Received July 11, 2019, accepted July 21, 2019, date of publication July 29, 2019, date of current version August 12, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2931531

Blockchain-Based Personal Health Records Sharing Scheme With Data Integrity Verifiable

SHANGPING WANG[1], DAN ZHANG[1], AND YALING ZHANG[2]
[1]School of Science, Xi'an University of Technology, Xi'an 710048, China
[2]School of Computer Science and Engineering, Xi'an University of Technology, Xi'an 710048, China

Corresponding author: Dan Zhang (1361716068@qq.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61572019, and in part by the Key Project of Natural Science Foundation of Shaanxi Province of China under Grant 2016JZ001.

ABSTRACT The sharing of personal health records can help to improve the accuracy of the doctor's diagnosis and to promote the progress of medical research. Currently, to reduce the maintenance cost of data, personal health records are usually outsourced to a third party such as the cloud service provider. In this case, patients may lose direct control over their personal health records and the semi-trusted cloud service provider may tamper with or reveal personal health records. Therefore, ensuring the privacy and integrity of personal health records and realizing the fine-grained access control are crucial issues when personal health records are shared. As a distributed architecture with decentralized and tamper-proof features, blockchain provides a new way to protect the personal health records sharing system. In this paper, we propose a new personal health records sharing scheme with data integrity verifiable based on blockchain. Aiming at the problems of privacy disclosure, limited keyword search ability and loss of control rights in the process of personal health record sharing, the new scheme uses searchable symmetric encryption and attribute-based encryption techniques to achieve privacy protection, keyword search, and fine-grained access control. Compared with the existing similar schemes, the new scheme allows patients to distribute attribute private key for users, avoiding many security problems caused by the existing of attribute authority in the scheme. Furthermore, the new scheme uses blockchain to manage keys in the scheme, avoiding the single point failure problem of centralized key management. In particular, the new scheme stores the hash values of encrypted personal health records in blockchain, and the related index set is stored in smart contract, which can further improve the efficiency of data integrity verification. Finally, performance evaluation and security analysis indicate that our scheme is secure and feasible for practical use.

INDEX TERMS Personal health records, blockchain, smart contract, searchable symmetric encryption, attribute-based encryption, data integrity verification.

I. INTRODUCTION

In recent years, the rapid development of network information technology and cloud technology has brought a huge impact on people's lifestyle. The emergence of personal health records sharing system based on electronic information and cloud technology enables patients to store, manage and share their health information conveniently, efficiently and accurately. As a kind of healthcare information recorded and managed by the patient, personal health records provide the patient with a complete and accurate personal medical history that can be obtained online. These personal health records are valuable resources, and reasonable sharing of personal health records can benefit patients, research institutions, pharmaceutical companies, and the whole healthcare system.

In the personal health records sharing system, patient's personal health records are often outsourced to the third party such as the cloud service provider in order to achieve resource sharing and reduce the maintenance costs of data center. Under the circumstances, one of the most controversial issues is how to ensure the security, privacy and searchability of personal health records while achieving fine-grained access control. An effective solution is to combine cloud storage, searchable symmetric encryption, and attribute-based encryption together. But this approach also presents a new

The associate editor coordinating the review of this manuscript and approving it for publication was Martin Reisslein.

set of challenges. First, when various encryption mechanisms are used to protect personal health records outsourced to the cloud server, centralized key management will lead to a single point of failure. Secondly, almost all attribute-based encryption schemes require a trusted authority to set up the system and distribute the private keys for system participants. However, it is very difficult to find a completely credible authority in reality. In addition, the system participants usually escrow their attribute private keys to the trusted authority in the attribute-based encryption scheme. This so-called key escrow problem can compromise the confidentiality of personal health records outsourced by patients to the cloud server, especially when the authorization center is threatened. Finally, the cloud platform may not be credible due to issues such as employee corruption. During the sharing of the personal health records, the cloud server may return tampered or partially qualified encrypted personal health records to users for its benefit. These erroneous or incomplete personal health records can mislead users (such as doctors, research institutions or other patients) into making false judgments that endanger the lives of patients or others.

Fortunately, the emergence of blockchain technology provides a new way to solve the above problems. The use of blockchain for key management and distribution makes key management and distribution easier and more secure. In addition, the blockchain has the characteristics of unforgeable and tamper-proof. Every event or transaction on the blockchain is timestamped and cannot be tampered with once it is recorded on the blockchain. Therefore, storing the hash values of encrypted personal health records onto the blockchain not only can effectively avoid the bad consequences caused by incorrect or partially satisfied encrypted personal health records returned by the malicious cloud server. Meanwhile, the cloud server can also be urged to honestly perform the operation according to the requirements of users.

The main contributions of this paper are summarized as follows:

(1) We propose a new personal health records sharing scheme, in which the cloud storage, searchable symmetric encryption, attribute-based encryption, blockchain technology and smart contract are combined together to achieve privacy protection, keyword search, fine-grained access control and data integrity verification. Compared with existing similar solutions, the new scheme allows patients to distribute attribute private keys to users, enabling fine-grained access control to be implemented without relying on any third party.

(2) In the new scheme, the single point failure problem that a single entity manages private keys in a centralized way is solved by managing and distributing private keys through blockchain. Furthermore, blockchain and smart contract work together to enable users to quickly and effectively verify the integrity and correctness of encrypted personal health records received from the cloud server without interacting with the cloud server.

(3) The data integrity verification contract is implemented in the new scheme and the contract is deployed to the official Ethereum test network Rinkeby. And we evaluate the performance of the new scheme in terms of smart contract cost and file encryption efficiency. Finally, the security analysis of the new scheme is given.

The rest of the paper is organized as follows. The related work is discussed in Section II. Preliminaries needed for the new scheme are reviewed in Section III. The model of our scheme is given in Section IV. The specific design of our scheme is described in Section V. Data integrity verification contract and performance evaluation and security analysis are given in Sections VI and VII, respectively. Finally, the conclusion is made in SectionVIII.

II. RELATED WORK

In this part, we briefly summarize the relevant researches on encryption technology and blockchain technology related to the scheme proposed in this paper.

A. SEARCHABLE SYMMETRIC ENCRYPTION

Searchable symmetric encryption mechanism is an important technology to realize the secure storage of data. It allows users to store encrypted document set to the cloud server and maintain the ability to perform a keyword search on the document without revealing document content and search information. In 2000, Song *et al.* [1] proposed the first searchable symmetric encryption scheme, which ensures that the cloud server could not learn any information about the plaintext of keyword and the plaintext of search results when searching with the ciphertext of keyword. However, the security of the scheme is relatively low, and it is secure under choose plaintext attack (CPA). In 2006, Curtmola *et al.* [2] proposed a new security definition for searchable symmetric encryption, and constructed two new searchable symmetric encryption schemes (SSE-1 and SSE-2) under the new security definition. In 2010, Wang *et al.* [3] proposed the definition of sortable searchable symmetric encryption and gave an effective design of symmetric key order-preserving encryption technique based on existing cryptographic primitives. In 2012, Premasathian and Choto [4] proposed a multi-keyword retrieval scheme for the first time. The scheme uses AND gate to connect different keywords, and hides user data and search token to ensure security. In 2017, Li *et al.* [5] proposed a searchable symmetric encryption scheme using blockchain technology, in which user data is split and stored on the blockchain. However, simulation experiments on the bitcoin blockchain show that it is not practical to store a large amount of data on the blockchain. In 2018, Zhang *et al.* [6] proposed a two-side verifiable trusted keyword search scheme based on blockchain. The scheme ensures that the encrypted data can be searched and the search results can be verified. At the same time, fair payment is achieved between the cloud server and the user.

B. ATTRIBUTE-BASED ENCRYPTION

As an extension of public key cryptography and identity-based cryptography, attribute-based encryption can realize fine-grained access control to data. In 2005, Sahai and Waters [7] first proposed the concept of attribute-based encryption. In 2006, Goyal *et al.* [8] divided the attribute-based encryption into key policy attribute-based encryption and ciphertext policy attribute-based encryption according to the differences of ciphertext and key expression forms and application scenarios, and constructed the first key policy attribute-based encryption scheme. In 2007, Bethencourt *et al*. [9] constructed the first ciphertext policy attribute-based scheme. In recent years, with the proposal of some basic schemes [10]–[14] and the continuous deepening of the research on attribute cryptography, researchers have made many achievements in attribute cryptography. For example, to improve communication efficiency and reduce the computing cost of users who want to access encrypted data stored in the cloud, Li *et al*. [15] proposed KSF-OABE: outsourced attribute-based encryption with keyword search function for cloud storage. Besides, Li *et al* have done a lot of work to solve the problems of user revocation, attribute revocation and side channel attack in attribute-based encryption schemes, and achieved some important results [16]–[19]. Especially, considering that too long ciphertext will cause high communication cost in practical application, many researchers begin to study attribute encryption scheme with fixed ciphertext length. Emura *et al.* [20] considered the attribute-based encryption scheme with constant ciphertext length, but the encryption policy in this scheme can only support the AND gate access structure. Attrapadung *et al.* [21] proposed a ciphertext policy attribute-based encryption scheme. The scheme supports threshold access structures and the length of ciphertext is constant.

C. BLOCKCHAIN TECHNOLOGY AND ITS APPLICATION

As the underlying technology of Bitcoin, the blockchain first appeared in Bitcoin: a peer-to-peer cash system [22] published by Satoshi Nakamoto in 2008, which describes in detail how to establish a new set of decentralized point-to-point trading system without trust foundation, and its realizability has been proved by the stable operation of bitcoin since 2009. As a decentralized ledger, blockchain records the experience of a subject from the original state to the current state in the form of an immutable log [23]–[25]. In recent years, inspired by the great success of blockchain in the financial field, many researchers have begun to actively explore the application of blockchain technology in other fields. For example, decentralized internet of things [26]–[29], decentralized data sharing [30]–[34], etc.

In particular, great achievements have been made in the exploration of the application of blockchain in the medical field. Yue *et al.* [35] proposed a blockchain-based medical data gateway that allows patients to easily and securely access their data without disclosing privacy. Azaria *et al.* [36] designed a new medical data management system based on blockchain. The system provides convenience for patients access their medical information across providers and medical sites. In recent years, many blockchain-based methods and systems [37]–[41] have been proposed for the acquisition, storage and management of patient medical data. Besides, many blockchain-based medical data sharing methods have been proposed. Xia *et al.* [42] proposed a blockchain-based electronic medical record sharing scheme. The solution addresses the access control issues associated with sensitive data stored in the cloud, utilizing tamper-proof and built-in autonomy attributes of the permissioned blockchain. However, the existence of some third parties such as key publishers and authenticators, makes the scheme have many security risks. Zheng *et al.* [43] proposed a personal health records sharing scheme based on blockchain, cloud storage and machine learning. The scheme ensures that patients can easily and safely share their personal health records and benefit from the process. But the scheme does not consider how to verify the correctness and integrity of the personal health records returned by cloud servers.

By analyzing the existing schemes, it can be seen that the existing personal health records sharing schemes based on cloud storage and blockchain can't achieve fine-grained access control of data well. Besides, there is the risk of privacy leakage in these centralized management systems. In particular, when a user wants to verify the correctness and the integrity of encrypted personal health records returned by the cloud server, it is necessary to frequently interact with the cloud server, which makes the scheme inefficient in practice. Aiming at these problems in the existing schemes, in this paper, we propose a new personal health records sharing scheme with data integrity verifiable based on blockchain.

III. PRELIMINARIES

In this section, we review some basic techniques used in our scheme, and the important symbols used in our scheme are summarized in Table 1.

A. ACCESS STRUCTURE IN THE ATTRIBUTE-BASED ENCRYPTION SCHEME

Consider a set of parties $P = \{P_1, \cdots, P_n\}$. A collection $\mathbb{A} \subset 2^P$ is said to be monotone if for $\forall B, B', B \in \mathbb{A}$, and $B' \subseteq B$, then $B' \in \mathbb{A}$. An access structure $\mathbb{A}$ [21] is the nonempty subset of P, i.e. $\mathbb{A} \subseteq 2^P \setminus \{\phi\}$. The sets in $\mathbb{A}$ are called authorized sets, and the sets that are not in $\mathbb{A}$ are called unauthorized sets. Particularly, an access structure $\mathbb{A}$ is called a threshold access structure if there exists a positive integer $t_{\mathbb{A}}$, such that $B \in \mathbb{A}$ if and only of $|B| \geq t_{\mathbb{A}}$, where $|B|$ denotes the number of elements in set B.

B. BLOCKCHAIN TECHNOLOGY

The blockchain [22] is essentially a distributed ledger with the characteristics of decentralization, trustless, and tamper-proof, etc. The blockchain is a chain of data blocks linked together, and the link pointer is the hash value generated

TABLE 1. Notations table.

Notation	Description
msk	The system master key in attribute-based encryption scheme
mpk	The system public parameter in attribute-based encryption scheme
(S,t)	The threshold access policy in attribute-based encryption scheme
sk_ω	The attribute private key of the user with attribute set ω
C_{sk_ω}	The ciphertext of sk_ω
K	The AES key that is used to encrypt personal health records
C_K	The ciphertext of K
F	The personal health record set to be outsourced to the cloud server
W	The keyword set extracted from F
C	The ciphertext set of F
TX	The id set of transactions that contain the hash values for files in C
F_w	The personal health record set containing the keyword w
C_w	The ciphertext set of F_w
TX_w	The id set of transactions that contain hash values of files in C_w
I_1	The keyword index set of encrypted personal health records
I_2	An index set of the hash values of encrypted personal health records
T_w	The token generated by the keyword w

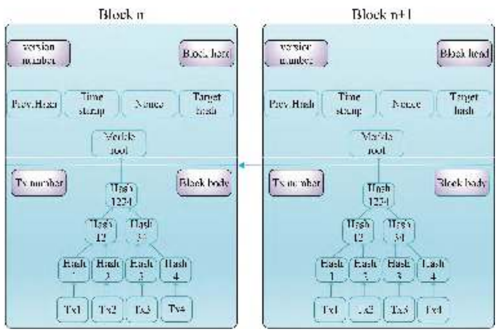


FIGURE 1. The data structure of blockchain.

by processing the block header with a cryptographic hash algorithm. Each data block in the blockchain records a set of tree-like transaction status information composed of hash values to ensure that the transaction data in each block cannot be tampered with, and the block linked in the blockchain cannot be tampered with.

The data structure of blockchain is shown in Figure 1, each data block in the blockchain contains two parts: block header and block body. The block header encapsulates the current version number, the hash value of the previous block, the timestamp, the random number, the target hash value of the current block, and the root value of the Merkle tree. The block body contains the Merkle tree of all transactions in the current block and the count of all transactions in the block.

C. ETHEREUM TRANSACTION

Ethereum transactions [30] are signed packets that store messages sent from external owned accounts (EOA). The message data is Ether or contract execution parameter.

The data structure of the Ethereum transaction is shown in Figure 2, the Nonce field indicates the number of transactions sent by this transaction sender. The Gas Limit field indicates the maximum gas allowed for this transaction. The Gas Price field indicates the price of the gas fee that the sender of the transaction is willing to pay. The To field

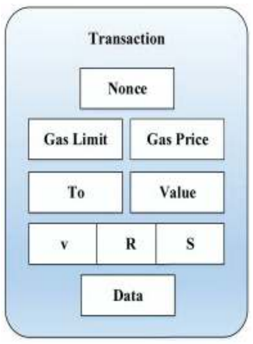


FIGURE 2. Ethereum transaction.

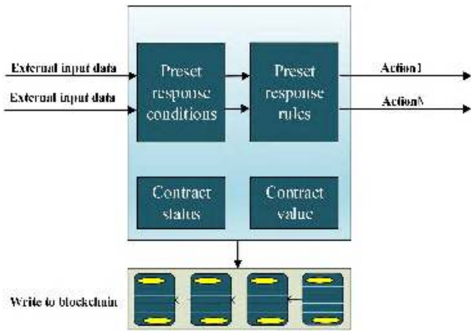


FIGURE 3. The model of smart contract.

represents the contract address when the contract is called and the destination user's address when the transfer is made. The Value field represents the number of Ether transferred from the sender to the receiver. The V R S field is the sender's signature information. The Data field is the optional data field where the transaction sender can store any data that he/she wants to store.

D. SMART CONTRACT

Smart contracts are modular, reusable, and automatically executed scripts that run on the blockchain. Smart contract itself is a participant in the blockchain network, which responds to the received information, receives and stores value, and sends out information and value. When the smart contract is deployed, the bytecode compiled by the compiler is stored in the blockchain, and there is a storage address corresponding to it. The model for the smart contract is shown in Figure 3, when a predefined condition in the contract occurs, a transaction is sent to the contract address, and all network nodes execute the opcodes generated by compiling the contract scripts, and finally write the execution results to the blockchain.

IV. THE MODEL OF OUR SCHEME

In this part, the model of our scheme is given, and the function of each entity in the scheme and the workflow of the scheme are briefly introduced.

A. THE SCHEME ENTITIES

As shown in Figure 4, our scheme mainly includes three entities: patient, user and cloud server.

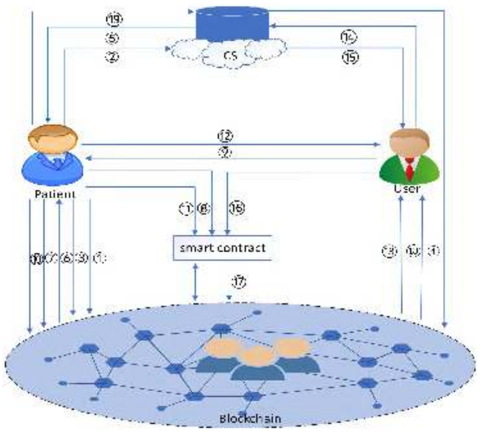


FIGURE 4. The model of the scheme.

Patient: The owner of personal health records. In order to achieve resource sharing and reduce the cost of data maintenance, the patient usually encrypts personal health records and uploads them to the cloud server. In our scheme, the patient is mainly responsible for deploying the smart contract, generating and distributing attribute private key for the user.

User: Individuals or organizations that access patient's personal health records for research or other useful purposes.

Cloud server: It stores encrypted personal health records and keyword indexes of encrypted personal health records uploaded by the patient, and provides the search service for users with access rights in the personal health records sharing stage.

B. THE WORKFLOW OF THE SCHEME

Our scheme includes four stages: initialization, personal health records storage, personal health records sharing and personal health records management.

1) INITIALIZATION STAGE

In this stage, the patient initializes some parameters for later use, and deploys the compiled data integrity verification contract to blockchain and records the contract address and ABI. As shown in step ① of Figure 4.

2) PERSONAL HEALTH RECORDS STORAGE STAGE

This stage includes three procedures: outsourcing preparation, storage enforcement and storage confirmation. Among them, outsourcing preparation corresponds to the steps ②③ in Figure 4, storage enforcement corresponds to the steps ④⑤ in Figure 4, and storage confirmation corresponds to the steps ⑥⑦⑧ in Figure 4. Each step is described as follows:

② The patient encrypts personal health record set to be outsourced to the cloud server with a searchable symmetric encryption scheme and sends the generated ciphertext set and the keyword index set to the cloud server.

③ The patient encrypts the key of searchable symmetric encryption scheme with an attribute-based encryption scheme, and saves the ciphertext of the key to the blockchain through a transaction and records the transaction id.

④ The cloud server stores personal health record ciphertext set and the keyword index set sent by the patient and builds a signature based on the stored personal health record ciphertext set. Then the signature is saved to the blockchain through a transaction, and the transaction id is recorded.

⑤ The cloud server sends the transaction id recorded in the step ④ to the patient.

⑥ The patient reads the signature of the cloud server from blockchain according to the transaction id sent by the cloud server. Once the signature is validated, the patient believes that the personal health record ciphertext set stored by the cloud server is correct.

⑦ The patient saves the hash values of all encrypted personal health records to the blockchain in the form of transactions and records all transaction ids.

⑧ The patient generates keyword indexes for the hash values of all encrypted personal health records and stores these indexes into the data integrity verification contract. After that, the patient deletes the local personal health record set and the corresponding ciphertext set.

3) PERSONAL HEALTH RECORDS SHARING STAGE

This stage includes four procedures: request access, access authorization, personal health records retrieval, and personal health records verification. Request access corresponds to steps ⑨⑩⑪⑫ of Figure 4. Access authorization corresponds to step ⑬ of Figure 4. Personal health records retrieval corresponding to steps ⑭⑮ of Figure 4. Personal health records verification corresponding to steps ⑯⑰⑱ of Figure 4. Each step is shown below:

⑨ The user sends an access request containing his/her identity information and Ethereum account address to the patient.

⑩ The patient selects an appropriate attribute set for the user and generates the corresponding attribute private key, and the attribute private key is encrypted with a symmetric key generated through the Diffie-Hellman key exchange protocol and stored to the blockchain through a transaction, and the transaction id is recorded.

⑪ The patient adds the user's Ethereum account address to the data integrity verification contract.

⑫ The patient sends the contract address and ABI recorded in step ①, the transaction ids recorded in steps ③⑩, and the attribute set selected in step ⑩ to the user.

⑬ The user first reads the ciphertext of the attribute private key and the ciphertext of the searchable symmetric encryption scheme key from blockchain according to the transaction ids in the message sent by the patient. Then, the user sequentially executes the decryption algorithm of the symmetric encryption scheme and the decryption algorithm of the attribute-based encryption scheme to restore the attribute private key and the searchable symmetric encryption scheme key.

⑭ The user generates a token using the key of the searchable symmetric encryption scheme obtained in step ⑬ and the keyword he/she is interested in, and then sends the token to the cloud server.

⑮ The cloud server performs a search according to the token received from the user and returns search results to the user.

⑯ The user calls the data integrity verification contract with the token that is generated in step ⑭.

⑰ The data integrity verification contract responds to the user's request, performs the search and saves the searched transaction id set to the blockchain in the form of events.

⑱ The user obtains the transaction id set returned by the data integrity verification contract through monitoring the blockchain, and reads the hash values of the target files from the blockchain according to the transaction id set. Later, the user verifies the integrity and correctness of the search results returned by the cloud server based on the hash values read from the blockchain.

4) PERSONAL HEALTH RECORDS MANAGEMENT STAGE

In the stage, the patient uploads the newly generated personal health records to the cloud server or deletes some encrypted personal health records stored in the cloud server, which corresponds to step ⑲ of Figure 4.

V. THE SPECIFIC DESIGN OF OUR SCHEME

A. DESIGN OBJECTIVES

Searchable encryption mechanisms and attribute-based encryption mechanisms can effectively address the problems of privacy leak, limited keyword search ability, and access control when sharing personal health records in the cloud storage. But it will also introduce a new set of challenges. Firstly, there are a large number of personal health records, although privacy protection and fine-grained access control can be achieved by directly encrypting them with an attribute-based encryption scheme, the scheme is inefficient. The combination of symmetric encryption and attribute-based encryption is an effective way to solve this problem. But the problem of the key security in the encryption scheme needs to be solved urgently. To solve this problem, our scheme uses blockchain to realize key management and distribution. In particular, considering the cost of storing data on the blockchain, our scheme adopts the attribute-based encryption scheme with constant ciphertext length, and security proof of the scheme is detailed in literature [21].

Secondly, in the attribute-based scheme, the attribute authority generates and manages the attribute private keys for the system participants may lead to problems such as key abuse and privacy leakage. To solve this problem, our scheme makes the patient act as the attribute authority to generate and distribute the attribute private key for the user.

In addition, providing authorized users with the ability to quickly retrieve data is something to consider when sharing personal health records. To achieve this goal, we construct an index building algorithm based on literature [2], and store the keyword index set of encrypted personal health records generated by the algorithm to the cloud server.

Finally, achieving efficient data integrity verification is also an urgent problem to be solved when sharing personal health records. In our scheme, the hash values of encrypted personal health records are stored in the blockchain, and the relevant index set is stored in the smart contract, which provides a new idea to solve this problem. Besides, in order to help patients manage their personal health records outsourced to the cloud server, index update and index deletion algorithms are constructed in our scheme.

B. THE SPECIFIC SCHEME

Our scheme is divided into four stages: initialization, personal health records storage, personal health records sharing and personal health records management. The specific design of each stage is as follows.

1) INITIALIZATION STAGE

In this stage, the patient first performs the following *Setup* algorithm to initialize some parameters for use in subsequent stages.

- **Setup**$(1^\lambda) \rightarrow (mpk, msk)$: The algorithm selects an appropriate encoding method τ, for each attribute $at \in U$, where U is the attribute universal set and $|U| = n$, such that $\tau(at) = x \in Z_p^*$, and the encoding is different from each other. It also chooses a bilinear group tuple (G, G_1) of prime order p, let $g, h \in_R G$ and $e : G \times G \rightarrow G_1$ is a bilinear map. Besides, it chooses a set $D = \{d_1, \cdots, d_{n-1}\}$, where $\{d_i\}_{i\in n-1} \in Z_p^*$ is different from each other, and also different to $\tau(at) = x$, for all $at \in U$. And for each $i \leq n-1$, let $D_i = \{d_1, \cdots, d_i\}$. Furthermore, it selects two hash functions $H : \{0,1\}^* \rightarrow \{0,1\}^l$ and $SHA-256$. Finally, the algorithm randomly chooses $\alpha, \gamma \in Z_p^*$, computes $u = g^{\alpha\gamma}$, $v = e(g^\alpha, h)$, and sets the system master key $msk = (g, \alpha, \gamma)$, the system public parameter $mpk = (U, n, u, v, \{h^{\alpha\gamma^i}\}_{i=0,\cdots,2n-1}, D, \tau, H, SHA-256)$.

After that, The patient compiles the created data integrity verification contract (given in section VI of this paper) into EVM bytecode and deploys it to the blockchain, recording the contract address and ABI locally.

2) PERSONAL HEALTH RECORDS STORAGE STAGE

In this stage, the patient realizes the outsourcing storage of personal health records through the three procedures of outsourcing preparation, storage enforcement and storage confirmation. The specific description of each process is as follows.

► Process1: outsourcing preparation

To achieve privacy protection, before outsourcing a personal health record set $F = \{F_q\}_{q\in[N]}$ to the cloud server, the patient encrypts the set F with the AES key K selected from the key space. That is, for each $q \in [N]$, the patient computes $C_q = AES.Encrypt(K, F_q)$, and gets the ciphertext set $C = \{C_q\}_{q\in[N]}$ of F.

Then, to ensure that encrypted personal health records can be searched by keywords, the patient builds the keyword index set I_1 for encrypted personal health records based on Algorithm 1. In Algorithm 1, $F_w = \{F_{s_j}\}_{j\in[m]}$ is the set of personal health records containing the keyword w in the set F, $C_w = \{C_{s_j}\}_{j\in[m]}$ is the ciphertext set of F_w, and m is the number of personal health records containing the keyword w in the set F. For each $C_{s_j} \in C_w, j \in [m]$, the keyword index set I_1 maps the ciphertext $T_w = SHA-256(K\|w)$ of the keyword w to the set $(ID_{C_{s_j}}, H(C_{s_j}))$, where $ID_{C_{s_j}}$ is the identifier of C_{s_j}, and $H(C_{s_j})$ is the hash value of C_{s_j}.

Algorithm 1 BuildIndex1

```
Input: AES key $K$, personal health record set $F$,
       ciphertext set $C$ of $F$, keyword set $W$ extracted
       from $F$
Output: keyword index set $I_1$
1 for each keyword $w \in W$ do
2     $T_w = SHA-256(K\|w)$
3     Suppose $F_w = \{F_{s_j}\}_{j\in[m]}$ and $C_w = \{C_{s_j}\}_{j\in[m]}$
4     for $j \in [m]$ do
5         $I_1(T_w) = (ID_{C_{s_j}}, H(C_{s_j}))$
6     end
7 end
8 return $I_1$
```

Finally, the patient sends ciphertext set C of F and keyword index set I_1 generated by the above operations to the cloud server. Meanwhile, to achieve fine-grained access control and solve the single point failure problem caused by a single entity centralized management key, the patient encrypts the AES key K with the following $CP-ABE.Encrypt$ algorithm, and saves its ciphertext C_K to blockchain through a transaction and records the transaction id.

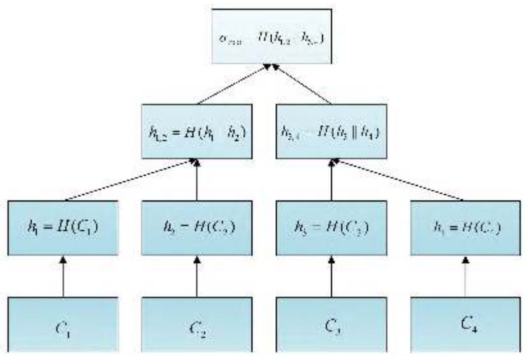
- **CP − ABE.Encrypt**$(msk, (S, t), K) \rightarrow C_K$: The algorithm inputs the system master key msk, threshold access policy (S, t), where the attribute set $S \subset U$ and $|S| = s \leqslant n$, threshold value t satisfies $1 \leqslant t \leqslant s$, and AES key K, it outputs the ciphertext C_K of K. Specifically, it randomly selects $k \in Z_p^*$ and computes

$$\begin{cases} C_{K_1} = u^{-k} \\ C_{K_2} = h^{k\cdot\alpha\cdot\prod_{at\in S}(\gamma+\tau(at))\prod_{d\in D_{n+t-1-s}}(\gamma+d)} \\ T = v^k = e(g^\alpha, h)^k \\ C_{K_3} = T \cdot K \end{cases}$$

Then, the ciphertext $C_K = (C_{K_1}, C_{K_2}, C_{K_3})$ of K.

► Process2: storage enforcement
The cloud server first stores the ciphertext set C of F and index set I_1 sent by the patient. Then, to facilitate patient verification of storage execution, the cloud server constructs a Merkle tree based on the stored ciphertext set C of F.

Here, we take the number $N = 4$ of personal health records and the ciphertext set $C = \{C_q\}_{q\in[4]}$ of F as an example to illustrate the construction process of the Merkle tree. The Merkle tree is built from the bottom to up. As shown in Figure 5, the hash value $H(C_q)$ of each encrypted personal health record in set C is stored to the corresponding leaf node $h_q = H(C_q)$, $q \in [4]$. Two adjacent leaf nodes are concatenated and hashed to generate their parent node, that is, $h_{1,2} = H(h_1\|h_2)$ and $h_{3,4} = H(h_3\|h_4)$. Continue the similar operation until the root $\sigma_{root} = H(h_{1,2}\|h_{3,4})$ of the Merkle tree is generated. It should be noted that constructing a Merkle tree requires an even number of leaf nodes. If the number N of personal health records is odd, then the ciphertext of the Nth personal health record will be copied to form an even number of leaf nodes, and the Merkle tree can be constructed in the way that is similar to $N = 4$.



$\sigma_{root} = H(h_{1,2} \| h_{3,4})$
$h_{1,2} = H(h_1 \| h_2)$
$h_{3,4} = H(h_3 \| h_4)$
$h_1 = H(C_1)$
$h_2 = H(C_2)$
$h_3 = H(C_3)$
$h_4 = H(C_4)$
C_1
C_2
C_3
C_4

FIGURE 5. The Merkle tree of personal health records ciphertext

Finally, the cloud server signs the root node σ_{root} of the Merkle tree with its Ethereum account private key, and saves the generated signature $I_{root} = sig_{CSP}(\sigma_{root})$ to the blockchain through a transaction and sends the transaction id to the patient.

► Process3: storage confirmation
According to the transaction id sent by the cloud server, the patient reads the signature $I_{root} = sig_{CSP}(\sigma_{root})$ of the cloud server from blockchain, and uses the Ethereum account public key of the cloud server to decrypt it and obtain σ_{root}. After that, the patient calculates σ'_{root} using the ciphertext set C of F stored locally. If $\sigma_{root} = \sigma'_{root}$, the patient can determine that the cloud server has performed the storage correctly.

After making sure that the cloud server performs the storage correctly, in order to enable the user to quickly verify the integrity and correctness of encrypted personal health records returned by the cloud server without interacting with the cloud server in the personal health records sharing stage, the patient first saves the hash values of the ciphertext of all personal health records in the set F into blockchain in the form of transactions, and records the transaction id set TX. The patient then generates the keyword index set I_2 for these hash values based on Algorithm 2. Similar to Algorithm 1, in Algorithm 2, the keyword index set I_2 maps the ciphertext $T_w = SHA-256(K\|w)$ of the keyword $w \in W$ to the set $TX_w = \{txid_{s_j}\}_{j\in[m]}$, where $TX_w = \{txid_{s_j}\}_{j\in[m]}$ is id set of transactions that save hash values of the ciphertext of the personal health records in the set $F_w = \{F_{s_j}\}_{j\in[m]}$. Finally, the patient calls the function addIndex() in the data integrity verification contract to store the keyword index set I_2 into the contract, and then deletes the local personal health record set F and ciphertext set C of F.

Algorithm 2 BuildIndex2

```
Input: AES key $K$, personal health record set $F$,
       transaction id set $TX$, keyword set $W$ extracted
       from $F$
Output: keyword index set $I_2$
1 for each keyword $w \in W$ do
2     $T_w = SHA-256(K\|w)$
3     Suppose $F_w = \{F_{s_j}\}_{j\in[m]}$, $TX_w = \{txid_{s_j}\}_{j\in[m]}$
4     for $j \in [m]$ do
5         $I_2(T_w) = txid_{s_j}$
6     end
7 end
8 return $I_2$
```

3) PERSONAL HEALTH RECORDS SHARING STAGE

This stage realizes the sharing of personal health records, and includes four processes: request access, access authorization, personal health records retrieval and personal health records verification.

► Process1: request access

To access the patient's personal health records, the user sends the access request containing his/her identity information and Ethereum account address to the patient. After the patient verifies the user's identity, an appropriate attribute set ω is selected for the user and the corresponding attribute private key sk_ω is generated through the following $CP-ABE.KeyGen$ algorithm.

- **CP−ABE.KeyGen**$(msk, \omega) \rightarrow sk_\omega$: The algorithm inputs the system master key msk, attribute set $\omega \in U$, and outputs attribute private key sk_ω. Concretely, it randomly picks two elements $r, z \in Z_p^*$, and computes the attribute private key

$$sk_\omega = \left(\left\{g^{\frac{r}{\gamma+\tau(at)}}\right\}_{at\in\omega}, \left\{h^{r\gamma^i}\right\}_{i=0,\cdots,n-2}, h^{\frac{r-z}{\gamma}}, z\right)$$

In particular, to avoid many security problems that exist in the centralized management of attribute private keys by a single entity, the patient negotiates a symmetric key K' with the user through the Diffie-Hellman key exchange protocol and uses it as the AES key to encrypt sk_ω, that is, $C_{sk_\omega} = AES.Encrypt(K', sk_\omega)$. After that, the patient saves the ciphertext C_{sk_ω} of sk_ω to blockchain through a transaction and records the transaction id.

After completing the above operations, the patient calls the function addUser() in the data integrity verification contract to add the user's Ethereum account address to the legitimate users set legalUsers[] of the contract, thus giving the user the right to call the data integrity verification contract.

Finally, the patient sends the address and ABI of the data integrity verification contract, the id of the transaction that saves the ciphertext C_{sk_ω} of sk_ω, the id of the transaction that saves the ciphertext C_K of the AES key K, and the attribute set ω to the user.

► Process2: access authorization

After receiving the message sent by the patient, the user first reads the ciphertext C_{sk_ω} of the attribute private key sk_ω and the ciphertext C_K of the AES key K from blockchain according to the transaction ids in the message. After that, the user executes the AES decryption algorithm to obtain the attribute private key sk_ω, that is, $sk_\omega = AES.Decrypt(K', C_{sk_\omega})$. Finally, the user determines whether his/her attribute set ω satisfies the ciphertext policy (S, t) of C_K, that is, whether $|\omega \cap S| \geq t$ is true. If so, the user can obtain AES key K by executing the following $CP-ABE.Decrypt$ algorithm.

- **CP−ABE.Decrypt**$(mpk, sk_\omega, \omega, C_K, (S, t)) \rightarrow K$: The algorithm inputs the system public parameter mpk, attribute private key sk_ω, attribute set ω, the ciphertext C_K of AES key K, threshold access policy (S, t), and outputs AES key K. Concretely, suppose $\omega_S = \omega \cap S$ and $|\omega_S| = t$. For $\forall at \in \omega_S$, it computes

$$Aggregate\left(\left\{g^{\frac{r}{\gamma+\tau(at)}}, \tau(at)\right\}_{at\in\omega_S}\right) = g^{\frac{r}{\prod_{at\in\omega_S}(\gamma+\tau(at))}}$$

With the output of the algorithm $Aggregate$, it also computes

$$X = e\left(g^{\frac{r}{\prod_{at\in\omega_S}(\gamma+\tau(at))}}, C_{K_2}\right) = e(g, h)^{k\cdot\alpha\cdot r\cdot\prod_{at\in S\setminus\omega_S}(\gamma+\tau(at))\prod_{d\in D_{n+t-1-s}}(\gamma+d)}$$

For simplicity, for $\forall d \in D$, let $\tau(d) = d$, and defines $P_{(\omega_S,S)}(\gamma)$ as

$$P_{(\omega_S,S)}(\gamma) = \frac{1}{\gamma}\left(\prod_{y\in(S\cup D_{n+t-1-s})\setminus\omega_S}(\gamma+\tau(y)) - \prod_{y\in(S\cup D_{n+t-1-s})\setminus\omega_S}\tau(y)\right)$$

Importantly, since $|\omega_S| \geq t$ and $degree(P_{(\omega_S,S)}(x)) \leq n-2$. Therefore, $h^{rP_{(\omega_S,S)}(\gamma)}$ can be calculated from the values included in sk_ω. After that, it calculates

$$e(C_{K_1}, h^{rP_{(\omega_S,S)}(\gamma)}) \cdot X = e(g, h)^{k\cdot\alpha\cdot r\cdot\prod_{y\in(S\cup D_{n+t-1-s})\setminus\omega_S}\tau(y)} \tag{1}$$

And from equation (1) can obtain

$$e(g,h)^{k\cdot\alpha\cdot r} = \left(e(C_{K_1}, h^{rP_{(\omega_S,S)}(\gamma)}) \cdot X\right)^{\frac{1}{\prod_{y\in(S\cup D_{n+t-1-s})\setminus\omega_S}\tau(y)}} \tag{2}$$

Then, it calculates

$$e(C_{K_1}, h^{\frac{r-z}{\gamma}}) = e(g, h)^{-k\cdot\alpha\cdot r} \cdot e(g, h)^{k\cdot\alpha\cdot z} \tag{3}$$

Finally, it substitutes equation (2) into equation (3) and multiplies it by $e(g, h)^{-z}$ to obtain $T = e(g, h)^{k\cdot\alpha}$, and then the AES key K is recovered by computing $K = C_{K_3}/T$.

► Process3: personal health records retrieval
When a user wants to access the patient's personal health records containing keywords w, the user generates a token T_w based on the following *TokenGen* algorithm and sends it to the cloud server.

- **TokenGen**$(K, w) \rightarrow T_w$: The algorithm inputs AES key K, keyword w, and outputs the token $T_w = SHA - 256(K \| w)$.

After the cloud server receives the token T_w, it performs the search according to Algorithm 3 and returns the search results to the user. More specifically, the cloud server first initializes two empty sets C_w and Tag_w. Next, it uses the token T_w to search the keyword index set I_1 and get the set $\{(ID_{C_{s_j}}, H(C_{s_j}))\}_{j\in[m]}$. Finally, it adds the encrypted personal health record C_{s_j} found by $ID_{C_{s_j}}$ to the set C_w, and the hash value $H(C_{s_j})$ of C_{s_j} is added to the set Tag_w, then returns $Result = C_w \cup Tag_w$ to the user.

Algorithm 3 Search
```
Input: keyword index set $I_1$, ciphertext set $C$ of $F$, token $T_w$
Output: $Result$
1 set $C_w = \phi$, $Tag_w = \phi$
2 phase $I_1(T_w) = \{(ID_{C_{s_j}}, H(C_{s_j}))\}_{j\in[m]}$
3 for $j \in [m]$ do
4     set $C_w = C_w \cup C_{s_j}$
5     set $Tag_w = Tag_w \cup H(C_{s_j})$
6     set $Result = C_w \cup Tag_w$
7 end
8 return $Result$
```

► Process4: personal health records verification
Since a semi-trusted cloud server may return tampered or partially satisfactory search results to the user. Therefore, it is necessary to verify the integrity and accuracy of encrypted personal health records in the set C_w returned by the cloud server before performing the decryption operation. To be specific, after receiving the search results $Result = C_w \cup Tag_w$ from the cloud server, the user calls the function search() in the data integrity verification contract with token T_w. The function search() uses the token T_w to search the keyword index set I_2 stored in the data integrity verification contract and saves the set TX_w of searched transaction id to blockchain in the form of events.

The user monitors the blockchain to obtain TX_w, and reads the hash values of the target files from blockchain according to the transaction ids in the set TX_w. Next, by comparing these hash values read from blockchain with the hash values in the set Tag_w returned by the cloud server, the user can determine which encrypted personal health records in the set C_w returned by the cloud server are complete and valuable for reference, and which have been changed. Finally, the user decrypts the untampered encrypted personal health records in the set C_w with the AES key K to obtain the original personal health records containing the keyword w.

4) PERSONAL HEALTH RECORDS MANAGEMENT STAGE
The stage is reached when the patient wants to upload a newly generated personal health record to the cloud server or delete an encrypted personal health record from the cloud server. Here, we consider the following two scenarios and assume that the cloud server will be honest in these two scenarios.
► Scenario 1: add a personal health record
When a new personal health record F_{N+1} is generated, the patient selects a keyword set $W' \subset W$ from F_{N+1}, encrypts F_{N+1} using the AES key K, that is $C_{N+1} = AES.Encrypt(K, F_{N+1})$, and builds the keyword index set $I'_1 = \{(T_{w'}, ID_{C_{N+1}}, H(C_{N+1}))\}_{w'\in W'}$ for C_{N+1} based on Algorithm 1. Later, the patient sends the ciphertext C_{N+1} of F_{N+1} and the keyword index set I'_1 to the cloud server.

After receiving the message sent by the patient, the cloud server first stores the ciphertext C_{N+1} of F_{N+1}, and then updates the keyword index set I_1 according to Algorithm 4. When the keyword index set I_1 is updated, the cloud server parses I'_1 to get the set $\{(T_{w'}, ID_{C_{N+1}}, H(C_{N+1}))\}_{w'\in W'}$. For each element $(T_{w'}, ID_{C_{N+1}}, H(C_{N+1}))$ in the set $\{(T_{w'}, ID_{C_{N+1}}, H(C_{N+1}))\}_{w'\in W'}$, the cloud server first uses $T_{w'}$ search the keyword index set I_1 to find the set $\{(ID_{C_{s_{j'}}}, H(C_{s_{j'}}))\}_{j'\in[m']}$, where $ID_{C_{s_{j'}}}$ is the identifier of the ciphertext $C_{s_{j'}}$ of the personal health record $F_{s_{j'}}$ containing keywords w' in the personal health record set F, $H(C_{s_{j'}})$ is the hash value of $C_{s_{j'}}$, and m' is the number of personal health records that contain keywords w' in the personal health record set F. Next, the cloud server adds $(ID_{C_{N+1}}, H(C_{N+1}))$ to the set $\{(ID_{C_{s_{j'}}}, H(C_{s_{j'}}))\}_{j'\in[m']}$.

Algorithm 4 UpdateIndex
```
Input: keyword index set $I'_1$, keyword index set $I_1$
Output: null
1 phase $I'_1 = \{(T_{w'}, ID_{C_{N+1}}, H(C_{N+1}))\}_{w'\in W'}$
2 for each $w' \in W' \subset W$ do
3     phase $I_1(T_{w'}) = \{(ID_{C_{s_{j'}}}, H(C_{s_{j'}}))\}_{j'\in[m']}$
4     add $(ID_{C_{N+1}}, H(C_{N+1}))$ to
5      $\{(ID_{C_{s_{j'}}}, H(C_{s_{j'}}))\}_{j'\in[m']}$
6 end
```

Finally, the patient saves the hash value $H(C_{N+1})$ of the ciphertext C_{N+1} of F_{N+1} to blockchain in the form of the transaction and generates the keyword index set $I'_2 = \{(T_{w'}, txid_{N+1})\}_{w'\in W'}$ for $H(C_{N+1})$ based on the Algorithm 2, then calls the function addIndex() in the data integrity verification contract to add I'_2 to the keyword index set I_2.
► Scenario 2: delete a personal health record
To delete the ciphertext C_q of personal health record $F_q \in F$ from the cloud server, the patient first generates a token $T_{w''} = SHA - 256(K\|w'')$ for each keyword $w'' \in W''$ according to the *TokenGen* algorithm. Note that, W'' is a keyword set extracted from F_q, and $W'' \subset W$. Later, the patient sends the identifier ID_{C_q} of C_q, the hash value $H(C_q)$ of C_q and the token set $T_{W''}$ to the cloud server.

After receiving the message sent by the patient, the cloud server first finds the ciphertext C_q of F_q according to ID_{C_q} and deletes it, and then performs Algorithm 5 to delete all keyword indexes of C_q from the keyword index set I_1. Specifically, for each token $T_{w''} \in T_{W''}$, the cloud server first uses $T_{w''}$ search the keyword index set I_1 to find the set $\{(ID_{C_{s_{j''}}}, H(C_{s_{j''}}))\}_{j''\in[m'']}$, where $ID_{C_{s_{j''}}}$ is the identifier of the ciphertext $C_{s_{j''}}$ of the personal health record $F_{s_{j''}}$ containing keywords w'' in personal health record set F, $H(C_{s_{j''}})$ is the hash value of $C_{s_{j''}}$, and m'' is the number of personal health records that contain keywords w'' in the personal health record set F. Next, the cloud server uses the hash value $H(C_q)$ of C_q to find the keyword index of C_q in the set $\{(ID_{C_{s_{j''}}}, H(C_{s_{j''}}))\}_{j''\in[m'']}$ and delete it.

Algorithm 5 DeleteIndex

```
Input: token set $T_{W''}$, keyword index set $I_1$, the hash
       value $H(C_q)$ of $C_q$
Output: null
1  for each $T_{w''} \in T_{W''}$ do
2      phase $I_1(T_{w''}) = \{(ID_{C_{s_{j''}}}, H(C_{s_{j''}}))\}_{j''\in[m'']}$
3      for $j'' \Leftarrow 1$ to $m''$ do
4          if $H(C_{s_{j''}}) = H(C_q)$ then
5              for $\epsilon \Leftarrow j'' + 1$ to $m''$ do
6                  $(ID_{C_{s_{\epsilon-1}}}, H(C_{s_{\epsilon-1}})) \Leftarrow$
7                  $(ID_{C_{s_\epsilon}}, H(C_{s_\epsilon}))$
8              end
9          end
10     end
11 end
```

Finally, the patient uses the token set $T_{W''}$ and the id of the transaction that saves $H(C_q)$ call the function deleteIndex() in the data integrity verification contract to delete all keyword indexes of $H(C_q)$ from the keyword index set I_2.

VI. IMPLEMENTATION OF SMART CONTRACT

In this part, the source code of the data integrity verification contract is given in Algorithm 6, and the function of each function in the contract is described in detail.

Algorithm 6 Data Integrity Verification Contract

```
pragma solidity ^0.4.19;
contract DataIntegrityVerification{
   bytes[] H;
   address public user;
   address public patient;
   mapping(address⇒bool) public legalUsers;
   struct indexSet{
     bytes32 txid;
   }
bytes32 token;
mapping(bytes32⇒indexSet[]) Index;
event searchEvent(bytes32 result);
function DataIntegrityVerification() public{
   patient=msg.sender;
   legalUsers[patient]=true;
}
modifier onlyPatient(){
     require (msg.sender == patient);
     _;
}
function storageH(bytes a) onlyPatient() public
returns (bool){
   H.push(a);
   return true;
}
function addUser(address nUser) onlyPatient()
public returns(bool){
   if (!legalUsers[nUser]) {
   legalUsers[nUser]=true;
   }
   return legalUsers[nUser];
}
functionremoveUser(address oUser) onlyPatient()
public returns(bool){
   legalUsers[oUser] = false;
}
function addIndex(bytes32 token, bytes32 txHash)
onlyPatient() public returns (bool){
   var temp = indexSet(txid: txHash);
   Index[token].push(temp);
   return true;
 }
function deleteIndex(bytes32 token, bytes32 txHash)
onlyPatient() public{
   uint indexlen=Index[token].length;
   for(uint i=0; i<indexlen; i++){
     if(Index[token][i].txid=txHash){
       for(uint j=i+1; j<indexlen; j++){
         Index[token][j-1]=Index[token][j];
       }
       delete Index[token][indexlen-1];
       Index[token].length- -;
       break;
     }
   }
}
```

Algorithm 6 (continued)

```
    modifier havedAuthorized(){
        require(legalUsers[msg.sender]);
        _;
    }
    function search(bytes32 token) havedAuthorized()
    public{
        uint len=Index[token].length;
        for (uint i=0; i<len; i++){
            searchEvent(result: Index[token][i].txid);
        }
    }
}
```

The contract provides the following nine function interfaces.

DataIntegrityVerification(): A constructor that is used to complete the contract's initialization variable assignment and is executed only once when the contract is created. In the data integrity verification contract above, this function holds the address of the external function caller and adds the patient's Ethereum account address to the legalUser[]. Note that, the name of the constructor must be the same as the name of the contract.

onlyPatient(): A function modifier that is the antecedent condition for function execution. When a function is decorated with onlyPatient(), the function must satisfy the conditions of onlyPatient() before it can run. The condition set by onlyPatient() in the data integrity verification contract above is that the function can only be called by the patient. In particular, "_" in the function modifier represents the function body to be executed.

storageH(): This function can only be called by the patient and is used to save the hash values of encrypted personal health records to the blockchain.

addUser(): This function can only be called by the patient and is used to add the Ethereum account address of the user authenticated by the patient to the legalUser[] of the data integrity verification contract.

removeUser(): This function can only be called by the patient. After the patient removes a user from the system, this function is called to revoke the user's permission to call the data integrity verification contract.

addIndex(): This function can only be called by the patient. It is responsible for saving keyword indexes of the hash values of encrypted personal healths records to the data integrity verification contract above.

deleteIndex(): This function can only be called by the patient. After the patient deletes an encrypted personal health record and its keyword indexes from the cloud server, the function is called to delete all keyword indexes of the hash value of its from the data integrity verification contract above.

TABLE 2. Smart contract fee test.

Operation	Gas Used	Ether Cost
deploy contract	874847	0.000874847
addUser	44305	0.000023948
removeUser	14322	0.000014322

havedAuthorized(): This is also a function modifier that restricts the execution of the function search(). The function search() is executed only if the caller of the function is a user in the legalUser[].

search(): This function can only be called by a user in the LegalUser[]. A legal user calls the function with token generated by a keyword. Then the function performs search and publishes the search results as searchEvents onto blockchain. The user monitors blockchain to get search results.

VII. PERFORMANCE EVALUATION AND SECURITY ANALYSIS

A. PERFORMANCE EVALUATION

We evaluated the performance of the scheme in terms of the cost of smart contract and the efficiency of file encryption.

1) THE COST OF SMART CONTRACT

For the smart contract part, our scheme uses Remix as the development tool of smart contract, writes smart contract with Solidity language and deploys the compiled contract in Ethereum test network Rinkeby.

We first tested the cost of deploy contract, addUser, and removeUser. In particular, we set $gasprice = 1Gwei$ in the experiment. The test results are shown in Table 2.

In Table 2, the deploy contract operation is performed during the initialization stage at a cost of 0.000874847Ether. The addUser operation is performed during the personal health records sharing stage, and the operation is performed after the patient authenticates the identity of the sender of the access request, and the cost of the operation is 0.000023948 Ether. In addition, after the patient deletes a user from the system, the removeUser operation is executed to revoke the user's permission to call the smart contract, and the cost of the operation is 0.000014322 Ether.

We then tested the cost of storageH, addIndex, deleteIndex, and search operations separately. Consider that the cost of these operations may increase as the number of personal health records increases. Therefore, we used the heart as a keyword to test the cost of these operations when a keyword corresponds to 1, 3, 6, 9 and 12 personal health records, and the test results are shown in Table 3.

As it can be seen from Table 3, the execution cost of storageH, addIndex, deleteIndex and search operations will increase as the number of personal health records increases. Among them, the execution costs of addIndex and storageH operations vary greatly as the number of personal health records increases. When the number of personal health

TABLE 3. Smart contract fee test.

Operation	File number	Gas Used	Ether Cost	Increase Ether
storageH	1	64907	0.000064907	0.000548913
	3	164721	0.000164721	
	6	314378	0.000314378	
	9	464099	0.000464099	
	12	613820	0.000613820	
addIndex	1	67118	0.000067118	0.000603933
	3	186482	0.000186482	
	6	343028	0.000343082	
	9	499510	0.000499510	
	12	42596	0.000042596	
deleteIndex	1	26450	0.000026450	0.000003329
	3	27832	0.000027832	
	6	29969	0.000029969	
	9	30524	0.000030524	
	12	31161	0.000031161	
search	1	24348	0.000024348	0.000020715
	3	29529	0.000029529	
	6	34707	0.000034707	
	9	39885	0.000039885	
	12	45063	0.000045063	

records increases from 1 to 12, the cost of performing these two operations increases by 0.000603933 Ether and 0.000548912 Ehter, respectively. In addition, although the execution cost of the search operation also increases with the number of personal health records increases, the increase is very small compared with addIndex and storageH operations, which is just 0.000020715 Ether. In particular, as the number of personal health records increases, the execution cost of the deleteIndex operation increases the least, only increasing 0.000003329 Ether.

By analyzing the test results in Table 2 and Table 3, it is not difficult to find that our data integrity verification contract requires less cost in deployment and invocation, and it is acceptable for patients and users. Moreover, although the cost of calling some of the functions in the contract increases as the number of personal health records increases, the increase is small. So our scheme uses the smart contract to verify the integrity and correctness of the search results returned by the cloud server is feasible in practice.

2) FILE ENCRYPTION EFFICIENCY

We first tested the time consumption of encrypting different files with the AES encryption algorithm on the computer. In the experiment, the CPU parameters of the computer are Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz, RAM is 4.00GB, the running system is Windows10 and the programming language is Java. The test results are shown in Figure 6.

According to Figure 6, the encryption time of AES encryption algorithm is proportional to the size of the file. As the

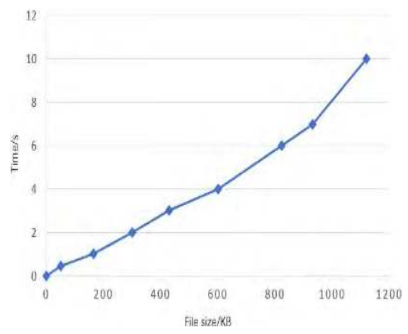


FIGURE 6. File encryption time.

file size increases, the encryption time of the AES encryption algorithm also increases. In addition, it also can be seen from Figure 6 that when AES encryption algorithm is used to encrypt files less than 100KB, the time spent is less than 1s. Even if the file size is increased to 1200KB, the additional time consumed is only about 10s. Since patient health records increase dynamically, we use AES encryption algorithm to encrypt the patient's personal health records, which can greatly improve the efficiency of the scheme.

Later, we compared the test results in Figure 6 with the time consumption of encrypting different files using the attribute-based encryption scheme in literature [41]. The comparison results are shown in Figure 7.

By analyzing Figure 7, it can be seen that the time consumed by AES encryption algorithm and attribute-based encryption algorithm in encrypting smaller files is not

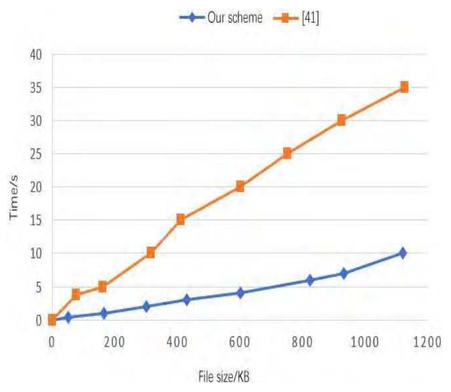


FIGURE 7. File encryption efficiency comparison.

very different. However, when the file is large, the encryption efficiency of the AES encryption algorithm is significantly higher than the attribute-based encryption algorithm. As the patient's health data contains many large image files, such as CT, X-ray, etc, our scheme is more applicable than traditional schemes that use attribute-based encryption scheme to encrypt personal health records.

B. SECURITY ANALYSIS

Security is a crucial issue in the personal health record sharing system. Here, we analyzed the security of our scheme from the following five different aspects.

1) TAMPER-PROOF

In our scheme, encrypted personal health records uploaded to the cloud server cannot be arbitrarily modified. Because the patient saves the hash values of encrypted personal health records to the blockchain through a transaction. The tamper-proof nature of blockchain ensures that data in the transaction cannot be arbitrarily changed unless someone has more than 51% computing power of the entire blockchain network.

2) PRIVACY PROTECTION

Personal health records are highly sensitive privacy data, no information about the patient's personal health records should be disclosed without his/her authorization. In our scheme, the patient encrypts the original personal health records and then outsources them to the cloud server. This not only solves the problem of limited blockchain storage capacity, but also greatly reduces the risk of privacy information disclosure in the original personal health records.

3) KEY MANAGEMENT

In our scheme, the key management and distribution problems in traditional cryptography schemes are effectively solved by using blockchain to realize key distribution and management.

4) FINE-GRAINED ACCESS CONTROL

In our scheme, the patient generates and distributes the attribute private key for the user. It not only solves many security risks caused by untrusted attribute authorities in attribute-based encryption schemes, but also fine-grained access control is implemented for patients to their personal health records without relying on any third party

TABLE 4. Functional comparison of similar schemes.

Function	[38]	[42]	[43]	Our scheme
Tamper-proofing	Y	N	Y	Y
Privacy protection	Y	Y	Y	Y
Distributed key management	N	N	Y	Y
Keyword search	N	N	N	Y
Data integrity verification	N	N	Y	Y
Fine-grained access control	N	N	N	Y

5) SEARCHABILITY AND VERIFIABILITY

Our scheme stores keyword indexes of encrypted personal health records to the cloud server, and keyword indexes of the hash values of encrypted personal health records are stored in the smart contract. In this way, keyword search and data integrity verification are achieved.

Based on the above security analysis, we made a functional comparison between our scheme and similar schemes cited in references, and the results of the comparison are shown in Table 4. Note that, in Table 4, Y represents that the literature supports this function, and N represents that the literature does not support this function.

As it can be seen from Table 4, our scheme is superior to the existing similar schemes in terms of functions such as tamper-proofing, privacy protection, distributed key management, keyword search, data integrity verification, and fine-grained access control. Moreover, our scheme provides an effective solution for improving data integrity, privacy and security of personal health record sharing system, while also ensuring the fast data retrieval capability of authorized users.

VIII. CONCLUSION

In this paper, a new personal health records sharing scheme is proposed. In the new scheme, the patient generates and distributes the attribute private key for the user, enabling the scheme to achieve fine-grained access control without relying on any third party. In addition, because of the blockchain has the characteristics of decentralization and tamper-proof, the use of the blockchain to maintain keys in the scheme makes the management and distribution of keys more secure. Furthermore, the hash values of encrypted personal health records are stored on the blockchain, and the related index set is stored in the smart contract, so that the personal health records receiver can conveniently and quickly verify the integrity of encrypted personal health records received from the cloud server.

Although our scheme solves some of the problems in the existing personal health records sharing schemes. However, how to verify whether the cloud server performs file update and file delete operations according to the patient's requirements during the personal health record management stage still needs further research.

REFERENCES

[1] D. Song, A. Perrig, and D. Wagner, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Secur. Privacy*, May 2000, pp. 44–55.
[2] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2006, pp. 79–88.
[3] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou, "Secure ranked keyword search over encrypted cloud data," in *Proc. IEEE 30th Int. Conf. Distrib. Comput. Syst.*, Jun. 2010, pp. 253–262.
[4] N. Premasathian and S. Choto, "Searchable encryption schemes: With multiplication and simultaneous congruences," in *Proc. 9th IEEE Int. ISC Conf. Inf. Secur. Cryptol.*, Tabriz, Iran, Sep. 2012, pp. 147–150.
[5] H. Li, F. Zhang, J. He, and H. Tian, "A searchable symmetric encryption scheme using blockchain," Nov. 2017, *arXiv:1711.01030*. [Online]. Available: https://arxiv.org/abs/1711.01030
[6] Y. Zhang, R. H. Deng, J. Shu, K. Yang, and D. Zheng, "TKSE: Trustworthy keyword search over encrypted data with two-side verifiability via blockchain," *IEEE Access*, vol. 6, pp. 31077–31087, 2018.
[7] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Advances in Cryptology—EUROCRYPT* (Lecture Notes in Computer Science), vol. 3494, R. Cramer, Ed. Berlin, Germany: Springer-Verlag, 2005, pp. 457–473.
[8] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. 13th ACM Conf. Comput. Commun. Secur.*, 2006, pp. 89–98.
[9] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symp. Secur. Privacy*, May 2007, pp. 321–334.
[10] M. Chase and S. S. Chow, "Improving privacy and security in multi-authority attribute-based encryption," in *Proc. 16th ACM Conf. Comput. Commun. Secur.*, 2009, pp. 121–130.
[11] N. Attrapadung and H. Imai, "Dual-policy attribute based encryption," in *Proc. Int. Conf. Appl. Cryptogr. Netw. Secur.*, vol. 5536. Springer, 2009, pp. 168–185.
[12] M. Green, S. Hohenberger, and B. Waters, "Outsourcing the decryption of ABE ciphertexts," in *Proc. USENIX Secur. Symp.*, 2011, no. 3, p. 34.
[13] J. Hur and D. K. Noh, "Attribute-based access control with efficient revocation in data outsourcing systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 7, pp. 1214–1221, Jul. 2011.
[14] P. Zhang, Z. Chen, K. Liang, S. Wang, and T. Wang, "A cloud-based access control scheme with user revocation and attribute update," in *Proc. Australas. Conf. Inf. Secur. Privacy.* Springer, 2016, pp. 525–540.
[15] J. Li, X. Lin, Y. Zhang, and J. Han, "KSF-OABE: Outsourced attribute-based encryption with keyword search function for cloud storage," *IEEE Trans. Services Comput.*, vol. 10, no. 5, pp. 715–725, Sep./Oct. 2017.
[16] J. Li, W. Yao, Y. Zhang, H. Qian, and J. Han, "Flexible and fine-grained attribute-based data storage in cloud computing," *IEEE Trans. Services Comput.*, vol. 10, no. 5, pp. 785–796, Jan. 2016.
[17] J. Li, W. Yao, J. Han, Y. Zhang, and J. Shen, "User collusion avoidance CP-ABE with efficient attribute revocation for cloud storage," *IEEE Syst. J.*, vol. 12, no. 2, pp. 1767–1777, Jun. 2018.
[18] J. Li, Q. Yu, and Y. Zhang, "Key-policy attribute-based encryption against continual auxiliary input leakage," *Inf. Sci.*, vol. 470, pp. 175–188, Jan. 2019.
[19] J. Li, Q. Yu, and Y. Zhang, "Hierarchical attribute based encryption with continuous leakage-resilience," *Inf. Sci.*, vol. 484, pp. 113–134, May 2019.
[20] K. Emura, A. Miyaji, A. Nomura, and K. Omote, "A ciphertext-policy attribute-based encryption scheme with constant ciphertext length," in *Proc. Int. Conf. Inf. Secur. Pract. Exper.* Berlin, Germany: Springer 2009, pp. 13–23.
[21] N. Attrapadung, J. Herranz, F. Laguillaumie, B. Libert, E. de Panafieu, and C. Ràfols, "Attribute-based encryption schemes with constant-size ciphertexts," *Theor. Comput. Sci.*, vol. 422, pp. 15–38, Mar. 2012.
[22] S. Nakamoto. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System.* [Online]. Available: https://bitcoin.org/bitcoin.pdf
[23] K. N. Griggs, O. Ossipova, C. P. Kohlios, A. N. Baccarini, E. A. Howson, and T. Hayajneh, "Healthcare blockchain system using smart contracts for secure automated remote patient monitoring," *J. Med. Syst.*, vol. 42, no. 7, pp. 130–138, 2018.
[24] C. Lin, D. He, X. Huang, K. K. R. Choo, and A. V. Vasilakos, "BSeIn: A blockchain-based secure mutual authentication with fine-grained access control system for industry 4.0," *J. Netw. Comput. Appl.*, vol. 116, pp. 42–52, Aug. 2018.
[25] C. Lin, D. He, X. Huang, M. K. Khan, and K.-K. R. Choo, "A new transitively closed undirected graph authentication scheme for blockchain-based identity management systems," *IEEE Access*, vol. 6, pp. 28203–28212, 2018.
[26] A. Dorri, S. S. Kanhere, and R. Jurdak, "Towards an optimized blockchain for IoT," in *Proc. 2nd Int. Conf. Internet-Things Design Implement.*, Pittsburgh, PA, USA, Apr. 2017, pp. 173–178.
[27] S. Huh, S. Cho, and S. Kim, "Managing IOT devices using blockchain platform," in *Proc. 19th Int. Conf. Adv. Commun. Technol. (ICACT)*, Feb. 2017, pp. 464–467.
[28] H. Shafagh, L. Burkhalter, A. Hithnawi, and S. Duquennoy, "Towards blockchain-based auditable storage and sharing of IOT data," in *Proc. Cloud Comput. Secur. Workshop.*, 2017, pp. 45–50.
[29] A. Boudguiga, N. Bouzerna, L. Granboulan, A. Olivereau, F. Quesnel, A. Roger, and R. Sirdey, "Towards better availability and accountability for IOT updates by means of a blockchain," in *Proc. IEEE Eur. Symp. Secur. Privacy Workshop*, Apr. 2017, pp. 50–58.
[30] S. Wang, Y. Zhang, and Y. Zhang, "A blockchain-based framework for data sharing with fine-grained access control in decentralized storage systems," *IEEE Access*, vol. 6, pp. 38437–38450, Jun. 2018.
[31] Y. Zhu, Y. Qin, Z. Zhou, X. Song, G. Liu, and W. C.-C. Chu, "Digital asset management with distributed permission over blockchain and attribute-based access control," in *Proc. IEEE Int. Conf. Services Comput. (SCC)*, San Francisco, CA, USA, Jul. 2018, pp. 193–200.
[32] Y. Zhu, Y. Qin, G. Gan, Y. Shuai, and W. C.-C. Chu, "TBAC: Transaction-based access control on blockchain for resource sharing with cryptographically decentralized authorization," in *Proc. IEEE 42nd Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, Jul. 2018, pp. 535–544.
[33] M. Baza, M. Nabil, N. Lasla, K. Fidan, M. Mahmoud, and M. Abdallah, "Blockchain-based firmware update scheme tailored for autonomous vehicles," Nov. 2018, *arXiv:1811.05905*. [Online]. Available: https://arxiv.org/pdf/1811.05905.pdf
[34] K. Fan, Y. Ren, Y. Wang, H. Li, and Y. Yang, "Blockchain-based efficient privacy preserving and data sharing scheme of content-centric network in 5G," *IET Commun.*, vol. 12, no. 5, pp. 527–532, Mar. 2018.
[35] X. Yue, H. Wang, D. Jin, M. Li, and W. Jiang, "Healthcare data gateways: Found healthcare intelligence on blockchain with novel privacy risk control," *J. Med. Syst.*, vol. 40, no. 10, pp. 218–225, 2016.
[36] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman, "MedRec: Using blockchain for medical data access and permission management," in *Proc. 2nd Int. Conf. Open Big Data (OBD)*, Aug. 2016, pp. 25–30.
[37] Y. Mei, "Research on blockchain method for secure storage of medical records," *J. Jiangxi Normal Univ.*, vol. 41, no. 5, pp. 481–487, 2017.
[38] H. Li, L. Zhu, M. Shen, F. Gao, X. Tao, and S. Liu, "Blockchain-based data preservation system for medical data," *J. Med. Syst.*, vol. 42, no. 8, pp. 141–153, 2018.
[39] K. Fan, S. Wang, Y. Ren, H. Li, and Y. Yang, "MedBlock: Efficient and secure medical data sharing via blockchain," *J. Med. Syst.*, vol. 42, no. 8, pp. 136–147, 2018.
[40] G. G. Dagher, J. Mohler, M. Milojkovic, and P. B. Marella, "Ancile: Privacy-preserving framework for access control and interoperability of electronic health records using blockchain technology," *Sustain. Cities Soc.*, vol. 39, pp. 283–297, May 2018.
[41] H. Wang and Y. Song, "Secure cloud-based EHR system using attribute-based cryptosystem and blockchain," *J. Med. Syst.*, vol. 42, no. 8, pp. 152–160, 2018.
[42] Q. Xia, E. Sifah, A. Smahi, S. Amofa, and X. Zhang, "BBDS: Blockchain-based data sharing for electronic medical records in cloud environments," *Information*, vol. 8, no. 2, pp. 44–59, 2017.
[43] X. Zheng, R. R. Mukkamala, R. Vatrapu, and J. Ordieres-Mere, "Blockchain-based personal health Data sharing system using cloud storage," in *Proc. IEEE 20th Int. Conf. e-Health Netw., Appl. Services (Healthcom)*, Sep. 2018, pp. 1–6.

SHANGPING WANG received the B.S. degree in mathematics from the Xi'an University of Technology, Xi'an, China, in 1982, the M.S. degree in applied mathematics from Xi'an Jiaotong University, Xi'an, in 1989, and the Ph.D. degree in cryptology from Xidian University, Xi'an. He is currently a Professor with the Xi'an University of Technology. His current research interests include cryptography and information security.

DAN ZHANG received the B.S. degree in mathematics from Luliang University, lu'liang, China, in 2017. She is currently pursuing the M.S. degree with the Xi'an University of Technology, Xi'an, China. Her current research interests include information security and blockchain technology.

YALING ZHANG received the B.S. degree in computer science from Northwest University, Xi'an, China, in 1988, the M.S. degree in computer science and the Ph.D. degree in the mechanism electron engineering from the Xi'an University of Technology, Xi'an, in 2001 and 2008, respectively, where she is currently a Professor. Her current research interests include cryptography and network security.