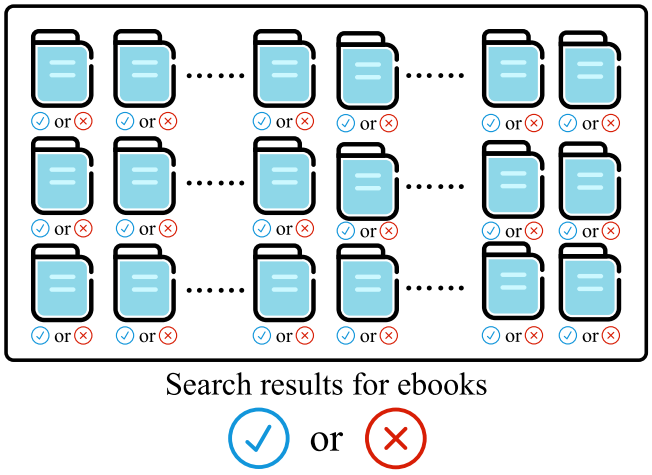

Fig. 12. Application diagram of MMDS scheme in electronic library.

(e.g., wireless sensor networks, etc.) via IoT devices (e.g., smartphones, tablets, laptops, etc.). The edge server in the area encrypts the ebooks and stores them on to cloud server according to keywords such as title and genre. At the same time, to prevent the malicious cloud server from deleting or tampering with the search results, the edge server uses a hash function to calculate the verification information stored on the blockchain.

When a reader wants to search for "secure" related ebooks, he/she can use his/her smartphone to send the search keyword: "secure" via nearby edge servers. The edge server encrypts the keyword to generate a trapdoor to the cloud server and blockchain at the same time. Cloud server and blockchain respectively respond to the search request, returning the corresponding encrypted ebook and verification information to the edge server. For the purpose of ensuring the usefulness of the search results, as shown in Fig. 12, the edge server first verifies that all qualified, correct encrypted ebooks are returned; otherwise, it further verifies the correctness of each encrypted ebook. Finally, the edge servers decrypt the valid ebook ciphertext and return the ebook to the user's smartphone. With the MMDS scheme, electronic libraries can efficiently complete the storage and search of ebooks while protecting data privacy. As described above, our next work will focus on the implementation of the MMDS scheme, which is proposed in other areas, and applied to more complex engineering problems that can help people enjoy a more convenient life.

8. Conclusion and the future work

In this paper, we propose a secure and verifiable multimedia data search scheme for cloud-assisted edge computing. It exploits the nature of bilinear pairings to achieve keyword search on encrypted data. It utilizes blockchain and hashing techniques to achieve fine-grained verification of search results. It reduces the burden on IoT devices by introducing edge servers. In terms of function, it supports more secure, flexible, and lightweight keyword search. It also supports more efficient fine-grained verification. The results of the security proof and analysis show that it is secure and effective. We have also evaluated its performance through several simulation experiments, indicating its efficiency and practicality. It can also be applied to solve practical problems in intelligent multimedia systems, such as electronic libraries. We will explore multi-keyword search and keyword sorting search schemes for cloud-assisted edge computing in future work.

CRediT authorship contribution statement

Shiwen Zhang: Conceptualization, Methodology, Data acquisition and processing, Analysis, Writing – original draft, Writing – review & editing, Project administration, Funding acquisition. **Jiayi He:** Conceptualization, Methodology, Data acquisition and processing, Analysis, Writing – original draft, Writing – review & editing. **Wei Liang:** Conceptualization, Analysis, Project administration, Writing – review & editing. **Keqin Li:** Conceptualization, Project administration, Writing – review & editing.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Shiwen Zhang reports financial support was provided by National Natural Science Foundation of China. Shiwen Zhang reports financial support was provided by Research Foundation of Education Bureau of Hunan Province.

Data availability

Data will be made available on request.

Acknowledgments

This work is supported in part by the Open Fund of Advanced Cryptography and System Security Key Laboratory of Sichuan Province under Grant SKLACSS-202206, the Natural Science Foundation of Hunan Province under Grant No. 2022JJ30267, the Natural Science Foundation of Fujian Province under Grant No. 2022J05106, and the Scientific Research Fund of Hunan Provincial Education Department under Grant No. 21B0493.

References

[1] Aparna Kumari, Sudeep Tanwar, Sudhanshu Tyagi, Neeraj Kumar, Michele Maasberg, Kim-Kwang Raymond Choo, Multimedia big data computing and internet of things applications: A taxonomy and process model, J. Netw. Comput. Appl. 124 (2018) 169–195.

[2] Pengjie Zeng, Anfeng Liu, Chunsheng Zhu, Tian Wang, Shaobo Zhang, Trust-based multi-agent imitation learning for green edge computing in smart cities, IEEE Trans. Green Commun. Netw. 6 (3) (2022) 1635–1648.

[3] Shaobo Huang, Zhiwen Zeng, Kaoru Ota, Mianxiong Dong, Tian Wang, Neal N. Xiong, An intelligent collaboration trust interconnections system for mobile information control in ubiquitous 5G networks, IEEE Trans. Netw. Sci. Eng. 8 (1) (2020) 347–365.

[4] Miaojiang Chen, Wei Liu, Tian Wang, Shaobo Zhang, Anfeng Liu, A game-based deep reinforcement learning approach for energy-efficient computation in MEC systems, Knowl.-Based Syst. 235 (2022) 107660.

[5] Muhammad Baqer Mollah, Md Abul Kalam Azad, Athanasios Vasilakos, Secure data sharing and searching at the edge of cloud-assisted internet of things, IEEE Cloud Comput. 4 (1) (2017) 34–42.

[6] Wei Wang, Peng Xu, Dongli Liu, Laurence Tianruo Yang, Zheng Yan, Lightweighted secure searching over public-key ciphertexts for edge-cloud-assisted industrial IoT devices, IEEE Trans. Ind. Inform. 16 (6) (2019) 4221–4230.

[7] Jiayi Li, Jianfeng Ma, Yinbin Miao, Lei Chen, Yunbo Wang, Ximeng Liu, Kim-Kwang Raymond Choo, Verifiable semantic-aware ranked keyword search in cloud-assisted edge computing, IEEE Trans. Serv. Comput. 15 (6) (2021) 3591–3605.
[8] Jawhara Aljabri, Anna Lito Michala, Jeremy Singer, ELSA: A keyword-based searchable encryption for cloud-edge assisted industrial internet of things, in: 2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing, CCGrid, IEEE, 2022, pp. 259–268.
[9] Ke Zhang, Jiahuan Long, Xiaofen Wang, Hong-Ning Dai, Kaitai Liang, Muhammad Imran, Lightweight searchable encryption protocol for industrial internet of things, IEEE Trans. Ind. Inform. 17 (6) (2020) 4248–4259.
[10] Jingwei Liu, Yating Li, Rong Sun, Qingqi Pei, Ning Zhang, Mianxiong Dong, Victor C.M. Leung, EMK-ABSE: Efficient multikeyword attribute-based searchable encryption scheme through cloud-edge coordination, IEEE Internet Things J. 9 (19) (2022) 18650–18662.
[11] Ye Tao, Peng Xu, Hai Jin, Secure data sharing and search for cloud-edge-collaborative storage, IEEE Access 8 (2019) 15963–15972.
[12] Huige Li, Haibo Tian, Fangguo Zhang, Jiejie He, Blockchain-based searchable symmetric encryption scheme, Comput. Electr. Eng. 73 (2019) 32–45.
[13] Jiaxing Li, Jigang Wu, Guiyuan Jiang, Thambipillai Srikanthan, Blockchain-based public auditing for big data in cloud storage, Inf. Process. Manage. 57 (6) (2020) 102382.
[14] Quanyu Zhao, Siyi Chen, Zheli Liu, Thar Baker, Yuan Zhang, Blockchain-based privacy-preserving remote data integrity checking scheme for IoT information systems, Inf. Process. Manage. 57 (6) (2020) 102355.
[15] Chengjun Cai, Jian Weng, Xingliang Yuan, Cong Wang, Enabling reliable keyword search in encrypted decentralized storage with fairness, IEEE Trans. Dependable Secure Comput. 18 (1) (2018) 131–144.
[16] Wenyuan Yang, Boyu Sun, Yuesheng Zhu, Dehao Wu, A secure heuristic semantic searching scheme with blockchain-based verification, Inf. Process. Manage. 58 (4) (2021) 102548.
[17] Dawn Xiaoding Song, David Wagner, Adrian Perrig, Practical techniques for searches on encrypted data, in: Proceeding 2000 IEEE Symposium on Security and Privacy, S&P 2000, IEEE, 2000, pp. 44–55.
[18] Eu-Jin Goh, Secure indexes, Cryptol. ePrint Arch. (2003).
[19] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, Public key encryption with keyword search, in: Advances in Cryptology-EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004. Proceedings 23, Springer, 2004, pp. 506–522.
[20] Reza Curtmola, Juan Garay, Seny Kamara, Rafail Ostrovsky, Searchable symmetric encryption: improved definitions and efficient constructions, in: Proceedings of the 13th ACM Conference on Computer and Communications Security, 2006, pp. 79–88.
[21] Wei Zhang, Yaping Lin, Sheng Xiao, Jie Wu, Siwang Zhou, Privacy preserving ranked multi-keyword search for multiple data owners in cloud computing, IEEE Trans. Comput. 65 (5) (2015) 1566–1577.
[22] Hui Yin, Zheng Qin, Jixin Zhang, Lu Ou, Fangmin Li, Keqin Li, Secure conjunctive multi-keyword ranked search over encrypted cloud data for multiple data owners, Future Gener. Comput. Syst. 100 (2019) 689–700.
[23] Qi Chai, Guang Gong, Verifiable symmetric searchable encryption for semi-honest-but-curious cloud servers, in: 2012 IEEE International Conference on Communications, ICC, IEEE, 2012, pp. 917–922.
[24] Jianfeng Wang, Xiaofeng Chen, Xinyi Huang, Ilsun You, Yang Xiang, Verifiable auditing for outsourced database in cloud computing, IEEE Trans. Comput. 64 (11) (2015) 3293–3303.
[25] Xueqiao Liu, Guomin Yang, Yi Mu, Robert H. Deng, Multi-user verifiable searchable symmetric encryption for cloud storage, IEEE Trans. Dependable Secure Comput. 17 (6) (2018) 1322–1332.
[26] Xiuxiu Jiang, Jia Yu, Jingbo Yan, Rong Hao, Enabling efficient and verifiable multi-keyword ranked search over encrypted cloud data, Inform. Sci. 403 (2017) 22–41.
[27] Wei Zhang, Yaping Lin, Gu Qi, Catch you if you misbehave: Ranked keyword search results verification in cloud computing, IEEE Trans. Cloud Comput. 6 (1) (2015) 74–86.
[28] Jie Zhu, Qi Li, Cong Wang, Xingliang Yuan, Qian Wang, Kui Ren, Enabling generic, verifiable, and secure data search in cloud services, IEEE Trans. Parallel Distrib. Syst. 29 (8) (2018) 1721–1735.
[29] Qin Liu, Yue Tian, Jie Wu, Tao Peng, Guojun Wang, Enabling verifiable and dynamic ranked search over outsourced data, IEEE Trans. Serv. Comput. 15 (1) (2019) 69–82.
[30] Qiuyun Tong, Yinbin Miao, Ximeng Liu, Kim-Kwang Raymond Choo, Robert H. Deng, Hongwei Li, VPSL: Verifiable privacy-preserving data search for cloud-assisted internet of things, IEEE Trans. Cloud Comput. 10 (4) (2020) 2964–2976.
[31] Hui Yin, Zheng Qin, Jixin Zhang, Lu Ou, Keqin Li, Achieving secure, universal, and fine-grained query results verification for secure search scheme over encrypted cloud data, IEEE Trans. Cloud Comput. 9 (1) (2017) 27–39.
[32] Shengshan Hu, Chengjun Cai, Qian Wang, Cong Wang, Xiangyang Luo, Kui Ren, Searching an encrypted cloud meets blockchain: A decentralized, reliable and fair realization, in: IEEE INFOCOM 2018-IEEE Conference on Computer Communications, IEEE, 2018, pp. 792–800.
[33] Lanxiang Chen, Wai-Kong Lee, Chin-Chen Chang, Kim-Kwang Raymond Choo, Nan Zhang, Blockchain based searchable encryption for electronic health record sharing, Future Gener. Comput. Syst. 95 (2019) 420–429.
[34] Loi Luu, Jason Teutsch, Raghav Kulkarni, Prateek Saxena, Demystifying incentives in the consensus computer, in: Proceedings of the 22Nd ACM Sigsac Conference on Computer and Communications Security, 2015, pp. 706–719.
[35] Jing Bai, Guosheng Huang, Shaobo Zhang, Zhiwen Zeng, Anfeng Liu, GA-DCTSP: An intelligent active data processing scheme for UAV-enabled edge computing, IEEE Internet Things J. 10 (6) (2022) 4891–4906.
[36] Zibin Zheng, Shaoan Xie, Hongning Dai, Xiangping Chen, Huaimin Wang, An overview of blockchain technology: Architecture, consensus, and future trends, in: 2017 IEEE International Congress on Big Data, BigData Congress, IEEE, 2017, pp. 557–564.
[37] Wenbo Wang, Dinh Thai Hoang, Peizhao Hu, Zehui Xiong, Dusit Niyato, Ping Wang, Yonggang Wen, Dong In Kim, A survey on consensus mechanisms and mining strategy management in blockchain networks, IEEE Access 7 (2019) 22328–22370.
[38] Yinghui Luo, Yiqun Chen, Qiang Chen, Qinglin Liang, A new election algorithm for dpos consensus mechanism in blockchain, in: 2018 7th International Conference on Digital Home, ICDH, IEEE, 2018, pp. 116–120.
[39] Tyler Crain, Vincent Gramoli, Mikel Larrea, Michel Raynal, DBFT: Efficient byzantine consensus with a weak coordinator and its application to consortium blockchains, 2017, arXiv preprint arXiv:1702.03068.
[40] Mihir Bellare, Ran Canetti, Hugo Krawczyk, Keying hash functions for message authentication, in: Advances in Cryptology—CRYPTO'96: 16th Annual International Cryptology Conference Santa Barbara, California, USA August 18–22, 1996 Proceedings 16, Springer, 1996, pp. 1–15.

Shiwen Zhang received his B.S. degree in Information and Computing Science from the University of Changsha, China, in 2010, and received his Ph.D. degree from the College of Computer Science and Electronic Engineering, Hunan University, in 2016. He is currently an associate professor at the School of Computer Science and Engineering, Hunan University of Science and Technology. He is a member of IEEE and CCF. His research interests include security and privacy issues in social networks, protection, and information security. Email: shiwenzhang@hnust.edu.cn.

Jiayi He received a B.S. degree from Hunan University of Arts and Science, and currently pursuing an M.S. degree from the school of Computer Science and Engineering at Hunan University of Science and Technology. His research interests include security and privacy issues in cloud computing and edge computing. Email: jiayihehnust@163.com.

Wei Liang is an Associate Professor at the College of Information Science and Engineering, Hunan University. He received his Ph.D. degree at Hunan University in 2013. He is a postdoctoral scholar at the Department of Computer Science and Engineering at Lehigh University in the USA in 2014–2016. His research interests include Networks Security Protection, embedded systems and Hardware/IP protection, Fog computing, and Security Management in WSN. Email: weiliang99@hnu.edu.cn.

Keqin Li (Fellow, IEEE) received the B.S. degree in computer science from Tsinghua University, Beijing, China, in 1985 and the Ph. D. degree in computer science from the University of Houston, Houston, Texas, USA, in 1990. He is currently a SUNY Distinguished Professor of Computer Science with the State University of New York, Albany, NY, USA. He is also a National Distinguished Professor with Hunan University, Changsha, China. He has authored or coauthored more than 850 journal articles, book chapters, and refereed conference papers. He holds over 70 patents announced or authorized by the Chinese National Intellectual Property Administration, Beijing, China. His current research interests include parallel and distributed computing, cloud computing security. He is currently an

Associate Editor for the ACM Computing Surveys and the CCF Transactions on High Performance Computing. He was on the Editorial Board of the IEEE TPDS, the IEEE ToC, the IEEE TCC, the IEEE TSC, and the IEEE TSUSC. He is a Fellow of the AAIA and a member of the Academia Europaea. Email: lik@newpaltz.edu.