Hindawi
Security and Communication Networks
Volume 2022, Article ID 9780188, 15 pages
https://doi.org/10.1155/2022/9780188

Research Article

A Blockchain-Based Normalized Searchable Encryption System for Medical Data

Qin Liu,[1,2] Yun Lian Liu,[1,3] Min Luo ,[1,2] Cong Peng,[1] and Xuan Xiao[4]

[1]*The School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China*
[2]*The Shandong Provincial Key Laboratory of Computer Networks, Qilu University of Technology (Shandong Academy of Sciences), Jinan 250014, China*
[3]*The Shanghai Key Laboratory of Privacy-Preserving Computation, Matrix-Elements Technologies, Shanghai 201204, China*
[4]*The Department of Ophthalmology, Renmin Hospital of Wuhan University, Wuhan, Hubei, China*

Correspondence should be addressed to Min Luo; mluo@whu.edu.cn

Received 15 May 2022; Accepted 12 July 2022; Published 23 August 2022

Academic Editor: Jie Cui

As COVID-19 continues to spread around the world, the healthcare industry has accelerated the transformation to digital healthcare services. In the era of big data, many hospitals prefer to use remote cloud servers to store and manage massive electronic medical data. However, cloud-assisted medical data systems cannot guarantee the confidentiality, integrity, and availability of data. Searchable encryption can effectively address the above challenges by enabling data search on the ciphertext, which achieves the availability of medical data while ensuring data security and privacy. However, the search server may return mismatched search results due to economic interests or single points of failure. Blockchain is a decentralized computing paradigm with public verifiability, which provides an efficient solution to this problem. However, the existing blockchain-based searchable encryption solutions do not consider the flexible search function of multiple users and the restriction of encrypted data for medical scenarios. Therefore, we propose a blockchain-based multiuser normalized searchable encryption (BNSE) scheme and design a blockchain-based normalized searchable encryption system for medical data (BNSEM) based on the scheme. To verify the practicality of the system, we evaluate the performance from both theoretical and experimental aspects.

1. Introduction

With the rapid spread of COVID-19 around the world, the healthcare industry has accelerated the shift to digital healthcare services [1, 2]. In the era of big data, many hospitals prefer to use remote cloud servers to store and manage huge amounts of electronic medical data. However, due to the inherent properties of the cloud such as centralization and openness, cloud-assisted medical data systems will face new privacy and security challenges [3]. Firstly, medical data outsourced to cloud servers may be accessed or tampered with by unauthorized users. The confidentiality and integrity of the medical data cannot be guaranteed. Secondly, centralized cloud servers may suffer from a single point of failure, which will result in the unavailability of medical data. Although the traditional encryption technology can ensure security, it is difficult to take into account the availability of outsourced medical data.

Searchable encryption (SE) is a critical cryptographic technique to achieve the availability of data while ensuring security and privacy, which enables users to search ciphertext data [4]. In a searchable encryption scheme, the data owner uploads the encrypted data to the cloud server. Then, the user needs to construct a trapdoor and submit it to the cloud server to search for data containing the target keywords. In most cases, the server is regarded as a semi-honest third-party entity [5, 6]; i.e., the server will perform the search operation correctly according to the protocol. However, in practical scenarios, the server may return mismatched search results due to economic interest or single point of failure.

Blockchain is a decentralized computing paradigm with public verifiability and tamper-proof features [7]. Applying blockchain to searchable encryption can effectively solve the problem of untrustworthy search results from the centralized servers [8–10]. Smart contracts deployed in the blockchain can perform search functions instead of third-party servers and automatically execute search protocols based on trigger conditions to produce correct results. In addition, blockchain nodes record transaction results in an immutable ledger, which guarantees the integrity of the results and eliminates the need for further validation of the results. Even if one or more nodes fail or are corrupted by malicious adversaries, the correctness of the results will not be affected due to the fault tolerance of the blockchain.

In the blockchain-based searchable system, data and search structure are stored in a ciphertext state, and thus, the legality of the data in the system cannot be guaranteed. In the medical scenario, the necessary supervision of encrypted medical data is needed to ensure the legality of medical data. For example, supervisors need to filter the search requests that contain illegal keywords to prevent the spread of false medical information. In addition, supervisors are supposed to check the legality of ciphertext data in the remote cloud when they suspect illegal data or when a user files a complaint with the supervisors. The controllability of medical data is key to maintaining a stable healthcare system, yet there is a lack of research related to the supervision of ciphertext data.

1.1. Motivation and Contributions. Inspired by the work [11], a blockchain-based searchable public key encryption with forward and backward privacy can be used to design a searchable encryption system for medical data. In medical scenarios, multiuser search functions and dynamic updates for authorized users need to be supported with data sharing requirements. In addition, to ensure the stability of the medical searchable encryption system, the ciphertext data in the system need to be legally supervised. As for the blockchain platform, the consortium chain hyperledger fabric is a good option considering the practicality and privacy. Based on the above analysis and practical application scenarios of the medical data, we propose a blockchain-based multiuser searchable encryption scheme supporting supervision and design a searchable encryption system for medical data based on the scheme. The main contributions of this study are as follows:

(i) We propose a blockchain-based multiuser normalized searchable encryption (BNSE) scheme, which achieves efficient retrieval of ciphertext data in multiuser scenarios and supports the supervision on the ciphertext data

(ii) We design a blockchain-based normalized searchable encryption system for medical data (BNSEM) based on the above scheme, which realizes the application of retrieval on the encrypted medical data

(iii) We evaluate the theoretical performance of the scheme and test the practical performance of the system to verify the availability

1.2. Organization. In Section 2, we review the existing research work related to the security and functionality of searchable encryption. In Section 3, we introduce the blockchain technology and broadcast encryption, as well as security definition. In Section 4, we describe the specific construction of our proposed scheme BNSE and prove its security. In Section 5, we present the design of our system BNSEM. In Section 6, we provide the security analysis of BNSEM. In Section 7, we conduct a performance evaluation of the system. Section 8 makes a conclusion of this study.

2. Related Work

In 2000, Song et al. [4] proposed the first SE scheme, which is a noninteractive single-keyword search scheme. The drawback of the scheme is that it is extremely inefficient when the number of documents is large. However, this pioneering work still greatly contributed to the research and development of searchable encryption. Later, many works [12, 13] focus on designing efficient security mechanisms to enhance the security. Meanwhile, some works also introduce searchable encryption schemes for functional extensions, including the multiuser SE scheme [14] and the dynamic SE scheme [15].

To balance security and efficiency, a practical SE scheme will leak some information to the adversary. However, the information leakage attack undermines the security of SE schemes [16]. Adaptive leakage exploit attacks have brought more attention to forward privacy [17]. Song et al. [12] proposed two schemes FAST and FASTIO, both of which have forward privacy. In addition, Bost et al. [6] presented a formal definition of backward privacy, and backward privacy gradually became a major security property of interest. Chamani et al. [18] proposed improvements in various aspects of performance to the work [6].

To avoid the problem of key management and distribution restrictions prevalent in symmetric searchable encryption (SSE) schemes, Boneh et al. [19] proposed the first searchable public key encryption (SPE) scheme, which is a noninteractive single-user search scheme. However, a significant limitation of SPE is that it contains a large number of time-consuming operations, such as bilinear pairs and exponential operations. In 2020, Chen et al. [11] proposed a lightweight SPE scheme with search performance close to the efficient SSE. However, the scheme does not implement multiuser search and cannot share data in multiuser scenarios.

From a functional point of view, most of the current research efforts focus on symmetric searchable encryption schemes that support only single-user search mode; i.e., the data user is the data owner. The few SSE schemes that support multiuser search also require the owner to calculate a search trapdoor [20] online. Multiuser searchable encryption (MUSE) [21] is a significant research content of SE

with practical research significance. In MUSE, a data owner uploads data to a cloud server and wants to share the data with multiple users. Attribute-based searchable encryption (ABSE) [22] can manage the retrieval of ciphertext data in a multiuser scenario, but it is computationally inefficient and lacks practicality.

Broadcast encryption (BE) [23] enables multiuser data sharing and is suitable for scenarios where data users are relatively fixed. Liu et al. [24] designed a multiuser searchable encryption scheme based on a single-user system prototype and inherited the functionality of adding, modifying, and deleting documents from the original dynamic scheme. However, scheme [24] requires online search trapdoor generation and multiple rounds of client-server interaction, which increases the communication overhead. Later, Liu et al. [25] combined public key authenticated encryption supporting keyword search with broadcast encryption BE and proposed a broadcast authenticated encryption primitive BAEKS supporting keyword search, while the scheme reaches a performance bottleneck when the number of users increases to a certain number.

In most existing schemes, the search server is regarded as an honest third party that performs the prescribed search protocol [26]. However, the search server may be a malicious third party that returns partial or even mismatched search results due to profit or random failures. The main reason for these problems is that centralized servers have complete control over the data and execute the protocols independently without supervision. In view of this, blockchain technology [7], a decentralized computing paradigm with public verifiability and invariance characteristics, combined with searchable encryption [14] can effectively solve the problem of untrustworthy third-party search results.

There are two ways to combine blockchain with searchable encryption, one of which is to use the blockchain for storing credentials and the other is to use the blockchain's smart contract to perform the search function. The first approach still follows the traditional server-side search by storing the transaction credentials on the blockchain [27]. Cai et al. [8] designed a dynamic and efficient searchable encryption scheme using blockchain. Tang [28] extends searchable encryption by saving essential messages on the blockchain and the scheme performs only a small number of operations on the blockchain, thus reducing the burden on the blockchain. When there are disputes and controversies, the misconduct of participants can be revealed through transactions on the blockchain. However, using the blockchain to store credentials still does not prevent the malicious behavior of servers.

Therefore, researchers have also proposed an alternative construction method to design smart contracts that include search functions instead of cloud servers to perform keyword search operation [14]. Chen et al. [29] used electronic medical record EHR file indexes to construct complex logical structures and store them on a blockchain so that data users can search the file indexes using these logical expressions. Hu et al. [20] enabled users to search private databases in a blockchain environment and implement dynamic access control for searches. However, all of the above schemes outsource complex operations or encrypted data to the blockchain, which greatly degrades the performance of the system. Chen et al. [11] designed a blockchain-based searchable public key encryption scheme with only lightweight hash operations.

3. Preliminaries

In this section, we introduce the blockchain technology, broadcast encryption, system model, security definition, and design goals.

3.1. Blockchain Technology. In 2008, blockchain technology received widespread attention following the publication of the Bitcoin white paper [7]. Blockchain provides a distributed, immutable, secure, transparent, and auditable ledger. The blocks in a blockchain store transactions at a specific time, and their hash values are recorded by a Merkle tree. The transaction data on the blockchain are shared in a P2P (peer-to-peer) network, and the security of the transaction data is ensured by cryptographic primitives (Merkle tree, asymmetric encryption, and digital signatures).

Since blockchain operates on a P2P network, a P2P network including a number of blockchain nodes (peer nodes, orderer nodes, etc.) needs to be created before deploying a blockchain platform. Each node provides two keys that can be used for encryption and signature. When a transaction is initiated, one node signs the transaction and broadcasts it to other peer nodes. When another node receives the signed transaction, it needs to verify the validity of the transaction before broadcasting it. The peer nodes (also known as miners) collect enough signatures of this transaction to pack it into a block and store it on the blockchain after passing consensus.

Smart contract: a smart contract contains a set of rules and logic, which is a decentralized, information-sharable program code deployed on the blockchain. The parties involve in signing a contract agree on the content of the smart contract and deploy it on the blockchain, which can automate the execution of the contract without relying on any third authority [30]. Smart contracts run automatically once started without the intervention of any contract signatory.

3.2. Broadcast Encryption. A public key broadcast encryption scheme consists of four algorithms, namely system setup (Setup), key generation (KeyGen), encryption (Encrypt), and decryption (Decrypt), defined as follows:

(i) $\text{Setup}(\kappa) \longrightarrow (N, \text{EK})$: with the security parameter κ as input, the maximum capacity N of the broadcast receiver group and the initial encryption key list EK are output.

(ii) $\text{KeyGen}(\kappa, \text{EK}) \longrightarrow (\text{pk}, \text{sk})$: with the security parameter κ and the encryption key list EK as input, the user's public-private key pair (pk, sk) is output and the public key pk is added to the key list EK.

(iii) $\text{Encrypt}(\text{EK}, S, m) \longrightarrow C_m$: the algorithm takes a subset of users $S \subseteq \{1, 2, \ldots, n\}$, encryption key list

EK, and a plaintext message m to be broadcast as input. The public keys $\mathrm{PK} = \{\mathrm{pk}_1, \ldots, \mathrm{pk}_n\}$ corresponding to the users in the subset S from the encryption key list EK are selected. The broadcast ciphertext C_m of the message m under encryption using the key set PK is output. Note that the broadcast ciphertext can only be correctly decrypted by the receiver in S.

(iv) $\mathrm{Decrypt}(\mathrm{sk}, C_m) \longrightarrow m$: taking user's private key sk and broadcast ciphertext C_m as input, if user $u_i \in S$ who has the private key sk, then the user u_i can use his private key sk to decrypt the broadcast ciphertext C_m and output the broadcast message m.

3.3. System Model. The system model of our BMNSE scheme is shown in Figure 1. It consists of six entities: trusted institute (TI), cloud server (CS), blockchain (BC), data owner (DO), data user (DU), and supervisor (SUP). Although the TI and SUP are not involved in the main process of data search, they are still two indispensable entities that play an important role in the execution of the scheme and the maintenance of the ecosystem. Before running scheme, TI first generates the parameters required for system initialization and issues public key certificates for users who join the system, and TI is offline the rest of the time.

After the initialization is completed, the program needs to perform five main steps, which are described as follows:

(1) *Encrypt File*. The DO first encrypts the data file using a symmetric encryption algorithm and then encrypts the symmetric key using a public key cryptography algorithm. Finally, DO uploads the ciphertext to CS.

(2) *Generate Searchable Encrypted Data Structures.* The DO extracts keyword-index pairs from files and generates searchable encrypted data structures. Then, DO uploads the structures to BC.

(3) *Search for the Files that Contain the Target Keyword.* DU generates a search trapdoor containing the target keyword and then sends the search request containing the trapdoor to a nearby blockchain node. The search request triggers the search process of the smart contract, which then returns the index of all matching encrypted files.

(4) *Access the Files*. The DU first decrypts the encrypted file index returned by the smart contract in step 3 and then accesses the data in the CS after getting the plaintext index.

(5) *Return the Encrypted Data.* Based on the file indexes submitted by DU, CS returns the corresponding files.

To ensure the legitimacy of the transactional data in the program, the necessary supervision of data cryptography by SUP is required. SUP has two main tasks: first, carrying out periodic audits of cryptographic data stored on CS, and second, scrutinizing the search requests of DU. The purpose of cryptographic data audit is to detect data files that contain illegal or sensitive keywords, timely revoke illegal files hosted on CS, and alert, warn, or punish the corresponding DU. The purpose of scrutinizing search requests is to monitor keyword search requests sent by DOs to the BC in real time and to intercept and warn the noncompliant search requests.

Based on the above system model, the following eight algorithms are defined in our scheme:

(i) $\mathrm{Setup}(\kappa) \longrightarrow \mathrm{Param}$: it is executed by TI and takes the security parameter κ as input and the system public parameter Param as output.

(ii) $\mathrm{KeyGen}(\mathrm{Param}) \longrightarrow (Q_u, \mathrm{d}_u)$: it is executed by TI and takes the public parameter Param as input and outputs the user's public-private key pair (Q_u, d_u).

(iii) $\mathrm{Encrypt}(\mathrm{Param}, \{Q_{u_i}\}, DB, \Sigma) \longrightarrow \mathrm{EDB}$: this algorithm is executed by DO. The input parameters contain the system public parameters Param, the public keys $\{Q_{u_i}\}$ of the authorized DUs, the database DB, and an empty mapping Σ. The algorithm outputs the searchable encrypted database EDB and the initialized mapping Σ.

(iv) $\mathrm{Update}(\mathrm{Param}, \{Q_{u_{i'}}\}, \overrightarrow{Z}, w_k, r, s) \longrightarrow \overrightarrow{Z'}$: this algorithm is executed by DO with input parameters including system parameter Param, public key set $\{Q_{u_{i_t}}\}$ of the users to be authorized, original broadcast cipher $\overrightarrow{Z}$, target keyword w_k, and secret values r, s saved by DO, where r is the secret value associated with version information and s is the secret value involved in the encryption calculation. The algorithm outputs the updated broadcast cipher $\overrightarrow{Z'}$.

(v) $\mathrm{Trapdoor}(\mathrm{Param}, d_{u_i}, w_k) \longrightarrow T_{w_k}$: this algorithm is executed by DU with the input of public parameter Param, authorized user's private key d_{u_i} and target keyword w_k, and the output of search trapdoor T_{w_k}.

(vi) $\mathrm{Search}(\mathrm{Param}, T_{w_k}, \mathrm{EDB}) \longrightarrow \mathbf{RS}(w_k)$: it is automatically executed by the smart contract, takes the system parameter Param, the search trapdoor T_{w_k} for the keyword w_k, and the encrypted database EDB as input, and outputs the matched search results $\mathbf{RS}(w_k)$.

(vii) $\mathrm{Decrypt}(\mathrm{Param}, \mathrm{d}_{u_i}, w_k, \mathbf{RS}(w_k)) \longrightarrow \{\mathrm{ind}_{w_k}\}$: this algorithm is executed by DU, which takes the private key d_{u_i} and the search result set $\mathbf{RS}(w_k)$ as inputs and outputs the decrypted file index set $\{\mathrm{ind}_{w_k}\}$.

(viii) $\mathrm{Supervise}(\mathrm{Param}, \{Q_{u_i}\}, \mathrm{d}_{\mathrm{sup}}, w_k) \longrightarrow T_{w_k}$: this algorithm is executed by SUP with input parameters including system parameter Param, public keys $\{Q_{u_{i_t}}\}$ of authorized users, illegal or sensitive keyword w_k and private key $\mathrm{d}_{\mathrm{sup}}$ of supervisor, and outputs search trapdoor T_{w_k} of sensitive words.

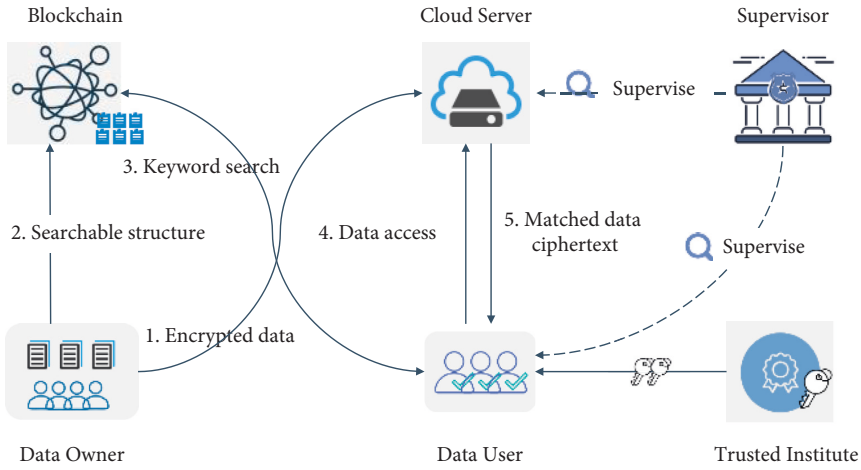


Figure 1: System model of BNSE.

3.4. Security Definition. Similar to [12], we demonstrate the confidentiality of our BMNSE scheme with a real/ideal simulation paradigm. To achieve higher operational efficiency, searchable encryption schemes will disclose some information to the server. The leakage information of our scheme is described by the leakage function $\mathscr{L} = \{\mathscr{L}_{\text{Setup}}, \mathscr{L}_{\text{KeyGen}}, \mathscr{L}_{\text{Encrypt}}, \mathscr{L}_{\text{Trapdoor}}, \mathscr{L}_{\text{Search}}, \mathscr{L}_{\text{Supervise}}\}$. The nonformal definition of the confidentiality of searchable encryption scheme is that no information about the database should be revealed other than the information leaked in the leak function $\mathscr{L}$. The formal definition of confidentiality can be presented by a reality/ideal simulation paradigm containing the game $\mathbf{Real}^{\text{SE}}$ and $\mathbf{Ideal}^{\text{SE}}$.

Definition 1. Let Π = (Setup, KeyGen, Encrypt, Trapdoor, Search, Supervise) denote the BMNSE scheme, $\mathscr{A}$ denote the adversary, and S be a simulator with a leakage function $\mathscr{L} = \{\mathscr{L}_{\text{Setup}}, \mathscr{L}_{\text{KeyGen}}, \mathscr{L}_{\text{Encrypt}}, \mathscr{L}_{\text{Trapdoor}}, \mathscr{L}_{\text{Search}}, \mathscr{L}_{\text{Supervise}}\}$ as an parameter. The following two probabilistic game experiments are defined:

(i) $\mathbf{Real}^{\Pi}_{\mathscr{A}}(\kappa)$: the game runs the system setup algorithm Setup (κ) to generate system parameters Param and the key generation algorithm KeyGen (Param) to generate the user's public-private key pair (Q_u, d_u). The game publishes the public message (Param, Q_u) and keeps the private key d_u secretly. Then, the adversary $\mathscr{A}$ selects a database DB and performs an encrypted query based on the information (Param, Q_{u_i}). Next, the game runs the encryption algorithm $\mathbf{Encrypt}(\text{Param}, Q_u, DB, \Sigma = \text{EDB})$ and returns the encrypted database EDB to $\mathscr{A}$. $\mathscr{A}$ chooses a keyword w_k for the trapdoor query, and the game runs the trapdoor generation algorithm $\mathbf{Trapdoor}(Param, \{d_{u_i}\}, w_k) = T_{w_k}$ and returns the trapdoor T_{w_k} to $\mathscr{A}$. Then, $\mathscr{A}$ selects a trapdoor T_{w_k} for the search query and the game runs the search algorithm $\mathbf{Search}(\text{Param}, T_{w_k}, \text{EDB}) = \mathbf{RS}$ and returns the result set **RS** to $\mathscr{A}$. The adversary $\mathscr{A}$ can repeat the above steps several times and finally output a bit $b \in \{0, 1\}$.

(ii) $\mathbf{Ideal}^{\Pi}_{\mathscr{A},S}(\kappa)$: the simulator S generates the system public parameter $\text{Param} \leftarrow S(\mathscr{L}_{\text{Setup}})$ using the leak function of system setup. Then, S generates the user's public-private key pair $(d_u, Q_u) \leftarrow S(\mathscr{L}_{\text{KeyGen}})$ based on the public parameter Param and the leak function $\mathscr{L}_{\text{KeyGen}}$ and publishes the public key list $\{Q_{u_i}\}$. Next, the adversary $\mathscr{A}$ launches an encrypted query and the simulator S generates an encrypted database $\text{EDB} \leftarrow S(\mathscr{L}_{\text{Encrypt}})$ and returns it to $\mathscr{A}$. Then, the simulator S uses the leak function of the trapdoor to generate a search trapdoor $T_{w_k} \leftarrow S(\mathscr{L}_{\text{Trapdoor}})$ in response to a trapdoor query from $\mathscr{A}$. After the adversary issues a search query, the simulator returns the result $\mathbf{RS} \leftarrow S(\mathscr{L}_{\text{Search}})$ using the leak function of the search. Finally, the adversary $\mathscr{A}$ outputs a bit $b \in \{0, 1\}$.

Scheme Π satisfies $\mathscr{L}$-adaptive security if for any probabilistic polynomial time (PPT) adversary $\mathscr{A}$, there exists a PPT simulator S such that

$$\left|\Pr\left[\mathbf{Real}^{\Pi}_{\mathscr{A}}(\kappa) = 1\right] - \Pr\left[\mathbf{Ideal}^{\Pi}_{\mathscr{A},S}(\kappa) = 1\right]\right| \leq \text{negl}(\kappa), \quad (1)$$

where $\text{negl}(\kappa)$ is a negligible function.

3.5. Design Goals. Combining the above system model and practical application requirements, our scheme should meet the following functional objectives:

(i) *Supervisibility.* Supervision can ensure the controllability of the cryptographic data. The SUP needs to supervise the encrypted data in the CS and the DU's search requests to ensure that the data can be stored and used in a legal and compliant manner.

(ii) *Multi-user Search.* Multiuser search is a basic function in data sharing scenarios. In this scenario, multiple DUs need to be authorized to access the encrypted data to provide easier data retrieval services.

(iii) *Dynamic Update*. It is an important function of the dynamic searchable encryption scheme. First, after DO generates data files containing preexisting keywords, the encrypted data structure corresponding to the keywords in BC and the data files in CS need to be updated. Secondly, in the multiuser scenario, dynamic update of authorized user DU needs to be implemented.

4. A Multiuser Normalized Searchable Encryption Scheme via Blockchain

In this section, we describe the specific construction of BNSE in detail and present its security proof. The algorithms are constructed as follows.

Setup (κ): the setup algorithm takes the security parameter κ as input. It generates parameters $(\mathbb{G}_1, \mathbb{G}_2, q, P, e)$ for the bilinear map system, where $\mathbb{G}_1$ is an additive group and $\mathbb{G}_2$ is a multiplicative group with the same prime order q, P is a generator of $\mathbb{G}_1$, and e: $\mathbb{G}_1 \times \mathbb{G}_1 \longrightarrow \mathbb{G}_2$ is a bilinear map. Then, the algorithm picks several secure hash functions $(H_0, H_1, H_2, H_3, H_4, h_1, h_2, h_3, h_4)$, where H_0, H_1, H_2, H_3, H_4 involve the elements on $\mathbb{G}_1$ or $\mathbb{G}_2$, H_0: $\mathbb{G}_1 \longrightarrow \mathbb{Z}_q^*$, H_1: $\{0,1\}^* \longrightarrow \mathbb{G}_1$, H_2: $\{0,1\}^* \times \{0,1\}^* \longrightarrow \{0,1\}^{\kappa+1}$, H_3: $\mathbb{G}_2 \longrightarrow \{0,1\}^{2\kappa}$, H_4: $\mathbb{G}_2 \longrightarrow \{0,1\}^{\kappa}$, h_1: $\{0,1\}^{\kappa} \longrightarrow \{0,1\}^{2\kappa}$, h_2: $\{0,1\}^{\kappa} \longrightarrow \{0,1\}^{2\kappa+\log M_{\max}}$, h_3: $\{0,1\}^{\kappa} \times \{0,1\}^{\log M_{\max}} \longrightarrow \{0,1\}^{2\kappa}$, and h_4: $\{0,1\}^{\kappa} \times \{0,1\}^{\log M_{\max}} \longrightarrow \{0,1\}^{\kappa+\log M_{\max}+1}$. Then, the algorithm selects a pseudorandom function F: $\{0,1\}^{\kappa} \times \{0,1\}^{\kappa} \longrightarrow \{0,1\}^{\kappa}$, with the inverse permutation F^{-1}. Finally, it outputs the public parameter

$$\text{Param} = \left(\mathbb{G}_1, \mathbb{G}_2, q, P, e, H_{0,1,2,3,4}, h_{1,2,3,4}, F, F^{-1}\right). \tag{2}$$

KeyGen (Param): $d_u \in \mathbb{Z}_q^*$ is generated randomly, and $D_u = d_u \cdot P$ is computed. The private key of date user is d_u, and D_u is a secret value to derive the public key. Given the public key of supervisor Q_{sup}, $t_u \in \mathbb{Z}_q^*$ is randomly selected and $Q_u^1 = t_u \cdot D_u$, $Q_u^2 = \mathrm{d}_u \cdot t_u \cdot Q_{sup} + D_u$, and $Q_u^3 = H_0(D_u) \cdot P$ are computed. Then, the key generation algorithm outputs the user's public key

$$Q_u = \left(Q_u^1, Q_u^2, Q_u^3\right). \tag{3}$$

Encrypt (Param, $\{Q_{u_i}\}$, DB, Σ): the input parameters of the encryption algorithm contain the system public parameter Param, the authorized data users' public key $\{Q_{u_i}\}_{i\in[1,n]}$, where n is the number of authorized users, DB = {OP, IND, W}, where OP = {add, del} (add means add file and del means delete files), IND = $\{\text{ind}_1, \text{ind}_2, \ldots, \text{ind}_m\}$, $W = \{w_1, w_2, \ldots, w_D\}$, $\text{Ind}_{w_k} = \{\text{ind}_{w_k}^1, \ldots, \text{ind}_{w_k}^{m_{w_k}}\}$ (m_{w_k} is the number of files containing the keyword w_k), and Σ[key] = value is a mapping that stores the keyword state pointer, which is able to trace back to the last update of the files including the keyword. Then, the following steps are performed:

(1) $r \in \mathbb{Z}_q^*$ is randomly selected, and the version information $VI = r \cdot P$ is computed for the encrypted database.

(2) Knowing that the authorized users of the encrypted data are $\{u_i\}_{i\in[1,n]}$ and each user's public key is $Q_{u_i} = (Q_{u_i}^1, Q_{u_i}^2, Q_{u_i}^3)$, $s \in \mathbb{Z}_q^*$ is chosen at random, and let the vector $\overrightarrow{Z} = (z_n, \ldots, z_0)$, where z_i is the coefficient of z^i in the polynomial $\prod_{i=1}^{n}(z - H(r \cdot Q_{u_i}^3)) + s$.

(3) For each keyword w_k in the keyword set W:

(1) The state pointer map $\Sigma[w_k] \longrightarrow (pt_{w_k}^c, c)$ of the keyword w_k is retrieved. If the retrieval result is empty, then the state of the current keyword is initialized. Let $c \leftarrow 0$ and $pt_{w_k}^c \leftarrow \{0,1\}^{\kappa}$, where $pt_{w_k}^0$ is not involved in information storage and c is the number of times the keyword w_k is updated. If the retrieval result is not empty, no initialization is required. The pseudorandom permutation key $k_{w_k}^{c+1} \leftarrow \{0,1\}^{\kappa}$ is randomly generated, and $pt_{w_k}^{c+1} = F(k_{w_k}^{c+1}, pt_{w_k}^c)$ is calculated. Subsequently, the local mapping $\Sigma[w_k] = (pt_{w_k}^{c+1}, c+1)$ is updated.

(2) Given the current keyword's state pointer $pt_{w_k}^{c+1}$ and the symmetric key key of the encrypted file, $\text{ref}_{pt_{w_k}^{c+1}} = h_1(pt_{w_k}^{c+1})$ and $\text{inc}_{pt_{w_k}^{c+1}} = (m_{w_k} \| k_{w_k}^{c+1} \| \text{key}) \oplus h_2(pt_{w_k}^{c+1})$ are computed, where $m_{w_k} = |\text{Ind}_{w_k}|$.

(3) For each file index $\text{ind}_{w_k}^j$ in Ind_{w_k}, $j \in [1, m_k]$, the encrypted index $EI_{w_k}^j = H_2(s, w_k \| j) \oplus (op \| \text{ind}_{w_k}^j)$ is computed, and then, $\text{ref}_{\text{ind}_{w_k}^j} = h_3(pt_{w_k}^{c+1}, j)$ and $\text{inc}_{\text{ind}_{w_k}^j} = (j \| EI_{w_k}^j) \oplus h_4(pt_{w_k}^{c+1}, j)$ are computed.

(4) The trapdoor $t_{w_k} = e(H_1(w_k), s \cdot P)$ is computed, and then, $\text{ref}_{t_{w_k}} = H_3(t_{w_k})$ and $\text{inc}_{t_{w_k}} = pt_{w_k}^{c+1} \oplus H_4(t_{w_k})$ are computed.

(4) The encrypted database is obtained through the above calculation.

$$\text{EDB} = \left\{ \langle \text{ref}_{pt_{w_k}^{c+1}}, \text{inc}_{pt_{w_k}^{c+1}} \rangle, \left\{ \langle \text{ref}_{\text{ind}_{w_k}^j}, \text{inc}_{\text{ind}_{w_k}^j} \rangle \right\}_{j \in \left[1, m_{w_k}\right]}, < \text{ref}_{t_{w_k}}, \text{inc}_{t_{w_k}} > \right\}. \tag{4}$$

The encrypted database is uploaded in the form of key-value pairs $\text{EDB}[\text{ref}_{pt_{w_k}^{c+1}}] = \text{inc}_{pt_{w_k}^{c+1}}$, $\text{EDB}[\text{ref}_{\text{ind}_{w_k}^j}] = \text{inc}_{\text{ind}_{w_k}^j}$,

and $\mathrm{EDB}[\mathrm{ref}_{t_{w_k}}] = \mathrm{inc}_{t_{w_k}}$ to the blockchain ledger via the smart contract as a searchable cryptographic data structure of keywords.

Update $(\mathrm{Param}, \{Q_{u_i}\}, \{u_{i_l}\}, w_k, r, s)$: the input parameters include the system parameter Param, the set of users to be authorized $\{u_{i_l}\}_{i_l \in [n, n_l]}$, where n' is the number of all authorized users, and the secret values r, s saved by the data owner, where the random number r involves the version information of keyword w_k and the secret value s is used to generate the trapdoor and encrypted index. Authorization update is performed on the file index set Ind_{w_k} containing the keyword w_k. The vector $\overrightarrow{Z'} = (z'_n, \ldots, z'_0)$ is computed, where z'_i is the coefficient of z^i in the polynomial $\prod_{i=1}^{n'}(z - H_0(r \cdot Q_{u_i}^3)) + s$ and n' is the total number of all authorized users.

To improve the computation efficiency, the original polynomial ciphertext $f(z)$ can be used to perform the computation by first subtracting the polynomial $f(z)$ from the secret value s and then multiplying $f(z) - s$ with the term generated by the public keys of the users to be authorized to get a new polynomial. Finally, the secret value s is embedded into this polynomial to get a new authorized polynomial $f\prime(z)$. The update process only needs to calculate the relevant terms of the user to be authorized based on the original secret text. In addition, the previously authorized users can still use the original vector $\overrightarrow{Z}$ to compute the trapdoor and decryption.

Trapdoor $(\mathrm{Param}, \{d_{u_i}\}, w_k)$: with the system parameter Param as input, only the authorized user $\{u_i\}_{i \in [1,n]}$ can use his private key d_{u_i} to compute the trapdoor of the keyword w_k. The steps of the trapdoor calculation are as follows:

(1) The version information $VI = r \cdot P$ of the keyword w_k is obtained. $D_{u_i} = \mathrm{d}_{u_i} \cdot P$ is computed such that $V = H_0(D_{u_i}) \cdot VI$, and $H_0(V)^0, H_0(V)^1, \ldots, H_0(V)^n$ are computed.

(2) Since $H_0(V)$ is a root of the polynomial $\prod_{i=1}^{n}(z - H_0(r \cdot Q_{u_i}^3))$, $\sum_{i=0}^{n} z_j H_0(V)^i = s$ is computed to get the secret value s.

(3) The trapdoor of the keyword w_k is output.

$$T_{w_k} = e(H_1(w_k), s \cdot P). \tag{5}$$

Search $(\mathrm{Param}, T_{w_k}, \mathrm{EDB})$: the search algorithm is the inverse process of the encryption algorithm, with the public parameter Param, the trapdoor T_{w_k} of the keyword w_k, and the encrypted database EDB as input parameters. An empty set $\mathbf{RS}(w_k) \leftarrow \varnothing$ is initialized to store the search results. Then, the following steps are performed:

(1) Given the trapdoor T_{w_k}, $\mathrm{ref}'_{t_{w_k}} = H_3(T_{w_k})$ is computed. $\mathrm{inc}_{t_{w_k}}{}' = \mathrm{EDB}[\mathrm{ref}'_{t_{w_k}}]$ is retrieved from the encrypted database. If $\mathrm{inc}'_{t_{w_k}} = \perp$, then the search algorithm is terminated and the search result $\mathbf{RS}(w_k) = \varnothing$ is returned. Otherwise, $pt_{w_k}^{c+1} = H_4(T_{w_k}) \oplus \mathrm{inc}_{t_{w_k}}{}'$ is computed.

(2) $\mathrm{ref}'_{pt_{w_k}^{c+1}} = h_1(pt_{w_k}^{c+1})$ is computed, and $\mathrm{inc}'_{pt_{w_k}^{c+1}} = \mathrm{EDB}[\mathrm{ref}'_{pt_{w_k}^{c+1}}]$ is retrieved. If $\mathrm{inc}'_{pt_{w_k}^{c+1}} = \perp$, the search algorithm is terminated and the search result set $\mathbf{RS}(w_k)$ is returned. Otherwise, $(m_{w_k} \| k_{w_k}^{c+1} \| \mathrm{key}) = \mathrm{inc}'_{pt_{w_k}^{c+1}} \oplus h_2(pt_{w_k}^{c+1})$ is computed.

(3) For each $j \in [1, m_{w_k}]$, $\mathrm{ref}'_{\mathrm{ind}_{w_k}^j} = h_3(pt_{w_k}^{c+1}, j)$ is computed and $\mathrm{inc}_{\mathrm{ind}_{w_k}^j}{}' = \mathrm{EDB}[\mathrm{ref}'_{\mathrm{ind}_{w_k}^j}]$ is retrieved. $(j, EI_{w_k}^j) = \mathrm{inc}_{\mathrm{ind}_{w_k}^j}{}' \oplus h_4(pt_{w_k}^{c+1}, j)$ is computed, and then, $(j, EI_{w_k}^j)$ is inserted into the search result set $\mathbf{RS}(w_k)$.

(4) Using the state pointer $pt_{w_k}^{c+1}$ of the keyword w_k and the pseudorandom permutation key $k_{w_k}^{c+1}$ obtained in step 2, the previous state pointer $pt_{w_k}^{c} = F^{-1}(k_{w_k}^{c+1}, pt_{w_k}^{c+1})$ is computed. Let $pt_{w_k}^{c+1} = pt_{w_k}^{c}$, and then, step 2 is proceeded.

Decrypt $(\mathrm{Param}, d_{u_i}, w_k, \mathbf{RS}(w_k))$: the decryption algorithm is used to decrypt the encrypted indexes in the search results $\mathbf{RS}(w_k)$. Using the secret value s computed in step 2 of the trapdoor algorithm Trapdoor, for each record $(j, EI_{w_k}^j)$ of the result set $\mathbf{RS}(w_k)$, $(op \| \mathrm{ind}_{w_k}^j) = EI_{w_k}^j \oplus H_2(s, w_k \| j)$ is computed. If op = add, the index $\mathrm{ind}_{w_k}^j$ is used to access the corresponding data ciphertext from the cloud server CS and decrypt the ciphertext using the key key obtained in step 2 of the search algorithm to get the plaintext data file. If op = add, it means this file index has been deleted and there is no need to access this file in the cloud server.

Supervise $\mathrm{Param}, \{Q_{u_i}\}, d_{\mathrm{sup}}, W^*\}$: the input parameters include system parameter *Param*, public keys $Q_{u_i} = (Q_{u_i}^1, Q_{u_i}^2, Q_{u_i}^3)$ of authorized users, the private key d_{sup} of supervisor, and the set of sensitive words W^*. $Q_{u_i}^2 - \mathrm{d}_{\mathrm{sup}} \cdot Q_{u_i}^1$ is computed to obtain D_{u_i}, and the steps are subsequently performed in the trapdoor algorithm Trapdoor to compute the secret value s. After obtaining a set of secret values $S = \{s\}$, the supervisor generates search trapdoors T_{w_k} for each secret value s of the sensitive word $w_k \in W^*$. Then, the hash value $H_4(T_{w_k})$ of the trapdoor is calculated and the hash values in the list L_h are stored and uploaded to the BC through the smart contract to realize the supervision of search requests. Second, the trapdoor set $\{T_{w_k}\}$ is used to get the matching file index ciphertext by executing the search smart contract and the ciphertext is decrypted using the secret value s to get the file index. Finally, the index is used to locate the illegal file containing the sensitive word $w_k \in W^*$ in CS to achieve the supervision of the ciphertext data in CS.

Correctness analysis: when generating the searchable encrypted data structure, a broadcast polynomial $f(z) = \prod_{i=1}^{n}(z - H_0(r \cdot Q_{u_i}^3)) + s$ is constructed. The authorized user $u_i \in \{u_1, \ldots, u_n\}$ is able to use his private key d_{u_i} to compute

$$\begin{aligned} H_0\left(H_0\left(\mathrm{d}_{u_i} \cdot P\right) \cdot VI\right) &= H_0\left(H_0\left(D_{u_i}\right) \cdot r \cdot P\right) \\ &= H_0\left(r \cdot Q_{u_i}^3\right). \end{aligned} \tag{6}$$

After obtaining the secret value s by substituting $H_0(r \cdot Q_{u_i}^3)$ into the broadcast ciphertext, the search trapdoor $T_{w_k} =$

$e(H_1(w_k), s \cdot P)$ is computed. The trapdoor search steps are described in the soundness proof of the security proof subsection. As for the ciphertext data supervision, given the private key d_{sup} of the supervisor and the partial public key $(Q_{u_i}^1, Q_{u_i}^2)$ of the authorized user, D_{u_i} is computed as follows:

$$\begin{aligned} Q_{u_i}^2 - \mathrm{d}_{\text{sup}} \cdot Q_{u_i}^1 &= \mathrm{d}_{u_i} \cdot t_{u_i} \cdot Q_{\text{sup}} + D_{u_i} - \mathrm{d}_{\text{sup}} \cdot t_{u_i} \cdot D_{u_i} \\ &= \mathrm{d}_{u_i} \cdot t_{u_i} \cdot Q_{sup} + D_{u_i} - \mathrm{d}_{u_i} \cdot t_{u_i} \cdot Q_{\text{sup}} \\ &= D_{u_i}. \end{aligned} \tag{7}$$

After getting D_{u_i}, the secret value s used for searching and decryption can be calculated as formula (1).

Theorem 1. *The BMNSE scheme* Π *is a* $\mathscr{L}$*-adaptive secure searchable encryption if* F *is a pseudorandom permutation function, the hash function is collision-resistant, the DBDH difficulty problem holds, and the polynomial-based broadcast encryption algorithm is adaptively secure.*

Proof. We demonstrate the adaptive security of the scheme through a sequence of games similar to reference [11]. The first game G_1 is the real-world game $\mathbf{Real}_{\mathscr{A}}^{\Pi}(\kappa)$. Each game is slightly different from the previous one, but they are indistinguishable from the adversary, finally reaching the last ideal world game $\mathbf{Ideal}_{\mathscr{A},S}^{\Pi}(\kappa)$. According to the transmission property of indistinguishability, it can be concluded that $\mathbf{Real}_{\mathscr{A}}^{\Pi}(\kappa)$ is indistinguishable from $\mathbf{Ideal}_{\mathscr{A},S}^{\Pi}(\kappa)$, thus completing the proof of confidentiality.

In the second game G_2, it maintains a list of state pointers PList for storing state pointers; i.e., $\text{PList}[w, c] = pt_c$. The state pointers are used in the encryption algorithm, and the game G_2 randomly chooses a string $pt_c \xleftarrow{R} \{0,1\}^{\kappa}$ to generate the state pointers instead of using the pseudorandom permutation function F. Because the pseudorandom substitution function F is indistinguishable from the actual random function, the games G_2 and G_1 are indistinguishable.

$$Pr[G_1 = 1] = \Pr[G_2 = 1]. \tag{8}$$

In the third game G_3, it models all hash functions as random oracles, where each oracle maintains a list to store input/output pairs. For example, given a random oracle H_1 with input x, the oracle randomly selects a string $y \xleftarrow{R} \{0,1\}^{l}$ as output, where l is the output length of the hash function, and stores (x, y) in the list H_1-List. Because the hash function is collision-resistant, the games G_2 and G_3 are indistinguishable.

$$Pr[G_2 = 1] = \Pr[G_3 = 1]. \tag{9}$$

In the fourth game G_4, it computes $st_{w_k} \in \mathbb{G}_2$ on the basis of $t_{w_k} = e(H_1(w_k), s \cdot P)$ by randomly choosing a secret value s in the encryption phase. Also, the game G_4 needs to maintain a list TList for storing (w_k, st_{w_k}) in response to the trapdoor query from the adversary $\mathscr{A}$. $(P, sP, H_0(w_k), t_{w_k})$ is a tuple based on the DBDH problem, and $(P, sP, H_0(w_k), t_{w_k})$ is a random tuple. If the adversary $\mathscr{A}$ can distinguish the games G_3 and G_4, it means that the adversary is able to distinguish the two tuples, i.e., solve the DBDH problem, which is contrary to the assumption of the hard problem. Thus, the games G_3 and G_4 are indistinguishable.

$$\left|\Pr[G_4 = 1] - \Pr[G_3 = 1]\right| \le \mathrm{Adv}_{\mathscr{A}}^{\text{DBDH}}(\kappa). \tag{10}$$

In the last game G_5, the simulator S maintains two lists, one for simulating random oracle queries and another counter that keeps track of the number of encryption updates since the system was initialized. For each encryption query, two random strings are selected. The simulator uses the encryption history to determine the encryption queries for the keyword w. Based on the encryption history, state pointers and keys can be generated and then the random oracle is updated. In the adversary's perspective, the view generated by the simulator S is completely indistinguishable from the view in the game G_4.

$$Pr[G_4 = 1] = \Pr[G_5 = 1] = \Pr\left[\mathbf{Ideal}_{\mathscr{A},S}^{\Pi}(\kappa)\right]. \tag{11}$$

Summing up, we can get

$$\left|\Pr\left[\mathbf{Real}_{\mathscr{A}}^{\Pi}(\kappa) = 1\right] - \Pr\left[\mathbf{Ideal}_{\mathscr{A},S}^{\Pi}(\kappa) = 1\right]\right| \le \mathrm{Adv}_{\mathscr{A}}^{DBDH}(\kappa), \tag{12}$$

where the advantage of solving the difficult DBDH problem $\mathrm{Adv}_{\mathscr{A}}^{\text{DBDH}}(\kappa)$ is negligible, so our proposed scheme Π is a $\mathscr{L}$-adaptive secure searchable encryption scheme. □

5. A Blockchain-Based Normalized Searchable Encryption System for Medical Data

In this section, we present our design of the BNSEM system based on the BNSE scheme presented in the preceding section.

5.1. System Architecture. We divide the BNSEM system into three layers: data collection layer, medical data processing layer, and medical data access layer. The system architecture is shown in Figure 2. The entities in the system are roughly the same as those in the BNSE scheme, and the difference is that the entities in the medical system are all medical service providers/users, including the medical data owner (MDO), medical data user (MDU), medical cloud server (MCS), and medical consortium blockchain platform (MCB).

In the medical data collection layer, medical data are mainly generated by doctors and patients. On the one hand, patients will generate corresponding medical data when they visit hospitals. On the other hand, the health data will be generated when patients use home medical tools or wearable medical monitoring devices, which can be used as reference indicators for the diagnosis of doctors.

In the medical data processing layer, the patients need to preprocess the data before uploading, including encrypting the medical data, establishing the index of medical file, extracting the keywords in the medical file, and constructing a searchable structure based on the file index and the keywords. Finally, the ciphertext of medical records are uploaded to MCS and the searchable structure are uploaded to MCB.

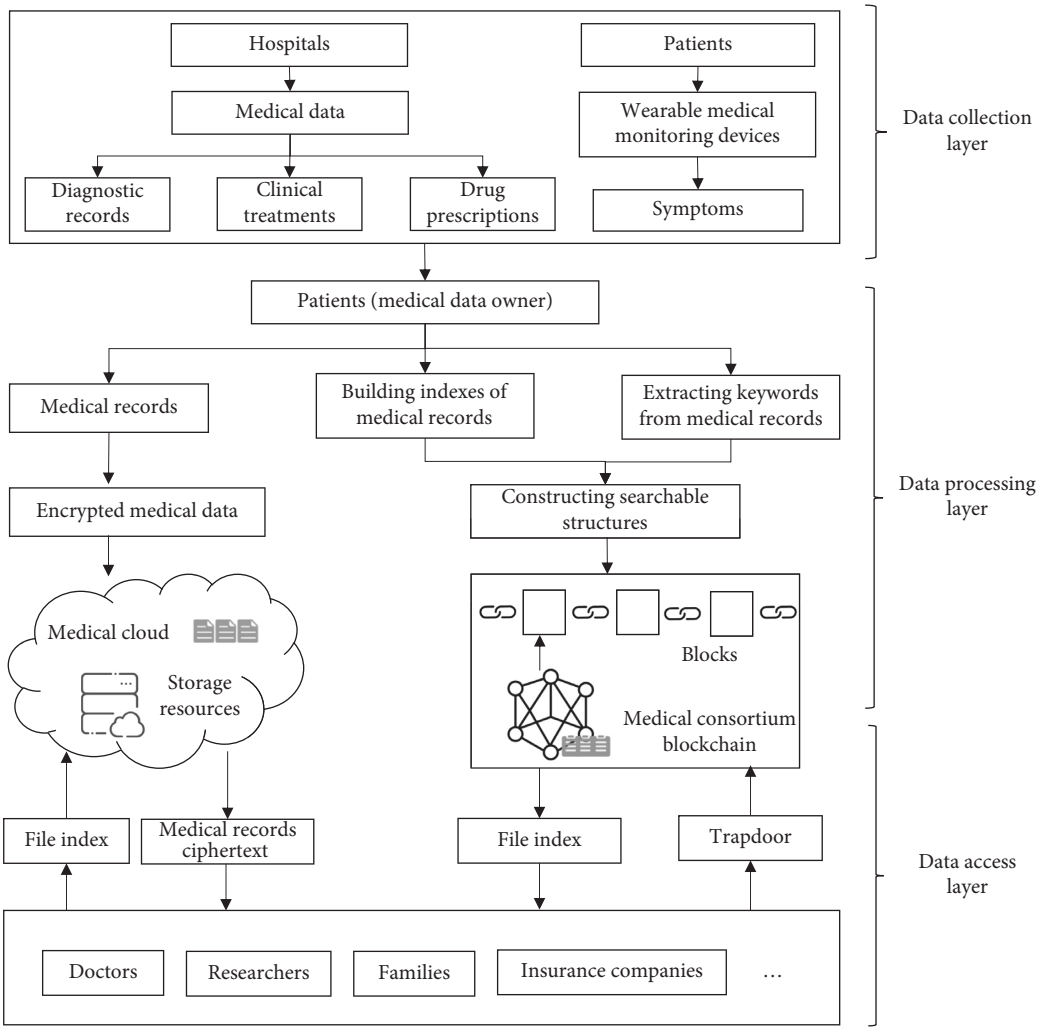


Figure 2: System architecture of BNSEM.

In the medical data access layer, only authorized medical data users can access the patient's medical data. First, the MDU generates a trapdoor for the target search keyword and sends the search request containing the trapdoor to MCB. Then, the smart contract matches the trapdoor with the searchable structure and returns the corresponding medical file index. Finally, the MDU uses the file index to access medical data in MCS and MCS returns the corresponding data to the MDU.

5.2. Medical Data Preprocessing. When a patient goes to the hospital, the doctor makes a diagnosis and generates an electronic medical record. The record includes the diagnosed disease, examination results (medical images, laboratories, etc.), medication prescriptions, and personal information (such as name, age, and gender). Each electronic medical record is treated as a file and has a unique file identifier. The doctor synchronizes the generated medical records to the patient to complete a disease diagnosis process.

5.2.1. Building the Indexes of Medical Records. When owning a specific number of medical data records, the patient can upload the record files. Before uploading, indexes corresponding to the files need to be constructed. For example, when the patient, i.e., MDO, receives m medical files $D = \{D_1, D_2, \ldots, D_m\}$, several indexes will be constructed for these files. The information related to the files can be embedded into the indexes according to the actual situation, such as the date and size of the files. The file indexes built for m medical data files D are $\mathrm{IND} = \{\mathrm{ind}_1, \mathrm{ind}_2, \ldots, \mathrm{ind}_m\}$.

5.2.2. Extracting Keywords from Medical Records. MDO performs keyword extraction for the keywords contained in each file in D. For medical data files, we mainly consider the keyword extraction of name, gender, and age in basic information, disease name, drug prescription in medical indicators, and doctor, hospital, and visit time in treatment information.

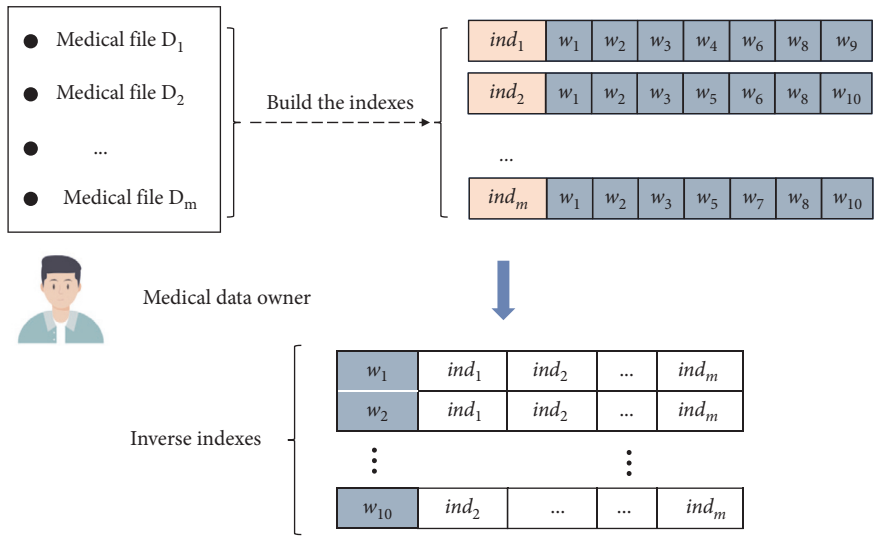


Figure 3: Process of constructing the inverse indexes.

5.2.3. Constructing the Inverse Indexes. The keywords $W_{\text{ind}_1}, W_{\text{ind}_2}, \ldots, W_{\text{ind}_m}$ extracted from different medical data files in D were integrated to obtain the keyword dictionary $W = \{w_1, w_2, \ldots, w_D\}$. Then, for each keyword w_k in the keyword dictionary W, the inverse index Ind_{w_k} containing the keyword is constructed. A specific construction of the inverse index of medical record files is shown in Figure 3.

5.3. Medical Consortium Block Chain Platform. In BNSEM system, Hyperledger Fabric is chosen as the medical consortium blockchain (MCB) platform. Because Fabric has a strict access mechanism, it can be managed collaboratively in a polycentric manner by entities from multiple organizations. In addition, the consortium blockchain can best balance the security and efficiency of the system compared with public and private blockchains. Initial access control can be achieved through the access mechanism of Fabric. By deploying smart contracts of Fabric, more fine-grained data access control can be realized.

MCB is a federation of multiple healthcare providers, which is built and maintained by different entities such as hospitals, research institutions, regulatory bodies (e.g., healthcare commissions), and insurance and pharmaceutical companies. Organizations with high trust level preselect some peer nodes as consensus nodes according to their management policies (e.g., supervision institutions and hospital management nodes). These designated consensus nodes are responsible for managing and updating the distributed ledger, while other peer nodes can only generate or contribute healthcare data transactions. Consensus nodes require a certain amount of computing power to perform consensus algorithms on transactions. In addition, if the number of consensus nodes increases, the degree of decentralization of the system increases and security and scalability can be improved.

MCB enables search structured storage and encrypted medical data retrieval by invoking predesigned and deployed smart contracts. Before MCB operates, the consortium members need to define a number of contracts developed by different organizations covering common terminology, data, rules, and processes to specify the model of data storage and sharing. A client application invokes a smart contract to execute the search protocol. When the execution is complete, the smart contract records the results (i.e., state changes) in the distributed ledger of MCB. Together with the ledger, smart contracts form the core part of the MCB system.

5.4. System Design

5.4.1. System Setup. Before the system runs, TI sets security parameters κ and generates system public parameters Param. The system parameters Param include bilinear operation parameters $(\mathbb{G}_1, \mathbb{G}_2, q, P, e)$, hash functions H/h with different output lengths, and pseudorandom permutation functions F/F^{-1} with reference to the setup algorithm in Section 4. The system selects AES algorithm as the pseudorandom permutation function to excrypt medical data. The medical data users in the system mainly include the data users' MDUs and the supervisory institution SUP. Before the users join the system, they need to generate a set of public-private key pairs for data authorization. The public-private key pair of SUP is $(d_{\text{sup}}, Q_{\text{sup}})$. The public-private key pair of MDU is (d_u, Q_u), which is computed in the setup algorithm.

5.4.2. Encryption and Updating of Medical Data. After the system is initialized, MDO will store the encrypted medical data and the corresponding searchable structure to authorize access by multiple MDUs. When patients visit the hospital and get multiple electronic medical records, these medical records will be preprocessed as described in subsection B. Next, MDO gets the file index set $\text{IND} = \{\text{ind}_1, \text{ind}_2, \ldots, \text{ind}_m\}$, the keyword dictionary $W = \{w_1, w_2, \ldots, w_D\}$, and the file index set $\text{Ind}_{w_k} = \{\text{Ind}^1_{w_k}, \text{Ind}^2_{w_k}, \ldots\}$. Let the database $\text{DB} = \left\{\text{IND}, W, \left\{\text{Ind}_{w_k}\right\}_{w_k \in W}\right\}$, and a mapping Σ stored locally for keeping the latest status of keywords (i.e., status

pointer) is initialized. Then, n MDUs are specified to be authorized, denoted as $\{u_i\}_{i\in[1,n]}$, whose public key is $\{Q_{u_i}\}$.

Taking the above parameters as input, the Encrypt data encryption algorithm in Section 4 is invoked to encrypt the medical record database DB to obtain the encrypted database; i.e., searchable data structure $\mathrm{EDB} = \left\{\langle \mathrm{ref}_{pt_{w_k}^{c+1}}, \mathrm{inc}_{pt_{w_k}^{c+1}}\rangle, \left\{\langle \mathrm{ref}_{\mathrm{Ind}_{w_k}^{j}}, \mathrm{inc}_{\mathrm{Ind}_{w_k}^{j}}\rangle\right\}_{j\in[1,m_{w_k}]}, \langle \mathrm{ref}_{t_{w_k}}, \mathrm{inc}_{t_{w_k}}\rangle\right\}$, where the variable ref is a reference to the subscript value and inc hides the information of the medical file indexes. After generating the encrypted database EDB, the key-value pairs $\mathrm{EDB}[\mathrm{ref}_{pt_{w_k}^{c+1}}] = \mathrm{inc}_{pt_{w_k}^{c+1}}, \mathrm{EDB}[\mathrm{ref}_{\mathrm{ind}_{w_k}^{j}}] = \mathrm{inc}_{\mathrm{Ind}_{w_k}^{j}}, \mathrm{EDB}[\mathrm{ref}_{t_{w_k}}] = \mathrm{inc}_{t_{w_k}}$ are uploaded to MCB through a smart contract.

Updates include medical data update and authorized user update. There are two types of medical data update: adding and deleting medical record files. When adding medical record files, MDO obtains the state pointer $pt_{w_k}^{c}$ for the same keyword w_k' as in the previous keyword dictionary W and invokes the Encrypt encryption algorithm in Section 4 to encrypt the newly added medical record database DB' to update it. When deleting medical record files, MDO performs the operation differently by selecting the del option from OP = {add, del}. The update of authorized medical data users is achieved by reconstructing the broadcast ciphertext. MDO adds the specified medical data user MDUs as $\{u_i'\}_{i'\in[n+1,n']}$, where n' is the total number of new and old authorized users, and the calculation method refers to the Update algorithm.

5.4.3. Retrieval of Encrypted Medical Data. When a patient (MDO) goes to another hospital for treatment, the authorized doctor (MDU) reviews the patient's past medical records to assist in the diagnosis. The search process for medical records is as follows:

Step 1. MDU selects a keyword w_k (e.g., hypertension) and generates the search trapdoor T_{w_k} by invoking the trapdoor algorithm using his private key.

Step 2. MDU sends a search request containing the search trapdoor T_{w_k} to the smart contract.

Step 3. The smart contract matches the trapdoor T_{w_k} with the search structure EDB stored in the blockchain to obtain the encrypted medical indexes $\{EI_{w_k}^{j}\}$ according to the search algorithm.

Step 4. MDU uses the secret value s to decrypt the ciphertext index $\{EI_{w_k}^{j}\}$ to get the plaintext index $\{ind_{w_k}^{j}\}$ and the option (add/del) corresponding to the index and the file decryption key, key.

Step 5. If the option is del, it indicates that the file has been deleted and no access is needed. On the contrary, MDU will access the medical data stored in the MCS with the indexes.

Step 6. MDU decrypts the medical record ciphertext returned from MCS to get the medical record file D using key.

5.4.4. Supervision of Medical Data and Search Requests. To ensure that the data in BNSEM system can be stored and used legally, supervisors such as the healthcare commission are required to regularly review the encrypted medical data in MCS and monitor the search requests of MDUs in real time. First, SUP maintains a sensitive word dictionary W', which includes sensitive keywords such as prohibited drugs, illegal hospitals, and fake doctors. Next, SUP invokes the supervise supervisory algorithm to locate the illegal files containing sensitive words in MCS using the private key d_{sup}. Then, SUP generates trapdoors for each sensitive word in W', and after hash calculation, a trapdoor hash list is obtained. Finally, SUP uploads the hash list to MCB through smart contracts to filter trapdoors in search requests and intercept the illegal requests containing sensitive words.

6. Security Analysis

6.1. Forward Privacy. The requirement of forward privacy is that given a previous search trapdoor, the update query does not reveal information about the keywords that were searched in the past; i.e., the previous keyword trapdoor cannot be used to search medical records newly added after the trapdoor was released. In the BNSEM system, the trapdoor T_{w_k} is equivalent to a state pointer of keyword w_k. With the help of this pointer, the smart contract will find the latest state $pt_{w_k}^{c+1}$ of the keyword w_k, which is used to locate the corresponding encrypted medical file index $EI_{\mathrm{ind}_j^{w_k}}$, where $j = \{1, \ldots, m_{w_k}\}$. The smart contract then computes the last updated state $pt_{w_k}^{c} = F^{-1}(k_{w_k}^{c+1}, pt_{w_k}^{c+1})$ to search the previously updated medical files.

When updating the medical files containing the keyword w_k, MDO will compute a new status pointer $pt_{w_k}^{c+2} = F(k_{w_k}^{c+2}, pt_{w_k}^{c+2})$, which is used to encrypt the file indexes and generate the searchable structure $\langle \mathrm{ref}_{t_{w_k}}, \mathrm{inc}_{t_{w_k}}\rangle$ corresponding to the latest version information VI. Due to the security of the pseudorandom permutation function VI, the adversary cannot predict the next state pointer based on the current state pointer $pt_{w_k}^{c+1}$ and the version information. Therefore, the previous search trapdoor cannot be used to search the medical data updated afterward, so forward privacy is guaranteed. The BNSEM system that implements forward privacy can effectively resist file injection attacks and avoid adversaries from inferring the keyword contained in a trapdoor.

6.2. Backward Privacy. Backward privacy limits the updated information of a keyword w that an adversary can obtain during a search query on the keyword w. That is, a searchable encryption system satisfies backward privacy if after a keyword-file index pair (w, ind) is added to the database and then deleted, and a search query on the keyword w will not disclose the index ind. In BNSEM system, encrypting a medical file index yields $EI_{w_k}^{j} = H_2(s, w_k \| j) \oplus (op \| \mathrm{ind}_{w_k}^{j})$, where the secret value s is broadcast encrypted using the authorized MDUs' public key and can only be decrypted by the authorized MDUs. Since the search result is in the form

of ciphertext, even if it is stored publicly on the MCB, the adversary cannot decrypt the broadcast ciphertext to recover the secret value s and cannot learn any useful information about the indexes of medical files. Therefore, the backward privacy of BNSEM can be achieved.

6.3. Distribution. Although BNSEM requires the use of a centralized MCS to store encrypted data, the search process is accomplished by smart contracts, which ensures the reliability and correctness of search results. First, to achieve the retrieval of encrypted medical data, the MDO uploads the searchable data structure to the distributed MCB platform by invoking the smart contract with storage function. Second, the MDU runs the trapdoor algorithm and uploads the trapdoor to trigger the smart contract with the search function. The correctness of the whole search process does not rely on the MCS, enabling decentralized search.

The blockchain is distributed, and each blockchain node is relatively independent and must be authenticated to join the system. It is difficult for the adversary to manipulate a large number of nodes at the same time to change the network rules and damage the blockchain system, which can effectively resist Sybil attack. In addition, since each search is recorded as an immutable transaction on the blockchain, the number of search requests sent by each MDU cannot be tampered with. The online keyword guessing attack (KGA) can be effectively resisted by setting an upper limit on the number of MDU's requests.

7. Performance Analysis

7.1. Performance Comparison. We compare the theoretical performance of our scheme with other multiuser searchable encryption schemes, where the MVSSE [24] and BAEKS [25] schemes are both based on public key cryptography, and Π^+ [20] is a symmetric searchable encryption scheme. In this study, we compare the computational overheads of the main algorithms of searchable encryption schemes, including encryption algorithm, trapdoor algorithm, and search algorithm. The results of the performance comparison are given in Table 1.

The notations in Table 1 are explained as follows: n denotes the number of authorized MDUs and m denotes the number of indexes containing the keyword w. Symbols h, exp, sm, mul_2, e, and mtp denote general hash functions (e.g., SHA-256 and SHA-3), exponential operation, scalar multiplication on the group $\mathbb{G}_1$, multiplication on the group $\mathbb{G}_2$, a bilinear pair from groups $\mathbb{G}_1$ to $\mathbb{G}_2$, and a map-to-point map. Although the hash functions $H_{0,2,3,4,}$, $h_{1,2,3,4}$ used in our scheme differ in input/output lengths, they can all be obtained by simple transformations of the general hash functions and will not add additional complexity. In addition, F/F^{-1} denotes pseudorandom permutation function (i.e., symmetric cryptography, e.g., AES and DES algorithms). The time overhead of the above operations is shown in Table 2.

It shows that the computational overhead of encryption algorithm in most schemes is linearly related to the number of indexes m in Table 1. The BAEKS scheme does not consider the number of indexes containing the keywords. In addition, the encryption computational complexity of BAEKS is linearly related to the number of users, so it is not shown in the computational overhead graph. The scheme Π^+ does not describe the broadcast encryption algorithm it uses, so the broadcast encryption overhead cannot be calculated. The encryption computation overheads of our scheme and the MVSSE scheme are $2 * sm + e + mtp + F + (2m + 4) * h$ and $sm + (2m) * \mathrm{mul}_2 + (2 + m) * F + (2m) * h$, respectively. Although our scheme contains additional time-consuming operations, they are independent of the number of indexes. The theoretical computational overhead of encryption algorithm for each scheme with respect to the number of file indexes is shown in Figure 4.

As for trapdoor algorithm, the computation overheads of MVSSE, Π^+, and our scheme are $2 * sm + 2 * \mathrm{mul}_2 + 2 * F$, $5 * F$, and $3 * sm + e + mtp + h$, respectively. Although the trapdoor computation overhead of our scheme is slightly higher than other schemes, we avoid key management and distribution operations compared with the symmetric scheme MVSSE. Moreover, the user in the MVSSE scheme cannot generate search trapdoors independently and it requires interactive communication with the server. Similarly, the Π^+ scheme requires the data owner to generate and distribute public-private key pairs for multiple recipients, which does not meet the key security specification. The theoretical computational overhead of trapdoor algorithm for each scheme is compared as shown in Figure 5. The computational overhead of our scheme is slightly higher than that of MVSSE scheme, and the trapdoor generation of Π^+ scheme only involves pseudorandom permutation operation with minimal time overhead.

When performing search operations, the MVSSE scheme contains multiple scalar multiplication operations, which will incur a large computation overhead. The search computation overhead of our scheme is lower than that of the symmetric searchable encryption scheme Π^+ because the computations in the main algorithm of our scheme are hash operations or symmetric cryptographic primitives. Therefore, our scheme is a searchable public key scheme with high search performance. Figure 6 shows the variation of the theoretical search computation overhead with the number of indexes for each scheme. Our scheme has the lowest computation overhead, and the MVSSE scheme has the highest time overhead with the number of indexes.

7.2. Prototype Implementation. We implement our BNSEM system using the MIRACL cryptographic library (C++) on a PC with 16 GB of RAM, Intel Core i5-7500 CPU, OS Windows 10, and a Fabric consortium blockchain on a PC with 16 GB of RAM, Intel Core i5-7500 CPU, and OS Ubuntu 16.04. In addition, we set the system security parameter κ to 128 bits, implement hash functions with different input and output lengths based on SHA-256, and use the AES algorithm in CBC mode as the pseudorandom permutation function with a key length of 128 bits. Finally, we choose a super-singular elliptic curve ($y^2 = x^3 - 3x$, $p =$

Table 1: Comparison of computational overheads.

Schemes	Encryption	Trapdoor	Search
MVSSE [24]	$sm + (2m) * mul_2 + (2 + m) * F + (2m) * h$	$2 * sm + 2 * mul_2 + 2 * F$	$3 * sm + (2m + 1) * mul_2 + m * F + 4m * h$
Π^+[20]	$2(m + 1) * F$	$5 * F$	$5m * F$
BAEKS [25]	$(2n + 5) * sm + n * e + (n + 1) * h$	$sm + e + mtp$	$(n + 4) * sm + 2 * e + 2 * mul_2 + h$
Ours	$2 * sm + e + mtp + F + (2m + 4) * h$	$3 * sm + e + mtp + h$	$(2m + 4)h$

Table 2: Time cost of basic operations.

Operations	e	mtp	sm	mul_2	F/F^{-1}	h
Time cost (ms)	43.669	86.316	15.995	0.024	0.002	0.001

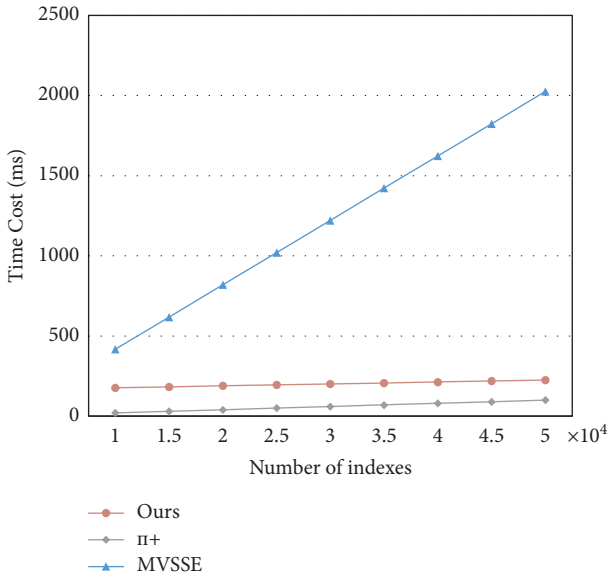
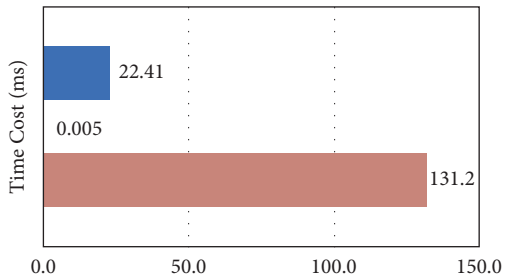


Figure 4: Comparison of encryption algorithm.



MVSSE	22.41
Π+	0.005
Ours	131.2

Figure 5: Comparison of trapdoor algorithm.

$2^{255} + 2^{41} + 1$) to achieve the ASE-128 security level. Next, we perform three simulation tests: the time cost of the encryption algorithm with the number of indexes, the time cost of the search algorithm with the number of indexes, and the time cost of all algorithms under a certain number of indexes of our system.

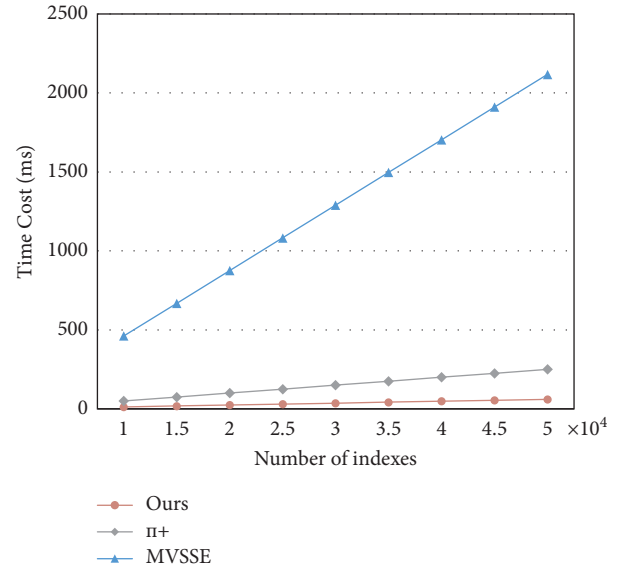


Figure 6: Comparison of search algorithm.

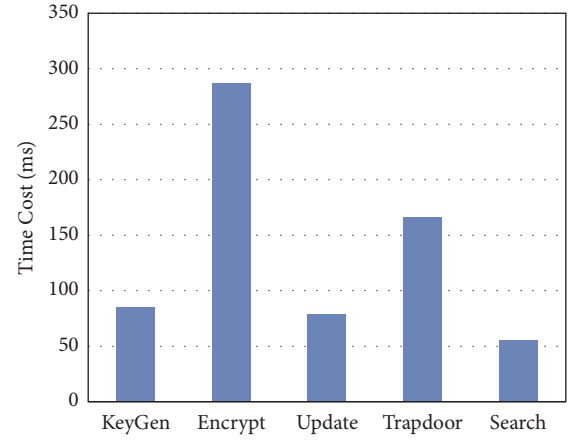


Figure 7: Time cost of each algorithm.

To overall evaluate the efficiency of our system, we test the average time overhead of all algorithms under the condition that the number of indexes containing the keyword is 10000, as shown in Figure 7. The key generation requires multiple scalar multiplication operations on the G1 group with a time overhead of about 85 ms. In addition, the time to generate a search trapdoor of the keyword is about 166 ms, while the time overhead to encrypt a search structure with 10000 file indexes is only 287 ms, mainly because the trapdoor algorithm requires the time-

consuming operations (bilinear pairs and mtp). The search algorithm in the smart contract is efficient with an average time overhead of about 55 ms for 10000 matched results.

8. Conclusion

In this study, we propose a blockchain-based searchable encryption scheme BNSE and design a searchable encryption system BNSEM for medical data based on the scheme. Firstly, the system adopts the smart contract of Fabric to guarantee the accuracy of search results. Secondly, we use polynomial-based broadcast cryptography to implement a multiuser search function. Then, the system achieves legal regulation of medical ciphertext data without violating the privacy of the private key. Finally, we provide the security analysis of BNSEM and perform a test of the time cost of each algorithm. For future work, we have considered functional extensions of multikeyword search and range queries on numerical data.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported the National Key Research and Development Program of China (2019QY0800), the Shandong Provincial Key Research and Development Program (2020CXGC010107 and 2021CXGC010107), the National Natural Science Foundation of China (U21A20466, 62172307, 61972294, and 61932016), the Blockchain Core Technology Strategic Research Program of Ministry of Education of China (2020KJ010301), the Special Project on Science and Technology Program of Hubei Province (2020AEA013), the Natural Science Foundation of Hubei Province (2020CFA052), the Wuhan Municipal Science and Technology Project (2020010601012187), the Foundation of Hangzhou Innovation Institute, Beihang University (2020-Y10-A-019), the Peng Cheng Laboratory Project (PCL2021A02), and the Foundation of Guangxi Key Laboratory of Trusted Software (kx202001).

References

[1] M. Fisk, A. Livingstone, and S. W. Pit, "Telehealth in the context of covid-19: changing perspectives in Australia, the United Kingdom, and the United States," *Journal of Medical Internet Research*, vol. 22, no. 6, Article ID e19264, 2020.

[2] S. Madhavan, L. Bastarache, J. S. Brown et al., "Use of electronic health records to support a public health response to the covid-19 pandemic in the United States: a perspective from 15 academic medical centers," *Journal of the American Medical Informatics Association*, vol. 28, no. 2, pp. 393–401, 2021.

[3] C. Esposito, A. D. Santis, G. Tortora, H. Chang, and K. R. Choo, "Blockchain: a panacea for healthcare cloud-based data security and privacy?" *IEEE Cloud Comput*, vol. 5, no. 1, pp. 31–37, 2018.

[4] D. X. Song, D. A. Wagner, and A. Perrig, *Practical techniques for searches on encrypted data IEEE Symposium on Security and Privacy*, pp. 44–55, University of califormia, Berkeley, CA, USA, 2000.

[5] P. Xu, Q. Wu, W. Wang, W. Susilo, J. Domingo-Ferrer, and H. Jin, "Generating searchable public-key ciphertexts with hidden structures for fast keyword search," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 9, pp. 1993–2006, 2015.

[6] R. Bost, B. Minaud, and O. Ohrimenko, *Forward and Backward Private Searchable Encryption from Constrained Cryptographic Primitives*, pp. 1465–1482, ACM, Times Square, NY, USA, 2017.

[7] S. Nakamoto, "Bitcoin: a peer-to-peer electronic cash system," *Decentralized Business Review*, Article ID 21260, 2008.

[8] C. Cai, J. Weng, X. Yuan, and C. Wang, "Enabling reliable keyword search in encrypted decentralized storage with fairness," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 1, pp. 131–144, 2021.

[9] W. Jiang, H. Li, G. Xu, M. Wen, G. Dong, and X. Lin, "PTAS: privacy-preserving thin-client authentication scheme in blockchain-based PKI," *Future Generation Computer Systems*, vol. 96, pp. 185–195, 2019.

[10] Y. Zhang, C. Xu, J. Ni, H. Li, and X. S. Shen, "Blockchain-assisted public-key encryption with keyword search against keyword guessing attacks for cloud storage," *IEEE Trans. Cloud Comput.*vol. 9, no. 4, pp. 1335–1348, 2021.

[11] B. Chen, L. Wu, H. Wang, L. Zhou, and D. He, "A blockchain-based searchable public-key encryption with forward and backward privacy for cloud-assisted vehicular social networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 6, pp. 5813–5825, 2020.

[12] X. Song, C. Dong, D. Yuan, Q. Xu, and M. Zhao, "Forward private searchable symmetric encryption with optimized I/O efficiency," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 5, pp. 912–927, 2020.

[13] J. Li, Y. Huang, Y. Wei et al., "Searchable symmetric encryption with forward search privacy," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 1, pp. 460–474, 2021.

[14] 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 *Proceedings of the IEEE INFOCOM 2018-IEEE Conference on computer comunications*, pp. 792–800, IEEE, Honolulu, Hi, USA, 2018.

[15] L. Chen, L. Qiu, K. Li, W. Shi, and N. Zhang, "DMRS: an efficient dynamic multi-keyword ranked search over encrypted cloud data," *Soft Computing*, vol. 21, no. 16, pp. 4829–4841, 2017.

[16] D. Cash, P. Grubbs, J. Perry, and T. Ristenpart, *Leakage-abuse attacks against searchable encryption*, pp. 668–679, ACM, Times Square, NY, USA, 2015.

[17] E. Stefanov, C. Papamanthou, and E. Shi, "Practical dynamic searchable encryption with small leakage," vol. 2013, p. 834, IACAR Cryptol, ePrint Arch, 2014.

[18] J. G. Chamani, D. Papadopoulos, C. Papamanthou, and R. Jalili, *New Constructions for Forward and Backward Private Symmetric Searchable Encryption*, pp. 1038–1055, ACM, Times Square, NY, USA, 2018.

[19] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," *EUROCRYPT, ser. Lecture Notes in Computer Science*, vol. 3027, pp. 506–522, 2004.

[20] S. Hu, C. Cai, Q. Wang, C. Wang, Z. Wang, and D. Ye, "Augmenting encrypted search: a decentralized service realization with enforced execution," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 6, pp. 2569–2581, 2021.

[21] R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky, *Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions*, pp. 79–88, ACM, Times Square, NY, USA, 2006.

[22] F. Zhao, T. Nishide, and K. Sakurai, "Multi-user keyword search scheme for secure data sharing with fine-grained access control," *ICISC, ser. Lecture Notes in Computer Science*, vol. 7259, pp. 406–418, 2011.

[23] A. Fiat and M. Naor, "Broadcast encryption," *CRYPTO, ser. Lecture Notes in Computer Science*, vol. 773, pp. 480–491, 1993.

[24] X. Liu, G. Yang, Y. Mu, and R. H. Deng, "Multi-user verifiable searchable symmetric encryption for cloud storage," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 6, pp. 1322–1332, 2020.

[25] X. Liu, K. He, G. Yang, W. Susilo, J. Tonien, and Q. Huang, "Broadcast authenticated encryption with keyword search," *ACISP, ser. Lecture Notes in Computer Science*, vol. 13083, pp. 193–213, 2021.

[26] Q. Wang, M. He, M. Du, S. S. M. Chow, R. W. F. Lai, and Q. Zou, "Searchable encryption over feature-rich data," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 3, pp. 496–510, 2018.

[27] P. Jiang, F. Guo, K. Liang, J. Lai, and Q. Wen, "Searchain: blockchain-based private keyword search in decentralized storage," *Future Generation Computer Systems*, vol. 107, pp. 781–792, 2020.

[28] Q. Tang, "Towards blockchain-enabled searchable encryption," *ICICS, ser. Lecture Notes in Computer Science*, vol. 11999, pp. 482–500, 2019.

[29] L. Chen, W. Lee, C. Chang, K. R. Choo, and N. Zhang, "Blockchain based searchable encryption for electronic health record sharing," *Future Generation Computer Systems*, vol. 95, pp. 420–429, 2019.

[30] Y. Yuan and F.-Y. Wang, "Blockchain: the state of the art and future trends," *Acta Automatica Sinica*, vol. 42, no. 4, pp. 481–494, 2016.