Hindawi
Security and Communication Networks
Volume 2022, Article ID 1103863, 15 pages
https://doi.org/10.1155/2022/1103863

Research Article

Time-Enabled and Verifiable Secure Search for Blockchain-Empowered Electronic Health Record Sharing in IoT

Xueli Nie,[1] **Aiqing Zhang**,[1,2] **Jindou Chen**,[1,2] **Youyang Qu**,[3] **and Shui Yu**[4]

[1]*School of Physics and Electronic Information, Anhui Normal University, Wuhu 241002, China*
[2]*Anhui Provincial Engineering Laboratory on Information Fusion, Wuhu 241002, China*
[3]*School of Information Technology, Deakin University, Geelong, VIC 3125, Australia*
[4]*School of Computer Science, University of Technology Sydney, Ultimo, NSW 2007, Australia*

Correspondence should be addressed to Aiqing Zhang; aqzhang2006@163.com

Received 5 June 2021; Revised 20 July 2021; Accepted 6 October 2022; Published 13 December 2022

Guest Editor: David Megias

The collection and sharing of electronic health records (EHRs) via the Internet of Things (IoT) can enhance the accuracy of disease diagnosis. However, it is challenging to guarantee the secure search of EHR during the sharing process. The advent of blockchain is a promising solution to address the issues, owing to its remarkable features such as immutability and anonymity. In this paper, we propose a novel blockchain-based secure sharing system over searchable encryption and hidden data structure via IoT devices. EHR ciphertexts of data owners are stored in the interplanetary file system (IPFS). A user with proper access permissions can search for the desired data with the data owner's time-bound authorization and verify the authenticity of the search result. After that, the data user can access the relevant EHR ciphertext from IPFS using a symmetric key. The scheme jointly uses searchable encryption and smart contract to realize secure search, time control, verifiable keyword search, fast search, and forward privacy in IoT scenarios. Performance analysis and proof demonstrate that the proposed protocol can satisfy the design goals. In addition, performance evaluation shows the high scalability and feasibility of the proposed scheme.

1. Introduction

Electronic health records (EHRs) are digital records that are the collection of patients' health records. An increasing number of EHR data are collected from Internet of Things (IoT) devices, such as intelligent wearable devices and smart watches. The electronic health records are stored electronically in a digital format which is maintained by a hospital. EHRs are highly private and have great financial value. As a consequence, more and more researchers pay wide attention to the sharing of EHR via IoT devices in recent years [1, 2]. The sharing of EHR can help doctors effectively assess patients' conditions and make correct disease diagnosis [3]. Moreover, it is beneficial to improve the quality of medical service [4].

Generally, data searching is the preliminary of data sharing. EHR searching via IoT devices brings the issues of security and privacy leakage. Firstly, protecting patients' private and sensitive information from leaking is important during the process of EHR searching as it includes personal benefits and reputation [5]. Secondly, only authentic data can improve the precision of the treatments and disease research [6]. Furthermore, patients should automatically revoke user access rights to protect his/her sensitive information.

To preserve data privacy, a data owner generally encrypts medical data before uploading it to the cloud and the cloud performed the encrypted keyword queries by using searchable encryptions [7–9]. However, the encrypted data searching based on the cloud may bring a great challenge for revoking users' access rights. Time-bound searchable encryption schemes were proposed to solve this issue [10, 11]. For dishonest behaviors of the cloud, a series of keyword-based verifiable searchable encryption schemes had been presented in [12, 13]. Even though these works combine different cryptographic algorithms and cloud computing for

EHR sharing to realize data security, there are still some security threats. The cloud is a semitrusted center to store, manage, and share EHRs, which may turn to loss, abuse, and leakage of data [9, 14]. Due to the centralization characteristic of a cloud server, it will cause single-point failure if the cloud server is attacked or lacks monitoring.

Fortunately, blockchain technology is proposed to be an advantageous solution for addressing the above problems [15–17]. Blockchain is considered a distributed public ledger to store patients' health records for sharing [18, 19]. It has significant features of immutability, decentralization, anonymity, transparency, and tamper resistance [20–22]. Search efficiency plays a key role in the blockchain. Blockchain-based searchable encryption had been proposed to support fast search [23]. Albeit blockchain-based searchable encryption for EHR sharing system is prospective, it still faces the following challenges:

(1) How to realize data security with time-bound and verifiable secure search in blockchain via IoT devices?

(2) How to guarantee that only eligible entities are authorized to access the EHR?

(3) How to achieve a fast search without disclosing patients' private information?

In order to tackle the abovementioned problems, we present a blockchain-based secure EHR sharing scheme with searchable encryption. It is tailor-made for IoT scenarios. In this work, an interplanetary file system (IPFS) [24] stores EHR ciphertext. All keywords with hidden star-like structures are encrypted and uploaded to the blockchain for ensuring data users quickly find out the intended EHR and protecting the security and privacy of the data. Besides, searchable encryption and smart contract are used to achieve that only eligible users are able to access health data after obtaining the data owner's authorization.

The main contributions of the proposed scheme are summarized as follows:

(i) We devise a novel framework that combines IPFS and blockchain to achieve a secure search for EHR sharing via IoT devices. The IPFS is used to store EHR ciphertext in a decentralized way. Blockchain is deployed to achieve data confidentiality and searchability.

(ii) We propose a time-bound and verifiable secure search protocol. Time control helps the data owner automatically revoke the user's access right and prevent the user from accessing future data. The verifiable encryption algorithm ensures that the search result is not falsified. The ciphertexts containing the same keyword have a hidden star-chain structure. The data user can send a trapdoor to the blockchain and then quickly find all matching files by the hidden star-chain structure during a limited time while prohibiting from predicting future states. In this way, search efficiency improves significantly.

(iii) We design smart contracts to achieve secure search. Furthermore, we deploy the smart contracts on the Ethereum platform and conduct extensive evaluations to demonstrate the feasibility and performance of the proposed scheme.

The remainder of the paper is organized as follows: an overview of the related work is conducted in Section 2. Preliminaries of the proposed protocol are presented in Section 3. In Section 4, we propose the system architecture, threat model, and design goals. In Section 5, the proposed protocol is formed and discussed in detail. Then, the performance analysis and proof of our scheme are introduced in Section 6. Section 7 evaluates the performance of the proposed protocol through extensive simulations. Finally, Section 8 concludes this work.

2. Related Work

In this section, we introduce the most relevant research on searchable encryption schemes based on blockchain.

2.1. EHR Sharing. EHR sharing can help to improve the accuracy of diagnosis [3]. However, EHR sharing brings some problems including security and privacy preservation in the system. In order to prevent sensitive data from leakage, some EHR sharing schemes based on searchable encryption algorithms were presented in [25, 26]. Zhang et al. [25] proposed an identity-based authorized searchable encryption scheme (IBASE) to encrypt diagnostic data with patients in cloud-assisted eHealth information systems. This article achieved patient's data sharing to different doctors. However, there exist multiple interactions between the patient and the doctor, which may reveal the patient's private information and increase his/her computational burden. Attribute encryption could be introduced to fulfill these requirements. Eltayieb et al. [26] proposed an attribute-based signcryption scheme to provide secure data sharing in the cloud environment. Furthermore, the smart contract solved the problem of cloud storage such as returning wrong results as in the traditional cloud server.

The ongoing trends are integrating the cloud with health blockchain to achieve a variety of security goals [27–30]. As EHRs were fragmented across decentralized hospitals, which hindered data sharing and puts patients' privacy at risk, [27] proposed a blockchain-based privacy-preserving data sharing for EHRs. In this article, the original EHRs were stored securely in the cloud and the indexes are reserved in a tamper-proof blockchain. Moreover, the zero-knowledge proof and the proxy reencryption technology provided strong privacy preservation in data sharing. In [28], the authors proposed a complete medical information system model based on blockchain technology, to realize the goal of safe storage and sharing of medical problems. This system designed an anonymous medical data sharing scheme based on cloud servers and proxy reencryption algorithm to improve the security of private medical data sharing. Zou et al. [29] proposed a blockchain-based medical data sharing and privacy-preserving eHealth system named SPChain. To

achieve quick retrieval, they devised special keyblocks and microblocksfor patients to store their EHRs. A secure cloud-assisted eHealth system was proposed to protect outsourced EHRs from illegal modification using blockchain technology [30]. In this work, the EHRs were outsourced by authenticated participants. In order to ensure tamper-proofing, each operation on outsourcing EHRs was integrated into the public blockchain as a transaction.

2.2. Searchable Encryption. Searchable encryption plays an important role in EHR sharing. Searchable encryption can be classified into two categories: symmetric key encryption [31] and public key encryption [32]. Searchable encryption was first introduced by Song et al. [33]. It was also the first scheme that supported keyword searching for encrypted data. In recent years, some cloud-assisted schemes have been proposed for searchable encryption [34]. A privacy-preserving sharing scheme for patient health information was proposed, which made use of the searchable encryption technique with keyword rang search and multikeyword search [35]. Since the cloud is untrustworthy, this article [35] used a bloom filter and message authentication code to classify health information files, filter fake data, and check data integrity. In order to verify whether the cloud has faithfully executed the search operations, a multiuser verifiable searchable symmetric encryption was presented in [31]. Authorized users could search data and verify the authenticity of the search result to improve the accuracy of the result. Since authorized users' access rights are always valid, it is insecure.

In order to automatically revoke a user's access right, a time key was introduced in [36]. The key seal was encapsulated in the ciphertext at the very beginning of the encryption. It implied that all users including data owner were constrained by the time period. Later, Yang and Ma [37] proposed a conjunctive keyword search with a designated tester and timing-enabled proxy reencryption function. It utilized a time server to generate a time token for the users. Moreover, it achieved time-controlled access right revocation to prevent authorized users from accessing the future EHR. The author [38] proposed a timed-release computational secret sharing and threshold encryption. This article used a time-release function instead of a time server to reduce overhead. The ongoing work [39] proposed 0-encoding and 1-encoding to generate the time key. However, these works had low search efficiency.

In order to improve the search efficiency, some schemes with hidden data structures were proposed in [23, 40, 41]. Users desired to find out more ciphertexts in one step. However, the scheme in [41] reduced the number of computation-intensive operations without searching for at least two matching ciphertexts in only one step. This work could not fulfill the need for a fast search. It could not prevent authorized users from accessing future data [40]. The scheme [23] did not consider dynamically revoking users' access rights.

Although all the above works realized search based on cloud technology, there is still a challenge: the cloud is not a completely trusted center that may collude with other entities to get the users' private information.

2.3. Searchable Blockchain. In order to address the above problems, some schemes adopted blockchain technology to achieve secure search [42, 43] due to its following advantages: decentralization, privacy preservation, immutability, fault tolerance, and the ability to implement smart contracts.

Due to the advantage of unforgeability, a smart contract cannot be modified or altered once it is deployed on blockchain [44]. The authors [45] proposed a blockchain-based searchable encryption scheme for EHR sharing, and used the complex Boolean expression to extract EHRs to construct the indexes. They designed smart contracts in blockchain to replace the centralized server for protecting users' sensitive information. Zheng et al. [46] utilized blockchain and sampling techniques with attributed-based encryption to realize fair outsourced decryption. Moreover, the work used smart contracts in blockchain to ensure that the proxy could always get the reward with the valid outsourced decryption.

Public key encryption with a keyword search scheme based on blockchain was presented in [47]. It employed multiple key servers to encrypt keywords for resisting off-line keyword guess attacks. Cai et al. [48] utilized hashing technique and searchable encryption algorithm with the assistance of blockchain to achieve secure search on-chain task-matching authorization. Their scheme reduced query communication overhead and storage overhead on the blockchain. Jiang et al. [49] introduced a blockchain-based architecture called SearchChain, a peer-to-peer keyword search system. It supported encrypted retrieval over an authorized keyword set while accessing a user to search his/her desired data and hiding the retrieval privacy in the decentralized environment. In order to reduce computation complexity and improve search efficiency, a blockchain-based searchable public key encryption scheme with forward and backward privacy was presented in [23]. The hidden data structure was used to achieve forward and backward privacy. They utilized smart contracts to guarantee that search results were correct and immutable. However, these works did not automatically revoke the user's access right and verify the authenticity of the search result.

The existing works proposed various blockchain-based searchable encryption schemes with security and privacy preservation. Actually, some schemes presented a concept or structure for a blockchain-based EHR sharing scheme without proposing a detailed solution to a specific application scenario. In this work, we propose a novel blockchain-based framework for EHR sharing by using searchable encryption and distributed storage technology to achieve privacy preservation and data security. Besides, we design the detailed protocol and implement smart contracts on the Ethereum test platform.

3. Preliminaries

We present the required preliminaries of this work in this section.

3.1. Complexity Assumptions

Definition 1. elliptic curve discrete logarithm problem (ECDLP)). Suppose E is an elliptic curve, P and Q are two primitive elements. Given $\#E$ the number of points on the curve, the ECDLP is getting the integer $d(1 \le d \le \#E)$ to satisfy as follows:

$$\underbrace{P + P + \cdots + P}_{d} = dP = Q. \tag{1}$$

In cryptosystems, the integer d is usually used as the private key and a point on the curve with coordinates $X = (x_X, y_X)$ is used as the public key X. ECDLPAssumption . Suppose that it is difficult to solve the ECDLP in polynomial time.

Definition 2. Computational bilinear Diffie–Hellman (CBDH) problem. We denote an elliptic curve as E and consider a cycling group G of prime order q. Let P be a random element in G and $a, b, c \in Z_q^*$. The CBDH problem is defined as follows: given an input tuple, $(P, aP, bP, cP) \in G$, and $e(P, P)^{abc}$ was computed. Assuming that an attacker A can calculate $e(P, P)^{abc}$ with the advantage $\mathrm{Adv}_A^{\mathrm{CBDH}}(1^\lambda)$, If the CBDH assumption holds, the advantage $\mathrm{Adv}_A^{\mathrm{CBDH}}(1^\lambda)$ must be ignored.

3.2. 0-Encoding and 1-Encoding. 0-encoding and 1-encoding are two types of encoding. They are used to turn the "greater than" problem into a "set intersection" problem [50]. In order to avoid duplication, more details of the two encodings can be seen in [51]. We only introduce some results of the algorithms.

Let $T_\varepsilon^1 = \varepsilon_{[l]}\varepsilon_{[l-1]} \ldots \varepsilon_{[1]}$ be an l-bit binary string, where $\varepsilon_{[i]}$ denotes the i-th bit of ε. Keyword search authorization and keyword generation time are encoded in a binary string in our work. A 0-encoding of ε is denoted by $T_\varepsilon^0 \leftarrow 0 - ENC(\varepsilon)$, defined as follows:

$T_\varepsilon^0 = \left\{\varepsilon_{[l]}\varepsilon_{[l-1]} \ldots \varepsilon_{[i+1]}1 | \varepsilon_{[i]} = 0, 1 \leqslant i \leqslant l\right\}$.

A 1-encoding of ε is denoted by $T_\varepsilon^1 \leftarrow 1 - \mathrm{ENC}(\varepsilon)$, defined as follows:

$T_\varepsilon^1 = \left\{\varepsilon_{[l]}\varepsilon_{[l-1]} \ldots \varepsilon_{[i]} | \varepsilon_{[i]} = 1, 1 \leqslant i \leqslant l\right\}$.

From the theorem in [51], we have $x > y$ if and only if T_x^1 and T_y^0 have a common element.

3.3. Forward Privacy. Forward privacy [52] requires that the previous search trapdoors do not search for the new updated files. An update query leaks no information about the keywords searched in the past. If a searchable encryption scheme has not forward privacy, a search token can be used to retrieve documents added after the token is issued. There will exist some attacks or data abuse (e.g., malicious users or servers may deduce keywords from this trapdoor). Thus, forward privacy is able to address the above issues. In our scheme, we utilize forward privacy to protect users' private information and prevent searching for future data using the previous trapdoor.

4. System Model

This section first proposes a general system architecture based on the searchable blockchain via IoT devices. On this basis, we highlight the threats model and security objectives.

4.1. System Architecture. A general searchable blockchain is composed of a data owner, data user, and blockchain network, as shown in Figure 1.

4.1.1. Data Owner (DO). Data owners contain patients who generate health records by interacting with doctors or obtaining data from smart devices, such as wearable devices and smart watches. Health records may draw interest from other institutions or companies. To protect their privacy and data security, data owners will encrypt their original data and send them to IPFS. Furthermore, the corresponding keywords and file identifiers with hidden data structures are encrypted and then formed into encrypted states. The data owners upload the encrypted states to the blockchain for searching and sharing, which will help patients to improve the accuracy of disease diagnosis. Only the data owner can authorize a data user to search for the intended keywords and decrypt the ciphertext.

4.1.2. Data User (DU). Data users include medical institutions or insurance companies who want to access patients' health records. They can search for intended keywords on the blockchain. They first need to send a search request to the data owner and get a search authorization token. Then, the DU generates a search trapdoor using his/her public key and token. After that, they call the search smart contract on the blockchain for searching. The results are sent to data users, who will verify the results by using the proof in the search authorization token. If the responses are true, the data user can access the data owners' data.

4.1.3. Blockchain Network. IPFS is used to store EHR ciphertext and return the corresponding file addressees to DO. In the blockchain, different entities have different access rights. Only authorized data users can use the trapdoor to search for desired data in the blockchain. The search transactions are packed into blocks. A smart contract is composed of data and code, which is an automatically executed script. We deploy a search smart contract in the proposed blockchain. Search smart contract helps authorized data users quickly find out all the matching keywords with hidden star-like structures.

4.2. Threat Model and Design Goals. In our scheme, all participating users honestly perform protocols and smart contracts. However, they are curious to obtain others'

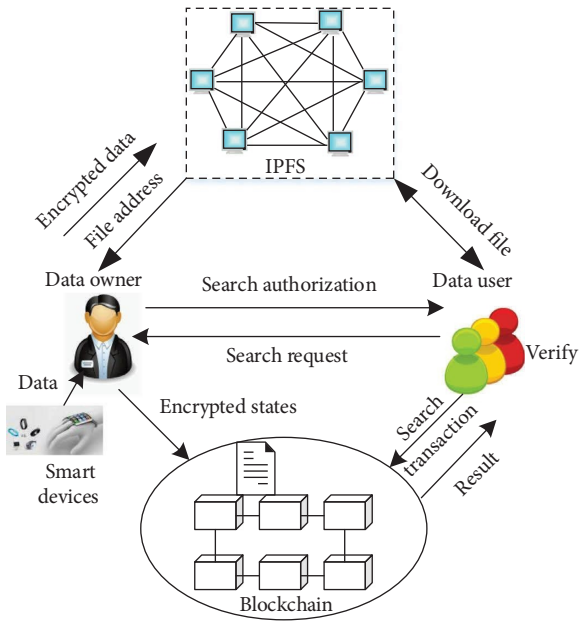

Figure 1: System architecture.

private information from encrypted states. Some malicious attackers may forge and modify the data during the transmissions. A revoked data user may attempt to search for encrypted states and access medical data. Based on the system model and threats, our scheme is designed to realize the following goals.

4.2.1. Secure Search. In this system, only authorized data user is allowed to search for the desired keyword or access the data owner's health records. The adversary cannot guess any private information from the search trapdoor. They cannot distinguish the encrypted keywords from the given keywords in the trapdoor.

4.2.2. Time Control. After obtaining a search token from a data owner, the eligible data user can search the intended file indexes and health records. Nevertheless, data users cannot utilize the same search trapdoor to access the data owner's future data. That is to say, a keyword search using a search token is time-bound.

4.2.3. Verifiable Search. Data users should be able to verify the correctness of the search results from the blockchain. The verification should be based on proof generated by the data owner. After passing the verification, the data user can access the requested data.

4.2.4. Fast Search and Forward Privacy. When the DU obtains the time-controlled authorization token from DO, he/she can generate a search trapdoor to fast search for all intended data by the hidden data structure in the smart contract. The attacker cannot utilize an old trapdoor to do some operations, such as searching for future updated files, testing the future updated files, and obtaining any other information about the future updated files. It means the old trapdoor is outdated.

5. The Proposed Protocol

In this section, we introduce the proposed protocol in detail.

5.1. Workflow. The data owner of a file (identified with f_i) extracts a keyword w for the file. It sets two secret keys k_1 and k_2 for the keyword w_i (the data owner sets the same keys k_1 and k_2 for the keyword w even though it may be extracted from different files at a different time). We suppose that the EHR and keyword generation time is t_g. The data owner sends EHR ciphertext with a symmetric key k and uploads it to IPFS. The data owner computes t_g as $\{t_{gj}\}_{j\in[1,l]} \leftarrow 0 - \mathrm{ENC}(t_g)$, encrypts w, $F(w)$, and ST with k_1, k_2, and pk_o into a state ciphertext I, and uploads the encrypted state I to the blockchain. All the state ciphertexts generated by different data owners at different times formulate the state ciphertexts II. If a data user with a public key pk_i wants to search for a state containing a keyword w' from the file collection of a data owner, the data user sends a search request to the data owner. The data owner will generate a keyword search authorization token T_1 for this data user. In this token T_1, the data owner assigns a valid time t_a for searching operations. Additionally, the data owner generates a proof P_{f_i} for the verification of search results. The data user can generate a search trapdoor T_w with the authorization token and its secret key. The data user with the trapdoor calls for search smart contract to fast search for the intended keywords from set II stored on the blockchain. The data user can verify the correctness of the search result with its secret key and proof P_{f_i}. If the search result is valid, the data owner will send the encrypted symmetric key the C_k to blockchain, and the data user will obtain the original EHR with the symmetric key k. The proposed protocol construction is presented below. The major notations used in the proposed protocol are listed in Table 1. The detailed process of the proposed protocol is shown in Figure 2.

5.1.1. Phase 1: System Setup. Given the system parameter λ, let G, G_T be groups of a λ-bit prime order p. P is a generator of G. $e: G \times G \longrightarrow G_T$ is a bilinear map. Let $H_0: \{0,1\}^* \longrightarrow G$, $H_1: Z_p^* \times Z_p^* \times \{0,1\}^* \longrightarrow Z_p^*$, $H_2: G_T \longrightarrow \{0,1\}^\lambda$, $H_3: G \times G \times Z_P^* \longrightarrow Z_P^*$, $H_4: G \times \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^* \longrightarrow Z_P^*$, $h_1: \{0,1\}^\lambda \longrightarrow \{0,1\}^\lambda$, and $h_2: \{0,1\}^\lambda \times N \longrightarrow \{0,1\}^\lambda$. The system chooses two random elements $\alpha, \beta \in Z_p^*$ and computes $H = \alpha P, T = \beta P$. Let L be a pseudorandom permutation (e.g., DES, AES), denoted as $L: \{0,1\}^\lambda \times \{0,1\}^\lambda \longrightarrow \{0,1\}^\lambda$. L^{-1} is the inverse permutation of F. The $\mathrm{Enc}(\cdot)$ and $\mathrm{Dec}(\cdot)$ are symmetric encryption algorithms and the corresponding decryption algorithm (e.g., AES), respectively.

The system parameter is $\mathrm{param} = (G, G_T, e, P, H_0, H_1, H_2, H_3, H_4, h_1, h_2, H, T, L, L^{-1})$. Furthermore, an empty map ST is initialized, denoted as $ST[\mathrm{key}] = \mathrm{value}$. The DO secretly stores the map.

Table 1: Notation table.

Notation	Description
w	Keywords extracted from EHR
$f_i(w)$	The $i-th$ file containing the keyword w
$F(w)$	All files containing the keyword w
t_g, t_a	Keyword generation time and trapdoor authorization time
I, II	State ciphertext and set of state ciphertext
C_k	Symmetric key ciphertext
C_m	EHR ciphertext
P_f	Proof for the verification of the search result
T_w	The search trapdoor of w
$S, \mathbb{S}$	Searched ciphertext and set of searched ciphertext
L, L^{-1}	a pseudorandom permutation and the inverse permutation
$ST[w], st_c, k_c, c$	An empty map, map state, map key, and a counter
V_c	The version information about the current encrypted file
$EI_{f_i(w)}$	An encrypted index containing the keyword w

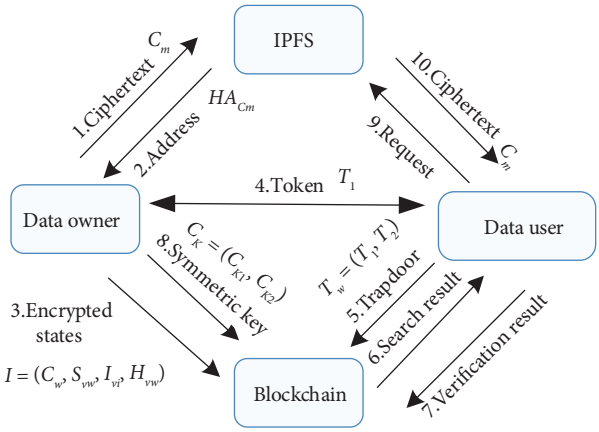

Figure 2: The proposed protocol.

5.1.2. Phase 2: KeyGen. A data user d_i randomly chooses $x_i \in Z_p^*$ and computes $X_i = x_i P$. The public and private key pair for d_i is $pk_i = X_i$ and $sk_i = x_i$. A data owner n_o randomly chooses $y_o \in Z_p^*$ and computes $Y_o = y_o P$. The public and private key pair for n_o is $pk_o = Y_o$ and $sk_o = y_o$.

5.1.3. Phase 3: Data Generation. (1) EHR ciphertext generation: the DO selects a symmetric key k randomly and executes the algorithm $\mathrm{Enc}(\cdot)$ to generate the EHR ciphertext $C_m = \mathrm{Enc}_k(m)$. Then, he/she uploads C_m to IPFS and obtains a hash address HA_{C_m} for each file.

(2) State ciphertext generation: the DO chooses a set of file identifiers $F(w) = (f_1(w), \ldots, f_n(w))$ containing the keyword w, where $n = |F(w)|$. Then, the DO performs Algorithm 1 to generate state ciphertext containing the keyword $I = (C_w, D_{vw}, I_{vi}, E_{v_w})$. Then, the DO sends I to the blockchain. The ciphertexts that contain the same keyword have a hidden relationship as shown in Figure 3. The state st_{c-1} contains i ciphertexts. By using a pseudorandom permutation L and a key k_c, the next state st_c is equal to $L(k_c, st_{c-1})$. The state st_c has i ciphertexts. If the DU wants to access desired data, he/she will generate a trapdoor. The trapdoor is considered a pointer. It points to a current state st_c and fast

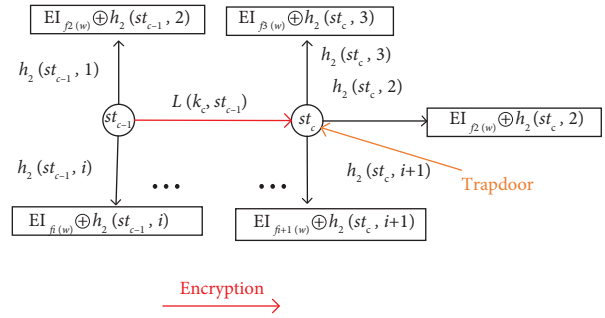

Figure 3: The hidden star-like structure of keyword ciphertext.

searches for i ciphertexts containing the same keyword in a limited time. It also can improve search efficiency.

5.1.4. Phase 4: Data Request. Time control means the DO controls the access time of the DU. That is to say, time control enables that the search trapdoor is valid before the authorized access time. In order to realize time control, we use 0-encoding and 1-encoding. In this system, the current time is t_a (authorization trapdoor time) which can be expressed as an integer. For example, the current time is "Jul. 04" which can be denoted as "$t_a = 0704$." t_a needs to be converted to the binary string "$t_a = 1011000000$" in the format of 1-encoding. We have $\{//1//, //101//, //1011//\} \leftarrow 1 - \mathrm{ENC}(t_a)$. We assume the data generation time t_g is yesterday "Jul. 03" denoted as "$t_g = 0703$." t_g also needs to be converted to the binary string "$t_g = 1010111111$" in the format of 0-encoding. We have $\{//11//, //1011//\} \leftarrow 1 - \mathrm{ENC}(t_g)$. We find "$t_a = 0704 > t_g = 070$," there is a common element "1011" in both sets. We assume the current time is t_a. The DU performs the following operations: $T_w = (T_1, T_2)$.

(1) Token Generation. when the DU wants to access the DO's EHR, he/she will send a request to the DO. Then, the DO generates a time-bound keyword authorization token T_1 for the DU. It has the property, if he/she also generates a proof

Input: a set of file identifiers $F(w) = (f_1(w) \ldots, f_n(w))$, keyword w, state map ST
Output: state ciphertext I
(1) Compute $\{t_{gj}\}_{j\in[1,l]} \leftarrow 0 - \mathrm{ENC}(t_g)$
(2) Randomly choose $\tau_w \in Z_p^*$ and compute $V_c = \tau_w P \in G$
(3) Retrieve from $ST[w]$ by w, obtain (st_c, k_c, c), and then sets $st_c \in \{0,1\}^*$, $k_c \in \{0,1\}^*$, and $c = 0$
(4) Compute $k_{c+1} = L(y_o, k_c)$ and $st_{c+1} = F(k_{c+1}, st_c)$, update $ST[w] = (st_{c+1}, k_{c+1}, c+1)$
(5) For each $t_{gj} \in \{t_{gj}\}_{j\in[1,l]}$, compute $v_j = H_1(k_1, t_{gj}, w)$
(6) Randomly choose $r \in Z_p^*$ and compute $V_j = rv_jP$ for each $j \in [1,l]$, $Z = rH_0(w)$, $C_w = [V_1, V_2, \ldots, V_l, Z, t_g]$
(7) Compute $c_{f_i(w)} = f_i(w) \oplus H_3(Z, k_2P, t_g)$ and $c_p = H_0(f_i(w)) \oplus H_4(Z, w, f_i(w), t_g)$
(8) Compute $D_{vw} = (n \| k_{c+1}) \oplus h_1(st_{c+1})$
(9) For $i \in \{1,n\}$, compute $EI_{f_i(w)} = (c_{f_i(w)} \| c_p \| HA_{c_m})$ and $I_{v_i} = h_2(st_{c+1}, i) \oplus EI_{f_i 0w}$
(10) Compute $E_{v_w} = H_2(e(H_0(w), Y_o)^{\tau_w}) \oplus st_c$
(11) Return $I = (C_w, D_{vw}, I_{vi}, E_{v_w})$

Algorithm 1: State ciphertext generation StatekeyCipGen.

P_{f_i} of keyword w' for the verification of the search result by executing Algorithm 2. The DO sends a token T_1 to the DU and proof P_{f_i} to the blockchain, respectively.

(2) Trapdoor Generation. after getting the DO's authorization token, the DU will generate a search trapdoor T_w with the public key X_i and a token T_1. The search trapdoor is calculated as follows: $T_2 = e(X_i, H_0(w'))e(V_c, H_0(w')^{y_o})$, $T_w = (T_1, T_2)$. The trapdoor T_w is sent to the blockchain for searching the desired data.

5.1.5. Phase 5: Data Access

(1) Search. the DU designs a smart contract to securely search the intended index in Algorithm 3. The DU extracts t_a from the trapdoor T_w. For each state ciphertext $I \in \mathbb{I}$, the search smart contract performs the following operations:

(i) It extracts t_g from I. If $t_g < t_a$, it computes $\{t_{gj}\}_{j\in[1,l]} \leftarrow 0 - \mathrm{ENC}(t_g)$ and $\{t_{aj}\}_{j\in[1,l]} \leftarrow 1 - \mathrm{ENC}(t_a)$.

(ii) It finds the integer b which satisfies $t_{ab} = t_{gb}$. It checks whether

$$e(U_b, Z) = e(V_b, x_i H_0(w')). \quad (2)$$

If the equation does not hold, it aborts. Otherwise, it obtains an encrypted keyword C_w and adds C_w into the set $\mathbb{S}$.

(iii) It computes $B = e(x_iP, H_0(w'))$.

(1) It computes $E_w^* = H_2(e(H_0(w'), Y_o)^{\tau_w}) = H_2(T_2/B)$, then retrieves $E_{v_w}^* = E_w^* \oplus st_c$, and adds $E_{v_w}^*$ into $\mathbb{S}$.
(2) If $E_{v_w}^* = \perp$, it terminates this algorithm. Otherwise, it obtains state information $st_c = H_2(T_2/B) \oplus E_{v_w}^*$.
(3) It computes $D_w^* = h_1(st_{c+1})$ and obtains $(n|k_{c+1}) = D_{v_w}^* \oplus D_w^*$, where $D_{v_w}^* = (n|k_{c+1}) \oplus D_w^*$, and adds $D_{v_w}^*$ into $\mathbb{S}$. If $D_{v_w}^* = \perp$, it returns $\mathbb{S}$ to the DS.
(4) For $i = n$ to 1, it computes $I_{a_i}^* = h_2(st_{c+1}, i)$, $EI_{f_i} = I_{a_i}^* \oplus I_{v_i}^*$, obtains index EI_{f_i}, and adds EI_{f_i} into $\mathbb{S}$.
(5) It computes $st_{c-1} = F^{-1}(k_c, st_c)$, sets $st_c = st_{c-1}$, and goes to step 1.

(iv) If $t_g > t_a$, it aborts.

(2) Verify. he DU obtains search results from the blockchain and verifies whether it is valid by performing Algorithm 4.

(3) Symmetric Key Ciphertext Generation. after DU sends a valid verification result to the blockchain, the DO will encrypt the symmetric key k under DU's public key X_i. The generated symmetric key ciphertext is uploaded to the blockchain. When the DU accesses the desired data, he/she will decrypt it with a symmetric key. The symmetric key ciphertext is calculated as follows: $C_{k1} = ke(H, X_i)$, $C_{k2} = \alpha T$. Then, the DO sends the symmetric key ciphertext $C_k = (C_{k1}, C_{k2})$ to the blockchain.

(4) Decryption. the DU obtains the EHR ciphertext from IPFS using the file address HA_{C_m}. The symmetric key is calculated as follows: it computes $D = (x_i/\beta)P$, $k = C_{k_1}/e(C_{k_2}, D)$. The DU uses the symmetric key k to get the original EHR by computing $m = \mathrm{Dec}_k(C_m)$.

6. Performance Analysis and Proof

In this section, we analyze how the proposed scheme achieves the design goals.

6.1. Secure Search. Secure search means that an adversary $\mathcal{A}$ cannot distinguish the keyword from the keyword ciphertext or search trapdoor. The proposed protocol is secure in the random oracle model assuming the CBDH assumption holds. According to the security game in [23], the security proof is as follows.

Proof. The random oracles of algorithm encryption and trapdoor are O_E and O_T, respectively. Suppose $\mathcal{A}$ is an

Input: keyword w' with two secret keys k_1' and k_2', version information V_c,
Output: token T_1, proof P_{f_i}
(1) Set the keyword search authorization time t_a and compute $\{t_{aj}\}_{j\in[1,l]} \leftarrow 1 - \mathrm{ENC}(t_a)$
(2) For each $t_{aj} \in \{t_{aj}\}_{j\in[1,l]}$, compute $u_j = H_1(k_1', t_{aj}, w')$
(3) For each $j \in [1, l]$, compute $U_j = u_j X_i$ and $T_1 = [U_1, U_2, \ldots, U_l, t_a]$
(4) Compute $P_{f_i} = k_2' P + y_o h_3 X_i$, $h_3 = H_3(X_i, Y_o, t_a)$
(5) Return T_1, P_{f_i}

Algorithm 2: Token generation TokenGen.

(1) function PUT$(CW, D_{vw}, I_{vi}, E_{v_w}))$ payable public returns()
(2) if keyword and state do not exist then
(3) store states ciphertext I
(4) else
(5) return false
(6) end if
(7) end function
(8) function SEARCH(T_w)
(9) if keyword authorization time is larger than keyword generation time then
(10) execute the process of search in Phase 5 of the protocol
(11) return matching file index EI_{f_i}
(12) else
(13) return false
(14) end if
(15) end function

Algorithm 3: Search smart contract construction algorithm.

Input: keywords ciphertexts set $\mathbb{S}$, token T_w, proof P_{f_i}
Output: 1/0.
(1) For each $S \in \mathbb{S}$, the data user parses $S = (C_w, EI_{f_i}, S^*_{v_w}, H^*_{v_w})$
(2) It computes $h_3 = H_3(X_i, Y_o, t_a)$, $A = P_{f_i} - x_i h_3 Y_o$ and $f_i' = c_{f_i} \oplus H_3(Z, A, t_g)$
(3) It checks $H_0(f_i') = c_p \oplus H_4(Z, w', f_i', t_g)$
(4) If the equation holds, it returns "1." Otherwise, it returns "0."

Algorithm 4: Verification.

attacker who has the advantage ε to attack the proposed chosen-plaintext attack game. We build a challenger $\mathscr{C}$. He/she plays a game with $\mathscr{A}$ to compute the solution to the CBDH problem as follows:

(i) Setup: given the CBDH parameters (P, aP, bP, cP),$e(P,P)^{abc}$ is computed. The challenge $\mathscr{C}$ randomly chooses $a, b \in Z_p$ and computes $Y = aP$ and $X = bP$. a and b are secretly stored by DO and DU, respectively. Then, it sends $\mathscr{A}$ the system parameter
param $= (G, G_T, e, P, H_0, H_2, h_1, h_2, F, F^{-1})$, the DO's public key $pk_o = aP$, and the DU's public key $pk_i = bP$.

H_0 queries: $\mathscr{C}$ maintains a list of tuples (w_i, h_i, x_{w_i}, c_i) called $H_0 - list$ for responding to the queries of H_0. $\mathscr{A}$ makes at most q_{H_0} hash function queries to H_0. $\mathscr{C}$ receives the queries and responds as follows:

(1) If the queries w_i are already in H_0 – list, $\mathscr{C}$ responds $h_i = H_i(w_i)$. Otherwise, it generates a random $c_i \in \{0, 1\}$ so that $\Pr[c_i = 1] = 1/(q_{\text{total1}} + 1)$, where $q_{\text{total1}} = q_E + q_T$.
(2) If $c_i = 0$, $\mathscr{C}$ randomly picks a number $x_{w_i} \in Z_q^*$ and computes $h_i = x_{w_i} P$. Otherwise, it sets $h_i = x_{w_i} cP$.
(3) $\mathscr{C}$ adds the tuples (w_i, h_i, x_{w_i}, c_i) into $H_0 - list$ and returns h_i to $\mathscr{A}$.

H_2 queries: H_2 queries are similar to the H_0 queries. Given an element $Q \in G_T$, $\mathscr{C}$ returns a random string $T \in \{0, 1\}^*$ and adds (Q, T) into $H_2 - list$.

(ii) Phase 1: $\mathscr{A}$ makes some queries.

Encryption queries: $\mathcal{A}$ queries the keyword $(w_i, F(w_i))$ to get encrypted file indexes. $\mathcal{C}$ responds as follows:

(1) $\mathcal{C}$ performs the aforementioned algorithms to respond H_0 and H_2 queries to get two lists (w_i, h_i, x_{w_i}, c_i) and (Q, E).
(2) $\mathcal{C}$ chooses $k^*_{c+1} \in \{0,1\}^*$ instead of computing $k_{c+1} = F(sk_o, k_c)$. Furthermore, $\mathcal{C}$ sends the encrypted data D_{vw}, $(I_{v_i})_{i=1,\ldots,n}$ to $\mathcal{A}$ and adds the tuple (st_c, k^*_{c+1}, c) into the lists $K[c+1] = k^*_{c+1}$ and $ST[w_i] = st_{c+1}$ secretly.
(3) If $c_i = 0$, $\mathcal{C}$ computes $Q = e(H_0(w), Y)^{\tau_{w_i}}$ and queries $H_2(Q)$ to get the value $E = H_2(Q)$, where $\tau_{w_i} \in Z^*_q$ is to generate the version $CV = \tau_{w_i} P$. $\mathcal{C}$ returns $E_{v_w} = E \oplus st_c$ to A, where $st_c + 1$ is retrieved by w_i from the map $ST[w_i] = st_c + 1$.
(4) Otherwise, it will report failure and end up.

Trapdoor queries: $\mathcal{A}$ queries the keyword w_i to obtain a trapdoor. $\mathcal{C}$ responds as follows:

(1) $\mathcal{C}$ performs the aforementioned algorithms to respond H_0 queries to get the list (w_i, h_i, x_{w_i}, c_i).
(2) If $c_i = 0$, $\mathcal{C}$ retrieves V_c and computes $T_2 = e(X, H_0(w_i))(V_c, H_0(w_i)^a)$. Then, $\mathcal{C}$ returns the trapdoor to $\mathcal{A}$.
(3) Otherwise, it will report failure and end up.

(iii) Challenge: $\mathcal{A}$ outputs two keyword-file pairs $(w_0, F(w_0))$ and $(w_1, F(w_1))$. Then, he/she sends them to $\mathcal{C}$. $\mathcal{C}$ executes as follows:

(1) $\mathcal{C}$ performs the aforementioned algorithms to respond H_0 queries to obtain the list (w_i, h_i, x_{w_i}, c_i).
(2) $\mathcal{C}$ runs the H_0 queries to get $H_0(w_0)$ and $H_0(w_1)$. If c_{i0} and c_{i1} are both equal to 0, then $\mathcal{C}$ will report failure and end up.
(3) $\mathcal{C}$ randomly chooses a bit $l \in \{0,1\}$. $\mathcal{C}$ performs encryption queries and maintains the tuple. The tuple denotes the latest state containing the keyword w_l.
(4) $\mathcal{C}$ computes $H_{\tau_w} = E^*_{w_l} \oplus st^*_{c+1}$, where $E^*_{w_l} \in \{0,1\}^*$, $E^*_{w_l} = H_2(e(H_0(w_l), Y_o)^{\tau_{w_l}})$.

(iv) Phase 2: the phase is the same as *Phase 1*. The restriction is that the adversary $\mathcal{A}$ cannot distinguish w_0 and w_1.

(v) Guess: $\mathcal{A}$ returns a guess $l' \in \{0,1\}$. $\mathcal{C}$ selects a random pair (Q, E) from the H_2 − list. Then, he/she returns $Q/e(P,P)^{x_{w_l}\tau_{w_l}}$ which is its guess for the CBDH problem. In addition, $\mathcal{C}$ uses x_{w_l} and τ_{w_l} in the challenge phase. Because $\mathcal{A}$ must make a query for either $H_2(e(H_0(w_0), Y_o)^{\tau_{w_0}})$ or $H_2(e(H_0(w_1), Y_o)^{\tau_{w_1}})$, its probability is 1/2. From the H_0 − list, $\mathcal{C}$ can obtain $Q = e(P,P)^{abcx_l\tau_{w_l}}$ and $E = H_2(e(H_0(w_l), Y_o)^{\tau_{w_l}})$. Then, $\mathcal{C}$ outputs $q/e(P,P)^{x_{w_l}\tau_{w_l}}$ as its guess.

In the guessing phase, if the challenger $\mathcal{C}$ can succeed in encryption queries and trapdoor queries at the same time, the probability is $(1 - 1/q_{\text{total1}} + 1)^{q_{\text{total1}}}$. Due to $(1 - 1/q_{\text{total 1}} + 1)^{q_{\text{total 1}}} \geq 1/\widehat{e}$, the probability that the challenger $\mathcal{C}$ does not abort during the game is greater than $1/\widehat{e}$, where $\widehat{e}$ is the base of the natural logarithm. In the challenge phase, if $c_{i_0} = c_{i1} = 0$, its probability is $(1 - (1/(q_{\text{total1}} + 1)))^2 \leq 1 - (1/(q_{\text{total1}} + 1))$. So, $\mathcal{C}$ must have probability $1/q_{\text{total 1}}$ at least in the challenge phase. In the guessing phase, $\mathcal{A}$ has the same probability to make H_2 queries with element $e(P,P)^{abcx_0\tau_{w_0}}$ and $e(P,P)^{abcx_1\tau_{w_1}}$. $\mathcal{C}$ has a probability of greater than $1/2\varepsilon$ to choose $Q = e(P,P)^{abcx_l\tau_{w_l}}$ correctly. Thus, the challenger $\mathcal{C}$ has the advantage that $\text{Adv}^{\text{CBDH}}_{\mathcal{A}}(1^\lambda) \geq (\varepsilon/2\widehat{e}(q_{\text{total1}} + 1)q_{\text{total2}})$, where $q_{total2} = q_{H_1} + q_{H_2}$. □

6.2. Time Control. The keyword ciphertext is encrypted by DO's public key. The eligible user can get a time-bound keyword search authorization trapdoor to search for a target keyword. As described in phase 5 of the protocol, we use 0-encoding and 1-encoding to get a time-bound trapdoor for controlling the access permission of the data user in a limited time. It finds the integer b, which can meet $t_{ab} = t_{gb}$. Especially, it checks whether $e(U_b, Z) = e(V_b, x_i H_0(w'))$ holds if trapdoor authorization time $t_a > t_g$. If the equation holds, the data user will find the desired data. The eligible data user is allowed to find the records of the data owner generated before the authorization time. Thus, other users cannot obtain any effective information during the process of keyword search.

6.3. Verifiable Search. In phase 3 of the protocol, the EHR ciphertext stored in IPFS is encrypted with the data owner's symmetric key. The corresponding keywords from the EHR are encrypted by DO's public key. In phase 4 of the protocol, the eligible users can get a search trapdoor from DO. He/she also gets a proof for the verification of search results. We recall that in proof $P_{f_i} = k'_2 P + y_o h_3 X_i$, the term $h_3 = H_3(X_i, Y_o, t_a)$ includes the public key X_i of the authorized data user. Therefore, the search result is only verified by the data user. In phase 5 of the protocol, when an eligible user sends a trapdoor to the search smart contract in the blockchain, he/she will obtain a search result. The search result can be verified by the eligible user who checks $H_0(f'_i) = c_p \oplus H_4(Z, w', f'_i, t_g)$. If the result of the verification is valid, the eligible user will decrypt the EHR ciphertext by obtaining the symmetric key. In this way, a verifiable search can be achieved.

6.4. Fast Search and Forward Privacy. In order to avoid repeatability, we refer to more details about forward privacy in [52]. We will only introduce a simple description here. In our proposed scheme, a DO maintains a state st for each keyword w. If a new file identifier containing the keyword w is added to the system, the DO will update st and send it to the blockchain. In order to better

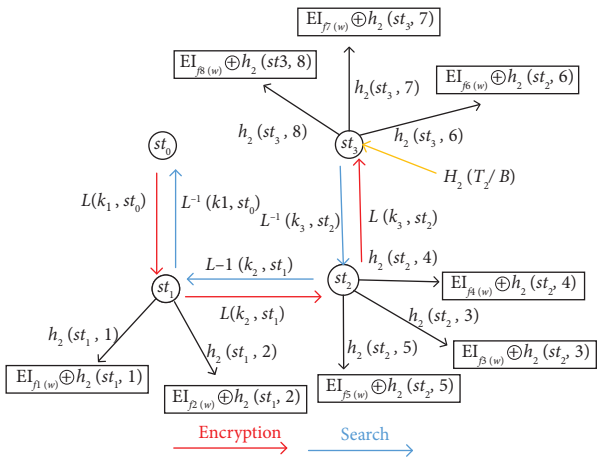

FIGURE 4: Keyword encryption and search keyword ciphertext with the hidden star-like structure.

understand fast search and forward privacy, we give a sample, as shown in Figure 4. There exists four states: st_0, st_1, st_2, st_3. st_0 is the initial state without ciphertext. The DO encrypts st_0 using a key k_1 to obtain the next state st_1. st_1 is linked to $EI_{f_1(w)} \oplus h_2(st_1, 1)$ ciphertext, $EI_{f_2(w)} \oplus h_2(st_1, 2)$ ciphertext, and $EI_{f_3(w)} \oplus h_2(st_1, 3)$ ciphertext with the same keyword w. Then, the DO utilizes st_1 and a key k_2 to get the next state st_2. st_2 has three ciphertexts containing the same keyword w. Similarly, the DO obtains st_3 which also has three ciphertexts. The DO uploads all encrypted states to the blockchain. When the DU wants to access intended data, he/she sends a trapdoor T_w to the blockchain. The trapdoor is a pointer to point to the current state st_3. Search smart contract computers $B = e(x_i P, H_0(w'))$ and $E_w^* = H_2(e(H_0(w'), Y_o)^{\tau_w}) = H_2(T_2/B)$ and obtains the state $st_3 = H_2(T_2/B) \oplus E_{v_w}^*$. Then, search smart contract can contain three indexes of the state st_3 and use a secret key k_3 to decrypt the state ciphertext $(D_w^* = h_1(st_{c+1}), (8 \| k_c) = D_{v_w}^* \oplus D_w^*)$. Then, the smart contract starts two threads. One is responsible to find the previous state $st_2 = L^{-1}(k3, st_3)$. Another one is responsible to search the indexes $\{EI_{f_8(w)}, EI_{f_7(w)}, EI_{f_6(w)}\}$ corresponding to the state st_3 by computing $h_2(st_3, 8), h_2(st_3, 7), h_2(st_3, 6)$ respectively. By repeating the above process, search smart contracts can find out all matching indexes $\{EI_{f_1(w)}, EI_{f_2(w)}, EI_{f_3(w)}, EI_{f_4(w)}, EI_{f_5(w)}, EI_{f_6(w)}, EI_{f_7(w)}, EI_{f_8(w)}\}$ quickly. From the above process, we know that the trapdoor as a pointer points to the latest state. The hidden star-like structure and smart contract improve search efficiency. Thus, it achieves fast search. In our scheme, the adversary cannot use the current state st_c and other information to obtain the future state by the pseudorandom permutation F. Thus, the old trapdoor cannot be used to search for future updated data. Forward privacy can be achieved.

7. Implementation and Evaluation

In this section, we utilize Java programming language and JPBC library to execute the proposed algorithms. We deploy the designed smart contract on the Ethereum test platform. Firstly, we give some parameter settings and platforms. Then, we compare security properties with other schemes. Furthermore, the communication and computational costs of our protocol are analyzed. Finally, we evaluate the performance of the smart contract on the blockchain.

7.1. Parameter Settings and Platform. The system security parameter is denoted as $\lambda = 128$. We use type A pairing on the elliptic curve $y^2 = x^3 + x$ over the field F_p for some prime $p = 3 \bmod 4$. We implement the cryptographic primitives by using JPBC library and Java on a laptop computer with Intel (R) Core (TM) i5-7400 CPU @3.00 GHz, 8 GB RAM, and Microsoft Windows 10 operating system. Additionally, Ganache (client version) is used to build a local test chain on a Linux system. Smart contract framework and solidity compilers are truffle @0.5.0 and sold @0.5.0, respectively. We utilize solidity language to write the data into a smart contract and then upload them to the blockchain. NodeJS's Web3js library interacting with smart contracts on the blockchain is achieved to directly obtain the time cost of sending a transaction. Due to the limited space, the detailed deployment process is skipped.

7.2. Comparisons of Security Properties. We compare the security properties of the proposed scheme with other works by Che et al. [23], Xu et al. [41], and Hu et al. [53] in Table 2. From Table 2, we conclude that our scheme achieves a verifiable search. Notably, some of the schemes have the properties of forward privacy and secure search, which are important security goals in EHR sharing systems. The comparison result shows that our proposed scheme can provide a promising solution to keyword search services.

7.3. Communication Overhead. The size of EHR data, an element in G, G_T, and Z_p are denoted by $|M|, |G|, |G_T|$ and Z_p bytes ,respectively. The hash address in IPFS is 46 bytes. The communication overhead includes five phases: data storage, data broadcast, search, data verification, and data access. In the data storage phase, DO sends EHR ciphertext C_m to IPFS for storage and the length is $|M|$ bytes. Then, IPFS sends HA_{c_m} to DO, where the length is $|G| + 46$ bytes. Additionally, DO broadcasts C_m, C_K, and HA_{c_m} to the blockchain. The total length is $((1 + 2n)|G| + |G_T| + 46$ bytes, where n is the amount of files containing the keyword$w$. DU searches the data, the generated communication overhead is $|G_T| + n|G|$ bytes during the process of data search. DU verifies the authenticity of the search result, the generated communication cost is $n|G|$ bytes in the data verification phase. At the data access phase, the generated communication cost is $|M| + |Q| + 46$ bytes, which is caused by k and C_m. The communication cost is shown in Table 3.

We compare our communication cost with other two works Chen et al. [23] and Xu et al. [41], in which the amount of keyword is denoted by y. As can be observed in Table 3, our scheme in the process of data broadcast is higher

Table 2: Comparison of security properties.

Properties	Chen et al. [23]	Xu et al. [41]	Hu et al. [53]	The proposed
Blockchain-based	√	×	√	√
Secure search	√	√	√	√
Verifiable search	×	×	×	√
Forward privacy	√	√	√	√

Table 3: Communication overhead of the proposed protocol.

Stage	The proposed	Chen et al. [23]	Xu et al. [41]
Data storage	$\|M\|$	$\|M\|$	—
Data broadcast	$(1+2n)\|G\| + \|G_T\| + 46$	$2y\|Q\| + yn\|Q\|$	$2y\|G_T\|$
Search	$\|G_T\| + n\|G\|$	$y(\|G\| + \|G_T\|) + 2y\|Q\| + yn\|Q\|$	$y(\|G\| + \|G_T\|)$
Data verification	$n\|G\|$	—	—
Data access	$\|M\| + \|Q\| + 46$	$\|M\| + yn\|G\|$	—

Table 4: Computational overhead of cryptographic algorithms (in *ms*).

	Algorithms	SystemInit	StateCipGen	Trapdoor	Search	Verify	SymkeyGen	Dec
$n = 10$	Average time	148	294	336	65	43	22	46
	Max time	357	332	352	76	46	25	48
	Min time	94	281	324	59	42	21	44
$n = 50$	Average time	153	1198	1227	67	42	23	45
	Max time	623	1228	1262	69	43	26	46
	Min time	93	1174	1000	58	42	22	44
$n = 100$	Average time	186	2310	2417	63	43	23	45
	Max time	542	2413	2569	69	44	26	47
	Min time	93	2305	1000	60	42	22	43

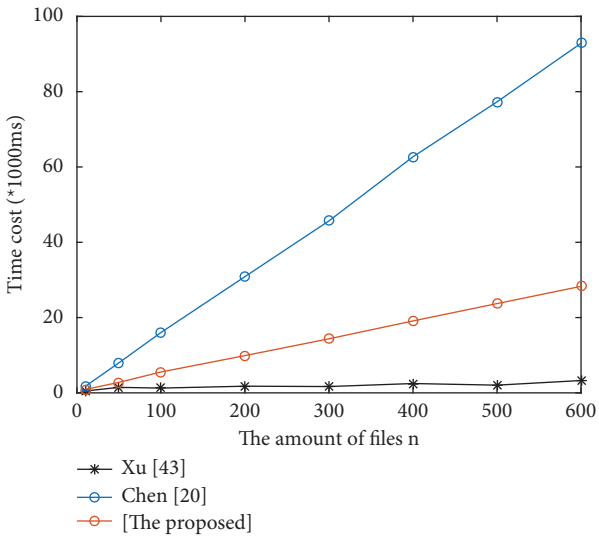

Figure 5: Comparisons of computational cost.

communication overhead as compared to Xu et al. [41]. Chen et al.'s [23] scheme in the process of data access has a higher communication cost than our scheme. Nevertheless, in the process of search, our communication cost of the process of data search is lower.

7.4. Computational Cost. We compare the computational overhead in Table 4. The algorithm SystemInit simulates the system initialization phase. A DO generates state ciphertext containing the keyword in StateCipGen. As for a DU, the algorithm Trapdoor generates a search token for him/her. The matching test of state ciphertext and trapdoor is executed in the Search algorithm. The DU receives the matching result and verifies its validity. Then, the DO generates symmetric key ciphertext performed by SymkeyGen. Finally, the symmetric key ciphertext is decrypted in the Dec algorithm.

As shown in Table 4, we implement the algorithms with file amounts 10, 50, and 100, respectively. We observe that the amount of files n affects the computational cost of StateCipGen and Trapdoor algorithms, because these algorithms contain file sets. However, other algorithms are not affected by file sets.

Besides, the comparison of computational cost is depicted in Figure 5. The result shows that our scheme becomes increasingly efficient than Chen et al. [23] scheme with the increasing amount of files. The proposed scheme is higher than the Xu et al. [41] scheme in terms of computational costs. This is because the proposed scheme has many cryptographic algorithms related to the amount of files. Xu et al.'s [41] scheme has only one pairing operation and a few cryptographic algorithms related to the amount of files, so

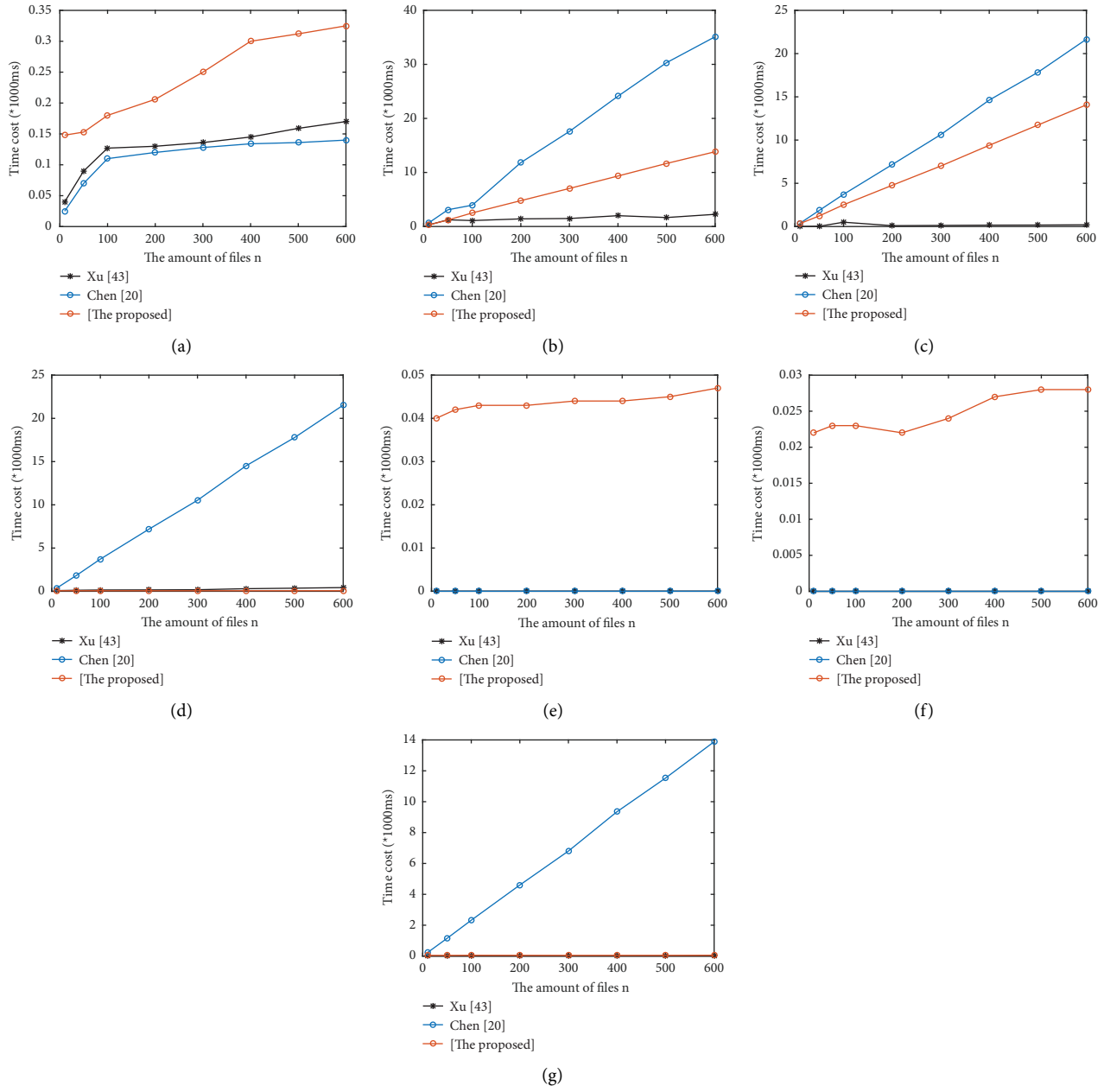

Figure 6: Computational cost taken by (a) SystemInit, (b) StateCipGen, (c) trapdoor, (d) search, (e) verify, (f) SymkeyGen, and (g) Dec.

the computational cost of Xu et al.'s [41] scheme is the smallest one of the three. In order to better show the results, we plotted the computation cost of each algorithm with the amount of files n, as shown in Figure 6.

Figure 6(a) indicates the computational cost taken by the system init algorithm of the proposed and other related works Chen et al. [23] scheme and Xu et al. [41] scheme. The system init computational cost for all the schemes changes linearly with the amount of files n. Figure 6(b) shows the computational cost of the data generation algorithm from the data owner. We can observe that it also increases linearly with the amount of files n, and Chen et al.'s [23] scheme has a higher computational cost compared to Xu et al.'s [41] scheme and the proposed scheme. Xu et al.'s [41] scheme is the smallest computational cost of the three. Because it has only one pairing operation, the computational cost of the data generation has hardly changed. Figure 6(c) shows the computational cost of the trapdoor generation algorithm from the data user. As can be observed from Figure 6(c), it also changes linearly with the amount of files n, and our scheme and Chen et al.'s [23] scheme have higher computational costs as compared to Xu et al.'s [41] scheme. Figure 6(d) presents the computational cost of the search algorithm at blockchain. It also varies linearly with the amount of files n, and Chen et al.'s [23] scheme has a higher computational cost compared to Xu et al.'s [41] scheme and the proposed scheme. Because the proposed search algorithm is achieved

Table 5: Time consumption of transactions.

Transactions	Tx_1 ($n = 10$)	Tx_1 ($n = 50$)	Tx_1 ($n = 100$)	Tx_2
Length (bytes)	1436	6556	12956	96
Average time (ms)	59.8	357.82	623.91	47.446
Max time (ms)	105	645	1130	98
Min time (ms)	48	329	614	37
Gas used (wei)	1277948	2466374	4544114	471536

by smart contract in a limited time, the computational cost of the search is almost unchanged.

Figure 6(e) presents the computational cost of verifying algorithm from the data user. As can be observed the Figure 6(e), the verify algorithm for the proposed scheme varies linearly with the amount of files n, while the other two schemes are constant. In the proposed scheme, he/she needs to verify the authenticity of the search result when the data user receives the search result from the blockchain. Figure 6(f) indicates the computational cost taken by the symmetric key algorithm at the data owner's end. From Figure 6(f), we observe that our scheme about the computational cost is higher than other schemes. This is because only when the data user obtains the right search result, he/she can get the encrypted symmetric key from the data owner. Figure 6(g) shows the computational cost taken by the decryption algorithm at the data user's end. From Figure 6(g), we can see that the computational cost of our scheme and Xu et al.'s [41] scheme are constant, while Chen et al. [23] scheme has a higher computational cost compared to Xu et al.'s [41] scheme and the proposed scheme, because Chen et al.'s [23] scheme has more hash operations and keywords.

7.5. Time Consumption Evaluation. The time consumption of sending a transaction to the blockchain is related to the length of the data package. According to the section computational cost, we know that the length of state ciphertext is $(1 + 2n)|G| + |G_T| + 46$ bytes and data access is $|M| + |Q| + 46$ bytes. The G, G_T, Q, and $|M|$ are 64 bytes, 32 bytes, 32 bytes, and 32 bytes, respectively. So, the size of two transactions is $Tx_1 = 128n + 156$ bytes and $Tx_2 = 96$ bytes, respectively. Because files amounts n affects the Tx_1, we set $n = 10$, $n = 50$, and $n = 100$ to implement the transactions on the Ethereum platform.

As can be seen from Table 5, the time consumption is related to a transaction's length for publishing a transaction in the blockchain. If the amount of files set is large, it will affect the speed of the transactions. In addition, a transaction's length also affects gas consumption.

8. Summary and Future Work

In this work, we present a blockchain-based EHR sharing scheme via IoT devices that combines searchable encryption and IPFS to realize the storage and sharing of EHR. The proposed scheme also realizes a time-bound and verifiable secure search mechanism with eligible users in the blockchain. Firstly, we propose an EHR sharing framework among multiple users. Secondly, we use searchable encryption and smart contract to ensure data confidentiality and employ a hidden star-chain structure to achieve fast search and forward private. Then, the performance analysis and proof of the proposed protocol testifies the achievement of design objectives. Furthermore, we also evaluate the performance of communication overhead and computational cost of the proposed protocol compared to other schemes.

Future work under progress is that we lay more focus on lightweight and dynamic searchable encryption schemes. In addition, we also plan to integrate federated learning [54], edge computing [55, 56], and IoT [57] in this scenario to better enhance privacy protection.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under grant 62072005, in part by the Natural Science Foundation of Anhui Province under grant 2108085Y22, and Anhui Provincial Engineering Laboratory on Information Fusion and Control of Intelligent Rabot (grant no. IFCIR2020008).

References

[1] K. Riad, R. Hamza, and H. Yan, "Sensitive and energetic IoT access control for managing cloud electronic health records," *IEEE Access*, vol. 7, pp. 86384–86393, Jul, 2019.

[2] P. P. Ray, B. Chowhan, N. Kumar, and A. Almogren, "BIoTHR: electronic health record servicing scheme in IoT-blockchain ecosystem," *IEEE Internet of Things Journal*, vol. 8, no. 13, pp. 10857–10872, Jan, 2021.

[3] A. Zhang and X. Lin, "Towards secure and privacy-preserving data sharing in e-health systems via consortium blockchain," *Journal of Medical Systems*, vol. 42, no. 8, p. 140, Jun. 2018.

[4] Y. Sharma and B. Balamurugan, "Preserving the privacy of electronic health records using blockchain," *Procedia Computer Science*, vol. 173, pp. 171–180, Jul. 2020.

[5] L. Guo, C. Zhang, J. Sun, and Y. Fang, "A privacy-preserving attribute-based authentication system for mobile health networks," *IEEE Transactions on Mobile Computing*, vol. 13, no. 9, pp. 1927–1941, Sep, 2014.

[6] Y. Su, Y. Li, K. Zhang, and B. Yang, "A privacy-preserving public integrity check scheme for outsourced EHRs," *Information Sciences*, vol. 542, pp. 112–130, Jan. 2021.

[7] X. Liu, X. Yang, Y. Luo, and Q. Zhang, "Verifiable multi-keyword search encryption scheme with anonymous key generation for medical Internet of Things," *IEEE Internet of Things Journal*, vol. 2021, pp. 1–13, Article ID 3056116, 2021.

[8] D. Zheng, B. Qin, Y. Li, and A. Tian, "Cloud-assisted attribute-based data sharing with efficient user revocation in the Internet of Things," *IEEE Wireless Communications*, vol. 27, no. 3, pp. 18–23, Jun. 2020.

[9] X. Wang, X. Cheng, and Y. Xie, "Efficient verifiable key-aggregate keyword searchable encryption for data sharing in outsourcing storage," *IEEE Access*, vol. 8, pp. 11732–11742, Dec, 2020.

[10] Q. Liu, G. Wang, and J. Wu, "Time-based proxy re-encryption scheme for secure data sharing in a cloud environment," *Information Sciences*, vol. 258, pp. 355–370, Feb. 2014.

[11] C. I. Fan, J. C. Chen, S. Y. Huang, J. J. Huang, and W. T. Chen, "Provably secure timed-release proxy conditional reencryption," *IEEE Systems Journal*, vol. 11, no. 4, pp. 2291–2302, Dec, 2017.

[12] W. Yang and Y. Zhu, "A verifiable semantic searching scheme by optimal matching over encrypted data in public cloud," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 100–115, Jun. 2021.

[13] Y. Liang, Y. Li, Q. Cao, and F. Ren, "VPAMS: verifiable and practical attribute-based multi-keyword search over encrypted cloud data," *Journal of Systems Architecture*, vol. 108101811 pages, Sep. 2020.

[14] T. Kanwal, A. Anjum, S. U. Malik, A. Khan, and M. A. Khan, "Privacy preservation of electronic health records with adversarial attacks identification in hybrid cloud," *Computer Standards & Interfaces*, vol. 78103616 pages, 2021.

[15] Y. A. Qadri, A. Nauman, Y. B. Zikria, A. V. Vasilakos, and S. W. Kim, "The future of healthcare Internet of Things: a survey of emerging technologies," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 1121–1167, Feb, 2020.

[16] E. J. De Aguiar, B. S. Faiçal, B. Krishnamachari, and J. Ueyama, "A survey of blockchain-based strategies for healthcare," *ACM Computing Surveys*, vol. 53, no. 2, pp. 1–27, 2021.

[17] P. P. Ray, D. Dash, K. Salah, and N. Kumar, "Blockchain for IoT-based healthcare: background, consensus, platforms, and use cases," *IEEE Systems Journal*, vol. 15, no. 1, pp. 85–94, Mar. 2021.

[18] J. Feng, F. Richard Yu, Q. Pei, X. Chu, J. Du, and L. Zhu, "Cooperative computation offloading and resource allocation for blockchain-enabled mobile-edge computing: a deep reinforcement learning approach," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6214–6228, Jul, 2020.

[19] L. Gao, T. Luan, B. Gu, Y. Qu, and Y. Xiang, "Blockchain based decentralized privacy preserving in edge computing," pp. 83–109, Wireless Networks, Apr. 2020.

[20] L. Liu, J. Feng, Q. Pei et al., "Blockchain-enabled secure data sharing scheme in mobile-edge computing: an asynchronous advantage actor–critic learning approach," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2342–2353, Feb, 2021.

[21] A. Zhang, P. Zhang, H. Wang, and X. Lin, "Application-oriented block generation for consortium blockchain-based IoT systems with dynamic device management," *IEEE Internet of Things Journal*, vol. 8, no. 10, pp. 7874–7888, 2021.

[22] 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.

[23] 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.

[24] J. Benet, "Ipfs-content addressed, versioned, p2p file system," *Eprint Arxiv*, 2014.

[25] X. Zhang, Y. Tang, S. Cao, C. Huang, and S. Zheng, "Enabling identity-based authorized encrypted diagnostic data sharing for cloud-assisted E-health information systems," *Journal of Information Security and Applications*, vol. 54, no. 3, 102612 pages, 2020.

[26] N. Eltayieb, R. Elhabob, A. Hassan, and F. Li, "A blockchain-based attribute-based signcryption scheme to secure data sharing in the cloud," *Journal of Systems Architecture*, vol. 102101711 pages, 2020.

[27] H. Huang, P. Zhu, F. Xiao, X. Sun, and Q. Huang, "A blockchain-based scheme for privacy-preserving and secure sharing of medical data," *Computers & Security*, vol. 99, pp. 102010–102013, Dec. 2020.

[28] Z. Chen, W. Xu, B. Wang, and H. Yu, "A blockchain-based preserving and sharing system for medical data privacy," *Future Generation Computer Systems*, vol. 124, pp. 338–350, Nov. 2021.

[29] R. Zou, X. Lv, and J. Zhao, "SPChain: blockchain-based medical data sharing and privacy-preserving eHealth system," *Information Processing & Management*, vol. 58, no. 4, pp. 102604–102618, Jul. 2021.

[30] S. Cao, G. Zhang, P. Liu, X. Zhang, and F. Neri, "Cloud-assisted secure eHealth systems for tamper-proofing EHR via blockchain," *Information Sciences*, vol. 485, pp. 427–440, Jun. 2019.

[31] 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, Nov, /Dec. 2020.

[32] 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, Sep, 2015.

[33] D. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proceedings of the 2000 IEEE Symposium on Security and Privacy*, pp. 44–55, Beijing China, August. 2000.

[34] R. Zhang, R. Xue, and L. Liu, "Searchable encryption for healthcare clouds: a survey," *IEEE Transactions on Services Computing*, vol. 11, no. 6, pp. 978–996, 2018.

[35] C. Xu, N. Wang, L. Zhu, K. Sharif, and C. Zhang, "Achieving searchable and privacy-preserving data sharing for cloud-assisted E-healthcare system," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8345–8356, Oct, 2019.

[36] K. Emura, A. Miyaji, and K. Omote, "A timed-release proxy re-encryption scheme," *IEICE - Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E94-A, no. 8, pp. 1682–1695, 2011.

[37] Y. Yang and M. Ma, "Conjunctive keyword search with designated tester and timing enabled proxy Re-encryption function for E-health clouds," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 4, pp. 746–759, Apr. 2016.

[38] Y. Watanabe and J. Shikata, "Timed-release computational secret sharing and threshold encryption," *Designs, Codes and Cryptography*, vol. 86, no. 1, pp. 17–54, Jan. 2018.

[39] K. Emura, T. Hayashi, and A. Ishida, "Group signatures with timebound keys revisited: a new model, an efficient construction, and its implementation," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 2, pp. 292–305, 2020.

[40] P. Xu, X. Tang, W. Wang, H. Jin, and L. T. Yang, "Fast and parallel keyword search over public-key ciphertexts for cloud-assisted IoT," *IEEE Access*, vol. 5, pp. 24775–24784, 2017.

[41] P. Xu, S. He, W. Wang, W. Susilo, and H. Jin, "Lightweight searchable public-key encryption for cloud-assisted wireless sensor networks," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 8, pp. 3712–3723, 2018.

[42] K. Gai, J. Guo, L. Zhu, and S. Yu, "Blockchain meets cloud computing: a survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 2009–2030, 2020.

[43] S. Shi, D. He, L. Li, N. Kumar, M. K. Khan, and K. K. R. Choo, "Applications of blockchain in ensuring the security and privacy of electronic health record systems: a survey," *Computers & Security*, vol. 97, pp. 101966–102020, Oct. 2020.

[44] G. Wood, "Ethereum: a secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, pp. 1–32, 2014.

[45] 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 Generation Computer Systems*, vol. 95, pp. 420–429, Jun. 2019.

[46] H. Zheng, J. Shao, and G. Wei, "Attribute-based encryption with outsourced decryption in blockchain," *Peer-to-Peer Networking and Applications*, vol. 13, no. 5, pp. 1643–1655, Jun, 2020.

[47] 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 Transactions on Cloud Computing*, vol. 9, no. 4, pp. 1335–1348, 2021.

[48] 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.

[49] 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, Jun. 2020.

[50] H. Lin and W. Tzeng, "An efficient solution to the millionaires' problem based on homomorphic encryption," *Proceedings of Applied Cryptography and Network Security*, pp. 456–466, Jun. 2005.

[51] C. Chu, J. K. Liu, X. Huang, and J. Zhou, "Verifier-local Revocation Group Signatures with Time-Bound Keys," *ASIACCS*, vol. 12, pp. 26-27, 2012.

[52] R. Bost, B. Minaud, and O. Ohrimenko, "Forward and Backward Private Searchable Encryption from Constrained Cryptographic Primitives," in *Proceedings of the Acm Sigsac Conference*, pp. 1465–1482, Texas, Dallas, USA, November 2017.

[53] 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 Communications*, pp. 792–800, NY China, April. 2018.

[54] Y. Qu, L. Gao, T. H. Luan et al., "Decentralized privacy using blockchain-enabled federated learning in fog computing," *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 5171–5183, Jun, 2020.

[55] B. Gu, X. Wang, Y. Qu, J. Jin, Y. Xiang, and L. Gao, "Context-aware privacy preservation in a hierarchical fog computing system," *IEEE International Conference on Communications*, pp. 1–6, Jul. 2019.

[56] B. Gu, L. Gao, X. Wang, Y. Qu, J. Jin, and S. Yu, "Privacy on the edge: customizable privacy-preserving context sharing in hierarchical edge computing," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 4, pp. 2298–2309, Oct, /Dec. 2020.

[57] Y. Qu, S. Yu, W. Zhou, S. Peng, G. Wang, and K. Xiao, "Privacy of Things: emerging challenges and opportunities in wireless Internet of Things," *IEEE Wireless Communications*, vol. 25, no. 6, pp. 91–97, Dec. 2018.