Future Generation Computer Systems 95 (2019) 420–429

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Blockchain based searchable encryption for electronic health record sharing

Lanxiang Chen [a], Wai-Kong Lee [b], Chin-Chen Chang [c], Kim-Kwang Raymond Choo [d,*], Nan Zhang [a]

[a] College of Mathematics and Informatics, Fujian Normal University, Fujian Provincial Key Laboratory of Network Security and Cryptology, Fujian Network & Information Security Industry Technology Development Base, Fuzhou, China
[b] Faculty of Information and Communication Technology, University Tunku Abdul Rahman, Kampar, Perak, 31900, Malaysia
[c] Department of Information Engineering and Computer Science, Feng Chia University, Taiwan
[d] Department of Information Systems and Cyber Security and Department of Electrical and Computer Engineering, The University of Texas at San Antonio, San Antonio, TX, USA

HIGHLIGHTS

- Blockchain based searchable encryption for electronic health record sharing.
- File encryption, index construction, transaction generation and searching.
- Designated smart contract in blockchain to facilitate monetary rewarding.

ARTICLE INFO

Article history:
Received 10 June 2018
Received in revised form 27 November 2018
Accepted 13 January 2019
Available online 19 January 2019

Keywords:
Searchable encryption
Electronic health records
Blockchain
Smart contract
Ethereum

ABSTRACT

Data leakage in electronic health records (EHRs) could result in the compromise of patient privacy (e.g. medical conditions). Generally most data in EHRs remain unchanged once they are uploaded to the system; thus, blockchain can be potentially used to facilitate the sharing of such data. Different participating medical organizations and individuals (e.g. medical practitioners, hospitals, medical labs and insurance companies) can then access EHRs stored on the blockchain with a higher level of confidence. In this paper, a blockchain based searchable encryption scheme for EHRs is proposed. The index for EHRs is constructed through complex logic expressions and stored in the blockchain, so that a data user can utilize the expressions to search the index. As only the index is migrated to the blockchain to facilitate propagation, the data owners have full control over who can see their EHRs data. The use of blockchain technology ensures the integrity, anti-tampering, and traceability of EHRs' index. Finally, the performance of the proposed scheme is evaluated from two aspects, namely in terms of the overhead for extracting the document IDs from EHRs and the overhead associated with conducting transactions on smart contract in Ethereum.

1. Introduction

Electronic health record (EHR) is a collection of individuals' health related information, which includes medical conditions (e.g. diseases), medication, medical images and personal information (e.g. name, age, gender, weight, and billing information). Such data are generally sensitive and needed to be protected against unauthorized access; thus, one of the biggest challenges in healthcare systems is to share medical data securely (i.e. without resulting in leakage of patient data).

One common or naïve method to share medical data is to construct index for EHRs and encrypt EHRs prior to uploading them to the public or community cloud. The drawback of this approach is that different data providers have their own ways of creating indices, which in turn impedes data sharing among different medical organizations and individuals. Moreover, the cloud server may not be fully trusted. A general verification mechanism that can be applied to all search schemes is also lacking in the current literature. There is also no effective countermeasure to penalize a misbehaving server or user.

Blockchain based solution is a viable approach that allows one to build upon cryptographic algorithms to ensure data integrity, standardized auditing, and some formalized contracts for data access. Therefore, a blockchain based searchable encryption scheme

* Corresponding author.
E-mail addresses: lxiangchen@fjnu.edu.cn (L. Chen), wklee@utar.edu.my (W.-K. Lee), alan3c@gmail.com (C.-C. Chang), raymond.choo@fulbrightmail.org (K.-K.R. Choo), 1427703523@qq.com (N. Zhang).

https://doi.org/10.1016/j.future.2019.01.018
0167-739X/

for EHRs sharing is proposed in this paper. This scheme is designed to facilitate different healthcare institutions to share medical records in a secure manner. Our proposed solution not only brings convenience to patients, but also allows efficient sharing of medical information among researchers. The data user can receive accurate/correct search results with assurance, and know that any malicious activity (e.g. by a malicious server) can be identified during an audit or other investigation. To enable doctors and researchers to access patients' health data without disclosing their personal data, the desensitization technique should be used before the information is shared among others.

Similar to the approach of Hu et al. [1], fairness is also introduced to our scheme through smart contracts to achieve a financially-fair search mechanism. In the proposed scheme, every participant is treated equally and incentivized to perform the correct computations. In this way, an honest party can always gain what (s)he deserves while a malicious one gets nothing.

In our proposed system, only the search index is added to the blockchain to facilitate the distribution of EHRs, while the real EHRs are stored in a public cloud server in an encrypted form. When users want to access these EHRs, they need to authenticate themselves to the data owner to obtain the authorization together with the decryption key. With this arrangement, the data owner has full control over who can see their data. The contributions of this paper are summarized as follows:

- A blockchain based searchable encryption scheme for EHRs sharing is proposed. Specifically, we demonstrate the potential of using smart contract such as Ethereum [2] in our application.
- The proposed scheme utilizes the complex Boolean expression to extract the EHRs to construct the index, which differs from the scheme in [1]. Specifically, the scheme in [1] only supports single keyword search, but our proposed scheme supports complex query that allows different healthcare agents to request permission to access and interact with the medical records. In other words, our proposed scheme is more practical and robust.
- The smart contract used in the proposed scheme is designed to trace monetary rewards, including transaction fees, among involved parties in the multi-user setting. It ensures that the data owner is paid as long as (s)he reveals the transcript, which allows other users to search the database. At the same time, other users can obtain accurate search results as long as they make the required payments. This property guarantees fairness among the data owners and users.
- Our proposed blockchain based solution also guarantees that data user can receive accurate search results without additional verification. Meanwhile, as only the index is migrated to the blockchain, access to the real EHRs needs to be authenticated by the data owner. Hence, the data owner has full control over who can see their data.

In a real-world setting, EHR data need to be desensitized in order to remove personal information, such as name, identity, and other identifying information. This paper mainly focuses on how to build the index from EHRs and deploy them in the smart contract. Data owner can easily employ conventional symmetric key cryptography to encrypt the relevant EHR data and then outsource the encrypted data to some decentralized file storage network like InterPlanetary File System (IPFS).

The rest of this paper is organized as follows. Related work is discussed in Section 2. Relevant background materials are presented in Section 3. The system model and design goals are described in Section 4. The proposed scheme is presented in Section 5, and its security and performance evaluations are presented in Section 6. Finally, conclusions are drawn in Section 7.

2. Related literature

Since the cloud server is not fully trusted, data should be encrypted before being uploaded to the public cloud. To search on encrypted data, a large number of searchable encryption schemes, such as searchable symmetric encryption (SSE) schemes [3], have been proposed in the literature.

The first SSE scheme, proposed by Song et al. [4], is a non-interactive single keyword based SSE scheme. Specifically, this scheme scans all encrypted files and compares these files with the encrypted searched keyword to determine whether the keyword exists in the scanned files. A key limitation is lack of efficiency, particularly when the data size is large. Since this seminal work, many other different functional SSE schemes have been proposed in the literature. Such schemes include multi-keyword SSE schemes [5–8], dynamic SSE schemes [9–12] and verifiable SSE schemes [13–17].

In most existing schemes, the cloud server is modeled as an honest-but-curious party who will perform the prescribed protocol honestly. In reality, a malicious server may return incomplete search results or modified data. To address this concern, a number of verifiable SSE schemes [13–17] designed to check the correctness and integrity of search results and data are presented in the literature. However, these verifiable SSE schemes are highly dependent on specific constructions.

More importantly, the majority of verifiable SSE schemes only detect malicious behaviors, but they do not include mechanisms to punish the dishonest server(s), should they be detected. In other words, the malicious server may benefit financially by cheating the users. For example, the server receives payment from users but does not fully execute the search protocol, so the user may not gain the expected benefit. Therefore, a more reliable SSE scheme with built-in fairness mechanism is required. We refer interested reader to [3] for a recent survey of SSEs and their properties.

In the context of this paper, for cloud-based electronic health record (EHR) applications, cloud storage is usually provided privately (to preserve privacy and ensure security). This can, however, impede data sharing among users. For instance, if a medical researcher wishes to observe the symptoms for patients infected by Hepatitis B to identify or explore possible treatments, (s)he needs to separately request for access to a (large) number of EHRs from different clouds. This can be challenging, in practice.

Earlier in 2011, Li et al. [18] proposed a fine-grained authorization scheme for users to gain access to the search capabilities from localized trusted authorities according to their attributes. They presented two authorized private keyword search (APKS) schemes, based on hierarchical predicate encryption (HPE), to search on encrypted EHRs in public cloud. Their schemes provide functionalities, such as multi-dimensional, multiple keyword searches with simple range query and delegation and revocation of search capabilities.

In 2014, Xhafa et al. [19] proposed a fuzzy keyword search scheme with anonymous-ABE, designed for a hybrid cloud environment. They used a private cloud as a proxy to securely deploy their EHR data to the public cloud. They also adopted attribute-based encryption (ABE) to achieve fine grained access control. Their fuzzy keyword search was realized by wildcard-based method, and the authors utilized a symbol-based trie-traverse algorithm to improve efficiency.

In 2016, Liu et al. [20] proposed a cloud-based EHR sharing system, which supports fuzzy keyword search. Their search scheme facilitates a medical practitioner (or other user) to quickly retrieve EHRs whose symptoms contain the queried keywords. The ABE algorithm is also adopted to authenticate users based on their attributes. Guo et al. [21] proposed an ABE based fine-grained authorization scheme on a relational database. More recently, Yang

et al. [22] proposed a dynamic SSE scheme that supports forward privacy and delegated verifiability for EHR data. The authors utilized cloud storage and Internet of Things (IoT) to facilitate remote patient monitoring.

The EHR search systems described above either do not support data sharing or they are designed for a specific construction/application. In addition, the majority of these schemes use ABE mechanism to perform fine-grained access control. As different schemes have different ways of creating indices, these are not general methods to share EHRs even though they are based on ABE.

Recently in 2017, Li et al. [23] proposed a blockchain based SSE scheme, wherein blockchain is utilized as a peer-to-peer network to store user data in a pay-per-use way. Each user has equal status in the decentralized system and (s)he requests other users to store his/her data by submitting a transaction. The user (also called a worker or miner) who adds these data as a block to the blockchain will get rewards, whereby these data are stored in a public chain. However, this scheme only supports single keyword search.

Hu et al. [1] proposed a smart contract based SSE scheme. In the proposed scheme, the index of user files is stored in smart contract in the peer-to-peer network, while these files can be stored to any public cloud storage system. They introduced the concept of fairness, which can guarantee that an honest party can always gain what (s)he deserves while a malicious one gets nothing. This approach does not require any verification but it can ensure that the user receives the correct search results. They implemented a prototype of the proposed scheme based on Ethereum to illustrate the practicability of the decentralized search scheme over encrypted data. However, this scheme also only supports single keyword search.

Unlike the approaches presented in both [1] and [23], a blockchain based searchable encryption scheme for EHRs is proposed in this paper. The complex logic expressions are utilized to construct the EHRs' index, and these expressions are used by the user to search the index stored in the blockchain. Prior to presenting the detailed construction of the proposed scheme in Section 4, we will present the relevant materials regarded in the understanding of the proposed scheme in the next section.

3. Background

In this section, the smart contract in Ethereum, the gas system and notations will be described.

3.1. Smart contract in Ethereum

Smart contract, first proposed by Nick Szabo [24], is a computer program designed to digitally enforce the negotiation of a contract. Recently, several cryptocurrencies have implemented different types of smart contracts by utilizing cryptographic algorithms and various security protocols. A smart contract is not necessarily related to the classical concept of a contract, but can be any kind of computer program. The digital smart contract facilitates the reliable execution of transactions without involving some third-party and all transactions are trackable and irreversible. In other words, smart contracts provide security that is superior to traditional contract law and reduces transaction costs associated with contracting.

Ethereum [2] is a decentralized platform that runs the smart contract. In Ethereum, the smart contract is used to perform some general purpose computation on a blockchain or distributed ledger. Because of the properties of blockchain, all operations are transparent and reliable in Ethereum. Ethereum is the first blockchain implementation to have a Turing Complete virtual machine built in it. This implies that an Ethereum smart contract can be used to perform any computational task, in theory.

There is an entity known as workers or miners in Ethereum, which validates and approves all transactions in the blockchain. They add all new transactions to the blockchain by solving a cryptographically challenging puzzle – a process referring to as mining of new blocks. Once a new block is successfully mined, the worker is rewarded with newly-created cryptocurrency; thus, such rewards incentivize third-party entities to mine more blocks. The data stored and computations executed on Ethereum must be consistent across miners and cannot be modified or denied. All the stored data and executed computations are transparent to any users. Each smart contract, identified by a special address, consists of script code, a currency balance, and storage space in the form of a key/value store. Therefore, Ethereum acts as a trusted base that is trusted for correctness and availability, but not for privacy.

3.2. Gas system

In Ethereum, the gas system is introduced to prevent the faulty or malicious programs from occupying computing resources, such as the dead loop program. It can resist Denial-of-Service (DoS) attack and achieve Turing complete smart contract. In Ethereum, each transaction has a limited gas consumption, and the system will terminate the transaction when the gas limit is exhausted. Gas is obtained by Ethereum currency exchange and gas consumption is the source of the income for the workers.

As remarked by Hu et al. [1],

> *[T]he contract script is compiled into Ethereum opcodes and stored in the blockchain. Each opcode will cost a certain pre-defined amount of gas. When initiating a smart contract through sending a transaction, the sender has to specify the available "gasLimit" that supports execution, and the corresponding "gasPrice" that the sender is willing to pay for each unit of gas. The transaction will get included in the blockchain successfully only when the balance of the sender is larger than gasLimit*gasPrice.*

Using the gas system, a worker in Ethereum can be financially compensated by successfully solving a designated cryptographic puzzle. If the gas consumption of a transaction is larger than the *gasLimit*, then the transaction will be terminated and the gas consumption will be given to the worker.

When a user invokes a smart contract function, (s)he must declare that there is sufficient gas in the account and be willing to pay that amount. Thus, developers should deploy some efficient code to optimize the use of deployed protocols for users and ensure that appropriate fee is paid for transactions submitted to the network.

3.3. Notations

The notations used in this paper are defined in Table 1.

Two pseudo-random functions are defined as follows and λ is the security parameter.

f: $\{0,1\}^{\lambda} \times \{0,1\}^{*} \rightarrow \{0,1\}^{\lambda}$.

g: $\{0,1\}^{\lambda} \times \{0,1\}^{\lambda} \rightarrow \{0,1\}^{*}$.

4. System model and design goals

In this section, the system model and design goals are presented.

4.1. System model

In the proposed scheme, there are three entities, namely: data owner, user and blockchain. The system model is illustrated in Fig. 1.

The data owner is an entity that creates the EHRs, and this entity can be human (e.g., patient) or an organization (e.g., hospital or

Table 1
Notations.

D	The plaintext documents collection of EHRs, denoted as a set of m documents $D = \{D_1, D_2, \ldots, D_m\}$.
C	The encryption document collection for D, denoted as $C = \{C_1, C_2, \ldots, C_m\}$.
id	The address or identifier of the document allocated by MongoDB.
I	The index of documents collection.
Q	The plaintext query.
T_Q	The trapdoor for the query request.
R	The identifier collection of queried results.
$d \models X$	It indicates that document d satisfies condition expression X.
$\lfloor \cdot \rfloor$	It is a floor function.
$\|$	It is a concatenation symbol.
$\perp$	It is used to denote "NULL".
$Get()$	It is used to get the designated data item from a dictionary.

clinic). The data owner then builds the index for the respective EHRs and creates the smart contract to describe how one can search the index. Once this is completed, the data owner sends both the smart contract and the index to the blockchain. After that, the data owner encrypts the EHRs using a symmetric encryption algorithm and stores them at the cloud server.

The user is an entity that is authorized by the data owner to search the index to obtain the required EHRs. The user can be human (e.g., medical practitioner) or an organization (e.g., hospital, medical research institute, or health insurance company). The blockchain is an entity that stores the index and all smart contracts. The authorized users search the blockchain for some specific EHRs, and the smart contract should yield a correct and immutable result, which requires no further verification by the users.

4.2. Design goals

The design goals of the proposed scheme are similar to the scheme in [1]. We introduce the blockchain to achieve a confidential, fair, sound and controllable searchable scheme for sharing of EHRs. There are three goals for the proposed scheme, namely fairness, soundness and confidentiality.

Fairness. The fairness defined in this paper is similar to [1,25,26]. The property of fairness guarantees that the user will receive accurate search results if the user pays for the query task, which will be performed by a miner or worker, and the miner or worker will be rewarded by running the protocol correctly. In addition, in order to access the EHRs the user will pay the data owner who sends the search token to the user. **Soundness**. It means that if there is a dishonest entity that does not perform the protocol in a predefined way, it will be detected and it obtains no reward. In previous schemes, this is achieved by using a verification approach.

Confidentiality. In our scheme, as the newly added document is independent of previous documents, there is no forward privacy problem. In addition, the proposed scheme is applied to index of the EHRs, and the actual EHR data are stored to any other public storage system and can be protected by any other privacy preserving methods. Thus, we only need to protect the confidentiality of the query expressions from the adversary.

In the proposed scheme, each query Q is a complex expression and different queries are independent of each other. The search algorithm will return all identifiers that satisfy the query, and we assume these identifiers will be transferred via a secure channel to the user. In addition, we can encrypt these identifiers.

5. The construction

In this section, the formal definition and specific construction of the proposed scheme are described.

Table 2
An example of EHRs.

User ID	Name	Gender	Age	Phone	Disease	Images	Medication
01	Zhang	1	25	*****	*****	*****	*****
02	Li	0	33	*****	*****	*****	*****
03	Wang	1	22	*****	*****	*****	*****
04	Chen	0	47	*****	*****	*****	*****

5.1. Formal definition

The formal definition of the proposed scheme is defined as follows.

Definition 5.1. The proposed scheme is composed of five polynomial-time algorithms, namely: *Setup*, *BuildIndex*, *Enc*, *Trapdoor*, *Search*:

$(mk, sk) \leftarrow Setup(1^\lambda)$: It is run by the data owner to setup the scheme. It takes a security parameter λ as input, and outputs the master key mk and the secret file encryption key sk.

$I \leftarrow BuildIndex(mk, D)$: It is run by the data owner to generate the index I. It takes the master key mk and file collection D as input, and outputs the searchable index I.

$C \leftarrow Enc(sk, D)$: It is run by the data owner to encrypt documents. It takes the secret file encryption key sk and file collection D as input, and outputs encrypted document collection C.

$T_Q \leftarrow Trapdoor(mk, Q)$: It is run by the data owner to generate the trapdoor for the authorized user. It takes mk and query Q as input, and outputs the trapdoor T_Q.

$R \leftarrow Search(T_Q, I)$: It is run by the smart contract. It takes the trapdoor T_Q and index I as input, and outputs the identifier list of relevant encrypted documents R.

5.2. Scheme construction

Suppose all EHRs are stored on a NoSQL database, MongoDB. The form of the plaintext electronic health records (EHRs) is illustrated in Table 2. In MongoDB, each EHR is seen as a document and has a unique document identifier.

In the proposed scheme, the processes of file encryption, index construction, transaction generation and search are illustrated in Fig. 2.

The data owner scans all electronic health records (EHRs) and extracts all records that satisfy the condition expression: $X =$ "(disease = 'disease name') AND ($num_1 \leq$ age $< num_2$) AND (gender = 'male or female') AND (hospital = 'hospital name') AND ($time_1 \leq$createTime $< time_2$)". We can define 'age' as between 0 to 100, and num_2- $num_1 = 10$. To make the expressions different from each other, the 'create time' should also be included in the expressions. The 'create time' interval is set to be one year.

For example, X_i = "(disease = 'hepatitis B') AND (20 $\leq$ age $<$ 30) AND (gender= 'female') AND (hospital = 'Union Medical College Hospital') AND ('2000-01-01T00:00:00Z' $\leq$ createTime $<$

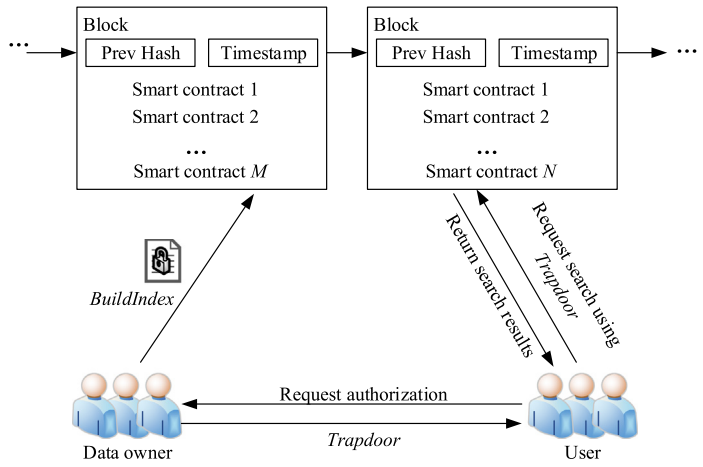

Fig. 1. System model.

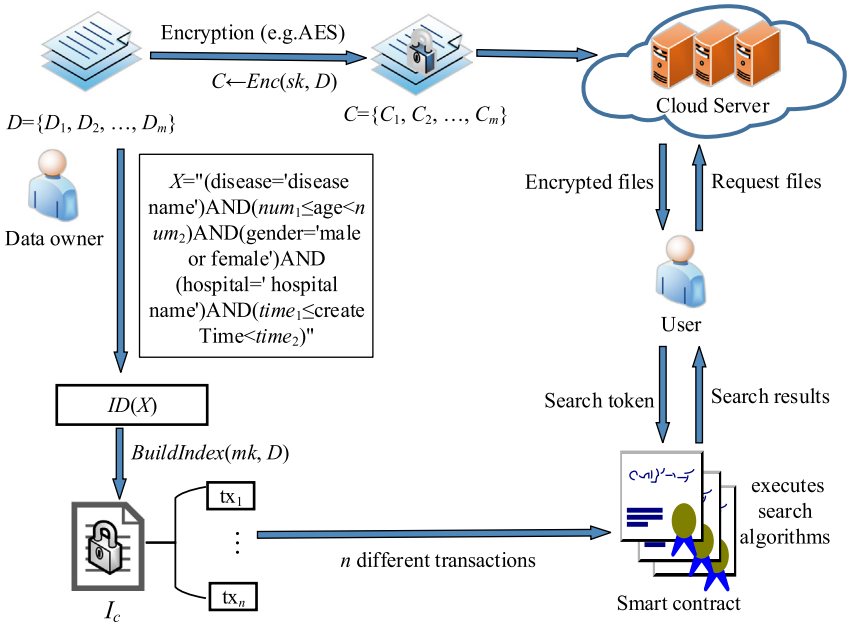

Fig. 2. Processes of file encryption, index construction, transaction generation and search.

'2001-01-01T00:00: 00Z')". To extract all records that satisfy the condition expression X_i, one can query the MongoDB as follows.

db.collection.find({"disease": "hepatitis B"}, {"age": {"$gte": 20, "$lt": 30}}, {"gender": " female"}, {"hospital": "Union Medical College Hospital"}, {"createTime": {"$gte": ISODate("2000-01-01T00: 00:00Z"), "$lt": ISODate("2001-01-01T00:00:00Z")}}).

All these expressions are denoted as $X = \{X_1, X_2, \ldots\ldots, X_m\}$. $ID(X_i) = \{id_{ij} | D_{ij} \models X_i\}$ denotes all document identifiers if the document satisfies the expression X_i. The data owner calls the *BuildIndex* algorithm to generate the index I based on the extracted identifiers. Then, the data owner encrypts all plaintext documents to obtain the encrypted documents. The encrypted document collection can be outsourced to any decentralized file storage network, such as InterPlanetary File System (IPFS) [27].

When the user wishes to search some EHRs, (s)he first authenticates himself/herself to the data owner and obtains the search token after authorization. Then, the user will utilize the token to request for the search to the smart contract. The smart contract will use the token to search the blockchain to get the corresponding results and return them to the user. The scheme construction is presented in Fig. 3. Here, $\$Bowner$ and $\$Buser$ denote the deposit account of the data owner and user, respectively. $\$deposit$ denotes the deposit currency by the user who will search the blockchain. $\$offer$ denotes the price for each search that will be rewarded to the data owner. $\$gasPrice$ denotes price for each unit of gas. GL_{srch} and G_{srch} denote gas limit and cost for calling *Search*() function, respectively.

In the setup phase, the reward for the data owner is predefined as $\$offer$ and the cost for mining a block is predefined as the amount of gas. The time limitation should be practical, so that the transaction can be processed appropriately. The identifier packing approach is similar to that of [1], which is inspired by [28]. This mechanism can ensure that each transaction consumes less gas than *gasLimit*. In Ethereum, each transaction has an upper bound of consumed gas, called *gasLimit*.

Assume $|id_i| = e$, $p^*e \leq \lambda$, p is a system parameter chosen by the data owner. We use concatenation to pack multiple file identifiers into one. To ensure confidentiality, the bit length of *id* should be less than that of the security parameter λ. Therefore, we have $p \leq \lambda/e$, where e is the bit length of the file identifier.

In the search phase, the cost of each transaction includes two parts, namely the reward to the data owner (i.e. $\$offer$) and the

***Setup*(1^λ):**

(1) The data owner generates the master secret key $mk \leftarrow \{0, 1\}^\lambda$ and document encryption key $sk \leftarrow \{0, 1\}^\lambda$ randomly.

(2) The data owner sets a price $\$offer$ for each search.

(3) The user makes a deposit $\$deposit$ from $\$Buser$.

(4) The user sets a time limitation T_1.

***I*←*BuildIndex*(*mk*, *D*):**

(1) Scan EHRs and extract all records that satisfy the condition expression $X=\{X_1, X_2, \ldots\ldots, X_m\}$, and get $ID(X_i)=\{id_{ij} \mid D_{ij} \models X_i\}$.

(2) Initialize an empty list L and an empty dictionary I. For each $X_i \quad X$:

(a) $k_1 \leftarrow f(mk, 1\|X_i)$; $k_2 \leftarrow f(mk, 2\|X_i)$;

(b) Set $a = \lfloor |ID(X_i)|/p \rfloor$, $c \leftarrow 0$, where p denotes the number of file identifiers that can be packed;

(c) Partition $ID(X_i)$ into $a+1$ segments. Pad the last segment to p entries if needed;

(d) For each segment in $ID(X_i)$:

– $id=id_1\|id_2\|\ldots\ldots\|id_p$, $r \leftarrow \{0, 1\}^\lambda$, $d \leftarrow id \quad g(k_2, r)$, $l \leftarrow f(k_1, c)$, c++;

– Add (l, d, r) to the list L.

(3) Partition L into n blocks L_i for $1 \le i \le n$, and send them to the smart contract one by one with n different transactions.

(4) The smart contract parses each entry in L_i into (l, d, r), and adds each $(l, d\|r)$ to I.

***C*←*Enc*(*sk*, *D*):**

It is executed by the data owner to encrypt the document collection. It uses a symmetric encryption algorithm (e.g. AES) with the secret key sk to encrypt the plaintext document collection D to obtain the ciphertext collection C.

***T_Q*←*Trapdoor*(*mk*, *Q*):**

A user sends a query request Q to the data owner. The owner first checks whether the request Q satisfy the format of the defined expression. If yes, she estimates t and $step$, then computes $k_1 \leftarrow f(mk, 1\|Q)$, $k_2 \leftarrow f(mk, 2\|Q)$ and sends $T_Q=(k_1, k_2, t, step)$ to the user. Otherwise, she will return "Expression Error" to the user.

***Search*(T_Q, *I*):**

(1) Assert current time $T < T_1$. Otherwise, turn to (6).

(2) Assert $\$deposit > GL_{srch} \times \$gasPrice + \$offer$.

(3) The user sets $c \leftarrow 0$ and sends search token (k_1, k_2, c) to the smart contract.

(4) For $i = 0$ to t:

(a) For $j = 0$ until Get returns $\perp$ or $j \ge step$:

– $l \leftarrow f(k_1, c)$; $d, r \leftarrow Get(I, l)$; $id \leftarrow d \quad g(k_2, r)$, c++; j++.

– Parse id into $(id_1, id_2, \ldots\ldots id_p)$ and save them to the list R.

(b) Set $\$cost \leftarrow \$offer + G_{srch} \times \$gasPrice$.

(c) Send $\$offer$ to $\$Bowner$ and $G_{srch} \times \$gasPrice$ to the worker who performs the transaction.

(d) Set $\$deposit \leftarrow \$deposit - \$cost$.

(e) The smart contract asserts that the estimated gas cost is lower than the balance. Assert $\$deposit > GL_{srch} \times \$gasPrice + \$offer$ and then go to (4a). Otherwise, turn to (5).

(5) Send $\$deposit$ to $\$Buser$.

(6) Assert current time $T > T_1$. Then send $\$deposit$ to $\$Buser$.

Fig. 3. Scheme construction.

reward to the worker (i.e. $G_{srch} \times \$gasPrice$). Within the predefined time limitation T_1, the data owner can obtain the reward from the trapdoor generation. Otherwise, the search request by the user is overdue and the user's deposit will be refunded. Note that each contract has a unique address in Ethereum. With the search token and previously stored index, the smart contract executes search algorithms and saves the search results (i.e., file identifiers) to its state, which is known publicly including the data owner.

The proposed scheme only focuses on the accuracy of query, but it does not consider access to the data of each EHRs. This can be achieved using existing file sharing schemes, for example using ABE algorithm to achieve fine-grained access control [18,19,29].

6. Security analysis and performance evaluation

The security analysis and performance evaluation are presented in this section.

6.1. Security analysis

Fairness: This goal is achieved by using the incentive mechanism of the blockchain. As in Ethereum, each user contributes his/her computing capability to add new block(s) to the blockchain and be rewarded (e.g., financially) from the completed work. In other words, all transactions are paid through buying gas. The malicious operation will be detected and the dishonest user will

get nothing in return. However, existing verifiable searchable encryption schemes [13–17] do not support this property, because the user already paid before the server starts searching. After they received the search results and found that these results are incorrect. However, there is no effective countermeasure to punish the dishonest entity. In addition, the time limitation T_1 specified by the user can ensure fairness from the aspect that the transaction should be completed in this time; otherwise, the user's deposit will be refunded.

Soundness: In the proposed scheme, the consensus characteristic of blockchain can guarantee that the user can obtain reliable and correct search results without verification. As long as the smart contract is correctly run on Ethereum, the search results will be stored as contract states permanently and publicly. Any change on the search results can be detected by each node in the Ethereum network.

Confidentiality: The proof of confidentiality for the query expressions in our scheme is similar to [1]. Since there is no updating operation (add and delete), it is simpler compared to the scheme of [1]. We also use real-ideal simulation game between a simulator S and an adversary A and introduce two stateful leakage functions $L = (L_1, L_2)$.

(1) L_1 is defined as $L_1(D) = (\sum_{X_i \in X} \left\lceil \frac{|ID(Xi)|}{p} \right\rceil, m, \{|D_j|\}_{Dj \in D}, \{id(|D_j|)\}_{Dj \in D})$. These notations are defined in Table 1.

(2) L_2 is defined as $L_2(D, Q) = (ID(Q) = \{id_j | D_j \models Q\}, T_Q)$. Input the document collection D and a query expression Q, it outputs the search pattern and the trapdoor of the expression T_Q.

Two simulation games $\text{Real}_A^{\Pi}(\lambda)$ and $\text{Ideal}_{A,S}^{\Pi}(\lambda)$ are conducted by a challenger with an adversary and a simulator respectively.

(1) $\text{Real}_A^{\Pi}(\lambda)$: The challenger runs the algorithm $Setup(1^\lambda)$ to generate the master secret key $mk \leftarrow \{0, 1\}^\lambda$ and document encryption key $sk \leftarrow \{0, 1\}^\lambda$, then the adversary A sends the document collection D to the challenger. The challenger executes algorithms $I \leftarrow BuildIndex(mk, D)$ and $C \leftarrow Enc(sk, D)$ to generate the index and ciphertexts and gives them to A. A makes a number of adaptive queries, and the adversary could receive the corresponding trapdoor T_Q from the challenger. Finally, A returns a bit as the output to indicate A wins or not.

(2) $\text{Ideal}_{A,S}^{\Pi}(\lambda)$: The adversary A chooses a document collection D. The simulator S outputs the index and ciphertexts and sends them to A when the simulator gets the leakage algorithm $L_1(D)$. A makes a number of queries Q. For each Q, the simulator makes the corresponding trapdoor and sends it to A when given the leakage algorithm $L_2(D, Q)$. Finally, A returns a bit as the output to indicate A wins or not.

We say that the proposed scheme Π is L-secure against non-adaptive attacks, if for any probabilistic polynomial-time (PPT) adversary A, there exists a PPT simulator S such that,

$$|\Pr[\text{Real}_A^{\Pi}(\lambda) = 1] - \Pr[\text{Ideal}_{A,S}^{\Pi}(\lambda) = 1]| \leq \text{negl}(\lambda),$$

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

Theorem 6.1. *If the pseudo-random functions f and g are pseudo-random, then the proposed scheme Π is L-secure against non-adaptive attacks.*

Proof. Similar to the security definition in [30], non-adaptive indistinguishability of SSE is equivalent to non-adaptive semantic security of SSE. It is equivalent to proving that for all PPT adversaries, there exists a PPT simulator S such that the advantage to distinguish the outputs of $\text{Real}_A^{\Pi}(\lambda)$ and $\text{Ideal}_{A,S}^{\Pi}(\lambda)$ is negligible.

$\text{Ind}_A^{\Pi}(\lambda)$ denotes a probabilistic experiment, if there exists all adversaries that $\Pr[\text{Ind}_A^{\Pi}(\lambda) = 1] \leq 1/2 + \text{negl}(\lambda)$. We can say that the scheme is secure in the sense of non-adaptive indistinguishability and then satisfies non-adaptive semantic secure. The adversary could win the game by analyzing the secret keys, the encrypted index, the encrypted documents, and the linkability of trapdoors.

As the generation of document encryption key sk and the encryption process both are performed by the data owner, if the encryption key can be kept secure, we consider the encrypted documents are secure. So we have

$$\Pr[\text{Ind}_A^{\Pi}(\lambda) = 1] = 1/2 + \text{Adv}(A(mk)) + \text{Adv}(A(I)) + \text{Adv}(A(T_Q)),$$

where $\text{Adv}(A(mk))$ and $\text{Adv}(A(I))$ denote the advantage of adversary A to distinguish the master secret key and the encrypted index from random strings. $\text{Adv}(A(T_Q))$ denotes the advantage of adversary to find out the relevance from the different trapdoors.

As the master key is generated randomly, there exists a negligible function $\text{neg}\, l_1(\lambda)$ such that

$$\begin{aligned}\text{Adv}(A(mk)) &= |\Pr[Setup(1^\lambda) \rightarrow mk] - \Pr[Random \rightarrow mk']| \\ &\leq \text{negl}_1(\lambda).\end{aligned}$$

In the algorithm $I \leftarrow BuildIndex(mk, D)$, the index is consisted of $(l, d \parallel r)$, and $r \leftarrow \{0, 1\}^\lambda$, $d \leftarrow id \oplus g(k_2, r)$, and $l \leftarrow f(k_1, c)$. If f and g are pseudo-random, then all of them are random. So we have

$$\begin{aligned}\text{Adv}(A(I)) &= |\Pr[BuildIndex(mk, D) \rightarrow I] - \Pr[Random \rightarrow I']| \\ &\leq \text{negl}_2(\lambda).\end{aligned}$$

In the algorithm $T_Q \leftarrow Trapdoor(mk, Q)$, $T_Q = (k_1, k_2, t, step)$, $k_1 \leftarrow f(mk, 1 \parallel Q)$, and $k_2 \leftarrow f(mk, 2 \parallel Q)$. For different queries Q_1 and Q_2, the trapdoors T_{Q1} and T_{Q2} are different and they are independent with each other. Thus we have

$$\begin{aligned}\text{Adv}(A(T_w)) &= |\Pr[Trapdoor(mk, Q) \rightarrow T_Q] - \Pr[Random \rightarrow T'_Q]| \\ &\leq \text{negl}_3(\lambda).\end{aligned}$$

Then, we have

$$\begin{aligned}\Pr[\text{Ind}_A^{\Pi}(\lambda) = 1] &= 1/2 + \text{Adv}(A(mk)) + \text{Adv}(A(I)) + \text{Adv}(A(T_Q)) \\ &= 1/2 + |\Pr[Setup(1^\lambda) \rightarrow mk] - \Pr[Random \rightarrow mk']| \\ &\quad + |\Pr[BuildIndex(mk, D) \rightarrow I] - \Pr[Random \rightarrow I']| \\ &\quad + |\Pr[Trapdoor(mk, Q) \rightarrow T_Q] - \Pr[Random \rightarrow T'_Q]| \\ &\leq 1/2 + \text{negl}_1(\lambda) + \text{negl}_2(\lambda) + \text{negl}_3(\lambda).\end{aligned}$$

Let $\text{negl}(\lambda) = \text{negl}_1(\lambda) + \text{negl}_2(\lambda) + \text{negl}_3(\lambda)$. Then we have

$$\Pr[\text{Ind}_A^{\Pi}(\lambda) = 1] \leq 1/2 + \text{negl}(\lambda).$$

Thus, the proposed scheme Π is L-secure against non-adaptive attacks. □

6.2. Performance evaluation

Findings from the performance evaluation is presented in this section. Our proposed scheme applies blockchain to the EHRs and uses a condition expression to extract document IDs to build index, instead of building single keyword based index as proposed in [1]. Overheads incurred by the proposed scheme are mainly due to extraction of the document IDs from the EHRs and the transactions on smart contract in Ethereum. Prior to presenting our evaluation findings, we will first provide a comparative summary of our proposed scheme and the scheme of Hu et al. [1].

From the perspective of query functionalities, the proposed scheme supports complex Boolean expression and range query, while the scheme of Hu et al. only supports single keyword search. As previously discussed, as data in EHRs typically remain unchanged once they are uploaded to the system; thus, we do not need to consider dynamic data updating operations. In addition, our scheme is designed specifically for EHRs unlike the general setting assumed in the scheme of Hu et al. [1] – see Table 3.

Table 3
Key differences between our proposed scheme and the scheme of Hu et al. [1].

Scheme	Query Support	Application Context
Hu et al. [1]	Single keyword	General
Ours	Boolean, range	EHRs

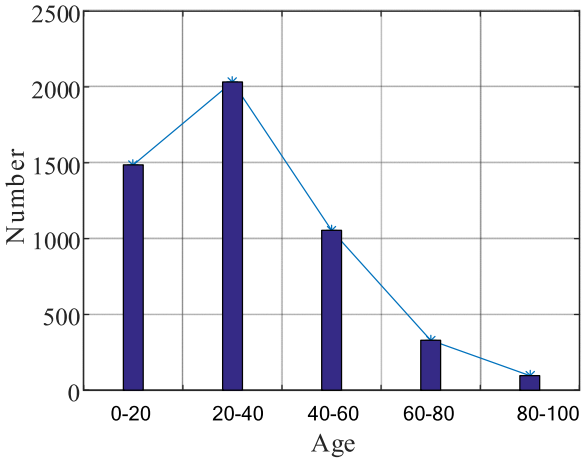

Fig. 4. Age distribution of 5000 EHR documents.

The index structures in both schemes are inverted index and the indices are stored to smart contracts. Also, both schemes support multi-user setting. During index construction, the proposed scheme will extract Boolean expression from EHRs, while the scheme of Hu et al. will extract single keyword from user data. Therefore, the overhead of index construction in both schemes differs.

6.2.1. Overhead incurred during the extraction of EHRs

Experiments are performed on a computing system with an Intel core i5-7200U 2.5 GHz processor, 8 GB memory, and Windows 10 (64 bit) operation system, coding implemented in Python 2.7 and MongoDB 3.4 enterprise edition.

As there is no publicly available EHR database, the Nursery dataset from University of California, Irvine (UCI) Machine Learning Repository [31] is selected as the test dataset. It has also been used in prior work on searchable encryption, such as [18].

The Nursery dataset has eight categorical attributes and one class attribute, and each attribute has up to 5 values. In the evaluation, each attribute is regarded as a keyword field and each attribute value as a keyword. The original dataset contains 12,960 records in total. A document identity field is also added to the Nursery dataset, which allows us to extract document identities after each MongoDB query. In addition, the original meaning of attributes is adjusted to the corresponding field of the EHRs.

To evaluate the overhead of extracting document IDs that satisfy the specific expression from the MongoDB, five groups of datasets that contain 1000, 2000, 3000, 4000, 5000 EHR documents are evaluated. For different size of EHR documents, the distribution of different age groups is similar and the number of people in the 20–40 age group is the most. The age distribution of 5000 EHR documents is illustrated in Fig. 4. As the size of the dataset is small, the gender is not divided further. The disease field is queried but it consists of the same disease.

The overhead of extracting all the EHR identifiers for different size of documents is presented in Fig. 5. The time for extracting the EHR identifiers from 1000 and 5000 documents is about 0.138 s and 0.471 s, respectively. Thus, it is clear that extracting document IDs from MongoDB is very efficient.

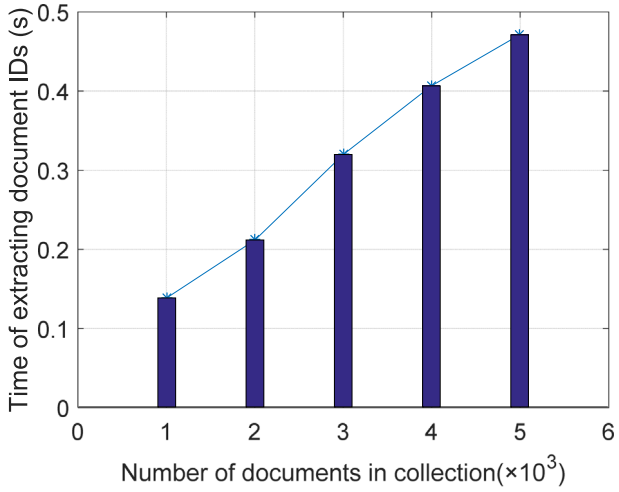

Fig. 5. Overhead of extracting all the EHR identifiers for different size of documents.

Table 4
Characteristics of datasets used in the experiments and [1].

Dataset name	Keyword–ID pairs	Distinct keywords	Dataset size
DS1	100 763	22 673	5.4 MB
DS2	300 617	54 980	14.1 MB
DS3	500 567	75 924	21.3 MB
DS4	1000 141	123 912	39 MB

6.2.2. Overhead incurred during the execution of smart contract in Ethereum

The smart contract used in the proposed scheme is the same as that of [1]; thus, the performance is also similar. In [1], a machine configured with 4 Intel cores i7-3770, a 16 GB RAM and an Ubuntu 16.04.2 operation system was used as the data owner, and the experiments implemented in Python Solidity and JavaScript.

To avoid exceeding *gasLimit*, the encrypted database is divided into n subsets and sent to the smart contract through n transactions. The pack number is set as $p = 8$, each search is completed with $t = 4$ transactions at most, and each of them returns $step = 47$ items at most. The datasets derived from Enron emails are used in [1] and these datasets are summarized in Table 4. Similar to our scheme, the keyword is our condition expression.

On the locally simulated Ethereum network *TestRPC*, the time for mining the block is set to be 0. We refer reader interested in the time overheads and transaction numbers for each phase on the different datasets to [1]. Unlike existing centralized search schemes, the time overhead of smart contract is much higher than that of the data owner. This is because storing all data requires thousands of transactions, with each transaction costing about 4 s on average. The search time for each located document with varying number of matching records is illustrated in Fig. 6. From the figure, one can observe a larger result set yields a lower search overhead, and the search algorithm is slower for larger dataset because a larger number of mined blocks leads to a longer time for loading.

On the official Ethereum test network *Rinkeby*, due to the limited balance, experiments are performed on the smallest database DS1. According to the parameters $p = 8$ and $t = 4$, there are 350 transactions for database DS1. The average block time for mining is 15 s and it consumes 88 min to complete the entire setup phase. It means that the time is mainly spent on smart contract, rather than the data owner in traditional SSE scheme.

After receiving a search token, the search time is about 20 s, 21 s, and 23.5 s for 100, 210, and 290 matched documents, respectively

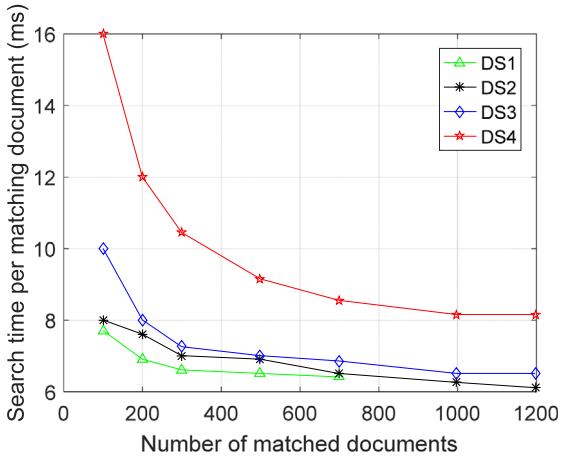

Fig. 6. Search time per matching document in *TestRPC*. *Source:* Data stem from [1].

in one transaction. The search time is about 39.5 s and 40 s for 430 and 515 matched documents, respectively in two transactions. The search time is about 57 s and 56 s for 630 and 708 matched documents, respectively in three transactions. It can be seen that the search time grows with the number of matched documents and transactions.

6.2.3. Other overheads

The process of index construction comprises extracting of document identifiers and packing of these identifiers for inclusion in the smart contract. Thus, the time overhead of index construction consists of two parts. However, without access to a publicly available EHR database or a collaborating hospital, we are unable to evaluate the actual performance of our proposed approach. It does appear from our evaluations that the index construction is efficient. For 5000 documents, the time for extracting the EHR identifiers is about 0.471 s, and 15 transactions to append them to the smart contract for $p = 8$ and $step = 47$. In practice, the time overhead is mainly incurred at the smart contract, and hence it is efficient for the data owner.

Overhead associated with trapdoor generation includes estimation of t and $step$, which will determine the number of transactions and two pseudo-random functions. Therefore, the overhead is independent of the number of documents and it is constant. Overhead incurred during the search (see Fig. 6) is mainly depends on the number of transactions, and this is dominated by the operations on the smart contract.

7. Conclusion

Blockchain is a relatively recent trend, which has promises in a number of applications for both civilian and military contexts. Inspired by the approach of Hu et al. [1], a blockchain based searchable encryption for EHRs sharing scheme was proposed in this paper. Utilizing a designated smart contract in blockchain to replace the centralized server, we were able to achieve a reliable and confidential search scheme without any verification mechanism. Leveraging blockchain, the proposed scheme attains fairness in the sense that honest users (and not the malicious entity) will be rewarded. The security analysis and performance evaluations suggested that the proposed scheme is feasible and effective.

Future research includes implementing a proof-of-concept of the proposed scheme and evaluating it in a real-world environment, for example in a smart campus setting of the authors. This will allow us to evaluate its utility in a real-world setting, as well as its scalability across different institutions and countries.

Acknowledgments

This work was supported by the Natural Science Foundation of China (No. 61602118, No. 61572010 and No. 61472074), Fujian Normal University, China Innovative Research Team (No. IRTL1207), Natural Science Foundation of Fujian Province, China (No. 2019J01329, No. 2017J01738), and the Cloud Technology Endowed Professorship, USA.

References

[1] S. Hu, C. Cai, Q. Wang, C. Wang, X. Luo, Kui. Ren, Searching an encrypted cloud meets blockchain: A decentralized, reliable and fair realization, in: Proceedings of IEEE INFOCOM, 2018.
[2] G. Wood, Ethereum: A secure decentralised generalised transaction ledger, Ethereum Project Yellow Paper, 151, 2014.
[3] G.S. Poh, J.-J. Chin, W.-C. Yau, K.-K.R. Choo, M.S. Mohamad, Searchable symmetric encryption: Designs and challenges, ACM Comput. Surv. 50 (3) (2017) 40:1–40:37.
[4] D.X. Song, D. Wagner, A. Perrig, Practical techniques for searches on encrypted data, in: Proceedings of IEEE S & P, 2000, pp. 44–55.
[5] N. Cao, C. Wang, M. Li, K. Ren, W. Lou, Privacy-preserving multi-keyword ranked search over encrypted cloud data, IEEE Trans. Parallel Distrib. Syst. 25 (1) (2014) 222–233.
[6] B. Wang, S. Yu, W. Lou, Y.T. Hou, Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud, in: Proceedings of IEEE INFOCOM, 2014, pp. 2112–2120.
[7] L. Chen, N. Zhang, K.-C. Li, S. He, L. Qiu, Improving file locality in multi-keyword top-k search based on clustering, Soft Comput. 22 (9) (2018) 3111–3121.
[8] L. Chen, L. Qiu, K.-C. Li, S. Zhou, A secure multi-keyword ranked search over encrypted cloud data against memory leakage attack, J. Internet Technol. 19 (1) (2018) 167–176.
[9] S. Kamara, C. Papamanthou, T. Roeder, Dynamic searchable symmetric encryption, in: Proceedings of ACM CCS, 2012, pp. 965–976.
[10] F. Hahn, Fl. Kerschbaum, Searchable encryption with secure and efficient updates, in: Proceedings of ACM CCS, 2014, pp. 310–320.
[11] M. Naveed, M. Prabhakaran, C.A. Gunter, Dynamic searchable encryption via blind storage, in: Proceedings of IEEE S & P, 2014, pp. 639–654.
[12] L. Chen, L. Qiu, K.-C. Li, W. Shi, N. Zhang, DMRS: an efficient dynamic multi-keyword ranked search over encrypted cloud data, Soft Comput. 21 (16) (2017) 4829–4841.
[13] Q. Chai, G. Gong, Verifiable symmetric searchable encryption for semi-honest-but-curious cloud servers, in: Proceedings of IEEE ICC, 2012, pp. 917–922.
[14] K. Kurosawa, Y. Ohtaki, How to update documents verifiably in searchable symmetric encryption, in: Proceedings of CANS, Springer, 2013, pp. 309–328.
[15] W. Sun, B. Wang, N. Cao, M. Li, W. Lou, Y.T. Hou, H. Li, Verifiable privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking, IEEE Trans. Parallel Distrib. Syst. 25 (11) (2014) 3025–3035.
[16] W. Sun, X. Liu, W. Lou, Y.T. Hou, H. Li, Catch you if you lie to me: Efficient verifiable conjunctive keyword search over large dynamic encrypted cloud data, in: Proceedings of IEEE INFOCOM, 2015, pp. 2110–2118.
[17] X. Chen, J. Li, J. Weng, J. Ma, W. Lou, Verifiable computation over large database with incremental updates, IEEE Trans. Comput. 65 (10) (2016) 3184–3195.
[18] M. Li, S. Yu, N. Cao, W. Lou, Authorized Private Keyword Search over Encrypted Data in Cloud Computing, in: Proceedings of IEEE ICDCS, 2011, pp. 383–392.
[19] F. Xhafa, J. Wang, X. Chen, J.K. Liu, J. Li, P. Krause, An efficient PHR service system supporting fuzzy keyword search and fine-grained access control, Soft Comput. 18 (9) (2014) 1795–1802.
[20] Z. Liu, J. Weng, J. Li, J. Yang, C. Fu, C. Jia, Cloud-based electronic health record system supporting fuzzy keyword search, Soft Comput. 20 (8) (2016) 3243–3255.
[21] C. Guo, R. Zhuang, Y. Jie, Y. Ren, T. Wu, K.K.R. Choo, Fine-grained database field search using attribute-based encryption for e-healthcare clouds, J. Med. Syst. 40 (11) (2016) 1–8.
[22] L. Yang, Q. Zheng, X. Fan, RSPP: a reliable, searchable and privacy-preserving e-Healthcare system for cloud-assisted body area networks, in: Proceedings of IEEE INFOCOM, 2017, pp. 1–9.
[23] H. Li, F. Zhang, J. He, H. Tian, A searchable symmetric encryption scheme using blockchain, 2017, arXiv preprint arXiv:1711.01030.
[24] N. Szabo, Smart contracts: building blocks for digital markets, extropy, 1996 [online]. Available: http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/.
[25] G. Asharov, Towards characterizing complete fairness in secure twoparty computation, in: Proceedings of TCC, 2014, pp. 291–316.
[26] A. Kosba, A. Miller, E. Shi, Z. Wen, C. Papamanthou, Hawk: The blockchain model of cryptography and privacy-preserving smart contracts, in: Proceedings of IEEE S & P, 2016, pp. 839–858.

[27] The IPFS Project. URL: https://ipfs.io/September, 2015.
[28] D. Cash, J. Jaeger, S. Jarecki, C.S. Jutla, H. Krawczyk, M.-C. Rosu, M. Steiner, Dynamic searchable encryption in very-large databases: Data structures and implementation, in: Proceedings of NDSS, 2014, pp. 23–26.
[29] Z. Liu, T. Li, P. Li, C. Jia, J. Li, Verifiable searchable encryption with aggregate keys for data sharing system, Future Gener. Comput. Syst. 78 (Part 2) (2018) 778–788.
[30] R. Curtmola, J. Garay, S. Kamara, R. Ostrovsky, Searchable symmetric encryption: improved definitions and efficient constructions, in: Proceedings of ACM CCS, 2006, pp. 79–88.
[31] University of California, Irvine, Nursery data set, 1997, http://archive.ics.uci.edu/ml/datasets/Nursery.

Lanxiang Chen received her M.S. and Ph.D. Degrees in Computer Architecture from Huazhong University of Science and Technology in China. She is currently an Associate Professor in Fujian Normal University. She is a member of the Computer Society of China. Her research interests include big data security, cloud computing, and cloud storage security.

Wai-Kong Lee received the B.Eng. in Electronics and M.Sc. degree from Multimedia University in 2006 and 2009 respectively. He had obtained the Ph.D. degree in Engineering from University Tunku Abdul Rahman, Malaysia in 2018. He was a visiting scholar to Carleton University, Canada (2017), Feng Chia University, Taiwan (2016, 2018) and OTH Regensburg, Germany (2015). He had served as reviewer for several international journals, such as IEEE Transactions on Dependable and Secure Computing (2016 and 2017) and Computer and Electrical Engineering (2017). His research interests are in the areas of cryptography, numerical algorithms, GPU computing, Internet of Things (IoT) and energy harvesting.

Chin-Chen Chang received his Ph.D. degree in computer engineering from National Chiao Tung University. His first degree is Bachelor of Science in Applied Mathematics and master degree is Master of Science in computer and decision sciences. Both were awarded in National Tsing Hua University. Dr. Chang served in National Chung Cheng University from 1989 to 2005. His current title is Chair Professor in Department of Information Engineering and Computer Science, Feng Chia University, from Feb. 2005. Prior to joining Feng Chia University, Professor Chang was an associate professor in Chiao Tung University, professor in National Chung Hsing University, chair professor in National Chung Cheng University. He had also been Visiting Researcher and Visiting Scientist to Tokyo University and Kyoto University, Japan. During his service in Chung Cheng, Professor Chang served as Chairman of the Institute of Computer Science and Information Engineering, Dean of College of Engineering, Provost and then Acting President of Chung Cheng University and Director of Advisory Office in Ministry of Education, Taiwan. Professor Chang has won many research awards and honorary positions by and in prestigious organizations both nationally and internationally. He is currently a Fellow of IEEE and a Fellow of IEE, UK. And since his early years of career development, he consecutively won Outstanding Talent in Information Sciences of the R. O. C., AceR Dragon Award of the Ten Most Outstanding Talents, Outstanding Scholar Award of the R. O. C., Outstanding Engineering Professor Award of the R. O. C., Distinguished Research Awards of National Science Council of the R. O. C., Top Fifteen Scholars in Systems and Software Engineering of the Journal of Systems and Software, and so on. On numerous occasions, he was invited to serve as Visiting Professor, Chair Professor, Honorary Professor, Honorary Director, Honorary Chairman, Distinguished Alumnus, Distinguished Researcher, Research Fellow by universities and research institutes. His current research interests include database design, computer cryptography, image compression and data structures.

Kim-Kwang Raymond Choo received the Ph.D. in Information Security in 2006 from Queensland University of Technology, Australia. He currently holds the Cloud Technology Endowed Professorship at The University of Texas at San Antonio (UTSA). In 2016, he was named the Cybersecurity Educator of the Year — APAC (Cybersecurity Excellence Awards are produced in cooperation with the Information Security Community on LinkedIn), and in 2015 he and his team won the Digital Forensics Research Challenge organized by Germany's University of Erlangen–Nuremberg. He is the recipient of the 2018 UTSA College of Business Col. Jean Piccione and Lt. Col. Philip Piccione Endowed Research Award for Tenured Faculty, IEEE TrustCom 2018 Best Paper Award, ESORICS 2015 Best Research Paper Award, 2014 Highly Commended Award by the Australia New Zealand Policing Advisory Agency, Fulbright Scholarship in 2009, 2008 Australia Day Achievement Medallion, and British Computer Society's Wilkes Award in 2008. He is also a Fellow of the Australian Computer Society, an IEEE Senior Member, and the Co-Chair of IEEE Multimedia Communications Technical Committee (MMTC)'s Digital Rights Management for Multimedia Interest Group.

Nan Zhang is M.S. candidate at Fujian Normal University. His research interests include cloud storage and information security.