MedShare: A Privacy-Preserving Medical Data Sharing System by Using Blockchain

Mingyue Wang, Yu Guo, *Member, IEEE*, Chen Zhang, Cong Wang, *Fellow, IEEE*, Hejiao Huang, *Member, IEEE*, and Xiaohua Jia, *Fellow, IEEE*

Abstract—Electronic Health Record (EHR) and its privacy have attracted widespread attention with the development of the healthcare industry in recent years. As locking medical data in a single healthcare center causes information isolation, healthcare centers are motivated to build medical data sharing systems. However, existing systems highly rely on the trusted centralized servers, which are vulnerable to distributed denial of service (DDoS) attacks and the single point of failure. Moreover, it is a non-trivial matter to authorize multiple users to search and access EHR in a privacy-preserving manner. In this article, we propose MedShare, a decentralized framework for secure EHR sharing. Our design utilizes the smart contract technique of blockchain to establish a trusted platform for healthcare centers to share their encrypted EHR. Considering that fine-grained access control is essential in practical EHR sharing service, we devise a constant-size attribute-based encryption (ABE) scheme, where the access policy is embedded in search result on the blockchain. Besides, we propose an efficient scheme that enables authorized MedShare users to perform multi-keyword boolean search operations over encrypted EHR. We formally analyze the security strengths and implement the system prototype on Ethereum. Evaluation results demonstrate that MedShare is efficient for EHR sharing.

Index Terms—Searchable encryption, decentralized framework, medical data sharing, attribute-based encryption, boolean search

1 Introduction

Healthcare industry has experienced transformations from Healthcare 1.0 to Healthcare 4.0 with the rapid development of medical technology [1], which helps to save and extend the patient's lives. In the past decade, many well-known healthcare startups, such as Flatiron Health,[1] iCarbonX,[2] and Datavant[3] have been founded. Along with this trend, Electronic Health Record (EHR) has always been playing a significant role in clinical diagnosis and treatment in the healthcare domain [2]. EHR sharing is primarily important to the improvement of medical services [3]. To reduce the maintenance cost, it is very common that EHR owners outsource their EHR to a cloud server [4].

With the increasing concern of EHR privacy, many privacy-preserving EHR sharing schemes have been proposed [5], [6]. In these schemes, the EHR owners encrypt their EHR before outsourcing to the cloud server with specified access control policy. The users (doctors or practitioners) with different attributes submit the encrypted search keywords to the cloud server. The cloud server performs the search operations over the ciphertexts of EHR with the search keywords. The users can only decrypt the ciphertexts that their attributes satisfy. However, this model relies on a centralized server to manage EHR and deal with query requests, which has two major disadvantages. First, the centralized model is vulnerable to DDoS attacks and the single point of failure, making the medical service unavailable. Second, the cloud server is not fully trustable and may not honestly perform all the required computations.

Another model for medical data sharing is the patient-centric model [7], [8]. This model lets patients collect and manage its EHR by themselves. The patients are responsible for constructing the ciphertexts of EHR with access control policy, which requires powerful computing resources. However, in practice, the EHR of a patient is stored in the healthcare centers that the patient has attended. It is actually geographically scattered when the patient goes to different healthcare centers for medical services. If a patient is transferred from one healthcare center to another, the doctor in the later center needs to refer to the EHR in the previous center. It is inefficient to access EHR for both doctors and patients due to the information isolation. It lacks a reliable platform for these distributed healthcare centers to share medical data.

1. https://flatiron.com/
2. https://www.icarbonx.com/en/
3. https://datavant.com/

- *Mingyue Wang and Xiaohua Jia are with the Department of Computer Science and Technology, Harbin Institute of Technology, Shenzhen 518055, China, and also with the and also with the Department of Computer Science, City University of Hong Kong, Hong Kong. E-mail: mingyue.wang@my.cityu.edu.hk, csjia@cityu.edu.hk.*
- *Yu Guo is with the School of Artificial Intelligence, Beijing Normal University, Beijing 100875, China. E-mail: yuguo@bnu.edu.cn.*
- *Chen Zhang is with the Department of Computer Science, City University of Hong Kong, Hong Kong. E-mail: c.zhang@my.cityu.edu.hk.*
- *Cong Wang is with the Department of Computer Science, City University of Hong Kong, Hong Kong, and also with the and also with the City University of Hong Kong Shenzhen Research Institute, Shenzhen 518057, China. E-mail: congwang@cityu.edu.hk.*
- *Hejiao Huang is with the Department of Computer Science and Technology, Harbin Institute of Technology, Shenzhen 518055, China. E-mail: huanghejiao@hit.edu.cn.*

Manuscript received 16 Dec. 2020; revised 3 Sept. 2021; accepted 19 Sept. 2021. Date of publication 24 Sept. 2021; date of current version 6 Feb. 2023. (Corresponding author: Xiaohua Jia.) Recommended for acceptance by I. Ray. Digital Object Identifier no. 10.1109/TSC.2021.3114719

The emergence of the blockchain [9] technique comes as a promising solution for EHR sharing because of its decentralization, transparency, immutability, and verifiability. The blockchain can be regarded as a decentralized ledger that records all of the history data (i.e., transactions) that can be seen by everyone. All transactions are validated by other blockchain nodes through the underlying consensus protocol, and they cannot be modified. By using blockchain, the healthcare centers can build their own indexes and post them to the blockchain, and the data users can use the smart contracts deployed on the blockchain to honestly perform the data search operations. Meanwhile, all data access operations are recorded on the blockchain and can be verified by all parties.

Recently, some blockchain-based schemes discussed in [10] have been proposed to support data sharing in healthcare. Typically, FHIRChain [11] and MeDShare [12] were both focused on managing authorization permissions through interoperability, where the behaviors of EHR were recorded on the blockchain. They highly rely on real-time interaction. That is, all the related entities need to be online all the time. Besides, a blockchain-based EHR sharing system also brings some new issues: 1) As EHR always contains sensitive information, data confidentiality is of top importance, which conflicts with the transparency of the blockchain. 2) It is still difficult to achieve encrypted fine-grained access control of the search result on the blockchain, where only lightweight storage is provided. The reason is that the ciphertext size of traditional attribute-based encryption (ABE) schemes [13], [14] increases linearly with the complexity of the access policy. 3) How to efficiently find the targeted EHR through blockchain is also a critical issue. It is challenging to enable secure on-chain multi-keyword conjunctive boolean search (boolean search) to achieve efficient EHR sharing services. Existing boolean search schemes [15], [16] require interaction between the server and data users to process the multiple keywords. However, it is inefficient for users to interact with the blockchain-based medical system for keywords in the boolean query.

In this paper, we propose MedShare, a new blockchain-based framework for privacy-preserving EHR sharing. To protect EHR privacy, healthcare centers publish their encrypted EHR indexes on the blockchain using smart contracts and then make these smart contracts to conduct reliable and secure search services. In our multi-user setting, we propose a fine-grained on-chain access control mechanism by using attribute-based encryption [17] with constant size, where the access control policy is embedded in the result. For efficient data retrieval, we design a non-interactive on-chain boolean search protocol in sublinear complexity based on the boolean search scheme [15]. In summary, our main contributions are listed as follows:

- We propose a new blockchain-based framework for secure EHR sharing, named MedShare. By utilizing the blockchain, MedShare provides the track of how the EHR is accessed and guarantees the integrity of query result during medical services.
- We design an on-chain lightweight attribute-based encryption mechanism to achieve fine-grained access control of EHR for multiple users, where the access policy is embedded in the encrypted result on the blockchain.
- We devise a new privacy-preserving on-chain boolean search scheme that enables efficient data retrieval before accessing the plaintext EHR.

The rest of this paper is organized as follows. Section 2 introduces the related work. Section 3 shows the background knowledge and cryptographic primitives. Section 4 presents the system architecture, threat assumptions, and design goals. Section 5 shows the design details. The implementation of our design is discussed in Section 6. Our security analysis is conducted in Section 7. Section 8 describes the experiment and performance evaluation. Finally, Section 9 concludes this paper.

2 Related Work

Our work involves plenty of techniques in applied cryptography and blockchain area. In this section, we will elaborate on representative studies in three principal fields.

2.1 Fine-Grained Access Control

Attribute-based encryption (ABE) technique enables the data owners with fine-grained access control for sensitive data access based on the attributes of users. Namely, only the users with attributes that satisfy the owner-enforced access policy can decrypt the encrypted messages under a specific structure. The concept of ABE was first proposed by Sahai *et al.* [18], in which the prototype was derived from identity-based encryption. Classical ABE schemes can be classified into two categories: Key-policy ABE (KP-ABE) and Ciphertext-policy (CP-ABE). In KP-ABE [19], the users' private keys were generated according to access policy, and the ciphertext of confidential data was related to the system attributes. For CP-ABE [13], [20], the roles of attributes and access policy are converted. All of these schemes have played an important role in fine-grained access control of data sharing systems. Nevertheless, it is inefficient when the authorized users want to find the targeted data among a large number of EHR directly encrypted by the ABE schemes. Several works [21], [22] have resorted to the combination of searchable encryption and ABE schemes, which is an effective method to address this issue. Moreover, when implementing an ABE system, schemes in [23], [24] suffered from efficiency drawbacks in both computation cost and ciphertext size because of the complex access structures. Several works [17], [25] have been proposed to solve this problem, which makes ABE more practical in practice. In our blockchain-based application, we utilize the ABE algorithm with constant ciphertext size and computation overhead, which can greatly reduce the storage overhead on each peer node of blockchain.

2.2 Searchable Encryption

Searchable encryption (SE) technology has always been one of the research hotspots in the cloud-based data outsourcing era. SE enables the semi-trusted server to search directly over encrypted data without decryption, where the data privacy is protected as well. Many schemes mainly focused on improving the functionality [26] and security [27] of the query in the single-writer/single-reader scenario. Namely,

the data owner, who is also the unique user, encrypts data and generates search tokens with the symmetric private key. With the continuously increasing demands of user scale in practice, the multi-user model was introduced to settle encrypted search among several separate users [28], [29]. Currently, researchers are concentrated on exploring rich functionality and secure SE schemes for large-scale data. Cash *et al.* [15] presented the first design that realized sublinear conjunctive keyword search over structured data, where boolean queries were also supported. This work has motivated several researches [30], [31] on search systems in different application scenarios. For example, Sun *et al.* devised a non-interactive scheme [32] that avoided per-query interaction between the data owner and users. To improve the efficiency of the OXT presented in [15], Kamara *et al.* [33] proposed the scheme with worst-case sublinear search. Jiang *et al.* [34] devised the privacy-preserving SE scheme to support multi-keyword search based on the low-frequent keyword on the blockchain. Recently, Zhang *et al.* [35] redesigned the boolean keyword search scheme for multiple clients with owner-enforced authorization. In dynamic databases, Patranabis *et al.* [36] developed the forward and backward private SE scheme for conjunctive boolean keyword searches. Moreover, we are aware that there are leakage abuse attacks in SE schemes in recent years. They can be mitigated by using the padding [37] or ORAM [38] technologies, which is not the focus of this paper. In this work, we aim to build an efficient and privacy-preserving on-chain boolean search scheme without relying on a trusted server.

2.3 Blockchain Applications

There are many blockchain-based schemes [39], [40] in decentralized applications, especially in access control and secure search. To achieve accountability and authentication, Azaria *et al.* [41] first devised a distributed prototype to manage EHR authorization by utilizing blockchain and smart contract. Each participant is bound to the specified smart contract to control real-time permissions by interoperability. Later, Laurent *et al.* [42] proposed an access control scheme based on blockchain, where the smart contract maintains a white/blacklisting to indicate permit/deny. These schemes only consider the blockchain as an access control moderator without concerning protecting sensitive information. To address privacy issues, Hu *et al.* [43] conceived to integrate SE technique with blockchain to achieve on-chain keyword search by storing the index on the blockchain. To enhance query performance, Cai *et al.* [44] utilized the smart contract to record the encrypted search process to ensure fair payment while the operations of the matching index are all off-chain. However, these works only focus on the single-user model, which ignores the issue of access control for multiple users. To settle this issue in medical data sharing, many blockchain-based schemes have been proposed. In [45], the authors proposed a patient-centric medical data management system with fine-grained access control on the blockchain. But it only supports the single keyword query, which is not efficient in practice. In Healthchain [46], Xu *et al.* devised a system fitting into large scale health data from the internet of things based on blockchain, where transactions were designed for key management. Besides, the proposed framework is a private blockchain by redesigning the fundamental architecture. Jiang *at al.* [47] proposed a medical information exchange system with the off-chain storage and on-chain verification design. Chen *et al.* [48] designed the EHR sharing system that supported logic expression query with integrity assured. But the access control in this scheme is to request permission through interaction. Motivated by the above works, we seek solutions to devise a verifiable and trustworthy framework supporting secure EHR sharing with fine-grained access control.

TABLE 1
Glossary

Acronym	Definition
λ	the security parameter
$id \in \{0,1\}^{\lambda}$	the file ID
$DB(w)$	set of files that contain keyword w
$\mathbf{w} = (w_1, \ldots, w_n)$	the authorized keyword set for users
L	the attribute list of users
$\mathbb{A}$	access policy of HCs
RDK	the retrieval decryption key to decrypt the encrypted EHR files
EDindex	on-chain encrypted database index
BSindex	on-chain boolean search index
PTindex	on-chain partial token map
$sterm$	the least frequent term among boolean search terms (or keywords)
H_0, H_1, H_2, H_3	cryptographic hash function
F, F_p	key-based pseudo-random function (PRF)
$\mathbf{P}$	pseudo-random permutation (PRP)
SK	private key set of users
ST_w	search token of keyword w

3 Preliminaries

In this section, we introduce some background knowledge and related techniques that will be utilized in this paper. The notations are summarized in Table 1.

3.1 Blockchain and Ethereum

Blockchain has developed into a very promising technology in recent years with the rise and booming of cryptocurrencies, such as Bitcoin [9], Ethereum [49]. It is essentially a linked chain of data blocks added by the consensus among most peer nodes. The link pointer between blocks is generated with the cryptographic hash functions that ensure transaction data in the blockchain immutable and the block linked tamper-proof. The characteristics of blockchain can be summarized as 1) Transparency: Everyone in the blockchain system can obtain all the transactions recorded. 2) Traceability: All of the status changes of data recorded in transactions can be accessed by tracing the linked blocks. 3) Immutability: All transactions added to the blockchain are the results of the cryptographic consensus. Thus, no one can change transactions and blocks.

Ethereum is a new decentralized platform with programmable smart contracts, which makes the blockchain more powerful. The platform supports Turing Complete, enabling users to create, deploy, and run the smart contract on the blockchain. Ideally, the contract can be executed automatically according to the predefined logic without any central party involved, and the result will be recorded on the

blockchain. In our system, we aim to leverage the smart contracts to honestly carry out the authorized search operations instead of using traditional central servers.

3.2 Cryptographic Primitives

3.2.1 Bilinear Pairing

Let $\mathbb{G}$, $\mathbb{G}_{\mathbb{T}}$ be two cyclic multiplicative groups of the same prime order p. g is the generator of $\mathbb{G}$. Denote $\hat{e}$ a bilinear pairing: $\mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_{\mathbb{T}}$ with the following properties:(1) Bilinear: $\hat{e}(g^a, g^b) = \hat{e}(g, g)^{ab}$ for all $a, b \in \mathbb{Z}_p^*$; (2) Non-degenerate: $\hat{e}(g, g) \neq 1$; (3) Computable: It is efficient to compute $\hat{e}(g_1, g_2)$ for all $g_1, g_2 \in \mathbb{G}$.

3.2.2 Hardness Assumptions

DDH Assumption. Let $\mathbb{G}$ be a cyclic group of prime order p, the decisional Diffie-Hellman(DDH) problem is to distinguish the ensembles $\{(g, g^a, g^b, g^{ab})\}$ from $\{(g, g^a, g^b, g^z)\}$, where the elements $g \in \mathbb{G}$ and $a, b, z \in \mathbb{Z}_p$ are chosen uniformly at random. We say that the DDH assumption holds if there is not any Probabilistic Polynomial Time (PPT) distinguisher $\mathcal{D}$ can solve the DDH problem in a non-negligible advantage

$$\left| \begin{array}{l} Pr[\mathcal{D}(\mathbb{G}, p, g, g^a, g^b, g^z) = 1] - \\ Pr[\mathcal{D}(\mathbb{G}, p, g, g^a, g^b, g^{ab}) = 1] \end{array} \right| < \epsilon.$$

Strong RSA Problem. Set $n = pq$, where p and q are two big prime integers. Choose a random element in $\mathbb{Z}_n^*$. We say that an efficient algorithm $\mathcal{A}$ solves the strong RSA problem if it receives as input the tuple (n, g) and outputs two elements (z, e) such that $z^e = g \bmod n$.

3.2.3 Pseudo-Random Functions

A function F transforms the element $x \in X$ to an output $y \in Y$ with a secret key $k_f \in K$. We say F is a pseudo-random function (PRF) if for all efficient adversaries $\mathcal{A}$, its advantage $Adv_{F,\mathcal{A}}^{prf}(\kappa) = |Pr[\mathcal{A}^{F(K,\cdot)}(1^\kappa)] - Pr[\mathcal{A}^{f(\cdot)}(1^\kappa)]| < negl(\kappa)$, where $negl(\kappa)$ is a negligible function in F and f is a truly random function from X to Y.

3.2.4 Access Policy in Attribute-Based Encryption

Access Structure. Given an attribute set L and an access policy $\mathbb{A}$, we consider that L satisfies $\mathbb{A}$ if and only if $\mathbb{A}$ returns 1 on L, represented by $L \models \mathbb{A}$. If L does not satisfy $\mathbb{A}$, we denote this case by $L \not\models \mathbb{A}$. In our scheme, we consider AND-gate policy AND_m^*. In formal, given an attribute list $L = [L_1, L_2, \ldots, L_n]$ and an access policy $\mathbb{A} = [\mathbb{A}_1, \mathbb{A}_2, \ldots, \mathbb{A}_n] = \bigwedge_{i \in \mathcal{I}_{\mathbb{A}}} \mathbb{A}_i$, where $\mathcal{I}_{\mathbb{A}}$ is a subscript index set and $\mathcal{I}_{\mathbb{A}} = \{i | 1 \leq i \leq n, \mathbb{A}_i \neq *\}$, we hold that $L \models \mathbb{A}$ if $\mathbb{A}_i = *$ or $L_i = \mathbb{A}_i$ for all $1 \leq i \leq n$ and $L \not\models \mathbb{A}$ otherwise. It is noted that the wildcard $*$ in $\mathbb{A}$ plays the role of "do not care" value.

4 Problem Statement

4.1 System Architecture

Fig. 1 shows the architecture of MedShare, containing four main entities: healthcare centers (HCs), EHR users, the blockchain, and attribute authority (AA) (not shown).

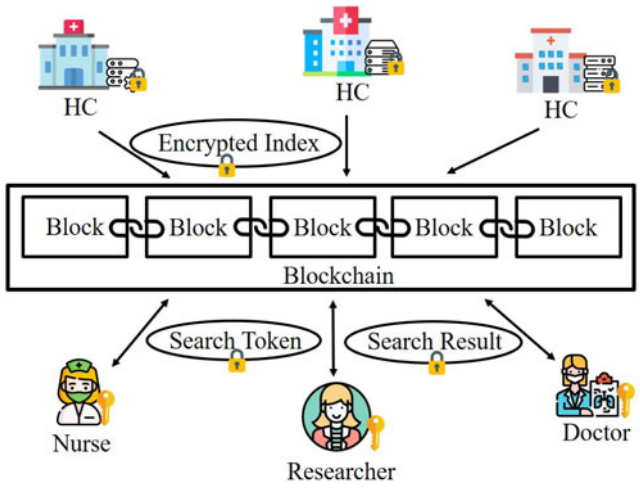


Fig. 1. The system architecture of MedShare.

- *HCs* are data owners of EHR that establish and collect medical data. To protect data confidentiality, they need to encrypt original EHR and store them locally (or local server). They generate and distribute authorized keyword search keys for users. Meanwhile, they are responsible for building encrypted indexes and deploying them on smart contracts.
- *Users* in MedShare are doctors, nurses, and so on, who are allowed to access the EHR. They are assigned with various attributes, so they have different EHR access permissions. After each query, they obtain the search result from the blockchain.
- *Blockchain* is the platform that provides decentralized and verifiable computation. In MedShare, the encrypted indexes of EHR are deployed on smart contracts by HCs in advance. When users submit authorized search tokens, the on-chain search algorithm is executed by smart contract automatically. Note that we adopt the consortium blockchain in MedShare to improve the throughput of the whole system, where the latest Blockchain-as-a-Service (BaaS) solutions can be applied, such as Microsoft Azure, Amazon, etc.
- *AA* is responsible for entitling attributes for users according to their identities or roles in our system. It also issues the encryption keys for HCs and the attribute secret keys for users.

In MedShare, we utilize a hybrid storage architecture, the same as in [44], to adapt to the limited storage capability of blockchain, i.e., storing encrypted EHR locally and maintaining indexes on the blockchain. Each HC maintains their EHR databases as the inverted index (denoted by $DB(w)$), the EHR files that include keyword w. It constructs the secure searchable indexes with fine-grained access control policy. The problem is that, given an authorized search token of multiple keywords, we aim to achieve verifiable and correct boolean search over on-chain EHR indexes securely. Meanwhile, for the search result on the blockchain, each user can only access the EHR permitted by its attributes.

4.2 Threat Assumptions

Our main goal is to protect data confidentiality and query privacy in data access by the blockchain-based system. We make the following threat assumptions for the entities involved in our system:

- *Peer nodes of the blockchain* are potential adversaries [50]. They honestly execute the designated search protocols. However, they intend to learn sensitive information from the on-chain indexes, query transactions, and search result.
- *HCs and users* are assumed trusted. We consider HCs are reputable institutions and honestly provide valid EHR indexes. Users submit search tokens generated from the predefined token generation protocol.
- *AA* is always trusted. It honestly generates keys for HCs and users, respectively.

4.3 Design Goals

Under the aforementioned system architecture and threat assumptions, the proposed MedShare scheme aims to achieve the following goals.

- *Privacy Preserving:* No other party can obtain the EHR information except the authorized users.
- *Correctness and Verifiability:* The correctness and completeness of search result in the medical services shall be guaranteed.
- *Fine-grained Access Control:* The access control of EHR files is decided by the access policy embedded in the encrypted indexes. Only if the attributes of the user satisfy the policy can it access the corresponding EHR files of search result.
- *Multi-keyword Conjunctive Boolean Search:* By submitting the search tokens encrypted from multiple keywords, the user will get encrypted result containing all of these keywords.

5 The Design of MedShare

In this section, we first introduce the overview design of MedShare. Our design can be regarded as two parts: on-chain access control and multi-keyword boolean search. We further present the design details of each part, respectively.

5.1 The Overview of Our Design

In our design, we adopt the blockchain as the underlying platform to conduct the search and access control procedures in EHR sharing. To find the targeted EHR efficiently, we propose the privacy-preserving on-chain boolean search scheme. As shown in Fig. 2, the HCs need to make their own access policy in advance for users with different attributes. All EHR is encrypted and stored locally by the HCs. They construct the searchable indexes based on the predefined access policy and then deploy them into the smart contracts on the blockchain, respectively. Under our design, access control is embedded in the search result stored on smart contracts. When users want to access EHR, they transform the search keywords into search token with the authorized keyword key independently. The smart contracts will perform the encrypted boolean search according to search protocol. Then, the result (encrypted EHR file IDs and decryption keys) is recorded on the blockchain, and users can decrypt it using their attribute keys assigned previously by the AA.

To enable privacy-preserving and efficient EHR sharing for HCs and users, there are still two challenges to be solved.

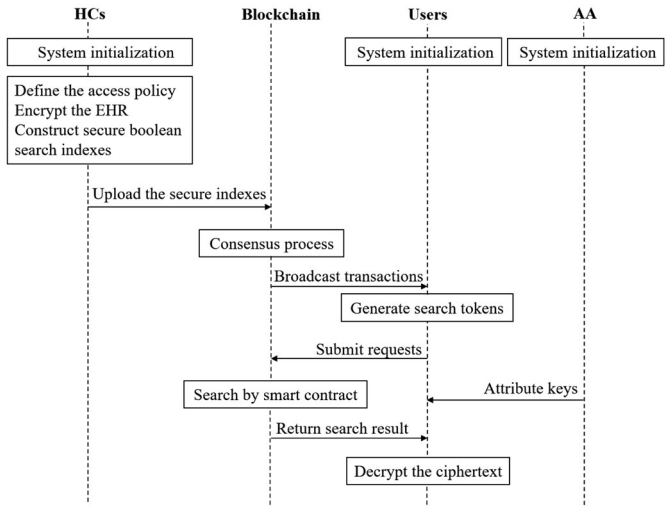


Fig. 2. The overview of proposed MedShare.

The first one is to devise a privacy-preserving on-chain access control scheme. Existing solutions [41], [42], [51] use the blockchain to realize real-time permit/deny access control, where the access policy is public to everyone. As the blockchain is transparent, it will lead to the disclosure of privacy if the access policy is sent to the blockchain directly in our medical system. In addition, fine-grained access control is more practical in such a system. However, it is challenging because blockchain can only support lightweight storage. The second challenge is how to enable users to perform efficient on-chain boolean search. Existing cloud-based boolean search schemes [15], [16] rely on the cloud to interact with users honestly. However, the interaction will greatly reduce the throughput of the whole system on the blockchain. Moreover, there is not such a trusted server anymore. To solve the two issues mentioned above, we propose the on-chain access control by adopting constant-size ABE and designing on-chain indexes to support efficient boolean search in a non-interactive manner. The details will be presented in the following subsections.

5.2 On-Chain Constant-Size ABE-Based Access Control

In MedShare, the first issue is how to achieve fine-grained access control for multiple users on the blockchain. To this end, we resort to ABE [19] technology, which is suitable for the data access control design of multiple users. However, it is not easy to apply ABE to our blockchain-based platform. The reason is that existing ABE schemes suffer from severe efficiency drawbacks due to huge ciphertext storage and large computation overhead. Whereas, blockchain can only support limited storage and has constrained computation power. In light of the aforementioned observations, we design a new solution through constant-size ciphertext ABE with constant computation cost.

In EHR sharing application scenarios, we consider that users have multiple attributes, by which the access policy is defined. Let $\mathcal{U} = \{\eta_1, \eta_2, \ldots, \eta_n\}$ denote n attributes in the medical system, where each of them has multiple values. Let $S_i = \{v_{i,1}, v_{i,2}, \ldots, v_{i,n_i}\}$ denote the multi-value set for the attribute η_i. For instance, a user may belong to department of medicine with the duty of nurse. Next, we will

present the on-chain fine-grained access control design in MedShare.

ABE.Setup. AA chooses two random numbers $x, y \in \mathbb{Z}_p^*$ as the master key $ABE.msk = \langle x, y \rangle$. H_0 is a collision-resistant hash function: $\{0,1\}^* \rightarrow \mathbb{Z}_p^*$. To bind each attribute value to its corresponding attribute, we set X_{i,k_i}, Y_{i,k_i} as

$$X_{i,k_i} = g^{-H_0(x||i||k_i)}, Y_{i,k_i} = \hat{e}(g,g)^{H_0(y||i||k_i)}, \tag{1}$$

where i $(1 \leq i \leq n)$ is the ith attribute η_i in $\mathcal{U}$ and k_i $(1 \leq k_i \leq n_i)$ is the kth value in the attribute value set S_i. The public key for attributes is $ABE.mpk = \langle g, \{X_{i,k_i}, Y_{i,k_i}\} \rangle$.

ABE.KeyGen. Let L denote the attribute list that the user has. For each attribute value in L, AA selects a secret value $sk \in_R \mathbb{Z}_p^*$ to bind each value to the attribute it belongs to. H_3 is another collision-resistant hash function: $\mathbb{Z}_p^* \rightarrow \mathbb{G}$. Then for the ith attribute $(1 \leq i \leq n)$ and the k_ith value in the attribute list L of the users, the corresponding attribute key $SK_L = \langle sk, \{\bar{\sigma}_i\} \rangle$ can be computed as

$$\bar{\sigma}_i = \sigma_{i,k_i} = g^{H_0(y||i||k_i)} H_3(sk)^{H_0(x||i||k_i)}. \tag{2}$$

ABE.Enc. In our design, the EHR file indexes are uploaded to the blockchain. To protect the privacy and achieve fine-grained access control, we resort to making the ciphertext of ABE with constant size, which reduces the storage overhead of blockchain. In detail, we aggregate all the attributes and corresponding values in access policy $\mathbb{A} = \bigwedge_{i \in \mathcal{I}_\mathbb{A}} \mathbb{A}_i$ by a continuous multiplication function

$$\langle X_\mathbb{A}, Y_\mathbb{A} \rangle = \left\langle \prod_{i \in \mathcal{I}_\mathbb{A}} \bar{X}_i, \prod_{i \in \mathcal{I}_\mathbb{A}} \bar{Y}_i \right\rangle,$$

where $\langle \bar{X}_i, \bar{Y}_i \rangle = \langle X_{i,k_i}, Y_{i,k_i} \rangle$ is shown in Eq. (1). In MedShare, files are encrypted with different symmetric AES key k_{id}, which is also the retrieval decryption key (RDK). The HCs store the encrypted files locally. Here, the file ID id and corresponding decryption key k_{id} $(id||k_{id})$ is what we want to protect under access policy. Then, each HC selects $s \in_R \mathbb{Z}_p^*$ and sets

$$e_{id} = \mathbf{ABE.Enc}(mpk, id||k_{id}, \mathbb{A}) = \langle \mathbb{A}, C_0, C_1, C_2 \rangle, \tag{3}$$

where $C_0 = (id||k_{id}) \cdot Y_\mathbb{A}^s, C_1 = g^s$, and $C_2 = X_\mathbb{A}^s$. Now, we have encrypted the sensitive information into constant size ciphertext with fine-grained access policy. We will discuss how to combine it into the on-chain searchable indexes in Section 5.3.2.

ABE.Dec. After retrieving search result from the blockchain, the user needs to decrypt them to get the plaintext EHR files. Similarly to that in*ABE.Enc*, secret key aggregation is also conducted by $\sigma_\mathbb{A} = \prod_{i \in \mathcal{I}_\mathbb{A}} \bar{\sigma}_i$, where the $\bar{\sigma}_i$ is the user's secret attribute key SK_L. Then the decryption phase is computed as

$$id||k_{id} = \frac{C_0}{\hat{e}(\sigma_\mathbb{A}, C_1) \cdot \hat{e}(H_3(sk), C_2)}, \tag{4}$$

where C_0, C_1, C_2 are the ciphertexts shown in Eq. (3). Only when the user's attributes satisfy the access policy (i.e., $L \models \mathbb{A}$) defined by the HC, the decryption in Eq. (4) can happen successfully. In our design, the decryption is also constant in computation, which is user-friendly. Note here, the constant computation means the decryption time is constant even if the number of attributes varies. Then, the user can obtain the encrypted EHR file with id and get the plaintext EHR by using symmetric RDK k_{id}.

5.3 Multi-Keyword Conjunctive Boolean Search

5.3.1 Non-Interactive Keyword Authorization

Before introducing the design of on-chain boolean search, we now present how to implement the multi-user search keyword authorization process in MedShare. Existing privacy-preserving solutions [5], [52] require HCs to be online all the time to provide search tokens for different users, which are not suitable for practical EHR sharing applications. To address this issue, we devise a non-interactive search authorization scheme based on [16]. In MedShare, the HCs use the hash functions to transform keywords to primes to improve storage efficiency. Here, all keywords are mapped to distinct primes. In this paper, we will use w to represent the corresponding prime of the keyword for ease of presentation. Afterward, by utilizing the RSA function, there will be only one interaction for HCs to issue the keyword secret keys in the initialization phase. Then, users can produce their targeted keyword search token independently, reducing the communication overhead between HCs and users significantly. By using our proposed scheme, users can only perform secure keyword search within the permitted keyword set.

With these primes of keywords, HCs will use $g^{\frac{1}{w_i}}$ to construct secure searchable indexes for keyword w_i. To authorize keyword set, each HC provides the potential users with a partial token $g^{\frac{1}{w_1 \ldots w_n}}$ for the keyword set $\mathbf{w}$ including $''w_1, w_2, \ldots, w_n''$, which are the keywords that the users will search with. Therefore, for specific users, their search capabilities are restricted at the given range rather than a random keyword search. When the user needs to search on keyword $w_i \in \mathbf{w}$, it can obtain the value $g^{\frac{1}{w_i}}$ as shown in Eq. (5) based on the strong RSA problem, which will be used to generate the encrypted search tokens

$$g^{\frac{1}{w_i}} = \left(g^{\frac{1}{w_1 \ldots w_n}}\right)^{\prod_{w \in \mathbf{w}/\{w_i\}} w}. \tag{5}$$

5.3.2 On-Chain Index Design

Under the above-mentioned design, we then consider how to achieve encrypted boolean search for the blockchain-based EHR sharing system. One straightforward method of performing boolean search is to repeat the single keyword search directly to retrieve all matched files. Then, users can get the common files via interaction operation of all these sets. This approach is simple but requires multiple rounds of interactions between the server and the users. Moreover, it will bring severe privacy leakage, where the server can access the matched file set of every keyword [31]. To address the above concerns, we seek an efficient and secure SE scheme for boolean search on the blockchain.

Motivated by [15], we resort to building three on-chain indexes achieving sublinear search complexity on the blockchain. The indexes are in the form of key-value pairs:

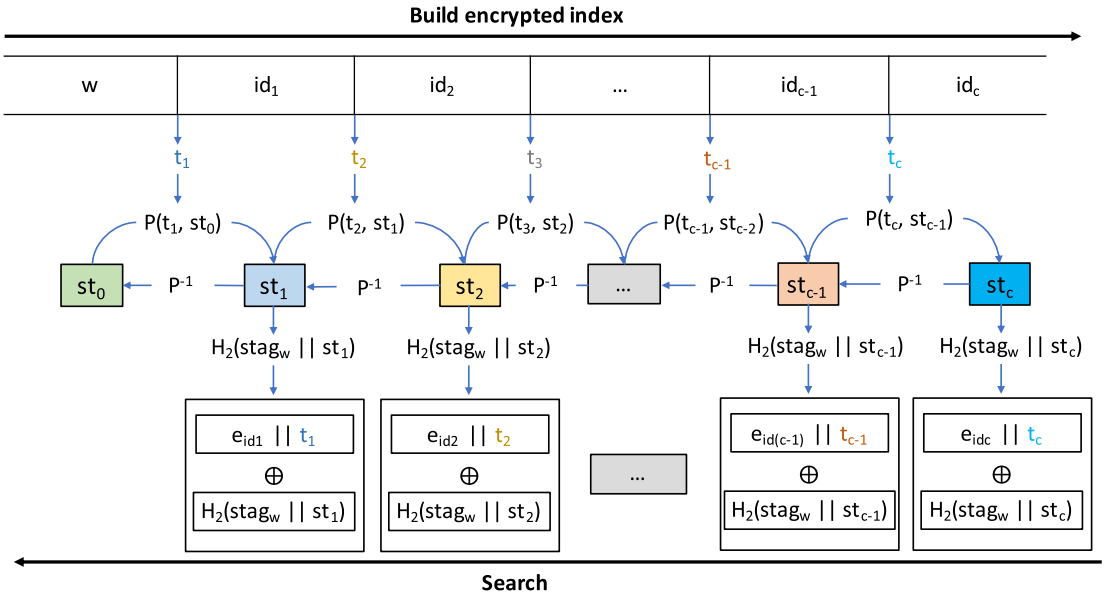


Fig. 3. The design of on-chain index EDindex.

- EDindex (encrypted DB index) is the encrypted form of mapping from keywords to all file IDs that contains them, respectively.
- BSindex (boolean search index) is a file-keyword index, where the key represents the containment of a file for its keyword while the value indicates the existence of the containment.
- PTindex (partial token map index) is used for generating search token (to be discussed in Section 6.1) without relying on the trusted server as in [16].

To protect the privacy of search keyword w, we first encrypt it to $stag_w$ (to be discussed in *IndexGen* of Section 6.1.). We derive $xind \leftarrow F_p(K_I, id)$ to hide the file ID id with PRF F_p. As shown in Fig. 3, in EDindex, we link each id with a counter c and a random nonce t_c as each keyword w is contained in multiple files in the inverted DB index. To produce new unique nonce for each new file id, we derive the search token nonce st_c from the random nonce t_c of current file id and the last st_{c-1} by a PRP $\mathbf{P}$. With this new search token nonce st_c, the relationships between the newly appended files and the previous search queries cannot be revealed. To generate the keys in EDindex, we associate each keyword with the latest random token nonce st_c by hash function H_1. Note that we use another different hash function H_2 to mask the ciphertexts and the random nonce t_c of file id as follows:

$$e \leftarrow (e_{id}||t_c) \oplus H_2(stag_w||st_c), \tag{6}$$

where e_{id} is the encrypted ciphertext of ABE in Eq. (3) to achieve on-chain fine-grained access control. Then, the access control is embedded in the result on the blockchain securely in MedShare. Moreover, to associate w with each file counter c (for multi-keyword boolean search process), we generate the blinding value z

$$z \leftarrow F_p\Big(K_Z, g_2^{1/w}||c\Big). \tag{7}$$

Then, we can calculate the blinded value $y \leftarrow xind \cdot z^{-1}$ that reveals the connection of w and id for later efficient boolean search services. In EDindex, we store the corresponding e and y for each file of keyword w.

In BSindex, to enable the boolean search, we need to build and pre-store the one-to-one relation, representing this file contains the keyword by computing $xtag \leftarrow g_3^{F_p(K_X, g_3^{1/w'}) \cdot xind}$. The $xtag$ value is to test whether this file ($xind$) contains the the keyword w'.

During the search process, there are multiple keywords $''w_1, w_2, \ldots''$ in query Q. Here, w_1 is the least frequent keyword in the query (sterm). Then, the blinding value z is used to blind the rest search keyword w_j $(j = 2, \ldots)$ to build $xtoken[i, j]$ as follows:

$$g^{F_p\left(K_Z, \left(sk_{\mathbf{w}}^{(2)}\right)^{\prod_{w \in \mathbf{w}/\{w_1\}} w}||i\right) \cdot F_p\left(K_X, \left(sk_{\mathbf{w}}^{(3)}\right)^{\prod_{w \in \mathbf{w}/\{w_j\}} w}\right)} = g^{F_p\left(K_Z, g_2^{1/w_1}||i\right) \cdot F_p\left(K_X, g_3^{1/w_j}\right)}, \tag{8}$$

where i is the counter of the ith file containing the keyword w_1 , and $sk_{\mathbf{w}}^{(2)}$, $sk_{\mathbf{w}}^{(3)}$ are parts of the search authorization key (as shown in Eq. (9)). To avoid the interaction between the server and users in [15], the index PTindex is constructed to link the counter c and search token nonce st_c with the current keyword. Here, only users with search keys can generate the label l to fetch the targeted partial token from PTindex on the smart contracts. Only when all the $xtoken[i, j]^y \in$ BSindex for the ith file of w_1, this id will be added into the final result. This protocol is correct because $xtoken[i, j]^y = g^{z \cdot F_p\left(K_X, g_3^{1/w}\right) \cdot (xind \cdot z^{-1})} = g^{F_p\left(K_X, g_3^{1/w}\right) \cdot xind}$, which means this $xtag$ in BSindex is correctly recomputed. If all $xtag$s are matched in BSindex, this file also contains all the rest keywords in query Q.

6 Implementation of MedShare

In this section, we first introduce the detailed construction of the secure multi-keyword search in MedShare. Next, we show how to extend it into general boolean search. Finally, we will give more discussions about the implementation prospect.

6.1 Construction of Secure Multi-Keyword Search

We now introduce the detailed implementation of our proposed MedShare. This scheme consists of five algorithms: *Initialization* , *DUKeyGen* , *IndexGen* , *DUTokenGen* , and *On-chain Search* , discussed as follows.

Initialization. Each HC executes the system initialization with security parameter λ. It first finds two big prime integers p, q, random keys $K = (K_I, K_Z, K_X)$ for a PRF F_p, and k for a PRF F. H_1, H_2 are two cryptographic hash functions. $\mathbf{P}$ is a symmetric pseudo-random permutation. Set $g_i \in_R \mathbb{Z}_n^*$ for $i = 1, 2, 3$, and $n = pq$. Then, the system public key for each HC is $PK = (n, F, H_1, H_2, F_p, \mathbf{P}, ABE.mpk)$, and the system master key is $MK = (p, q, k, K, g_1, g_2, g_3, ABE.msk)$.

DUKeyGen. This process is conducted by the HCs and AA, authorizing what the users can search and access, respectively. Assuming that an authorized user with attribute list L is permitted to search over keywords $\mathbf{w} = (w_1, w_2, \ldots, w_n)$, the HC generates the private key SK for each of its authorized user. For each user, the secret key consists of two parts, i.e., search keyword authorization key $SK_{\mathbf{w}}$ and access authorization key SK_L. In detail, the $SK_{\mathbf{w}}$ is computed as

$$SK_{\mathbf{w}} = (sk_{\mathbf{w}}^{(1)}, sk_{\mathbf{w}}^{(2)}, sk_{\mathbf{w}}^{(3)}), \tag{9}$$

where $sk_{\mathbf{w}}^{(t)} = g_t^{1/\prod_{j=1}^{n} w_j}$ for $t \in [1, 2, 3]$, which is determined by all authorized keywords w_j in $\mathbf{w}$. The key $SK_L = \langle sk, \{\bar{\sigma_i}\}_{1 \le i \le n} \rangle$ is originated from their attributes and relevant values according to Eq. (2). The secret key set $SK = (k, K, SK_{\mathbf{w}}, SK_L)$ together with the permission set $\mathbf{w}$ are provided to the users in a secure channel.

Algorithm 1. Encrypted Index Generation

Input: Public key PK, master key MK, database DB, the retrieval decryption key array RDK, a set of access policy $\mathbb{A}$
Output: EDindex, BSindex, and PTindex

Initialize EDindex and PTindex to empty arrays;
Initialize BSindex to an empty set;
for each $w \in DB$ **do**
  Initialize a counter $c \leftarrow 0$;
  Generate a random nonce $st_0 \xleftarrow{\$} \{0,1\}^{\lambda}$;
  Calculate $stag_w$ according to Eq. (10);
  for all $id \in DB(w)$ in random order **do**
    Set the counter $c \leftarrow c + 1$;
    Generate a random nonce $t_c \xleftarrow{\$} \{0,1\}^{\lambda}$;
    Update $st_c \leftarrow \mathbf{P}(t_c, st_{c-1})$;
    $u \leftarrow H_1(stag_w || st_c)$;
    Calculate e_{id} (Eq. (3)), e(Eq. (6));
    Compute z(Eq. (7));
    Set $xind \leftarrow F_p(K_I, id)$; $y \leftarrow xind \cdot z^{-1}$;
    Append EDindex[u] = (e, y) to EDindex;
    Set $xtag \leftarrow g^{F_p\left(K_X, g_3^{1/w}\right) \cdot xind}$ and add $xtag$ to BSindex;
  Set $l \leftarrow H_1(stag_w)$ and append $st_c || c$ to PTindex[l];
Deploy EDindex, BSindex, PTindex to the blockchain batch by batch.

IndexGen. To support efficient and secure boolean search on the blockchain, HCs need to execute Algorithm 1 independently to generate encrypted databases, which includes three on-chain indexes: EDindex, BSindex, and PTindex (as discussed in Section 5.3.2). They take $DB(w)$, the retrieval decryption key array RDK, and the access policy $\mathbb{A}$ as the input. In EDindex, each keyword w in the inverted index is calculated as

$$stag_w \leftarrow F\left(k, g_1^{1/w}\right), \tag{10}$$

where the F is a key based PRF, and k is the master key. Then, for every file id of w, the key-value pairs $u \leftarrow (e, y)$ is constructed. To build BSindex, $xtag$ is constructed, indicating that the current keyword is contained in the file. The PTindex maps keywords to the encrypted partial tokens, which is used to assist the users in generating search tokens. After completing the procedure of index generation, the HCs deploy the EDindex, BSindex, and PTindex to smart contracts, respectively. In this way, the smart contract can access the indexes submitted by this HC.

Discussions. Since the EHR files need to be updated frequently, the HCs are responsible for building new search indexes as desired. When the HC adds keyword-file pair in the database, it first calculates l to fetch PTindex[l] from the smart contract. If PTindex[l] is empty, it conveys that this keyword has not appeared in the previously posted DB. Then, it needs to rebuild the EDindex, BSindex, PTindex of the corresponding keyword as discussed in Section 5.3. The HCs then update all the new indexes on the blockchain.

DUTokenGen. To perform a boolean search Q on keyword set $\bar{\mathbf{w}} \subseteq \mathbf{w}$, the authorized user need to generate the search token with its secret key SK. For simplicity, we take the conjunctive query $Q = w_1 \wedge w_2 \wedge \cdots \wedge w_m$ as an example, and the general boolean search will be presented in following Section 6.2. Above all, it chooses sterm $\bar{\mathbf{s}} \subseteq \bar{\mathbf{w}}$ in Q to generate $stag$ as

$$stag \leftarrow F\left(k, \left(sk_{\mathbf{w}}^{(1)}\right)^{\prod_{w \in \mathbf{w}/\{w_1\}} w}\right), \tag{11}$$

supposing w_1 is the sterm here. Then, it can access the corresponding PTindex. Note that, the counter c implies the cth file that includes current keyword. It then generates $xtoken$ that signifies the relation between sterm and the rest keyword in Q. The complete search token ST_w can be obtained as elaborated in Algorithm 2.

Algorithm 2. Search Token Generation

Input: Private key of authorized users SK, boolean query Q
Output: Search Token ST_w

Initialize ST_w and $xtoken$ to empty arrays;
Initialize $\bar{\mathbf{s}}$ to empty set;
Set $\bar{\mathbf{s}} \leftarrow \bar{\mathbf{s}} \cup \{w_1\}$; $\mathbf{x} \leftarrow \bar{\mathbf{w}}/\bar{\mathbf{s}}$;
Calculate the tag $stag$ of sterm (Eq. (11));
Set $l \leftarrow H_1(stag)$;
Fetch $st_c || c \leftarrow$ PTindex[l] from smart contract;
for $i = c$ to 1 **do**
  for all j in $\mathbf{x}$ **do**
    Compute each $xtoken[i, j]$(Eq. (8));
Set $ST_w \leftarrow (stag, xtoken)$;
return ST_w.

On-Chain Search. The boolean search scheme following the on-chain index design is illustrated in Algorithm 3. To

search for targeted EHR files satisfying interested query conditions, each authorized user first generates the search token ST_w independently. Next, it posts it to the blockchain through a query transaction so that the peer nodes on the blockchain can execute the smart contracts to match the qualified encrypted file indexes. Particularly, after getting the ith file, it calculates $xtag' = xtoken[i,j]^y$ and judge if $xtag' \in$ BSindex. If all of the $xtag$'s are contained, this encrypted file ID (with fine-grained access control) is a part of the final result R. Then, users can obtain their search results on the blockchain.

Algorithm 3. Search on Smart Contract

Input: Search token ST_w, partial token $st_c||c$
Output: Result R
Initialize R to empty sets;
for $i = c$ to 1 **do**
  Set $u \leftarrow H_1(stag_w||st_i)$;
  Get $(e, y) \leftarrow$ EDindex.$find(u)$;
  $e_{id}||t_i \leftarrow e \oplus H_2(stag_w||st_i)$;
  if $xtoken[i,j]^y \in$ BSindex for all j **then**
    Add e_{id} to R;
  Compute $st_{i-1} \leftarrow \mathbf{P}^{-1}(t_i, st_i)$;
return R.

For the user with private key SK_L, it then decrypts the encrypted ciphertext in R to get the matching file ID and the retrieval key k_{id}. That is, if $L \models \mathbb{A}$, the user computes $\sigma_{\mathbb{A}} = \prod_{i \in \mathcal{I}_{\mathbb{A}}} \bar{\sigma}_i$ to get the original message as shown in Eq. (4). At last, the user can get the plaintext medical files by decrypting encrypted file with the retrieval key k_{id}.

6.2 Extension to General Boolean Search

As mentioned, we have only explained our protocol for the case of conjunctive keyword queries. Similar to [15], [16], our scheme can also be extended to support the form $w_1 \wedge \psi(w_2 \ldots, w_m)$, where ψ is a boolean formula of keywords w_i belonging to the permitted keyword set $\mathbf{w}$. Thus, the authorized users consider w_1 as the sterm to calculate $stag$ and xtoken for the rest of keywords in the query. Differently, they need to submit the search token together with the boolean formula ψ to the blockchain. In the on-chain search procedure, after retrieving the tuple (e, y), smart contract will record a new series of boolean variables $v_2, \ldots, v_m$ as defined in Eq. (12) instead of checking whether all $xtoken \in$ BSindex or not

$$\begin{cases} v_i = 1, & xtoken[c,i]^y \in \text{BSindex} \\ v_i = 0, & otherwise \end{cases}. \tag{12}$$

Then, the value of boolean formula $\psi(v_2, \ldots, v_m)$ can be evaluated easily. If it is true, which means the tuple satisfies the query condition, the encrypted index e will be added into the final result. By following the similar design, we can achieve any boolean query expressions.

6.3 More Discussions

Comparing to the centralized model, blockchain technology shows great advantages to facilitate the scattered EHR sharing, while it lacks privacy protection. Since the patients' EHR files are personal and highly sensitive, it is indispensable to design a secure medical data sharing scheme. Moreover, who can access the plaintext EHR file is also an essential issue. In this paper, we design an efficient and secure EHR sharing protocol with fine-grained access control customized to the blockchain. In the implementation, we are also aware of the defect of blockchain's efficiency. There are already many excellent works [53], [54] that concentrated on improving the scalability and throughput of the blockchain. The advancement of BaaS technology further provides promising scalable solutions. HCs can utilize the distributed storage system, for example, InterPlanetary File System (IPFS),[4] to reduce the local storage overhead. Based on these observations, we can foresee that the blockchain-based, secure, and feasible EHR sharing system will be applied in the near future.

7 Security Analysis

In this section, we demonstrate the security guarantees of the proposed scheme for secure search over the encrypted on-chain indexes. Recall that we technically combine the SE and ABE schemes to design the secure boolean search protocols with access control. With the SE scheme, the smart contracts can perform search operations securely. Meanwhile, the ABE scheme enables credible fine-grained authorization for EHR sharing in multi-user scenarios. To formally analyze the security guarantees, we follow the adopted cryptographic primitives in the framework of SE [55]. Specifically, we first define the setup leakage $\mathcal{L}^{\text{Stp}}$ as

$$\mathcal{L}^{\text{Stp}} = (|\text{EDindex}|, \langle|\mathcal{L}_1|, |\mathcal{P}_1|\rangle_n, |\text{BSindex}|, \langle|xtag|_x\rangle, |\text{PTindex}|, \langle|\mathcal{L}_2|, c\rangle_m),$$

where $|\text{EDindex}|$ and $|\text{BSindex}|$ are the size of index EDindex and corresponding BSindex, respectively. $|\text{PTindex}|$ is the size of PTindex map. $\langle|\mathcal{L}_1|, |\mathcal{P}_1|\rangle_n$ are the ciphertext lengths of n key-value structures in EDindex. $\langle|xtag|_x\rangle$ is the length of x elements in BSindex set. $\langle|\mathcal{L}_2|, c\rangle_m$ are the ciphertext lengths of m key-value pairs in PTindex, in which c indicates the file number. When a user sends a search transaction for the keyword set $\mathbf{w} = \{w_1, \ldots, w_n\}$, the view of an adversary is defined in the leakage $\mathcal{L}^{\text{Srch}}$ as

$$\mathcal{L}^{\text{Srch}} = (ST_{w_1}, \text{PTindex}[w_1], \langle\mathcal{L}, \mathcal{P}\rangle_s),$$

where ST_{w_1} is the search token, $\text{PTindex}[w_1]$ is the matched entry of PTindex map of keyword w_1, and $\langle\mathcal{L}, \mathcal{P}\rangle_s$ are s matched EDindex entries. When updating keyword w in EDindex index, the leakage function $\mathcal{L}^{\text{Updt}}$ captured by an adversary is defined as

$$\mathcal{L}^{\text{Updt}} = (op, \text{EDindex}[w], \text{PTindex}[w]),$$

where $op \in \{add\}$ denotes update operations for (kw, id) pairs, $\text{EDindex}[w]$ is the updated entry of the EDindex index, and $\text{PTindex}[w]$ represents the keyword w entry in PTindex map. Apart from above leakages, we also define the leakage $\mathcal{L}^{\text{Rpt}}$ for repeated queries

$$\mathcal{L}^{\text{Rpt}} = (M_{q \times q}, R_q),$$

4. https://github.com/ipfs/ipfs

where $M_{q\times q}$ is the symmetric bit matrix that records repeated q queries and R_q denotes the result set. Given the simulation-based security definition [15], we provide the formal definition:

Definition 1 *Let* $\Omega = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Search}, \mathsf{Update})$ *be our scheme for secure request-search services. Given leakage functions* $\{\mathcal{L}^{\mathrm{Stp}}, \mathcal{L}^{\mathrm{Srch}}, \mathcal{L}^{\mathrm{Updt}}, \mathcal{L}^{\mathrm{Rpt}}\}$*, we define the following probabilistic experiments* $\mathbf{Real}^{\Omega}_{\mathcal{A}}(\lambda)$ *and* $\mathbf{Ideal}^{\Omega}_{\mathcal{A},\mathcal{S}}(\lambda)$ *with a probabilistic polynomial time (PPT) adversary* $\mathcal{A}$ *and a PPT simulator* $\mathcal{S}$*:*

$\mathbf{Real}^{\Omega}_{\mathcal{A}}(\lambda)$*: the experiment calls* $KGen(1^{\lambda})$ *to generate a private key* k*.* $\mathcal{A}$ *selects a sample set* (w, id) *to build the encrypted indexes via* Setup *and* Update *algorithms with the private key* k*. Then* $\mathcal{A}$ *adaptively conducts a polynomial number of queries with authorized keyword via* Search *algorithm. Finally,* $\mathcal{A}$ *returns a bit as the output.*

$\mathbf{Ideal}^{\Omega}_{\mathcal{A},\mathcal{S}}(\lambda)$*:* $\mathcal{A}$ *selects a sample set* (w, id)*, and* $\mathcal{S}$ *simulates indexes* $\widetilde{EDindex}$, $\widetilde{BSindex}$ *for* $\mathcal{A}$ *based on* $\mathcal{L}^{\mathrm{Stp}}$*. From* $\mathcal{L}^{\mathrm{Updt}}$*,* $\mathcal{S}$ *can update the encrypted indexes. Then,* $\mathcal{A}$ *adaptively conducts a polynomial number of queries.* $\mathcal{S}$ *simulates search tokens and ciphertexts from* $\mathcal{L}^{\mathrm{Srch}}$ *and* $\mathcal{L}^{\mathrm{Rpt}}$ *in each query. Finally,* $\mathcal{A}$ *returns a bit as the output.*

Ω *is a* $(\mathcal{L}^{\mathrm{Stp}}, \mathcal{L}^{\mathrm{Srch}}, \mathcal{L}^{\mathrm{Updt}}, \mathcal{L}^{\mathrm{Rpt}})$*-secure scheme, if for all PPT adversaries* $\mathcal{A}$*, there exists a simulator* $\mathcal{S}$ *such that:* $Pr[\mathbf{Real}^{\Omega}_{\mathcal{A}}(\lambda) = 1] - Pr[\mathbf{Ideal}^{\Omega}_{\mathcal{A},\mathcal{S}}(\lambda) = 1] \leq negl(\lambda)$*, where* $negl(\lambda)$ *is a negligible function in* λ*.*

Theorem 1 Ω *is an adaptively secure scheme with* $(\mathcal{L}^{\mathrm{Stp}}, \mathcal{L}^{\mathrm{Srch}}, \mathcal{L}^{\mathrm{Updt}}, \mathcal{L}^{\mathrm{Rpt}})$ *leakages under the random-oracle model if* $\{F, F_p, H_1, H_2\}$ *are secure PRFs.*

Proof. The objective is to prove that all PPT adversaries $\mathcal{A}$ cannot distinguish the output of $\mathbf{Real}^{\Omega}_{\mathcal{A}}(\lambda)$ and $\mathbf{Ideal}^{\Omega}_{\mathcal{A},\mathcal{S}}(\lambda)$ as defined in Definition 1. We first define random oracles $\{\mathcal{H}_F, \mathcal{H}_{F_p}, \mathcal{H}_{H_1}, \mathcal{H}_{H_2}\}$. From leakage $\mathcal{L}^{\mathrm{Stp}}$, the simulator $\mathcal{S}$ simulates the indexes, which have the same size as the real one. The indexes contain a dictionary with n entries and a set with x elements. The dictionary $\mathcal{D}$ uses $\mathcal{L}_1$-bit and $\mathcal{P}_1$-bit random strings as the key-value pairs in each entry. Each element in the set $\mathcal{E}$ consists of $|xtag|$-bit random string.

When the first query sample (v_s, v_f, v_x) is sent, $\mathcal{S}$ generates simulated tokens $\widetilde{t_1} = \mathcal{H}_F(\widetilde{k_1}||v_s)$ and $\widetilde{t_2} = \mathcal{H}_{F_p}(\widetilde{k_2}||v_s||i||v_x)$ from c to 1, where $\widetilde{k_1}, \widetilde{k_2}$ are random strings and c is the number of matched index from $\mathcal{L}^{\mathrm{Stp}}$. After that, the random oracle $\mathcal{H}_{H_1}$ is operated in the way of $\widetilde{l} = \mathcal{H}_{H_1}(\widetilde{t_1}||\alpha)$ to find the matched entry, where α is the random token from $\mathcal{L}^{\mathrm{Srch}}$. For each accessed index value (β, γ) in $\mathcal{D}$, another random oracle $\mathcal{H}_{H_2}$ is operated as $\widetilde{R} = \mathcal{H}_{H_2}(\widetilde{t_1}||\alpha) \oplus \beta$ to obtain $\widetilde{R}$ inside, where $\widetilde{R}$ has the same length as the real one. With random string $\widetilde{k_3}$, another token $\widetilde{t_3} = \mathcal{H}_{F_p}(\widetilde{k_3}||\widetilde{t_2}||\gamma)$ is calculated to check whether $\widetilde{t_3}$ lies in the simulated set $\mathcal{E}$. From $\mathcal{L}^{\mathrm{Rpt}}$, $\mathcal{S}$ updates $\widetilde{M_{1,1}} = 1$ in a matrix $\widetilde{M_{q\times q}}$, and adds $(\widetilde{l}, \widetilde{R})$ into result $\widetilde{R_q}$.

For the subsequent queries, if $\mathcal{L}^{\mathrm{Rpt}}$ indicates the query appearing before, $\mathcal{S}$ will select the same tokens and return the result simulated before. Meanwhile, it will update the corresponding symmetric element in matrix to " 1 ". Otherwise, $\mathcal{S}$ will simulate the tokens and result set by following the procedure as the first query. Due to the pseudo-randomness and semantic security of PRF, $\mathcal{A}$ cannot distinguish the simulated tokens and result from the real tokens and result. □

TABLE 2
Time Cost of ABE Algorithm(ms)

Attributes number	Setup cost	KeyGen cost	Encrypt cost	Decrypt cost
2	21.81	11.50	2.99	4.09
4	34.35	18.45	2.86	4.33
8	47.12	33.42	2.76	4.48
16	63.42	64.15	2.59	4.01

8 Evaluation Results

8.1 Prototype Implementation

The primary goal of MedShare is to achieve a secure and reliable medical data sharing system among distributed medical databases. We implemented the system prototype[5] and deployed it on the Ethereum blockchain network $TestRPC$ [56] to evaluate the performance of the whole process. We comprehensively analyzed the overhead of both latency and storage through different database scales and various potential influence factors in each procedure.

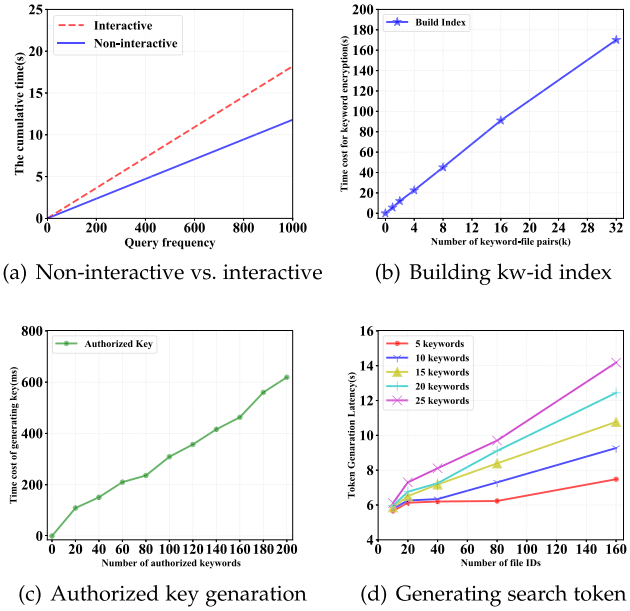
MedShare was implemented in Python, and the Ethereum smart contract was deployed by Solidity.[6] Solidity is an object-oriented programming language designed for implementing functions in Ethereum. Moreover, MedShare interacts with the blockchain platform based on web3. Specifically, we employ HMAC-SHA256 from the python cryptography library for the key-based cryptography functions: F, F_p, and $\mathbf{P}$. The first and second parameters of HMAC are set as the key and input, respectively. For the hash function set $\{H_1, H_2\}$, we choose the standard hash implementation SHA-256 [57]. The experiment was conducted on a desktop computer with Ubuntu 18.04.4 LTS operation system, Intel (R) Core (TM) i7-6700 CPU, 16 GB of RAM. The evaluation data set, consisting of 7.6×10^4 keyword-file ID pairs, was synthesized from the Enron email database.[7] The corresponding file IDs of the keywords were picked randomly from a set of strings. Different numbers of keywords were included in each file, varying from 40 to 400. The various numbers of file IDs and keywords were set in the evaluation.

For the on-chain access control, we adopt the constant-size ABE algorithm proposed in [17] based on the Type A elliptic curve to test its performance. Here, we show its latency overhead with distinct attribute sizes. Table 2 shows the performance when ranging attribute size from 2, 4, 8, 16, where we ran the algorithm ten times, and each time cost was the averaged running time, separately. From this table, we can conclude that the time cost of the *Setup* and *KeyGen* phase grows linearly with the attribute scale while the *Enc* and *Dec* phase features constant computation cost, which is significantly user-friendly.

8.2 Local Performance Evaluation

The data outsourcing process includes system setup and data processing procedures. In the setup phase, the computing cost concentrates on system parameter configuration

5. https://github.com/groupJia/MedShare2021
6. Online at https://solidity.readthedocs.io/en/develop/
7. Online at http://www.cs.cmu.edu/~enron/



(a) Non-interactive vs. interactive (b) Building kw-id index

(c) Authorized key genaration (d) Generating search token

Fig. 4. Evaluation for MedShare off-chain performance.

and EHR preprocessing. We were first required to produce the relevant system parameters for the ABE scheme and RSA. Then, in the data preprocessing procedure, each keyword was mapped to available prime integers for generating search token non-interactively. In MedShare, we design the non-interactive mechanism where the HCs are not required to stay online to produce the search token for each query from users. To demonstrate the advantage of our design, we compared the latency with that from the interactive method [52], where the users need to interact with the HCs before each query. Fig. 4a depicts the comparison of cumulative time overhead between two mechanisms. Note here, we both performed 1000 times query under the same setting that the least frequent keyword contained one file ID. From Fig. 4a, we can see that the interactive mechanism will lead to 6.4s latency when compared to our non-interactive approach. This is because the users have to rely on the HCs to send the search token to them in each query. In contrast, in MedShare, there is only a one-time interaction. After that, the users can produce authorized search tokens independently, without bothering the HCs.

The data processing procedure refers to building the secure index (generating EDindex, BSindex, PTindex) and secret key generation. To evaluate the index generation procedure, we chose approximately 3.2×10^4 keyword-file ID pairs in the plaintext form. We assumed the HCs could fetch the corresponding prime integer of each keyword from the mapping table to cut down the total time overhead. As shown in Fig. 4b, the time cost of generating encrypted index was almost linear with the number of keyword-ID (kw-id) pairs, and the total overhead of all the above pairs were about 171s and 5.4ms per pair. The secret key generation phase in MedShare can be separated into two parts: attributes key and search keyword authorization key. The computation time cost of the former one has been presented in Table 2. The cost of the latter part is shown in Fig. 4c, from which we can see that the overhead was linear with the number of authorized keywords. Note here, this is the

TABLE 3
Local Query Time Cost

Keyword-ID pairs	BuildIndex time(s)	TokenGen time(ms)	Search time(ms)
(1572, 2959)	18.78	18.73	6.21
(2861, 6790)	36.53	35.67	11.73
(4278, 13682)	72.96	44.01	14.71
(7798, 31845)	169.68	61.41	20.42
(12835, 75852)	405.97	87.34	29.09

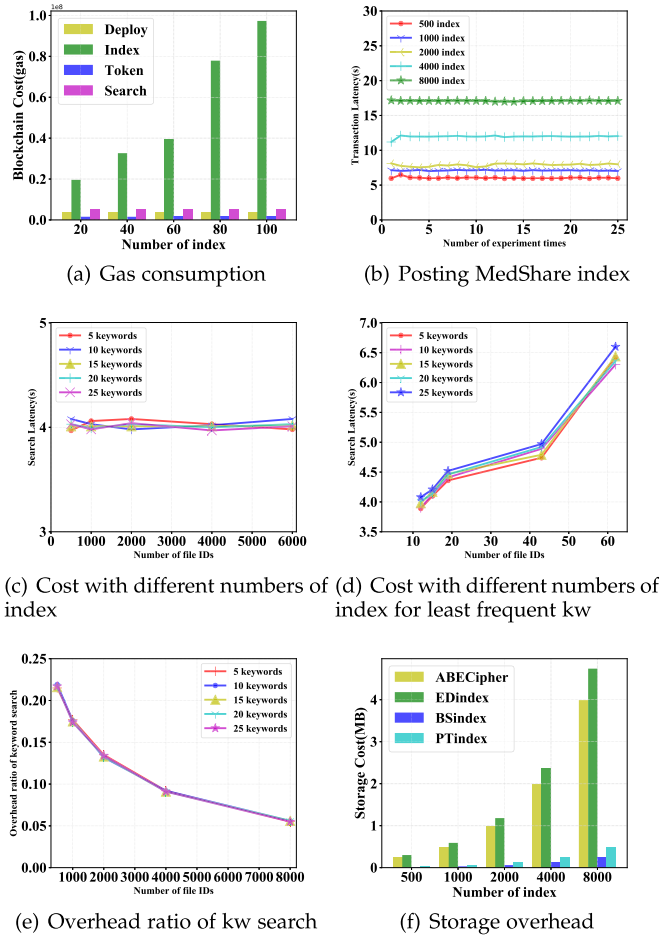
core design of achieving the non-interactive mechanism, which has dramatically optimized the search token production process. It is also worth mentioning that both the index and secret key generation are one-time procedures within an acceptable time cost level. As shown in Table 3, we measured the local time cost of all the procedures. In our design, it just took around 5.3ms to encrypt one pair of keyword-ID. When the search keywords number was set to six, the time cost of the search token was also quite appreciable. Furthermore, merely 30ms were cost to retrieve the search result under the setting of 7.6×10^4 index, which is extremely fast for privacy-preserving applications.

Before performing the boolean search, the users need to generate search tokens. For a set of search keywords in the query, they can be regarded as two parts: the first keyword and the rest. In particular, the first keyword is the least frequent keyword ($sterm$), which means the number of file IDs containing it is the least comparing to the other keywords in the query. Analyzing the core design of the token generation, we can realize that the factors that influence the latency are the number of file IDs of the least frequent keyword and the number of the keyword in the query set. As shown in Fig. 4d, the time cost grew as the number of file IDs of the first keyword increased under the same set of search keywords. For example, when the query set consisted of 20 keywords, the latency boosted from 6.07s to 12.44s as the file number was configured as 10 and 160, respectively. Furthermore, the number of keywords in the query also played a great role in the search token generation. The latency raised from 7.48s to 14.18s when the search keyword changed from 5 to 25 under the same number of files as 160.

8.3 On-Chain Performance Evaluation

After all the preparations, the HCs need to post the EDindex, BSindex, and PTindex on the blockchain platform for boolean search services. To assess the feasibility of MedShare, we measured the gas consumption of implementing the smart contract on Ethereum, including the deployment of smart contracts, posting indexes, uploading the search token, and search procedure. Specifically, the gasPrice is set to 1 Gwei, where 1 Gwei = 10^{-9} Ether. The cost of Ether is computed under the Ethereum gas rules: $etherCost = gasCost \times gasPrice \ / \ 10^9$. As shown in Fig. 5a, deploying our smart contract on Ethereum cost about \$0.74 with an exchange rate of 1 Ether = \$200 USD.[8] Meanwhile, it took \$7.87 USD to post 60 batches of the index, which has confirmed that the capital cost

8. Here is the exchange rate in July, 2020. It fluctuates widely, which is up to 1 Ether = \$2100 USD in Jul, 2021. The gas fees change accordingly.



(a) Gas consumption (b) Posting MedShare index

(c) Cost with different numbers of index (d) Cost with different numbers of index for least frequent kw

(e) Overhead ratio of kw search (f) Storage overhead

Fig. 5. Evaluation for the MedShare on-chain performance.

is not a burden for the HCs. Moreover, the gas cost for deployment, uploading search token, and the search process almost remained constant with certain search capabilities, which is also acceptable for users. For the EHR file update process, the reconstruction of these indexes is off-chain, where the evaluation of update costs are the same as that in Fig. 5a.

In Fig. 5b, we further measured the transaction confirmation latency of posting different numbers of indexes to the smart contract. The average time for mining block is set to 2s. The experiment results show that posting index is considerably fast. It took approximately 5.97s to deploy 500 index entries while merely 17.10s was spent in recording more than 8000 pairs of indexes on the blockchain. Furthermore, we assessed the efficiency of on-chain search and obtaining the result from the blockchain. Fig. 5c demonstrates the latency varied with the number of index entries and search keyword set. From this figure, we can conclude that search latency is not dominated by either the deployed index number or the keyword number. This is because the time complexity of both the key-value match of EDindex and the existence judgment of BSindex is $O(1)$. The result confirms that boolean search complexity is independent of database size. With this characteristic, MedShare has excellent scalability of boolean search. The search cost here is determined by the number of the least frequent keyword and the number of the final result to be recorded on the chain. Taking this into consideration, we set the deployed index entries as 8000 and the number of the least frequent keyword ranging from 10, 20, 40, 60, respectively. Ignoring the impact of transaction confirmation, Fig. 5d shows that the latency increases almost linearly with the file number of the least frequent keyword. This confirms the reality that search complexity is decided by the number of the files containing the least frequent keyword.

To comprehensively evaluate the performance of MedShare, we further compare the overhead ratio of on-chain search by smart contracts. As shown in Fig. 5e, the introduced overhead ratio will be moderated with the expansion of the keyword-ID scale. For instance, the overhead ratio decreased from more than 20% to nearly 5% when the number of index extends from 500 to 8000. The underline reason is that the index generation and uploading will take up the most latency with the data set expansion. In Fig. 5f, we then investigated the storage overhead of different parts of the indexes. The result indicates that the storage cost accumulates linearly with the amount of data set. For example, 0.3MB is taken up with the index number of 500, roughly half of that for encrypting 1000 pairs of the index. Note here, the ABECipher is included in the EDindex. In a nutshell, all the results above show that cryptographic design in MedShare provides an acceptable balance between time latency and storage overhead.

9 Conclusion

In this paper, we propose MedShare, a new blockchain-based EHR sharing system with fine-grained access control. To preserve EHR privacy and search confidentiality, we construct a new design to generate encrypted indexes published to the smart contracts. In MedShare, a constant-size ABE scheme is designed to keep data privacy while achieving fine-grained access control on the blockchain. We present how to build encrypted EHR indexes, generate search tokens, and achieve efficient multi-keyword boolean search on a blockchain platform with reliable search result. To show the security guarantee, we provide a thorough security analysis to illustrate that our proposed scheme can protect the confidentiality of both indexes and search tokens. Finally, a system prototype is implemented and deployed on Ethereum to evaluate the feasibility and efficiency of MedShare. One promising direction of future work is to extend our design to support dynamic on-chain access control authorizations, where the access policy can be changed (revocation) as desired.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 61732022, in part by the Research Grants Council of Hong Kong under GRF projects CityU under Grants 11213920, 11217819, 11217620, and R6021-20F, in part by the Innovation and Technology Commission of Hong Kong under ITF Project ITS/145/19, in part by the Shenzhen Municipality Science and Technology Innovation Commission under Grant SGDX20201103093004019, CityU), in part by the Fundamental Research Funds for the Central Universities under Grant 310421108, and in part by the National Key R&D Program of China under Grant 2017YFB0803002.

REFERENCES

[1] J. J. Hathaliya and S. Tanwar, "An exhaustive survey on security and privacy issues in healthcare 4.0," *Comput. Commun.*, vol. 153, pp. 311–335, 2020.

[2] Y. Liu, Z. Ma, X. Liu, S. Ma, and K. Ren, "Privacy-preserving object detection for medical images with faster R-CNN," *IEEE Trans. Inf. Forensics Secur.*, to be published, doi: 10.1109/TIFS.2019.2946476.

[3] W. Lin, W. Dou, Z. Zhou, and C. Liu, "A cloud-based framework for home-diagnosis service over big medical data," *J. Syst. Softw.*, vol. 102, pp. 192–206, 2015.

[4] J. Liu, X. Huang, and J. K. Liu, "Secure sharing of personal health records in cloud computing: Ciphertext-policy attribute-based signcryption," *Future Gener. Comput. Syst.*, vol. 52, pp. 67–76, 2015.

[5] H. Li, Y. Yang, Y. Dai, J. Bai, S. Yu, and Y. Xiang, "Achieving secure and efficient dynamic searchable symmetric encryption over medical cloud data," *IEEE Trans. Cloud Comput.*, vol. 8, no. 2, pp. 484–494, Apr.–Jun. 2020.

[6] L. Xu, S. Sun, X. Yuan, J. K. Liu, C. Zuo, and C. Xu, "Enabling authorized encrypted search for multi-authority medical databases," *IEEE Trans. Emerg. Top. Comput.*, vol. 9, no. 1, pp. 534–546, Jan.–Mar. 2021.

[7] W. J. Gordon and C. Catalini, "Blockchain technology for healthcare: Facilitating the transition to patient-driven interoperability," *Comput. Struct. Biotechnol. J.*, vol. 16, pp. 224–230, 2018.

[8] Y. Zhuang, L. R. Sheets, Y.-W. Chen, Z.-Y. Shae, J. J. Tsai, and C.-R. Shyu, "A patient-centric health information exchange framework using blockchain technology," *IEEE J. Biomed. Health Inform.*, vol. 24, no. 8, pp. 2169–2176, Aug. 2020.

[9] The Bitcoin Project, 2009. [Online]. Available: https://bitcoin.org/en/

[10] A. H. Mayer, C. A. da Costa, and R. d. R. Righi, "Electronic health records in a blockchain: A systematic review," *Health Inform. J.*, vol. 26, no. 2, pp. 1273–1288, 2020.

[11] P. Zhang, J. White, D. C. Schmidt, G. Lenz, and S. T. Rosenbloom, "Fhirchain: Applying blockchain to securely and scalably share clinical data," *Comput. Struct. Biotechnol. J.*, vol. 16, pp. 267–278, 2018.

[12] Q. Xia, E. B. Sifah, K. O. Asamoah, J. Gao, X. Du, and M. Guizani, "MeDShare: Trust-less medical data sharing among cloud service providers via blockchain," *IEEE Access*, vol. 5, pp. 14 757–14 767, 2017.

[13] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symp. Secur. Privacy*, 2007, pp. 321–334.

[14] K. Yang and X. Jia, "Expressive, efficient, and revocable data access control for multi-authority cloud storage," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 7, pp. 1735–1744, Jul. 2014.

[15] D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M.-C. Roşu, and M. Steiner, "Highly-scalable searchable symmetric encryption with support for boolean queries," in *Proc. Annu. Cryptol. Conf.*, 2013, pp. 353–373.

[16] S.-F. Sun, J. K. Liu, A. Sakzad, R. Steinfeld, and T. H. Yuen, "An efficient non-interactive multi-client searchable encryption with support for boolean queries," in *Proc. Eur. Symp. Res. Comput. Secur.*, 2016, pp. 154–172.

[17] Y. Zhang, D. Zheng, X. Chen, J. Li, and H. Li, "Computationally efficient ciphertext-policy attribute-based encryption with constant-size ciphertexts," in *Proc. Int. Conf. Provable Secur.*, 2014, pp. 259–273.

[18] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2005, pp. 457–473.

[19] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. Proc. 13th ACM Conf. Comput. Commun. Secur.*, 2006, pp. 89–98.

[20] A. Lewko and B. Waters, "Decentralizing attribute-based encryption," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2011, pp. 568–588.

[21] Y. Miao, J. Ma, X. Liu, X. Li, Q. Jiang, and J. Zhang, "Attribute-based keyword search over hierarchical data in cloud computing," *IEEE Trans. Serv. Comput.*, vol. 13, no. 6, pp. 985–998, Nov./Dec. 2020.

[22] A. Michalas, "The lord of the shares: Combining attribute-based encryption and searchable encryption for flexible data sharing," in *Proc. 34th ACM/SIGAPP Symp. Appl. Comput.*, 2019, pp. 146–155.

[23] J. Li, X. Chen, J. Li, C. Jia, J. Ma, and W. Lou, "Fine-grained access control system based on outsourced attribute-based encryption," in *Proc. Eur. Symp. Res. Comput. Secur.*, 2013, pp. 592–609.

[24] S. Wang, J. Zhou, J. K. Liu, J. Yu, J. Chen, and W. Xie, "An efficient file hierarchy attribute-based encryption scheme in cloud computing," *IEEE Trans. Inf. Forensics Secur.*, vol. 11, no. 6, pp. 1265–1277, Jun. 2016.

[25] A. Ge, R. Zhang, C. Chen, C. Ma, and Z. Zhang, "Threshold Ciphertext policy attribute-based encryption with constant size Ciphertexts," in *Proc. Australasia Conf. Inf. Secur. Privacy*, 2012, pp. 336–349.

[26] M. Wang, Y. Miao, Y. Guo, C. Wang, H. Huang, and X. Jia, "Attribute-based encrypted search for multi-owner and multi-user model," in *Proc. ICC IEEE Int. Conf. Commun.*, 2021, pp. 1–7.

[27] S.-F. Sun *et al.*, "Practical backward-secure searchable encryption from symmetric puncturable encryption," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2018, pp. 763–780.

[28] A. Kiayias, O. Oksuz, A. Russell, Q. Tang, and B. Wang, "Efficient encrypted keyword search for multi-user data sharing," in *Proc. Eur. Symp. Res. Comput. Secur*, 2016, pp. 173–195.

[29] Y. Yang, X. Liu, and R. H. Deng, "Multi-user multi-keyword rank search over encrypted data in arbitrary language," *IEEE Trans. Dependable Secure Comput.*, vol. 17, no. 2, pp. 320–334, Mar.–Apr. 2017.

[30] J. Wang, X. Chen, S.-F. Sun, J. K. Liu, M. H. Au, and Z.-H. Zhan, "Towards efficient verifiable conjunctive keyword search for large encrypted database," in *Proc. Eur. Symp. Res. Comput. Secur.*, 2018, pp. 83–100.

[31] X. Yuan, X. Yuan, Y. Zhang, B. Li, and C. Wang, "Enabling encrypted boolean queries in geographically distributed databases," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 3, pp. 634–646, Mar. 2020.

[32] S.-F. Sun *et al.*, "Non-interactive multi-client searchable encryption: Realization and implementation," *IEEE Trans. Dependable Secure Comput.*, to be published, doi: 10.1109/TDSC.2020.2973633.

[33] S. Kamara and T. Moataz, "Boolean searchable symmetric encryption with worst-case sub-linear complexity," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2017, pp. 94–124.

[34] S. Jiang *et al.*, "Privacy-preserving and efficient multi-keyword search over encrypted data on blockchain," in *Proc. IEEE Int. Conf. Blockchain*, 2019, pp. 405–410.

[35] K. Zhang, M. Wen, R. Lu, and K. Chen, "Multi-client sub-linear boolean keyword searching for encrypted cloud storage with owner-enforced authorization," *IEEE Trans. Dependable Secure Comput.*, to be published, doi: 10.1109/TDSC.2020.2968425.

[36] S. Patranabis and D. Mukhopadhyay, "Forward and backward private conjunctive searchable symmetric encryption," in *Proc. NDSS Symp.*, 2021.

[37] L. Xu, X. Yuan, C. Wang, Q. Wang, and C. Xu, "Hardening database padding for searchable encryption," in *Proc. Conf. Comput. Commun.*, 2019, pp. 2503–2511.

[38] E. Stefanov *et al.*, "Path ORAM: An extremely simple oblivious RAM protocol," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2013, pp. 299–310.

[39] M. Li *et al.*, "CrowdBC: A blockchain-based decentralized framework for crowdsourcing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 6, pp. 1251–1266, Jun. 2019.

[40] Y. Peng, M. Du, F. Li, R. Cheng, and D. Song, "FalconDB: Blockchain-based collaborative database," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2020, pp. 637–652.

[41] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman, "MedRec: Using blockchain for medical data access and permission management," in *Proc. 2nd Int. Conf. Open Big Data*, 2016, pp. 25–30.

[42] M. Laurent, N. Kaaniche, C. Le, and M. Vander Plaetse , "A blockchain-based access control scheme," in *Proc. 15th Int. Conf. Secur. Cryptography*, 2018, pp. 168–176.

[43] S. Hu, C. Cai, Q. Wang, C. Wang, X. Luo, and K. Ren, "Searching an encrypted cloud meets blockchain: A decentralized, reliable and fair realization," in *Proc. IEEE Conf. Comput. Commun.*, 2018, pp. 792–800.

[44] C. Cai, J. Weng, X. Yuan, and C. Wang, "Enabling reliable keyword search in encrypted decentralized storage with fairness," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 1, pp. 131–144, Jan./Feb. 2021.

[45] S. Wang, D. Zhang, and Y. Zhang, "Blockchain-based personal health records sharing scheme with data integrity verifiable," *IEEE Access*, vol. 7, pp. 102887–102901, 2019.

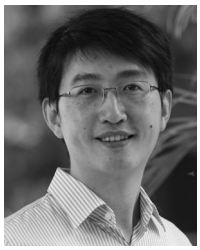
[46] J. Xu *et al.*, "Healthchain: A blockchain-based privacy preserving scheme for large-scale health data," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8770–8781, Oct. 2019.

[47] S. Jiang, J. Cao, H. Wu, Y. Yang, M. Ma, and J. He, "BlocHIE: A BLOCkchain-based platform for healthcare information exchange," in *Proc. IEEE Int. Conf. Smart Comput.*, 2018, pp. 49–56.
[48] L. Chen, W.-K. Lee, C.-C. Chang, K.-K. R. Choo, and N. Zhang, "Blockchain based searchable encryption for electronic health record sharing," *Future Gener. Comput. Syst.*, vol. 95, pp. 420–429, 2019.
[49] The Ethereum Project, 2014. [Online]. Available: https://ethereum.org
[50] Y. Guo, H. Xie, Y. Miao, C. Wang, and X. Jia, "FedCrowd: A federated and privacy-preserving crowdsourcing platform on blockchain," *IEEE Trans. Serv. Comput.*, to be published, doi: 10.1109/TSC.2020.3031061.
[51] D. D. F. Maesa, P. Mori, and L. Ricci, "Blockchain based access control," in *Proc. IFIP Int. Conf. Distrib. Appl. Interoperable Syst.*, 2017, pp. 206–220.
[52] S. Jarecki, C. Jutla, H. Krawczyk, M. Rosu, and M. Steiner, "Outsourced symmetric private information retrieval," in *Proc. ACM Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2013, pp. 875–888.
[53] H. Yu, I. Nikolić, R. Hou, and P. Saxena, "Ohie: Blockchain scaling made simple," in *Proc. IEEE Symp. Secur. Privacy*, 2020, pp. 90–105.
[54] C. Li *et al.*, "A decentralized blockchain with high throughput and fast confirmation," in *Proc. USENIX Annu. Tech. Conf.*, 2020, pp. 515–528.
[55] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2012, pp. 965–976.
[56] Npmcommunity, "Ethereumjs-testrpc," 2017. [Online]. Available: https://www.npmjs.com/package/ethereumjs-testrpc
[57] *Secure Hash Standard (SHS), Std.* no. 180–4, Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Oct. 2008.

Mingyue Wang received the BE degree from the Department of Computer Science and Technology, Harbin Engineering University in 2018. She is currently working toward the PhD degree with the Department of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, and the City University of Hong Kong. Her research interests include applied cryptography and blockchain technology.

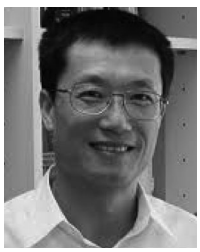
Yu Guo (Member, IEEE) received the BE degree in software engineering from Northeastern University in 2013, and the MSc degree in electronic commerce and the PhD degree in computer science from the City University of Hong Kong, in 2014 and 2019, respectively. He is currently a lecturer with the School of Artificial Intelligence, Beijing Normal University. He was a postdoctor and a research fellow with the City University of Hong Kong. His research interests include cloud computing security, network security, privacy-preserving data processing, and blockchain technology.

Chen Zhang received the BE degree in network engineering from the Harbin University of Science and Technology in 2017 and the ME degree in computer technology from the Harbin Institute of Technology, Shenzhen, in 2019. She is currently working toward the PhD degree with the Department of Computer Science, City University of Hong Kong. Her research interests include mobile edge computing and blockchain.



Cong Wang (Fellow, IEEE) is currently a professor with the Department of Computer Science, City University of Hong Kong. His research interests include data and network security, blockchain and decentralized applications, and privacy-enhancing technologies. Since 2017, he has been one of the founding members of the Young Academy of Sciences of Hong Kong and was conferred the RGC Research Fellow in 2021. He was the recipient of the Outstanding Researcher Award (junior faculty) in 2019, the Outstanding Supervisor Award in 2017, and the President's awards in 2019 and 2016 from City University of Hong Kong. He was the co-recipient of the Best Paper Award of IEEE ICDCS 2020, ICPADS 2018, MSN 2015, the Best Student Paper Award of IEEE ICDCS 2017, and the IEEE INFOCOM Test of Time Paper Award 2020. His research has been supported by multiple government research fund agencies, including the National Natural Science Foundation of China, Hong Kong Research Grants Council, and the Hong Kong Innovation and Technology Commission. He was an associate editor for the *IEEE Transactions on Dependable and Secure Computing*, *IEEE Transactions on Services Computing*, *IEEE Internet of Things Journal*, *IEEE Networking Letters*, and *Journal of Blockchain Research*, and a TPC co-chair for a number of IEEE conferences and workshops. He is a member of the ACM.

Hejiao Huang (Member, IEEE) received the graduation degree from the City University of Hong Kong and the PhD degree in computer science in 2004. She is currently a professor with the Harbin Institute of Technology, Shenzhen, and was an invited professor at INRIA, France. Her research interests include cloud computing, trustworthy computing, and formal methods for system design and wireless networks.



Xiaohua Jia (Fellow, IEEE) received the BSc and MEng degrees from the University of Science and Technology of China in 1984 and 1987, respectively, and the DSc degree in information science from the University of Tokyo, in 1991. He is currently an adjunct with the Harbin Institute of Technology, Shenzhen. He is currently the chair professor with the Department of Computer Science, City University of Hong Kong. His research interests include cloud computing and distributed systems, computer networks, wireless sensor networks, and mobile wireless networks. He is the editor of the *IEEE Transactions on Parallel and Distributed Systems from 2006 to 2009, Wireless Networks, Journal of World Wide Web*, and *Journal of Combinatorial Optimization*. He is the general chair of ACM MobiHoc 2008, the TPC co-chair of IEEE MASS 2009, the area chair of IEEE INFOCOM 2010, the TPC co-chair of IEEE GlobeCom 2010-Ad Hoc and Sensor Networking Symposium, and the panel co-chair of IEEE INFOCOM 2011. He is a fellow of the IEEE Computer Society.