Non-Repudiation Storage and Access Control Scheme of Insurance Data Based on Blockchain in IPFS

JIN SUN, XIAOMIN YAO, SHANGPING WANG, AND YING WU

School of Science, Xi’an University of Technology, Xi’an 710054, China

Corresponding authors: Jin Sun (oksunjin@xaut.edu.cn) and Xiaomin Yao (1556259441@qq.com)

This work was supported in part by the National Natural Science Youth Foundation of China under Grant 61303223, and in part by the National Natural Science Foundation of China under Grant 61572019.

ABSTRACT  The insurance business plays a quite significant role in people’s lives, but in the process of claim settlement, there are still various frauds such that the insurance companies’ refusal to compensate or customers’ malicious fraud to obtain compensation. Therefore, it is very important to ensure fair and just claims. In this paper, by combining the blockchain technology and the ciphertext-policy attribute-based encryption system, we build a scheme for secure storage and update for insurance records under the InterPlanetary File System (IPFS) storage environment in the insurance system. In this scheme, we use the fog node to outsource encryption of insurance records to improve the efficiency of the staff; In addition, we store encrypted insurance records on IPFS to ensure the security of the storage platform and avoid the single point failure of the centralized mechanism. In addition, we use the immutability of the blockchain to achieve the non-repudiation of both insurance companies and the client. The security proof shows that the proposed scheme can achieve selective security against selected keyword attacks. Our scheme is efficient and feasible under performance analysis and real data set experiments.

INDEX TERMS  Attribute-based encryption, blockchain, ciphertext update, fog-computing, interplanetary file system.

I. INTRODUCTION

With the improvement of the people’s life quality, various insurances have emerged at the historic moment. However, in the process of participating in insurance, there are always various unfair events. On the one hand, insurance company maliciously deceives the client or refuse to compensate for damaging the rights of the client; On the other hand, the client will maliciously scam the insurance company for compensation. Therefore, how to achieve a fair and just insurance business between insurance company and the client is particularly important. The immutability and traceability of blockchain technology can solve this problem well.

Blockchain is a decentralized, immutable and trusted distributed ledger, which provides a secure, stable, transparent, auditable and effective way to record transactions and data information interactions. Therefore, storing the information of insurance records on the blockchain can not only ensure the originality of the data, but also solve the security problem of the central authorization mechanism. In this way, a fair and just insurance business can be achieved between the insurance company and the client. However, To ensure client’s privacy, the insurance staffs need to encrypt the insurance records before storage. The heavy computing operations will reduce the efficiency of the staff, for the purpose of improving the staff’s efficiency, and reducing the client’s computing burden, therefore, it is a good choice to outsource the encryption and decryption of insurance records to fog nodes.

In 2014, Cisco proposed the definition of fog computing[1]. The fog node is located on the layer between the cloud and the end user. Compared with the cloud, it is closer to the end user. Therefore, the fog node can provide a large amount of computing for resource-constrained devices and reduce the computing burden of resource-constrained devices, so it is very suitable as an outsourcing agent.

Therefore, we outsource great majority computing operations to fog nodes for processing. However, fog nodes are incompletely trusted and vulnerable to attack, which will
seriously threaten the security of insurance records. An effective solution is that encrypts the insurance records before outsourcing encryption. Attribute-based encryption (ABE)\[2\] is an one-to-many encryption technology, it has an access control mechanism that allows access control of encrypted data using access policies and attributes. In particular, ciphertext-policy attribute-based encryption (CP-ABE) is the data owner to define the access strategy, and only the user who meet the access policy can decrypt, so it can ensure data security and fine-grained access control.

However, the existing ciphertext-policy attribute-based encryption scheme still cannot solve the situation of non-repudiation in the insurance system. Therefore, we propose a storage and access control scheme based on blockchain in the insurance system. In this scheme, we store the insurance record on IPFS to ensure the security of the storage platform. Secondly, the hash of the insurance record is recorded on the blockchain, and it can take effect only after the client issues a confirmed transaction. This effectively prevents repudiation between the insurance company and the client. In addition, we also outsource a large amount of encryption and decryption computations to the fog node, improving the computational efficiency of the staff. Finally, we update each claim of the insurance records to prevent insurance companies and the client from infringing on each other’s rights.

Our contribution: In our scheme, we use the non-tamperable of blockchain technology to achieve non-repudiation between the insurance company and the client. Our scheme has the following advantages:

Secure content storage: We stored insurance records on the InterPlanetary File System (IPFS). IPFS is a distributed storage protocol that is content addressable, it not only can ensure the security of the stored content, but also generate a unique hash value for the stored file to avoid duplicate storage of the file.

Client confirmation: The staff of the insurance company first records the hash of the insurance information on the blockchain, and then the client verifies whether the hash value is consistent with the hash of the insurance records. If they are consistent, a confirmation transaction is issued on the blockchain by the client, and at this time, the insurance policy is officially effective.

Outsourcing encryption: We outsource a large amount of computations in the encryption and decryption process to the fog node, which not only reduces the computation amount of the staff and the client but also improves work efficiency.

Update of ciphertext: After each claim, we update the content of the claim to our insurance record to ensure fair transactions between the insurance company and the client.

The specific structure of our scheme is as follows: In the section II, we analyze the existing scheme, in the section III, we list the knowledge points that may be used in the scheme, in the section IV, we overview the system model of our scheme, the algorithm in our scheme is described in detail in the section V, and the security of our scheme is proved in the section VI, the performance of our scheme is analyzed in the section VII, and finally, we are in the Section VIII summarizes our scheme.

II. RELATED WORK
In Table 1, we give a brief overview the outsourcing computing, searchable and ciphertext update of existing schemes.

Zhang et al. [3] proposed an access control scheme that supports fog computing outsourcing and attribute updating. The heavy calculation operations of encryption and decryption are outsourced to the fog nodes. Therefore, the calculation in the encryption and decryption process is independent of the number of attributes. However, this scheme cannot provide keyword search, so the saved data cannot be retrieved in real-time. Huang et al. [4] proposed a fine-grained access control scheme with ciphertext update and computing outsourcing for fog computing in the Internet of Things. It first encrypts data based on attributes and then outsources to cloud storage. Therefore, only the users who meet the access policy can decrypt the stored data. However, this scheme only focuses on data encryption, decryption, update processes, but the data is not searchable, so it cannot be applied to various practical scenarios. Green et al. [5] proposed a new paradigm for ABE which largely eliminates the defect that the time required for decryption increases with the complexity of the access formula. However, in this scheme, the stored data cannot be searched and does not have the ciphertext update function. Asim et al. [6] proposed a new scheme for attribute-based encryption based on ciphertext policy, which allows outsourcing of computationally intensive encryption and decryption operations. In this scheme, the agent cannot learn any information about the ciphertext from the part encrypted text. The user can decrypt the stored ciphertext only if he has the secret key associated with a set of attributes that satisfy the association policy. However, although this scheme reduces the decryption operation for users with limited resources, the stored data cannot be retrieved and updated at any time, so it is not suitable for practical scenarios. Zuo et al. [7] first proposed a CCA security model with outsourced decryption ABE, and then it proposes a specific ABE solution for CCA security with outsourced decryption. However, in this scheme, data encrypted based on attributes cannot be efficiently retrieved, and the ciphertext cannot be updated. Zhang et al. [8] proposed a CP-ABE scheme with a fixed amount of calculation and ciphertext length. Because the access policy of this scheme is an AND gate with multiple attribute values, so, it is a good choice for data sharing in the cloud. However, effective retrieval cannot be performed in the shared data, and the shared content cannot be updated. Zhang et al. [9] proposed a technique to add a matching stage before the decryption stage. This ciphertext is used to test whether the attribute key matches the access policy in the ciphertext. This scheme will generate a special key component during the decryption process, which can be quickly decrypted by pairing, although the decryption speed of this scheme is very fast, the data in this scheme does not have search capabilities. Zheng et al. [10] proposed a
TABLE 1. Symbols table.

| Scheme       | Two-side verifiability | Outsourcing encryption | Searchable | Outsourcing decryption | Ciphertext update | Access policy |
|--------------|-------------------------|------------------------|------------|------------------------|------------------|--------------|
| Zhang et al. | No                      | Yes                    | No         | Yes                    | Yes              | Access tree  |
| Huang et al. | No                      | Yes                    | No         | Yes                    | No               | Access tree  |
| Green et al. | No                      | Yes                    | No         | Yes                    | No               | LSSS         |
| Asim et al.  | No                      | Yes                    | No         | Yes                    | No               | AND          |
| Zuo et al.   | No                      | Yes                    | No         | Yes                    | No               | Access tree  |
| Zhang et al. | No                      | Yes                    | No         | Yes                    | No               | AND          |
| Zheng et al. | No                      | Yes                    | No         | Yes                    | No               | Access tree  |
| Zhao et al.  | No                      | Yes                    | No         | Yes                    | No               | AND          |
| Chen et al.  | Yes                     | Yes                    | Yes        | No                     | Yes              | —            |
| Zhang et al. | Yes                     | No                     | Yes        | No                     | No               | Access tree  |
| Li et al.    | Yes                     | No                     | No         | No                     | No               | LSSS         |
| Li et al.    | No                      | NO                     | No         | Yes                    | No               | LSSS         |
| Li et al.    | No                      | Yes                    | Yes        | NO                     | No               | LSSS         |
| Miao et al.  | No                      | Yes                    | Yes        | No                     | Yes              | Access tree  |
| Ours         | Yes                     | Yes                    | Yes        | Yes                    | Yes              | Access tree  |

The scheme outsources cumbersome search operations to the cloud, and can verify that the cloud honestly performed the search operations. In addition, users whose attributes meet the access control policy specified by the data owner can search outsourced data. However, this scheme does not have the ciphertext update function, so it cannot be used in medical, logistics, insurance and other practical scenarios. Zhao et al. [11] proposed a novel data sharing protocol by combining and using attribute-based encryption (ABE) and attribute-based signature (ABS). In this agreement, the data owner outsources heavy encryption and decryption computations to the fog node, and supports outsourced data retrieval and ciphertext update. However, the user cannot verify the source of the data. Chen et al. [12] first introduced the rational laziness and partially dishonest workers in the outsourcing computing model. In addition, the scheme proposes a new fair conditional payment scheme based only on traditional electronic cash systems for outsourcing calculations. The proposed construction uses semi-trusted third parties to achieve fairness and efficiency. However, only in exceptional cases (i.e. in case of dispute) participate in the agreement. In addition, since neither the secret sharing / split scheme nor the cut selection protocol be used to generate or verify payment tokens, this scheme has certain advantages in terms of efficiency. However, the protocol in this scheme does not have search capabilities. Zhang et al. [13] introduced a reliable keyword search scheme for encrypted data without any trusted or untrusted third parties. In this scheme, the digital signature-based encrypted data index allows users to search outsourced encrypted data and check whether the search results returned by the cloud meet predetermined search requirements. In particular, the scheme achieves server-side verifiability for the first time, which protects honest cloud servers from malicious data owners during the data storage phase. In addition, even if the user or cloud is malicious, the blockchain technology and hash function can be used to achieve the fairness of the payment of search fees without introducing any third party. However, this solution focuses on the search of outsourced data and fair payment, without considering the update of outsourced data. After outsourcing encryption and decryption operations to fog nodes in most documents, the outsourced data does not have search capabilities, and in general, the outsourced data will not be updated. Li et al. [14] proposed a ciphertext strategy that can realize the fine-grain access control of encrypted IOT data on the cloud based on an attribute encryption scheme, which can ensure user privacy. However, this solution does not involve outsourcing operations of encryption and decryption so that it owning huge computational and storage burden for cloud, nor does it support keyword search. Li et al. [15] proposed a verifiable ABE scheme for outsourcing decryption, which not only provides authorized and unauthorized users with outsourcing decryption services, but also verifies the correctness of outsourcing decryption. However, in this scheme, keyword search and ciphertext update are not involved. Li et al. [16] proposed an attribute-based encryption scheme with outsourcing key release and outsourcing encryption and decryption, which supports keyword search. However, this solution does not have the ciphertext update function. Miao et al. [17] proposed an attribute-based keyword search scheme that hides the access strategy in shared multi-owners and demonstrates improvements to it. However, this scheme does not outsource heavy encryption and decryption calculations to fog nodes, so it owning higher computational and storage burden for cloud than our scheme. Therefore, we propose a secure storage and ciphertext update scheme in the insurance system. In this scheme, we outsource the encryption and decryption operations to the fog node for processing, and the outsourced data can be searched and updated at any time. In addition, the scheme can achieve client-side verification, which makes our scheme more suitable for various practical scenarios.

III. PRELIMINARIES

A. BILINEAR MAP

Definition 1 (Bilinear Map) [18]: Let $G_1$ and $G_2$ be two multiplicative cyclic groups of prime order $p$, $g$ is the generator.
of $G_1$ and $e : G_1 \times G_1 \rightarrow G_2$ be the bilinear map which has several properties:

- Bilinearity: $\forall g \in G_1, h \in G_1, \forall x, y \in \mathbb{Z}_p^*, e(g^x, h^y) = e(g, h)^{xy}$.
- Non-degeneracy: $\exists g \in G_1, h \in G_1, e(g, h) \neq 1$.
- Computability: for all $g \in G_1, h \in G_1, e(g, h)$ can be effectively calculated.

B. HARDNESS PROBLEM

Definition 2 (Decision Bilinear Diffie-Hellman (DBDH) Assumption) [19]: Given the bilinear map parameters $(G_1, G_2, \beta, g, e)$ and elements $a, b, c, Z \in \mathbb{Z}_p^*$, the DBDH assumption states that no probabilistic time adversary $A$ can distinguish the tuple $(g, g^a, g^b, g^c, e(g, g)^{abc})$ from the tuple $(g, g^a, g^b, g^c, e(g, g)^{-abc})$ with a non-negligible advantage, where the advantage of adversary $A$ is defined as

$$\text{Adv}_{\text{DBDH}}^{\beta}(A) = \left| \Pr \left[ A \left( g, g^a, g^b, g^c, e(g, g)^{abc} \right) = 1 \right] - \Pr \left[ A \left( g, g^a, g^b, g^c, e(g, g)^{-abc} \right) = 1 \right] \right|.$$ 

C. ACCESS STRUCTURE

Definition 3 (Access Structure) [20]: Let $\{P_1, P_2, \cdots, P_n\}$ be a group of attributes. For $\forall B, C$, if $B \in A$ and $B \subseteq C$, then $C \in A$, we say that $A \subseteq 2^{|P_1, P_2, \cdots, P_n|}$ is monotone. An access structure (or monotone access structure) $A$ is a non-empty subset of $\{P_1, P_2, \cdots, P_n\}$, the set in $A$ is an authorized set, and the set not in $A$ is called an unauthorized set.

Definition 4 (Access Tree): As in literature [21], let $T$ be an access tree, we first define some functions to simplify the use of $T$. $x$ is a node in $T$, $num_x$ is the number of children of node $x$, $I_x$ represents the threshold where $I_x \in \{1, num_x\}$, and $\text{attr}(y)$ represents the attribute related to a leaf node $y$, $\text{parent}(x)$ represents the parent node of node $x$ in the tree, and $\text{index}$ returns the label associated with node $x$, these index values ??are uniquely assigned to the nodes in the access tree in any way. Each non-leaf node of $T$ represents a threshold gate, which can be described by its child nodes $num_x$ and a threshold $I_x$, and each leaf node represents an attribute, and the threshold value of each leaf node is $I_x = 1$, when $I_x = 1$, it means OR gate, and when $I_x = num_x$, it means AND gate.

IV. OVERVIEW OF OUR SYSTEM

A. SYSTEM MODEL

The model includes five entities, namely, blockchain, IPFS, fog nodes, insurance company system, and clients.

When a client needs to purchase insurance, the first registers in the insurance company system and generates his own public-private key pair based on the system's private key. For the generated insurance record, the staff stores the insurance record for future claims. Because the insurance record involves the privacy of the client, so it should be encrypted before saving.

In our system, the client and the insurance company staff first generate the insurance record through negotiation, as shown in step 1; The staff first signs the insurance record, then extracts the keywords, and generates an index for the file, finally he saves it to the insurance company system. For efficiency, we outsource the insurance record to the fog node for encryption. However, the insurance record involves the privacy of the client, so the insurance record needs to be encrypted before outsourcing, and then the encrypted insurance record is outsourced to the fog node. The fog node returns the encrypted insurance record to the staff, as shown in step 2, finally, the staff signed the insurance record returned by the fog node and uploaded it to IPFS for storage, and then IPFS returned a hash address for it, as shown in step 4, 5; Then, the staff broadcasts the Hash value of the insurance record and the Hash address returned by IPFS to the blockchain by a transaction, and obtains the block ID, as shown in step 6, 7; In order to facilitate client verification, the staff will send the ID to the client, as shown in step 8; After the client obtains the transaction ID, he checks whether the staff has correctly saved the insurance record according to the transaction ID. If the hash is correct, a confirmation transaction is issued by the client on the blockchain, as shown in step 9; At this point, the data storage phase is complete. When a claim is needed, the client first sends a claim request to the insurance company. After the system smart device of the insurance company matches the index, the system smart device sends a token containing a hash address for it, as shown in step 10, 11; The client downloads the corresponding insurance record on IPFS according to the hash address, as shown in step 12, 13; The client sends the insurance records to the fog node for decryption, as shown in step 14, 15; Then, the client negotiates the details of the claim with the staff, and then updates the insurance record, as shown in step 16; After updating, storing the insurance policy in encrypted form, and repeat steps 2 − 7.

B. SECURITY MODEL

Now, we define the selected plaintext security of our scheme, and the polynomial time probability adversary will develop an access structure $T^*$ at the beginning of the security game.

- **Init:** The PPT adversary $A$ specifies an access structure $T^*$ and sends it to the challenger $C$.
- **Setup:** The challenger $C$ gets the public key $pk$ by calling the Setup algorithm and sends it to the adversary $A$.
- **Phase 1:** The adversary $A$ can commit any attribute set $S$ to the challenger $C$, and query the private key for $S$. The only limitation is that all the queried attribute sets do not satisfy $T^*$. For the attribute set $S$, the challenger $C$ calls the KeyGen algorithm and sends the corresponding secret key $sk$ to the adversary $A$.
- **Challenge:** The adversary $A$ commits two messages of equal length $m_0$ and $m_1$ to the challenger $C$. After receiving the messages, the challenger $C$ first selects $\theta \in \{0, 1\}$ at random, and then calls the $\text{DO-Encrypt}$ and $\text{Fog-Encrypt}$ algorithms to encrypt $m_\theta$ and generates the ciphertext $CT^*$. Finally, the challenger $C$ sends the challenge ciphertext $CT^*$ to the adversary $A$. 
Phase 2: The adversary A can query more secret keys of other attribute sets. The only limitation is that the secret key cannot decrypt the challenge ciphertext $CT^*$.  

Guess: The adversary A outputs a guess $\theta^*$. In this security model, the adversary A’s advantage is $\text{Adv}(A) = |\Pr(\theta' = \theta) - \frac{1}{2}|$.

During the search process, the adversary is allowed to submit the search token during the challenge phase, but he cannot learn any content with keywords hanging from the search token. We formalize the security model by selectively chosen-keyword game and keyword secret games.

Setup: The challenger C inputs the security parameter $k$, and calls the $\text{KeyGen}$ algorithm to get the secret key $sk$, at this time, the adversary A just know the public key $pk$, and $mk$ is only known by the challenge C.

Phase 1: The adversary A queries Trap oracle for a series of keyword sets $(w_1, \ldots, w_\tau)$, as follows:

- $\text{Trap}(sk, w)$: The challenger C generates the trapdoor $T_w$ by calling $\text{Trap}(sk, w)$ algorithm, and then sends it to the adversary A.

Challenge: The adversary A first chooses two keyword set $w_0, w_1$, and sends them to the challenger C, then the challenger C selects a bit $\theta \in \{0, 1\}$ at random, while it is required that $w_0$ and $w_1$ cannot be issued in the Trap oracle. Finally the challenger C sets $I_b = \text{Enc}(pk, w_b, W)$ and sends it to the adversary A.

Phase 2: The adversary A queries Trap oracle for keyword set $\{w_{i+1}, \ldots, w_\tau\}$ as follow:

- $\text{Trap}(sk, w_i \neq w_0, w_1)$. The challenger C first calls $\text{Trap}(sk, w_i \neq w_0, w_1)$ algorithm to generate the trapdoor $T_{w_i}$, and then sends it to the adversary A.

Guess: The adversary A outputs a guess $\theta \in \{0, 1\}$, it will win the game if $\theta' = \theta$. In this game, the advantage of the adversary A in breaking our scheme is defined as $\text{Adv}(1^k) = |\Pr(\theta' = \theta) - \frac{1}{2}|$.

In order to ensure keyword security, our scheme must ensure that malicious adversary will not get any content from encrypted keywords and search tokens. Specifically, if there is no time probability polynomial adversary A can obtain any content related to the keyword from the index and search token, the keyword security can be ensured. Keyword security games are as follows:

Setup: The challenger C inputs a security parameter $k$, and then calls $\text{setup}(k, U)$ algorithm to get the public-master key pair $(pk, mk)$.

Phase 1: The adversary A queries $\text{KeyGen}$ algorithm for many times.
V. OUR SCHEME CONCRETE

In table 2, we give some symbol description which we will use in our scheme.

| Symbol | Description |
|--------|-------------|
| M      | Insurance records |
| I      | The index of insurance records |
| H      | The hash address returned by IPFS |
| T      | Access tree |
| W      | Access control |
| S      | The set of attribute |
| ID     | Transaction number on the blockchain attribute |

- **Setup**: The system takes two multiplicative cyclic groups $G_1, G_2$ of the order $p$. $g \in G_1$ is generator of $G_1$ and $e : G_1 \times G_1 \rightarrow G_2$ is bilinear map. Two collision resistant hash function are named as $H_1 : \{0, 1\}^* \rightarrow Z_p^*, H_2 : \{0, 1\}^* \rightarrow G_1$, the system randomly selects $a,b \in Z_p^*$, $h \in G_1$, for each attribute $j \in S$, the system selects $u_j \in Z_p^*$ at random. Finally, the system outputs the public key $mpk = \{G_1, G_2, p, g, H_1, H_2, e, h\}$ and the master private key $msk = \{a, b, [u_j]_{1 \leq j \leq m}\}$.

- **KeyGen**: According to the system private key $msk$, the client and staff generate their public-private key pair separately. The system first computes $Y = e(g, g)^h, Y_1 = g^{a, Y}$. The client selects $\alpha, \beta, \gamma \in Z_p^*$, he computes $A = g^{\alpha}, B = g^{\beta}, Z = g^{\gamma}$ and gets their public and private key pair $PK_i = \{\alpha, \beta, \gamma\}$. The staff selects $t \in Z_p^*$ at random, he computes $\lambda = g^{\alpha t, r}$, for each attribute $j$, he randomly selects $t_j \in Z_p^*$, and computes $\delta_j = g^{H_2(t)^h}, \xi_j = g^{b, \delta_j}$. He first selects a unique secret $r$, then it also selects $\sigma \in Z_p^*$, and computes $D = g^{\beta + \alpha r}, D_1 = g^{\alpha, h^n}, D_2 = g^{\beta, D_1} = g^{\beta, \sigma \lambda t}$. Finally, he gets the private key pair $SK_i = \{\lambda, [\delta_j, \xi_j]_{1 \leq j \leq m}\}, SK_{h} = \{D, D_1, D_2, [D_j]_{1 \leq j \leq m}\}$, where $SK_h$ is the search private key and $SK_i$ is the secret private key.

- **Encrypt**: For insurance documents $M$, the staff first signs the file $M$ as: $\text{sig}_M = (H(id_M) \cdot g^{H(M)})^d$, where $id_M$ is the identifier of $M$, and then selects a random key $ck$, and he encrypts $M$ with the symmetric encryption algorithm to get $E_{ck}(M, \text{sig}_M)$. The staff defines an access tree $T$ and sends it to the fog node, and then the staff and the fog node run the Encrypt algorithm to encrypt $ck$. The Encrypt algorithm includes two sub-algorithms, namely DO-Encrypt and fog-Encrypt.

- **fog-Encrypt**: For each node $x$ in the access tree $T$, the fog node first selects a polynomial $q_x$, which starts with the root node $R$ and is selected in a top-down manner. For each node $x$ in $T$, its threshold value $k_x$ needs to have 1 more order than the degree $d_x$ of $q_x$, that is $k_x = d_x + 1$.

Starting from the root node $R$, the fog node is randomly selected $s_1 \in Z_p^*$ and set $q\theta(0) = s_1$, and other points in the random selection are used to completely define the polynomial $q_R$. For each node $x$, it sets $q(x) = q_{\text{parent}(x)}(\text{index}(x))$, and randomly chooses other points of $d_x$ and $q_x$ to fully define it. In the tree $T$, let $X$ be the set of attributes related to leaf node. Fog node computes $C_1 = g^{s_1}, C_2 = h^{s_1}, C_j = g^{q_x(\text{index}(x))}, \forall j = \text{attr}(x) \in X$.

Finally, the fog node sends the ciphertext $CT_1 = \{T, C_1, C_2, [C_j]_{\forall j = \text{attr}(x) \in X}\}$ to the staff.

- **DO-Encrypt**: The staff randomly selects $s_2 \in Z_p^*$ and computes $C = ck \cdot e(g, g)^{\beta s_2}, C_1 = g^{s_1}, C_1' = C_1 \cdot g^{s_2}, C_2' = C_2 \cdot h^{s_2}$, and he gets the ciphertext set $CT = \{T, CT(x), C_1, C_2', [C_j]_{\forall j = \text{attr}(x) \in X}\}$. The staff first stores the ciphertext $CT$ on the IPFS, and gets a hash address $H$ returns by the IPFS. Second, the staff uses the SHA256 algorithm to hash the file $M$ and gets $h' = \text{SHA256}(M)$, then he broadcasts $H, h'$ to the blockchain and gets the block ID, and sends the block ID to the client. The client verifies the hash of $M$ according to the block ID. If the verification is passed, the client issues a confirmation transaction. At this time, the data storage phase is completed.

- **Index**: The staff first picks the keyword set $w = \{w_1, w_2, \cdots, w_k\}$ out from the records, and he randomly selects $r_1, r_2 \in Z_p^*$, when $l = 1$, it represents the keyword of length $j$ in $w$, otherwise, it means that the file does not include the keyword. Then he sets $\phi_j = A^{r_1 H_2(w)}$, where $i \in \{1, \cdots, m\}, E_0 = A^2, E_1 = B^2, E_2 = Z^t, \phi_y = g^{\lambda t, 0}, \delta' = H_1(\text{attr}(y)\phi_y(0))$, finally, he gets the index $I = \{\phi_i, [\phi_y, \delta']_{1 \leq y \leq t}, E_0, E_1, E_2\}$.

- **Trapdoor**: When applying for a claim, in order to search the keyword set $w' = \{w_1', w_2', \cdots, w_k'\}$, the client first selects a $r \in Z_p^*$ at random, and he computes $T_1 = \prod_{i=1}^{k} g^{\tau a_i(w_i)}, T_2 = g^{\tau Y}, T_3 = \lambda^t$, and then he computes $T_1' = \delta_j = \delta_j' = \xi_j^t = \xi_j^r$, where $j \in S$. Finally, he gets the trapdoor $\text{Trap} = \{S, T_1, T_2, T_3, [\delta_j', \xi_j^t]_{j \in S}\}$ and sends the Trap to the staff of the insurance company.

- **search**: The staff verifies that the user satisfies the access structure.

1) If $x$ is a leaf node, he sets $j = \text{attr}(x)$, and if $j \in S$, he computes $D_x = e(\delta_j, \phi_y) = e(g, g)^{r a_j(0)}$, otherwise, return $D_x = \bot$. 

---

**J. Sun et al.: Non-Repudiation Storage and Access Control Scheme of Insurance Data Based on Blockchain in IPFS**

VOLUME 8, 2020
2) If \( x \) is a non-leaf node, for all the child nodes \( x' \) of \( x \), let \( \omega_x \) be a set of any size of the child node, so that \( D_x \parallel \omega_x \). If there is no such set, then let \( D_x = \perp \), otherwise he sets \( D_x = \prod_{x' \in \omega_x} D_{x_i}^\Delta \omega_{x_i}(0) = e(g, g)^{r_{q_i}(0)} \) where \( i = \text{index}(x') \). \( \omega_x = \{ \text{index}(x')(x' \in \omega_x) \} \). 

3) The staff verification 1) if it is true, then the relevant Hash value is found to the client, otherwise, it returns \( \perp \).

- **Decrypt**: If the client’s attributes satisfy the access policy \( W \), the client can decrypt the file using the Decrypt algorithm. The Decrypt algorithm consists of two parts: fog-Decrypt and user-Decrypt.

1) If \( x \) is a leaf node, he sets \( j = \text{attr}(x) \), and if \( j \in S \), he computes \( e(D_j, C_j) = e(g^{\Delta \omega_x(x')}, g^{\lambda(0)}) = e(g, g)^{r_{q_i}(0)} \), and if \( j \in X \), it returns \( \perp \).

2) If \( x \) is a non-leaf node, the recursive algorithm is defined as: for all the child nodes \( x' \) of \( x \), let \( \omega_x \) be a set of any size of the child node, it returns \( F_x = \perp \), otherwise, \( F_x \) can be computed as \( F_x = \prod_{x' \in \omega_x} F_{x'}^{\Delta \omega_{x'}(0)} \): 

\[
F_x = \prod_{x' \in \omega_x} F_{x'}^{\Delta \omega_{x'}(0)} \\
= \prod_{x' \in \omega_x} (e(g, g)^{r_{q_i}(0)} \Delta \omega_{x'}(0)) \\
= \prod_{x' \in \omega_x} (e(g, g)^{r_{q_i}(0)} \Delta \omega_{x'}(0)) \\
= \prod_{x' \in \omega_x} (e(g, g)^{r_{q_i}(0)} \Delta \omega_{x'}(0)) \\
= (e(g, g)^{r_{q_i}(0)} \Delta \omega_{x'}(0)) \\
= (e(g, g)^{r_{q_i}(0)} \Delta \omega_{x'}(0))
\]

where \( i = \text{index}(x'), \omega_x = \{ \text{index}(x')(x' \in \omega_x) \} \), then the Decrypt algorithm can be defined by calling the root node \( R \) in the \( W \). If \( S \) satisfies \( W \), the fog node can get: \( F_R = e(g, g)^{r_{q_i}(0)} \).

Fog node can compute \( k = e(D_1, C_1) + e(D_2, C_2) = e(g^{\lambda(0)}, g^{\lambda(0)}) = e(g, g)^{r_{q_i}(0)} \). All fog node in the fog node sends \( G = \{ e(C_i(M, \text{sig}(M)), C', C, H) \} \) to the client. Upon receive \( G \) from the fog node, the client can run the following sub-algorithms to obtain the symmetric key \( c_k \).

- **Update**: The client sends an update request to the staff, and the staff first generates a global key \( GK = \{ K_x = g^{q_i(0)} \} \), \( K_x = H_1(\text{attr}(x))^{q_y(0)} \), where \( x \in X \), and \( X \) is the set of leaf nodes in the tree \( T \). For each attribute \( j \in X \), the client randomly selects \( t_j \in \mathbb{Z}_p^* \) and computes:

\[
1): \text{If } j \in S \cap X, \text{then he computes } S_j = (D_j \cdot H_1(j)^{q_y})^r = g^{\frac{r}{\tau}} \cdot H_1(j)^{q_y} \\
2): \text{If } j \in X \cap (S \cap X), \text{then he computes } S_j = (H_1(j)^{q_y})^{r_q} = H_1(j)^{q_y} \cdot S_j = (g^{q_y})^{r_q} = g^{r_q}.
\]

Finally the fog node sends the attribute to the fog node and sends it to the fog node to verify.

1) If \( x \) is the leaf node, then the fog node sets \( j = \text{attr}(x) \), and if \( j \in S \cap X \), then:

\[
\text{verify } = e(S_{x}, k_x) = e(S_{x}, k_x) \\
= e(g, g)^{r_{q_i}(0)} \cdot H_1(\text{attr}(x))^{q_y(0)} \\
= e(g, g)^{r_{q_i}(0)} \cdot H_1(\text{attr}(x))^{q_y(0)} \\
= e(g, g)^{r_{q_i}(0)} \cdot H_1(\text{attr}(x))^{q_y(0)} \\
= e(g, g)^{r_{q_i}(0)}.
\]

2) If \( x \) is non-leaf node, for all child nodes \( x' \) of the node \( x \), the fog node lets \( \omega_x \) be a set of any size for this child node \( x' \), such that \( S_x \parallel \perp \), if there is no such set, it returns \( \perp \), otherwise, lets:

\[
I_x = \prod_{x' \in S_x} F_{x'}^{\Delta \omega_{x'}(0)} \\
= \prod_{x' \in S_x} (e(g, g)^{r_{q_i}(0)} \Delta \omega_{x'}(0)) \\
= \prod_{x' \in S_x} (e(g, g)^{r_{q_i}(0)} \Delta \omega_{x'}(0)) \\
= (e(g, g)^{r_{q_i}(0)} \Delta \omega_{x'}(0))
\]

If the attribute \( S \) satisfies the update policy \( W \), we set the overall evaluation result of the update tree to \( I \) = \( e(g, g)^{r_{q_i}(0)} = e(g, g)^{r_{q_i}(0)} \). Finally, the staff will check whether the equation holds: \( e(g, g)^{r_{q_i}(0)} = e(g, g)^{r_{q_i}(0)} \). If this equation holds, then it means that the staff receives the signature, allowing the user to update the ciphertext.

**VI. SECURITY PROOF**

**A. SECURITY OF THE ENCRYPT PROCESS**

**Theorem 1**: If there is a PPT adversary A who can win the proposed solution with a non-negligible advantage \( e(\varepsilon > 0) \), then there is a PPT simulator B that can distinguish DBDH tuples from random tuples with advantages \( \frac{e(\varepsilon)}{2} \).
Proof: Let $G_1, G_2$ be bilinear group of the order $p$, $g$ is the generator of $G_1$, and $e : G_1 \times G_1 \rightarrow G_2$ is an efficient and computable bilinear map. Firstly, the challenger randomly selects $\alpha, \beta, \gamma, \in Z_p^*$, and $b \in \{0, 1\}$, as well as a random element $T \in G_2$. We let $T = e(g, g)^{\beta p}$, and if $b = 0$, then $T$ is a random element, otherwise, the challenger sends $\{g, A, B, S, T\} = \{g, g^\alpha, g^\beta, g^\gamma, T\}$ to the simulator $B$. Now, the simulator $B$ plays the challenger in the following games.

Init: The adversary $A$ specifies a challenge access policy $W^*$ and sends it to the simulator $B$.

Setup: In order to provide the adversary a public key $mpk$, the simulator $B$ selects $b' \in Z_p^*$ at random, and computes $b = b' + \alpha \beta$, then he sets $\mu = e(g, g)^\beta = e(g, g)^{\beta} \cdot e(g, g)^{\mu}, \nu = g^\mu = g^\alpha = A, h = g^\beta$. For each attribute $j \in S$, the simulator $B$ randomly chooses $t_j$, if $j \in W^*$, he computes $pk_j = g^\alpha t_j^{-1} = g^\beta$, otherwise, he computes $mpk_j = g^\mu t_j = g^\nu$ where $t_j = v_j$ where $\theta$. Finally, the simulator $B$ sends the public key $\{G_1, G_2, p, g, H_1, H_2, e, h, \mu\}$ to the adversary $A$.

Phase1: At this stage, the simulator $B$ responds to the adversary $A$‘s attribute key, and the adversary $A$ can selectively submit any attribute set $L \in S$ to the simulator $B$. First, the simulator $B$ randomly selects $\eta_j \in Z_p^*$ and computes $\eta = \eta_j - \beta$, then he sets $D = g^{b' \cdot g^{\eta_j}} = g^{(b' + \alpha \beta)} \cdot e(g, g)^{\eta_j} = g^{\beta + \alpha \beta} = g^\beta, A'' = g^{\eta_j} h^\beta = A'' h^\beta, D_1 = g^{\eta_j} h^\beta = A'' h^\beta, D_2 = g^{\eta_j} A''$. For each attribute $j \in S$, if $j \in W^*$, the simulator $B$ computes $D_j = g^{\eta_j} h^\beta = \eta_j g^\beta = A'' h^\beta, \eta_j = \eta_j$ where $\eta_j \in S^j$ otherwise computes $D_j = g^{\beta + \eta_j} h^\beta = g^\beta h^\beta\eta_j = A'' h^\beta$. Finally, the simulator $B$ sends the key to the adversary $A$, if the attribute $j$ needs to be updated into the attribute list, and the new attribute set does not satisfy $W^*$, then the simulator $B$ lets the update key is $K_j \rightarrow m = v_j / v_m$.

Challenge: The adversary $A$ submits two message $M_0, M_1$, after receiving the message, the simulator $B$ sends $W^*$ to the fog node first, then the fog node randomly selects $u \in Z_p^*$ for all related attribute $j$, then fog node applies the linear secret sharing scheme to construct the shares $\omega_j$ of $u_j$. Then, for each attribute $j$, the fog node generates the challenge ciphertext by computing $g^{\alpha U}, h^{\alpha U}$ and $C_j = g^{\alpha \omega_j}$, and sends the challenge ciphertext $CT^* = \{g^{\alpha U}, h^{\alpha U}, \{C_j|\forall j \in W^*\}\}$ to the simulator $B$. Then the simulator $B$ randomly selects $u \in Z_p^*$ and generates the part ciphertext $CT^*_2$ as: $C' = m_0 \cdot e(g, g)^{\mu} = m_0 \cdot e(g, g)^{\mu} \cdot e(g, g)^{\mu} = m_0 \cdot Z \cdot e(g, g)^{\mu}, C = g^U, C_j = g^{\alpha \omega_j} = g^{U \cdot \alpha \omega_j}$ and $C_h = h^{\alpha \omega_j} = g^{\alpha \omega_j} h^{\alpha \omega_j} = A'' h^{\alpha \omega_j}$, finally the simulator $B$ returns the challenge ciphertext $CT = \{W^*, C', C_1, C_2, C_j|\forall j \in W^*\}$ to the adversary $A$.

Phase2: Same as Phase 1.

Guess: The adversary $A$ outputs a guess $\theta'$, if $\theta' = \theta$, the simulator $B$ outputs 0, it means that $T = R$; Otherwise, the simulator $B$ outputs 1 and guess that $T = e(g, g)^{\theta p}$. If $T = e(g, g)^{\theta p}$, then $CT^*$ is a valid ciphertext. In this case, the advantage of the adversary $A$ is $\delta_2 = \text{Pr} [B(g, g^\alpha, g^\beta, g^\gamma, T = e(g, g)^{\theta p}) = 0] = \frac{1}{2} + \epsilon$. If $T = R$, the perspective of the adversary $A, CT^*$ is completely random, so $\text{Pr} [B(g, g^\alpha, g^\beta, g^\gamma, T = R)] = \frac{1}{2}$.

B. SECURITY OF THE SEARCH PROCESS

Theorem 2: In view of the general bilinear group model, our scheme is selective secure against the selective keyword attacks. Where the hash function $H_1$ is a random oracle model, and $H_2$ is a one-way function.

Proof: We can prove that our scheme is secure against selective choose keyword attacks (SCKA) under a random oracle model. In the SCKA game, the adversary $A$ tries to distinguish $A'_{H_2(0)}$ from $A'_{H_2(1)}$. For an arbitrary element $s \in Z_p^*$, the probability of distinguishing $A'_{H_2(0)}$ and $g^s$ is the same as the probability of distinguishing $A'_{H_2(0)}$ and $A'_{H_2(1)}$, where $A = g^s$. The SCKA games as follows:

Setup: The challenger first selects $a, b, c \in Z_p^*$ at random, and generates the public parameter $(e, g, g^a, g^b, g^c)$ for the adversary $A$. Then the adversary $A$ selects an access policy $W^*$ and sends it to the challenger. $H(j)$ can be simulation as: if the attribute $j$ has been asked before, the challenger randomly selects $t_j$ and adds $(j, t_j)$ to the $OH_j$, then it returns $g^j$, otherwise, the challenger retrieve $t_j$ from and returns $g^j$.

Phase1: The adversary $A$ query $O_{keyGen}$ and $O_{Trap}$ as follow:

- $O_{keyGen}(S, msrk, W)$: The challenger firstly selects $t \in Z_p^*$ and computes $\lambda = g^{b' \cdot t}$, then, for each attribute $j$, the challenger randomly selects $t_j \in Z_p^*$ and computes $\delta_j = g^t H_1(j)^{y_j}$, $\xi_j = g^\beta$. Finally, the challenger returns the tuple $(S, \lambda, \{\delta_j, \xi_j|j \in S\})$.

- $O_{Trap}$: The challenger issues a oracle model $O_{keyGen}(sk_S, S)$ in order to get the attribute key $sk_S = \{S, \lambda, \{\delta_j, \xi_j|j \in S\}\}$. Then the challenger selects a random element $r \in Z_p^*$ and computes $T_1 = \prod_{i=1}^l g^\beta, T_2 = g^c, T_3 = \lambda^r$, if the attribute set $S$ satisfies the access policy, the challenger adds $w^*_{S}$ to the keyword set list $L_w$.

Challenge: If two keywords set $w_0$ and $w_1$ are given, and both $w_0$ and $w_1$ are in the not keyword set list $L_w$, the challenger randomly selects $s_1, s_2 \in Z_p^*$ and computes the secret of $s_2$ in the tree $T$. Then, the challenger outputs a random number $\theta' \in \{0, 1\}$. If $\theta' = 0$, he outputs $\phi_i = g^t_{H_2(s_i)}$ where $i \in \{1, \ldots, m\}, E_0 = A^a, E_1 = B^b, E_2 = Z^c, \phi_i = g_i(\theta), \delta_i = H_1(att(y))^{\mu_i(0)}$, otherwise, the challenger returns $\phi_i = A'_{H_2(\mu_i)}$ where $i \in \{1, \ldots, m\}, E_0 = A^a, E_1 = B^b, E_2 = Z^c, \delta_i = H_1(att(y))^{\mu_i(0)}$ where $att(y) = j \in S$.

Phase2: Same to the Phase1.

Theorem 3: Our scheme can ensure the privacy of keywords in a random oracle model, and $H_2$ is a one-way hash function.

Proof: We first construct a challenger to execute the following keyword privacy game.

Setup: The challenger first selects $a, b, c \in Z_p^*, h \in G_1$ and then selects a hash function $H_2 : \{0, 1\}^* \rightarrow Z_p^*$, finally, the challenger gets the public key $PK = (e, g, g^a, g^b, g^c, h)$ and private key $SK = \{a, b, c\}$, the challenger simulations the random oracle $O_{H_2(i)}$ as follow: If the attribute $j$ has not been queried before, the challenger selects $t_j \in Z_p^*$ at random,
TABLE 3. storage cost.

| Algorithm | VABKS | VKSE | EVPKE | Ours |
|-----------|-------|------|-------|------|
| Setup\Inst | \(|G| + |Z_p|\) | \(|G| + |G_0| + |Z_p|\) | \(6|G| + 6|Z_p|\) | \(2|G_1| + (2 + |S|) |Z_p|\) |
| KeyGen | \((2|S| + 1)|G|\) | \((2|S| + 3)|G| + |Z_p|\) | \((1 + |S|)|G|\) | \(2|G_2| + (9 + 3|S|)|Z_p| + 3|Z_p|\) |
| Trapdoor | \((2|S| + 3)|G|\) | \((2|S| + 3)|G|\) | \((1 + |S|)|G| + 2|S| |Z_p|\) | \((3 + 2|S|)|G_1|\) |
| Search | \((2|S| + 1)|G|\) | \((3|S| + 1)|G_0|\) | \((1 + |S|)|G|\) | \(3|G_1|\) |

TABLE 4. computational cost.

| Algorithm | VABKS | VKSE | EVPKE | Ours |
|-----------|-------|------|-------|------|
| Setup \ Inst | \(3O_E\) | 0 | \(5O_E + O_p + O_{E_0}\) | 0 |
| KeyGen | \((2 + 2|S|)O_E\) | \((2|S| + 4)O_E\) | \((2 + |S|)O_E\) | \((O_p + (10 + 3|S|)O_E)\) |
| Trapdoor | \((4 + 2|S|)O_E\) | \((2|S| + 4)O_E\) | \((1 + 3|S|)O_E\) | \((3 + 2|S|)O_E\) |
| Search | \((1 + 2|S|)O_E + 4O_p\) | \((2|S| + 1)O_p + |S|O_{E_0}\) | \((1 + 3|S|)O_E + 4O_p\) | \(O_p + O_E\) |

FIGURE 2. Comparison of storage cost and computation cost of KeyGen algorithm, Trapdoor algorithm and Search algorithm of the four schemes.

Phase 1: The adversary A publishes two random prediction models, as follows:

---

- \(O_{keyGen}\): The challenger runs the KeyGen algorithm and gets \(SK\), then he sends \(SK\) to the adversary A, finally, the challenger adds \(T\) to the list \(L_{keyGen}\).

- \(O_{trapdoor}\): The challenger runs the random oracle \(O_{keyGen}(W)\) and gets the attribute key \(SK = \{T, \{\varphi_y, \theta_y\} \mid y \in Ln(T)\}\), then he calls the Trap algorithm and gets \(W\), finally, he sends \(W\) to the adversary A.

Challenge: The adversary A first selects an attribute set \(S'\) and the challenger specifies a access policy \(W'\), then he applies the KeyGen algorithm to compute \(SK'\). Given the attribute set \(S'\) and private key \(SK'\), the adversary A selects a keyword set \(w'\) and computes \(C_1\) and \(W\), where \(S'\) needs satisfy the requests which specified in the keyword security game.

Phase 2: Same as Phase 1.

Guess: The adversary A first outputs a keyword \(w'\) and sends it to the challenge, the challenge applies the Encrypt algorithm to compute \(C_2\), if \((W', I') = 1\), then the adversary A wins the game.

VII. PERFORMANCE ANALYSIS

We analyze the performance of our scheme from three aspects, namely storage cost, computation cost and experimental simulation, separately.

We first analyze the storage cost of our scheme by comparing with the VABKS scheme[11], VKSE scheme [22], and EVPKE scheme[23]. Before comparing, we first define some symbols. \(|G|\) represents the element of bit length in \(G\); \(G_i\) represents the element of bit length in \(G_i\); \(|Z_p|\) represents the element of bit length in \(Z_p\); \(S\) is the user’s attribute set. In Table 3, we give the storage costs of the four schemes.

As shown in Table 3, we can see that except for the KeyGen algorithm, our storage cost is higher than the VABKS scheme, VKSE scheme, and EVPKE scheme. In other algorithms,
the storage cost of our scheme is lower than the other three schemes. This is because that we outsourced the insurance records to fog nodes for partial encryption and decryption operations, and this storage cost is still within acceptable limits.

Next, we will analyze the computational cost of our program by comparing with the VABKS scheme, VKSE scheme, and EVPKE scheme. We first need to define some time-consuming operations, as follows: \( O_p \) is the bilinear pairing operation, \( O_e \) is the exponential operation, In Table 4, we give the computation cost of the four schemes.

As shown in Table 4, we can see that except for the KeyGen algorithm, our computational cost is higher than the VABKS scheme, VKSE scheme, and EVPKE scheme. In other algorithms, the computational cost of our scheme is lower than the other three schemes. This is because we not only generate the client’s key in the KeyGen algorithm, and a public-private key pair is generated for the fog node, which reduces the computational cost of client and staff when encrypting and decrypting files, and improves the staff efficiency.

Finally, we analyze the actual performance of our scheme through experimental simulations. We leveraged a real data set to perform an experimental simulation on an Ubuntu server 15.4 with a processing core i5 processor, which was performed in a pairing-based cryptography (PBC) laboratory. As we all know, in this experiment, \(|Z_p| = 160 \text{ bit}, |G| = 1024 \text{ bit}\). Since all four schemes are affected by attributes, we set \( S \) as the impact factor and let \( S \in [10, 50] \).

Since we outsource a large amount of encryption and decryption computations to the fog node, the storage cost and computation cost of our scheme in the KeyGen algorithm are higher than the VABKS scheme, VKSE scheme, and EVPKE scheme, but it is still within the acceptable range, as shown in Figure 2 (a) and (d). In the trapdoor algorithm, our scheme has the same performance as the VABKS and VAKE schemes, as shown in Figure 2 (b) and (e). However, our scheme has obvious advantages in the search algorithm, as shown in Figure 2 (c) and (f).

VIII. CONCLUSION

In the insurance system, it is a very important issue to ensure the fairness and justice of the insurance business. Therefore, we build an security storage and update scheme for insurance records under IPFS storage environment in the insurance system by combining the blockchain technology and the ciphertext policy attribute-based encryption system. The scheme realizes the non-repudiation for both insurance companies and customers, and uses distributed IPFS as a storage platform to ensure the security storage of insurance records. In addition, the stored insurance records are easy to retrieve. However, our scheme is still not perfect. The storage of expired user attributes and their keys requires a certain amount of storage space. Therefore, in future work, we will consider leveraging smart contracts to deploy our scheme on the blockchain for implementation.

REFERENCES

[1] S. Mhatre and A. V. Nimkar, “Secure cloud-based federation for EHR using multi-authority ABE,” in Advanced Computing and Intelligent Engineering, Berlin, Germany: Springer, 2019, pp. 3–15.

[2] H. Li and T. Jing, “A lightweight fine-grained searchable encryption scheme in fog-based healthcare IoT networks,” Wireless Commun. Mobile Comput., vol. 2019, May 2019, Art. no. 1019767.

[3] P. Zhang, Z. Chen, J. K. Liu, K. Liang, and H. Liu, “An efficient access control scheme with outsourcing capability and attribute update for fog computing,” Future Gener. Comput. Syst., vol. 78, pp. 753–762, Jan. 2018.

[4] H. Huang, X. Chen, Q. Wu, X. Huang, and J. Shen, “Bitcoin-based fair payments for outsourcing computations of fog devices,” Future Gener. Comput. Syst., vol. 78, pp. 850–858, Jan. 2018.

[5] M. Green, S. Hohenberger, and B. Waters, “Outsourcing the decryption of ABE ciphertext,” in Proc. 20th USENIX Conf. Secur., 2011, p. 3.

[6] M. Asim, M. Petkovi and T. Ignatenko, “Attribute-based encryption with encryption and decryption outsourcing,” in Proc. 20th USENIX Conf. Secur., 2011, pp. 1–4.

[7] C. Zuo, J. Shao, and G. Wei, “CCA-secure ABE with outsourced decryption for fog computing,” Future Gener. Comput. Syst., vol. 78, pp. 720–738, Oct. 2018.

[8] Y. Zhang, D. Zheng, X. Chen, J. Li, and H. Li, “Efficient attribute-based data sharing in mobile clouds,” Pervas. Mobile Comput., vol. 28, pp. 135–149, Jun. 2016.

[9] Y. Zhang, X. Chen, J. Li, D. S. Wong, H. Li, and I. You, “Ensuring attribute privacy protection and fast decryption for outsourced data security in mobile cloud computing,” Inf. Sci., vol. 379, pp. 42–61, Feb. 2017.

[10] Q. Zheng, S. Xu, and G. Ateniese, “VABKS: Verifiable attribute-based keyword search over outsourced encrypted data,” in Proc. IEEE Conf. Comput. Commun., Apr. 2014, pp. 522–530, doi: 10.1109/INFOCOM.2014.6847976.

[11] F. Zhao, T. Nishide, and K. Sakurai, “Realizing fine-grained and flexible access control to outsourced data with attribute-based cryptosystems,” Inf. Secur. Pract. Exper., vol. 15, pp. 83–97, Oct. 2011, doi: 10.1007/978-3-642-21037-1.

[12] X. Chen, J. Li, and W. Susilo, “Efficient fair conditional payments for outsourcing computations,” IEEE Trans. Inf. Forensics Security, vol. 7, no. 6, pp. 1687–1694, Dec. 2012.

[13] 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.

[14] L. Zhang, “Attribute-based encryption with privacy protection and accountability for IoT,” IEEE Trans. Cloud Comput., early access, Feb. 19, 2020, doi: 10.1109/TCC.2020.2975184.

[15] J. Li, Y. Wang, Y. Zhang, and J. Han, “Full verifiability for outsourced decryption in attribute based encryption,” IEEE Trans. Services Comput., vol. 13, no. 3, pp. 478–487, May 2020, doi: 10.1109/TSC.2017.2710190.

[16] 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. 2017, doi: 10.1109/TSC.2016.2542813.

[17] Y. Miao, x. Liu, and. “Privacy-preserving attribute-based keyword search in shared multi-owner setting,” IEEE Trans. Dependable Secure Comput., early access, Feb. 5, 2019, doi: 10.1109/TDSC.2019.2897675.

[18] R. Su, Z. Zhu, L. Sun, and N. Pan, “Practical searchable CP-ABE in cloud storage,” in Proc. 2nd IEEE Int. Conf. Comput. Commun. (ICCC), Chengdu, China, Oct. 2016, pp. 180–185.

[19] S. E. Wang and B. G. Lin, “A scheme of attribute-based encryption access policy used in mobile cloud storage for personal health records,” in Proc. Int. Conf. Inf. Netw. Secur., Beijing, China, 2014, pp. 59–68.

[20] J. Bethencourt, A. Sahai, and B. Waters, “Ciphertext-policy attribute-based encryption,” Proc. IEEE Symp. Secur. Privacy, Berkeley, CA, USA, May 2007, pp. 321–334.

[21] L. Cheung and C. Newport, “Provably secure ciphertext policy ABE,” in Proc. 14th ACM Conf. Comput. Commun. Secur., 2007, pp. 456–465.

[22] Y. Miao, “Verifiable keyword search over encrypted cloud data in smart city,” Comput. Elect. Eng., vol. 65, pp. 90–101, Jan. 2018.

[23] P. Liu, J. Wang, H. Ma, and H. Nie, “Efficient verifiable public key encryption with keyword search based on KP-ABE,” in Proc. 5th Int. Conf. Broadband Wireless Comput. Commun. Appl., Nov. 2014, pp. 584–589, doi: 10.1109/BWCCA.2014.119.
JIN SUN received the B.S. degree from Shaanxi Normal University, in 2000, the M.S. degree from the Xi’an University of Technology, in 2005, and the Ph.D. degree from Xidian University, in 2012. She is currently an Associate Professor with the Xi’an University of Technology. Her main research include cryptography, information security, network security, and blockchain.

XIAOMIN YAO received the B.Sc. degree from Tianshui Normal University, in 2018. She is currently pursuing the master’s degree with the Xi’an University of Technology. She mainly studies cryptography, data security, and blockchain.

SHANGPING WANG received the B.S. degree from the Xi’an University of Technology, in 1982, the M.S. degree from Xi’an Jiaotong University, in 1989, and the Ph.D. degree from Xidian University, in 2003. He is currently a Professor with the Xi’an University of Technology. His main research areas include cryptography, information security, blockchain, and the Internet of Things.

YING WU received the B.Sc. degree from Qufu Normal University, in 2019. She is currently pursuing the master’s degree with the Xi’an University of Technology. She mainly studies cryptography, data security, and blockchain.