Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH

Open Access

Access control scheme based on blockchain and attribute-based searchable encryption in cloud environment

Liang Yan[1,2], Lina Ge[1,2,3*], Zhe Wang[1,2,3], Guifen Zhang[1,2], Jingya Xu[2,4] and Zheng Hu[1,2]

Abstract

With the rapid development of cloud computing technology, how to achieve secure access to cloud data has become a current research hotspot. Attribute-based encryption technology provides the feasibility to achieve the above goal. However, most of the existing solutions have high computational and trust costs. Furthermore, the fairness of access authorization and the security of data search can be difficult to guarantee. To address these issues, we propose a novel access control scheme based on blockchain and attribute-based searchable encryption in cloud environment. The proposed scheme achieves fine-grained access control with low computation consumption by implementing proxy encryption and decryption, while supporting policy hiding and attribute revocation. The encrypted file is stored in the IPFS and the metadata ciphertext is stored on the blockchain, which ensures data integrity and confidentiality. Simultaneously, the scheme enables the secure search of ciphertext keyword in an open and transparent blockchain environment. Additionally, an audit contract is designed to constrain user access behavior to dynamically manage access authorization. Security analysis proves that our scheme is resistant to chosen-plaintext attacks and keyword-guessing attacks. Theoretical analysis and experimental results show that our scheme has high computational and storage efficiency, which is more advantageous than other schemes.

Keywords Access control, Attribute-based encryption, Blockchain, Secure search, Attribute revocation

Introduction

With the connection of the global mobile Internet and the rapid development of cloud computing, more and more communication academia and industry are committed to shaping a safe and effective resource sharing method in the cloud environment [1]. Cloud storage technology has been widely used due to its high performance and low cost. To ensure the security of private data, data is usually stored in cloud services in encrypted form. However, the traditional public key encryption technology has been unable to meet the current needs of cloud data privacy protection. In this context, how to achieve access authorization and accurate retrieval of encrypted cloud data has become a new challenge.

Access control (AC) is a key technology to maintain data security and privacy [2]. The AC provides a solution to the above problem by constraining user access rights to ensure legitimate access to sensitive data. Attribute-based searchable encryption based on ciphertext policy not only enables fine-grained access control of encrypted data, but also supports users to retrieve ciphertext based on keywords. Ciphertext Policy Attribute-Based Encryption Algorithm (CP-ABE) [3, 4] allows data owners to

*Correspondence:
Lina Ge
66436539@qq.com
[1] School of Artificial Intelligence, Guangxi Minzu University, 530006 Nanning, China
[2] Key Laboratory of Network Communication Engineering, Guangxi Minzu University, 530006 Nanning, China
[3] Guangxi Key Laboratory of Hybrid Computation and IC Design Analysis, Guangxi Minzu University, 530006 Nanning, China
[4] School of Electronic Information, Guangxi Minzu University, 530006 Nanning, China

autonomously set data access policies according to a set of attributes, and associate data access policies with ciphertexts. When the user's attribute set satisfies the access policy, the ciphertext can be decrypted using the corresponding attribute private key, while the specific identity of the decryptor remains unknown, which is suitable for "one-to-many" access scenarios. In recent years, a large number of studies have applied attribute-based encryption technology to cloud data access control to improve the privacy and security of cloud data [5, 6]. However, the traditional CP-ABE algorithm consumes a lot of computational cost and the security of the access policy is often ignored because the access policy is embedded in the ciphertext. In addition, attribute access expiration and permission change are also urgent issues to be addressed.

Currently, most access control schemes typically use a centralized management model, which makes them susceptible to system-wide failure in the event of a single malfunction. Furthermore, traditional solutions rely on trusted third parties for access decisions, which not only incur high trust overhead but also unfair service fee payments. Therefore, designing secure and fair searchable access control schemes remains a pressing challenge. Blockchain is a distributed ledger technology characterized by decentralization, openness, transparency, tamper resistance, and traceability [7]. It supports the secure storage and transaction of data without the involvement of third parties, and users no longer have to worry about the high trust and security risks posed by third parties. This means that blockchain technology can be used to replace traditional third parties for access authorization management, enabling a fair and trusted distributed access control framework.

Based on the analysis of the above problems, we combine blockchain technology with attribute-based searchable encryption technology to propose a novel distributed data-sharing scheme. This scheme focuses on achieving fine-grained searchable access to encrypted cloud data while taking into account low computational cost, policy privacy, attribute revocation and dynamic authorization.

The main contributions of this study are as follows:

(1) A distributed fine-grained access control scheme is developed by combining blockchain and attribute-based searchable encryption. The scheme stores the data ciphertext in the distributed IPFS (Inter Planetary File System), and facilitates the secure distribution of metadata ciphertext via blockchain smart contract, thus avoiding the high trust cost and low security of storage caused by third-party intervention in traditional access systems.
(2) An improved attribute-based searchable encryption algorithm based on policy hiding is designed to prevent the leakage of user privacy attributes. Proxy encryption and decryption are introduced to reduce the user's computational consumption. Simultaneously, the algorithm supports attribute revocation and decryption verification.
(3) The proposed scheme realizes the secure search of encrypted keywords on the blockchain. At the same time, we design a smart contract with a search audit function to dynamically manage access rights based on user access behavior and accessibility period to prevent illegal access.
(4) Security analysis, performance comparison, and simulation experiments indicate that the proposed scheme is both feasible and advantageous.

Related work

Access control is an important security mechanism to protect sensitive information and system resources [8]. Blockchain as an underlying technological architecture that features trust, low cost, and high value. Many researchers have tried to combine access control and blockchain technology to make up for the drawbacks of traditional centralized access control. The existing blockchain access control schemes are mainly based on on-chain storage and execution of permission verification. Reference [9] proposes an access control scheme to facilitate the deletion, update, and access of lightweight IoT devices on the blockchain, however, this scheme is not suitable for large file sharing. Reference [10] employs blockchain technology to enable the secure sharing of vehicle information. Although the scheme in question calculates the reputation value of a vehicle based on the weight model to prevent unauthorized sharing, it does not account for privacy. Reference [11] proposes a blockchain data access control scheme based on digital certificates. This scheme designs an identity authentication protocol that does not require verification of third-party signatures, and proposes to use signature technology to protect the sensitive information of contracts and the user's identity information, but does not consider the secure storage of data. Focusing on a situation where the traditional IoT data sharing model is heavily reliant on a third party, reference [12] employs blockchain technology to design a trustless sharing model. However, this scheme does achieve fine-grained control using symmetric encryption. References [13–15] realize user attribute matching access control in a non-encrypted state, and deploy an access policy in the smart contract. Owing to the openness and transparency of the blockchain, a user

may send the access policy and attributes to a smart contract, thus compromising privacy.

Access control schemes featuring attribute-based encryption and cloud storage have been proposed to strengthen the security of data access and alleviate the storage bottleneck of blockchains. The CP-ABE algorithm enables superior data security and fine-grained access control for external storage. In the scheme proposed by [16–18], shared data are encrypted with CP-ABE and stored in the cloud service. The security of a cloud service directly affects the security of data, which creates a risk of user data loss. In [19], a distributed storage IPFS is used to implement a data-sharing scheme based on blockchain and secret sharing. This scheme stores plaintext data in the IPFS system. An obvious drawback is that when an address stored in the IPFS is leaked, any unauthorized user can obtain the corresponding plaintext data. References [20, 21] all use proxy computing to reduce the computational overhead of CP-ABE and ensure the traceability of the secret key. However, they do not consider the privacy of access policies. Since the access policy is embedded in ciphertext in plaintext form, it can be exploited by attackers to infer private attributes. Reference [22] proposes a smart grid data sharing scheme based on policy hiding to prevent the exposure of the access structure, and introduces proxy decryption to reduce the user decryption overhead. Reference [23] proposes an attribute-based encryption scheme for keyword searches based on policy hiding to prevent keyword-guessing attacks. Reference [24] proposes a cloud storage scheme with attribute policy hiding based on the "AND" gate access structure, which implements obfuscated attributes into the original access policy for user authentication. However, [22–24] are not blockchain-based, and ignore the security threats caused by third parties. Reference [25] proposes a policy-hiding blockchain access control scheme based on CP-ABE. The scheme uses polynomials to express the access structure. Data users can perform attribute policy matching locally and verify using homomorphic encryption. But this scheme needs to consume a lot of computing resources.

In order to realize the searchability of ciphertext, references [26, 27] propose an attribute-based encryption scheme based on keyword search. They store encrypted data on cloud services, and perform search verification on keywords on cloud services to ensure that the obtained ciphertext matches. In practical applications, attributes may be revoked due to permissions changes and other factors. To this end, references [28–30] propose data-sharing schemes that support attribute revocation. Reference[28] proposes a multi-authority searchable access scheme for cloud data. Although this scheme uses multiple authorizations to help keep user information confidential, it also easily leads to security bottlenecks. Reference [31] improves on the basis of reference [29] and implements policy hiding. However, in the above solution, the user's search behavior is uncontrollable, and there may be a phenomenon of unfair payment of search service fees. Reference [32] uses blockchain technology to solve the above problems, but this scheme does not support policy hiding.

Based on the above analysis, this study proposes a secure access control scheme based on blockchain and attribute-based searchable encryption in cloud environment. The scheme stores encrypted files with IPFS, which alleviates the storage pressure of blockchain and the single point of failure problem of traditional storage model. At the same time, the scheme supports attribute policy hiding and attribute revocation, which improves the security and flexibility of access. Proxy encryption and decryption are also introduced to reduce user computation consumption. In addition, the solution implements a secure search of encrypted keywords on the blockchain, while monitoring user access behavior and time limits through smart contracts to prevent unauthorized access.

Preliminaries

In this section, we describe the relevant background and basic information pertaining to this scheme. Table 1 introduces some of the notation used throughout this scheme.

Table 1 Notations in this paper

Symbol	Description
λ	safe parameter
PK	system public key
MK	system master key
SK	user private key
W	keyword
ck	symmetric key
S	user attribute set
U	system attribute set
CT	key ciphertext
CT_1	proxy encrypted ciphertext
CT_2	proxy decrypted ciphertext
SK'	intermediate key
P	access policy
Ω_i	hiding access policy
FID	file identification
HD	file storage address
I_w	keyword index
T_w	keyword trapdoor
$E(M)$	encrypted file

Blockchain

Blockchain was first used as the underlying supporting technology for Bitcoin. In essence, it is a distributed shared database that serves as a revolutionary low-cost credit technology solution, and is widely used in resource sharing, data traceability, and access control [33, 34]. Ethereum, a decentralized application platform based on the blockchain, provides an Ethereum virtual machine (EVM) [35] environment to support the operation of scripting languages. Users can write and deploy smart contracts and decentralized applications in Ethereum using programming languages such as Solidity and JavaScript to create more possibilities for blockchains [36, 37]. Smart contracts are codes written in accordance with transaction rules [38, 39]. Once generated, smart contracts cannot be tampered with, and run permanently on the blockchain. EVM creates a secure operating environment. An Ethereum account consists of a pair of public and private keys, and may be classified as an externally owned account or a contract account. An externally owned account is an account that is not associated with the contract, and is controlled by the user's account private key. A contract account is an account associated with the contract code generated upon deployment of a contract.

Smart contracts are deployed on the blockchain and users can interact with them by calling the corresponding address or interface in the contract [40]. Compared to that traditional contracts, the code of smart contracts features legal benefits and can execute automatically when the relevant conditions are met without interruptions in the execution process. Simultaneously, smart contracts can conduct secure transactions in a blockchain environment without a third-party arbiter. Before contract execution, all parties must submit the preset funds. Regardless of whether the contract is breached, it is implemented according to the mandatory automatic execution result.

Attribute-based searchable encryption

This section describes the basics of ciphertext policy attribute-based searchable encryption.

Bilinear mapping

Suppose p is a large prime number, G and G_T are two multiplicative cyclic groups of order p, and there is a bilinear mapping $e : G \times G \longrightarrow G_T$ that satisfies the following properties:

- Bilinear: For any $g_1, g_2 \in G$, and $u, v \in Z_p$, we have $e(g_1^u, g_2^v) = e(g_1^v, g_2^u) = e(g_1, g_2)^{uv}$.
- Non-degeneracy: There exists $g, h \in G$ such that $e(g, h) \neq 1$.
- Computability: For any $a, b \in G$, there is an efficient computation $e(a, b)$.

Access structure

This scheme uses the access control tree as the access structure. Assuming a system attribute set $U = \{att_1, att_2, att_3, att_4\}$ and access policy $P = ((att_1$ OR att_2) AND att_3 AND $att_4)$. Υ is defined as an access tree used to describe access policy P. The leaf nodes of Υ represent corresponding attributes, whereas the other nodes use the threshold to represent the "AND" and "OR" gates. Suppose that c represents the number of child nodes of node x. Let V represents the threshold of x, where $V \in [1, c]$. When $V = c$, the threshold gate is an "AND" gate, and when $V = 1$, the threshold gate is an "OR" gate. The value of each leaf node relates to user attributes and thresholds. If the user attribute set satisfies the access policy, then the secret value of the root node can be obtained by recursive calculation. In this case, the user attributes satisfy the access policy P in either of the following cases: (att_1 AND att_3 AND att_4) or (att_2 AND att_3 AND att_4).

DBDH assumption

Decisional Bilinear Diffie-Hellman (DBDH) assumption: Given a bilinear mapping $e : G \times G \longrightarrow G_T$ of order p, the challenger randomly selects $a, b, c \in Z_p$. Setting $T \in G_T$, with $(Q = (g, g^a, g^b, g^c), T)$ as the input, the attacker guesses. If it is judged that $T = e(g, g)^{abc}$, output 1. Otherwise, T is a random element in G_T, output 0. The advantage of defining an algorithm Γ to solve this problem is as follows: $\varepsilon = \left| Pr\left[\Gamma(Q, e(g, g)^{abc}) = 1\right] - Pr[\Gamma(Q, T \in G_T) = 1] \right|$.

Definition 1 If no attacker can solve the DBDH problem with a non-negligible advantage ε in polynomial time, the DBDH assumption holds.

DL problem

Discrete Logarithm (DL) problem: Let G be a multiplicative cyclic group of order p, and g be a generator of G. For $\forall f \in G$, there exists an exponent t such that $f = g^t (\mathrm{mod} p)$ holds. Then, it is difficult to solve the discrete logarithm t according to the values of f, g, and p.

IPFS

IPFS is a distributed storage system with decentralized cloud computing capabilities [41]. Compared with traditional centralized storage systems, IPFS does not have a

single point of failure, and the nodes do not trust each other, which provides higher security and access efficiency. With the development of cloud computing, it has become a trend for IPFS to replace traditional local storage technology [42, 43]. When a file is uploaded to the IPFS, it is split into multiple blocks for storage, and the system returns a unique hash value. The user does not need to know the storage path, and the unique hash value can determine data tampering. To download a file from the IPFS, the user can retrieve it through a hash address.

System model

This scheme uses blockchain technology, attribute-based searchable encryption, and IPFS to propose a fine-grained access control scheme with a hidden policy and distributed storage. Figure 1 illustrates the system model of the proposed scheme.

System composition

The primary interacting entities involve the data owner, data user, IPFS, blockchain, and proxy server.

Data Owner (**DO**): DO is the publisher of data. Its main responsibilities are to develop attribute sets, set private keys and access time periods for accessing users, and set access policies and keyword indexes for shared data. It also uploads the file cipher to IPFS and the metadata cipher to the smart contract.

Data User (**DU**): DU requests cryptographic metadata ciphertext and storage address from the smart contract according to the search token, and the address obtains the ciphertext file from IPFS. When the attribute set of DU satisfies the access policy, it can be accessed through the attribute private key to decrypt the ciphertext.

IPFS: IPFS is mainly responsible for the distributed storage of ciphertext data uploaded by DO and returning the ciphertext according to the hash storage address.

Blockchain (**BC**): The blockchain is responsible for the distribution of the user's private key and ciphertext, and for making appropriate audit decisions based on user access behavior. Simultaneously authenticate user attributes and perform keyword searches.

Proxy encryption server (**ES**): Mainly responsible for proxy encryption calculation.

Proxy decryption server (**DS**): Mainly responsible for proxy decryption calculation.

Algorithm description

The formal description of the proposed algorithm is as follows:

$Setup(1^{\lambda}) \longrightarrow (PK, MK)$: The system initialization algorithm is performed by DO. The algorithm takes a parameter λ as input, and outputs the system public key PK and the system master private key MK. Publish the PK and keep the MK secret.

$ProEnc(PK, P) \longrightarrow CT_1$: ES executes the proxy encryption algorithm, takes the system public key PK and the access policy P as input, and outputs the proxy encrypted ciphertext CT_1.

$Encrypt(PK, ck, W, CT_1) \longrightarrow (CT, \Omega_i, I_w)$: The encryption algorithm takes the public key PK, the symmetric key ck, the keyword W and the proxy encrypted ciphertext CT_1 as input, and outputs the key ciphertext CT, the hidden policy Ω_i and the keyword index I_w.

$KeyGen(PK, MK, S) \longrightarrow SK$: The attribute private key generation algorithm takes the public key PK, the

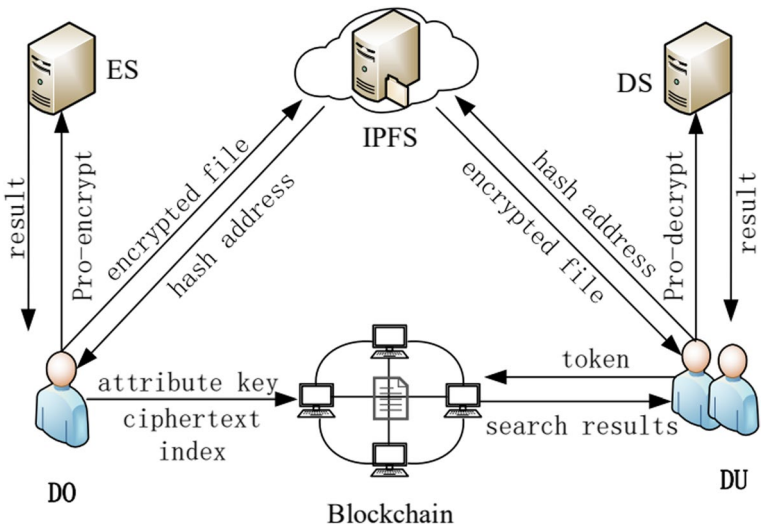

Fig. 1 System model

master private key MK and the user attribute set S as input, and outputs the attribute private key SK.

$Token(PK, SK, W') \longrightarrow (\Omega_i', T_w)$: DU executes the token generation algorithm. The algorithm takes the public key PK, the attribute private key SK, and the query key W' as input, and outputs the token $\{\Omega_i', T_w\}$.

$Search(I_w, T_w) \longrightarrow (0, 1)$: The search algorithm takes the keyword index I_w and the search trapdoor T_w as input. If the keyword match, the blockchain returns the ciphertext CT and the storage address HD to the user.

$ProDec(SK', CT) \longrightarrow CT_2$: The proxy decryption algorithm takes the intermediate key SK' and the ciphertext CT as input, and outputs the proxy decrypted ciphertext CT_2. Then, the DS returns CT_2 to the DU.

$Decrypt(d, CT_2) \longrightarrow ck'$: The user takes the secret parameter d and the proxy decrypted ciphertext CT_2 as input, and outputs the symmetric key ck'.

$Revocation(\overline{S}) \longrightarrow (\overline{AK_i}, \overline{S_i}, \overline{C_y})$: The attribute revocation algorithm takes a revocation attribute set $\overline{S}$ as input. Obtain the updated public key component $\overline{AK_i}$, attribute private key component $\overline{S_i}$ and ciphertext component $\overline{C_y}$ are obtained.

Security model

We give the definition of the security of the target scheme under the chosen plaintext attack. The security of the algorithm is proven through a game between the challenger and the attacker. The game consists of the following process:

- **Initialization**: The attacker selects a target challenge policy P^* and sends it to challenger.
- **Parameter setting**: The challenger runs the initial algorithm to obtain the public key and private key, and sends the public key to the attacker.
- **Phase 1**: The attacker selects attribute set S and sends it to the challenger to request the corresponding private key. The challenger runs the private key generation algorithm to generate the private key, and sends it to the attacker.
- **Challenge**: The attacker randomly selects two messages (M_0, M_1) of equal length and a keyword W^* to send to the challenger. The challenger then randomly selects $\mu \in \{0, 1\}$, executes the encryption algorithm to encrypt M_μ and W^*. Finally, the challenge ciphertext CT_μ is obtained and returned to the attacker.
- **Phase 2**: Equivalent to Phase 1. The attacker can perform Phase 1 multiple times to test the attack.
- **Guess**: The attacker guesses $\mu' \in \{0, 1\}$; if $\mu' = \mu$, the attacker wins the game, and the winning advantage is $\varepsilon = \left|Pr[\mu' = \mu]\right| - \frac{1}{2}$.

Definition 2 The attribute-based encryption algorithm is considered secure if the attacker's advantage in winning the game described above is negligible in polynomial time.

Scheme construction

The goal of this study is to implement a secure access control scheme based on an improved attribute-based searchable algorithm and blockchain technology. In this section, we elaborate on the structure of the attribute-based searchable algorithm and the design of the smart contract.

Algorithm design

Figure 2 illustrates the workflow of the proposed algorithm. The proposed scheme consists of the following six phases: system initialization, data encryption, key generation, search, data decryption, and attribute revocation.

Phase 1: System initialization. $Setup(1^\lambda) \longrightarrow (PK, MK)$. Let G be a multiplicative cyclic group of order prime p and g to be a generating element of G. There exists a bilinear mapping $e : G \times G \longrightarrow G_T$ with a safe parameter λ as the group size. Then, set two hash functions $H_1 : \{0,1\}^* \longrightarrow G$ and $H_2 : \{0,1\}^* \longrightarrow Z_p$. Simultaneously, we define the Lagrangian coefficient $\bigtriangleup_{i,S}(x) = \prod_{j \in S, i \neq j} (x - j)/(i - j)$, where $i \in Z_p$, S is the set of elements in Z_p. The initialization algorithm takes the security parameter λ as input, and randomly selects $\alpha, \beta \in Z_p$. For each attribute $a_i \in U$, randomly select $v_i \in Z_p$, where U is the system attribute set. Finally, DO executes the initialization algorithm and obtains the system public key PK and master private key MK:

$$\begin{aligned} PK &= \left\{g, g^\alpha, g^\beta, e(g,g)^\alpha, \left\{AK_i = g^{v_i} \mid \forall a_i \in U\right\}\right\}, \\ MK &= \{\alpha, \beta, \{v_i \mid \forall a_i \in U\}\}. \end{aligned} \tag{1}$$

Phase 2: Data encryption. In the encryption phase, DO first encrypts the file M using AES and obtains the encrypted ciphertext $E_{ck}(M)$, where ck is the symmetric key. Then upload the ciphertext to the IPFS for distributed cloud storage, and obtain the storage hash address HD.

$ProEnc(PK, P) \longrightarrow CT_1$. The DO formulates the access policy P according to the system attribute set U, and sends it to the ES. The ES obtains access tree Υ according to policy P. Set a polynomial q_x for each node x in the tree in a top-down order, starting from the root node R. The degree d_x of the highest term of each node's polynomial satisfies $d_x = t_x - 1$, where t_x is the threshold of x. Starting from root node R, the algorithm randomly selects a number $k_1 \in Z_p$ as the secret number, and sets $q_R(0) = k_1$. It then randomly selects other coefficients $_{d_R}$ of q_R to fully define the polynomial. For

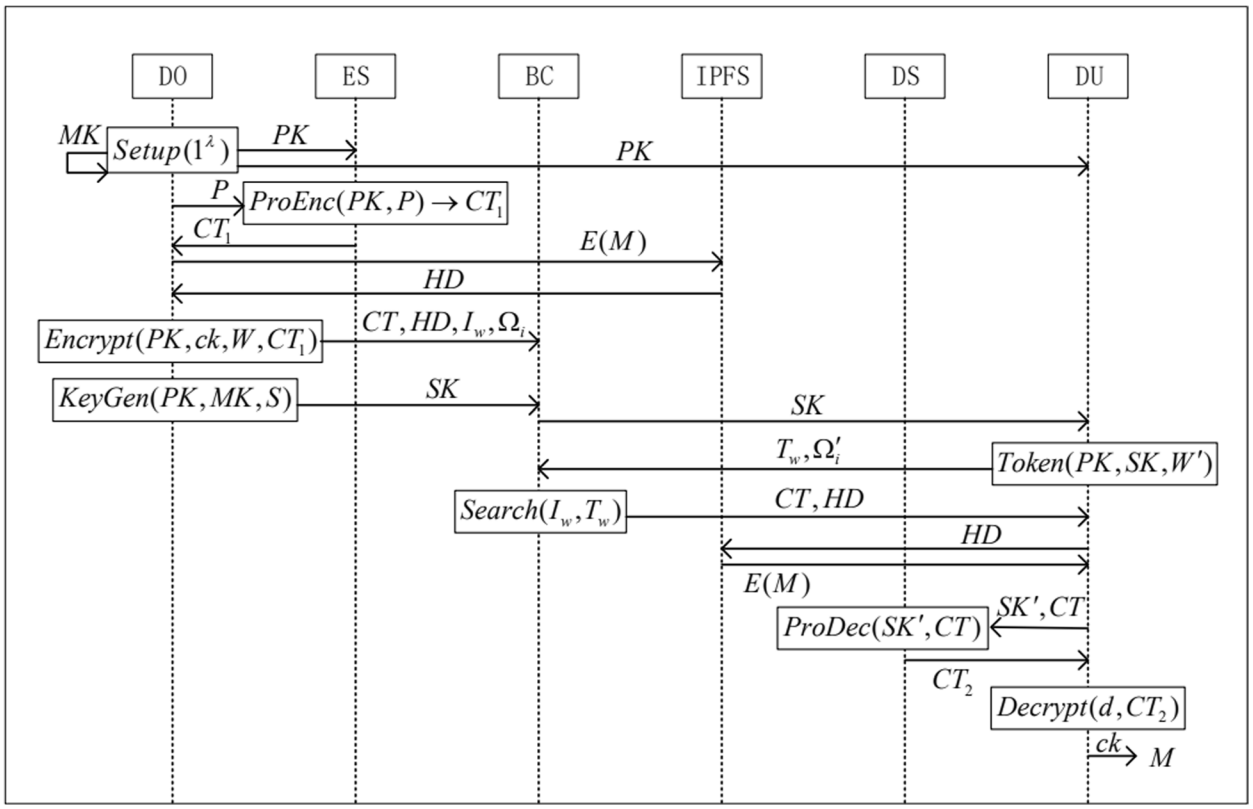

Fig. 2 The workflow of the proposed algorithm

other nodes x in Υ, we have $q_x(0) = q_{parent(x)}(index(x))$, where $index(x)$ represents the index of node x under parent node $parent(x)$. Randomly selected coefficients of d_x are used to get the full polynomial q_x. Letting Y be the attribute set of leaf nodes in the access tree Υ, the proxy encryption ciphertext can be obtained by the following calculation:

$$CT_1 = \left\{ \begin{array}{l} C_1' = g^{k_1}, \\ \left\{ C_y' = g^{v_i q_y(0)} \mid \forall a_i = att(y) \in Y \right\} \end{array} \right\}. \quad (2)$$

$Encrypt(PK, ck, W, CT_1) \longrightarrow (CT, \Omega_i, I_w)$. This algorithm is performed by the DO. After obtaining CT_1, DO randomly selects a number $k_2 \in Z_p$ and calculates: $C = ck \cdot e(g,g)^{\alpha k_2}$, $C_0 = g^{k_2}$, $C_1 = g^{\beta k_2} \cdot C_1' = g^{\beta k_2 + k_1}$, $C_2 = g^{H_2(ck)}$, $C_y = C_y'$. Finally, it outputs the key ciphertext:

$$CT = \left\{ C, C_0, C_1, C_2, \left\{ C_y \right\}_{att(y) \in Y} \right\}. \quad (3)$$

After that, DO sets the keyword W for the ciphertext, randomly selects $\sigma \in Z_p$, and calculates the keyword index:

$$I_w = \left\{ I_1 = g^{\frac{\beta\sigma}{H_2(W)}}, I_2 = g^{\alpha\sigma}, I_3 = g^{\sigma} \right\}. \quad (4)$$

In order to prevent the security threat caused by the disclosure of privacy attributes in the access policy. For each $att(i) \in P$, DO calculates $\Omega_i = e(H_1(att(i)), g^{\alpha})$ to hide the policy attribute. Then, DO creates the identity FID for the encrypted file and uploads $\{FID, CT, HD, I_w, \Omega_i\}$ to the blockchain.

Phase 3: Private key generation. $KeyGen(PK, MK, S) \longrightarrow SK$. The key generation algorithm is performed by the DO. DU sends the attribute set S to DO, and submits an access application. DO selects a random number $r \in Z_p$ and executes the following algorithm to generate the attribute private key:

$$SK = \left\{ \begin{array}{l} S_1 = g^{\alpha + \beta r}, S_2 = g^{r}, \\ \forall a_i \in S : S_i = g^{\frac{r}{v_i}}, S_{3,i} = H_1(a_i)^{\alpha} \end{array} \right\}. \quad (5)$$

Phase 4: Search. $Token(PK, SK, W') \longrightarrow (\Omega_i', T_w)$. If DU wants to get the ciphertext data, then DU needs to generate a search token. DU randomly chooses $\pi \in Z_p$ and computes the search trapdoor T_w based on the query keyword W':

$$T_w = \left\{ T_1 = g^{r\pi H_2(W')}, T_2 = g^{(\alpha+\beta r)\pi}, T_3 = g^{\pi} \right\}. \quad (6)$$

For each attribute $a_i \in S$, DU calculates $\Omega_i{}' = e\left(g, S_{3,i}\right)$. Eventually, DU sends the access token $\left\{\Omega_i{}', T_w\right\}$ to the blockchain to perform the search query.

$Search(I_w, T_w) \longrightarrow (0,1)$. After receiving the search request, the BC needs to determine whether the user's access behavior is legal. Then verify $\Omega_i = \Omega_i{}'$. If the equation holds, it means that the user attribute set satisfies the access policy. After that, BC executes the keyword search algorithm:

$$e(I_1, T_1)e(I_2, T_3) = e(T_2, I_3). \tag{7}$$

Correctness verification:

$$\begin{aligned} e(I_1, T_1)e(I_2, T_3) &= e(g^{\frac{\beta\sigma}{H_2(W)}}, g^{r\pi H_2(W')})e\left(g^{\alpha\sigma}, g^{\pi}\right) \\ &= e\left(g,g\right)^{(\alpha+\beta r)\pi\sigma} \\ e(T_2, I_3) &= e\left(g^{(\alpha+\beta r)\pi}, g^{\sigma}\right) = e\left(g,g\right)^{(\alpha+\beta r)\pi\sigma} \end{aligned}$$

If the search verification passes, return $\{CT, HD\}$ to the user. Otherwise output 0.

Phase 5: Data decryption. $ProDec\left(SK', CT\right) \longrightarrow CT_2$. The proxy decryption algorithm is executed by the proxy decryption server DS. DU randomly selects $d \in Z_p$, calculates the intermediate key:

$$SK' = \left\{ S_1{}' = S_1^{\frac{1}{d}}, S_2{}' = S_2^{\frac{1}{d}}, \left\{ S_i{}' = S_i^{\frac{1}{d}} \right\}_{\forall a_i \in S} \right\}. \tag{8}$$

Then, send $\left\{SK', CT\right\}$ to DS. Proxy decryption is performed in a state where the user's private key information is non-public, and the process is recursive. Given that the root node in access tree Υ is R, for each child node y in set Y, we set $p(y)$ to be the set of all nodes in the path from y to R. For node x in the access tree, let S_x be the set of sibling nodes (including itself). We have: $\bigtriangleup y = \prod_{x \in p(y), x \neq R} \bigtriangleup_{i,S}(0)$, where $i = index(x)$, $S = \left\{index(z) \mid z \in S_x\right\}$. If the user's attribute set satisfies the access policy, according to SK', for the root node R in the access tree Υ, compute

$$\begin{aligned} F_R &= \prod_{y \in Y, i = att(y)} e\left(C_y, S_i{}'\right)^{\bigtriangleup y} \\ &= \prod_{y \in Y, i = att(y)} e\left(g^{v_i q_y(0)}, g^{\frac{r}{dv_i}}\right)^{\bigtriangleup y} \\ &= e\left(g,g\right)^{\frac{rq_R(0)}{d}} \\ &= e\left(g,g\right)^{\frac{rk_1}{d}}. \end{aligned} \tag{9}$$

Otherwise, $F_R = \perp$. If F_R can be calculated, DS performs the next calculation:

$$\begin{aligned} D &= \frac{F_R \cdot e\left(S_1{}', C_0\right)}{e\left(S_2{}', C_1\right)} \\ &= \frac{e\left(g,g\right)^{\frac{rk_1}{d}} \cdot e(g^{\frac{\alpha+\beta r}{d}}, g^{k_2})}{e\left(g^{\frac{r}{d}}, g^{\beta k_2 + k_1}\right)} \\ &= e\left(g,g\right)^{\frac{\alpha k_2}{d}}. \end{aligned} \tag{10}$$

Then, secretly send $CT_2 = \{D, C, C_2\}$ to DU.

$Decrypt(d, CT_2) \longrightarrow ck'$. The DU obtains the ciphertext file $E(M)$ from the IPFS system according to the storage address HD, and performs the final decryption operation:

$$ck' = \frac{C}{D^d} = \frac{ck \cdot e\left(g,g\right)^{\alpha k_2}}{e\left(g,g\right)^{\alpha k_2}}. \tag{11}$$

To verify the correctness of decryption, DU computes $C_2 = g^{H_2(ck')}$. If the equation holds, DU decrypts the ciphertext using ck to obtain the plaintext file M.

Phase 6: Attribute revocation. $Revocation\left(\overline{S}\right) \longrightarrow \left(\overline{AK_i}, \overline{S_i}, \overline{C_y}\right)$. Assuming that there is a revocation attribute set $\overline{S}$. For each $a_i \in \overline{S}$, the DO randomly selects $\overline{v_i} \in Z_p$, calculates $vk_i = \overline{v_i}/v_i$. The DO updates the public key of a_i as $\overline{AK_i} = \left(g^{v_i}\right)^{vk_i}$. Then send vk_i to authorized users. The attribute revocation algorithm is accomplished through the following two steps:

- After gaining vk_i, legitimate users update the private key element corresponding to the attribute $a_i \in \overline{S}$: $\overline{S_i} = S_i^{vk_i^{-1}}$;
- For $\forall a_i = att\left(y\right) \in \overline{S}$, DO updates the ciphertext component: $\overline{C_y} = C_y^{vk_i}$. Afterwards, under the prewritten smart contract logic rules, the updated ciphertext is re-uploaded to the blockchain, and the access deadline of the revoked user is updated.

Smart contract design

In our scheme, the smart contract is primarily responsible for the storage and distribution of metadata ciphertexts. We created and deployed a smart contract on the Ethereum platform using solidity language. This section focuses on the following four functional module algorithms:

$Storage(Address_{DO}, FID, CT, HD, I_w, \Omega_i)$. This interface can only be called by DO. The function stores metadata related to the ciphertext, with the execution logic shown in Algorithm 1. First, it needs to verify whether DO is the caller of the method, where *msg.sender* refers to the current caller. Second, returns false if the *FID* already exists. Then, it forms a structure mapping: $FID \longrightarrow (CT, HD, I_w, \Omega_i)$, which is stored in the smart

contract. A transaction is eventually formed and broadcast in the blockchain.

SetUsk($Address_{DO}$, $Address_{DU}$, $E(SK)$, *deadline*). This function interface can only be called by DO. The execution logic is described in Algorithm 2. The DO encrypts the *SK* with the user's Ethereum account public key, and sets a valid access period for the user, uploading them into the smart contract. It generates a structure mapping: $Address_{DU} \longrightarrow (E(SK), deadline)$, and forms a storage transaction.

GetUsk($Address_{DU}$). This function interface can only be called by DU. The execution logic is shown in Algorithm 3. If the user information exists, the corresponding $E(SK)$ and *deadline* will be obtained according to the user's Ethereum address. Otherwise, an exception will be thrown. Subsequently, DU decrypts $E(SK)$ with its own Ethereum account private key. If the user's attribute set satisfies the access policy, the ciphertext can be decrypted by the attribute private key.

Search($Address_{DU}$, *token*). This function interface can only be called by DU. The algorithm takes the user address and search token as input to realize dynamic management of user access rights. The execution logic is shown in Algorithm 4. It is necessary to obtain the current access time *nowTime* and user access log. The access log contains the number of violations *lockNum*, lock time *lockTime*, and last successful access time *lastTime*. If the current time is less than the user's access deadline and the user's access lock time is less than the current time, then analyzes the user's access behavior. If the user's continuous access interval is less than the specified threshold, it is determined as an illegal behavior and the user's access rights are locked through the following penalty mechanism: $lockTime = nowTime + 2^{lockNum-1}$. Conversely, if the user's access behavior is legal, the latest access time of the user is recorded. Finally, BC verifies the user attributes and executes the keyword search algorithm, returning $\{CT, HD\}$ to the user.

Input: $Address_{DO}, FID, CT, HD, I_w, \Omega_i$
Output: $true/false$
1: **if** $msg.sender! = Address_{DO}$ **then**
2: **return** $false$;
3: **end if**
4: **if** FID already exists **then**
5: **return** $false$;
6: **end if**
7: Mapping: $FID \longrightarrow (CT, HD, I_w, \Omega_i)$;
8: **return** $true$;

Algorithm 1 Storage

Input: $Address_{DO}, Address_{DU}, E(SK), deadline$
Output: $true/false$
1: **if** $msg.sender! = Address_{DO}$ **then**
2: **return** $false$;
3: **end if**
4: **if** user information already exists **then**
5: **return** $false$;
6: **else**
7: Mapping: $Address_{DU} \longrightarrow (E(SK), deadline)$;
8: **return** $true$;
9: **end if**

Algorithm 2 SetUsk

Input: $Address_{DU}$
Output: $E(SK), deadline$
1: **if** $msg.sender! = Address_{DU}$ **then**
2: **throw** ("Only DU can call it. ");
3: **end if**
4: **if** private key does not exist **then**
5: **throw** ("Not qualified. ");
6: **end if**
7: **return** $E(SK), deadline$;

Algorithm 3 GetUsk

Input: $Address_{DU}, token$
Output: CT, HD
1: get the current time $nowTime$;
2: get the user's access log;
3: **require** $(msg.sender == Address_{DU})$;
4: **require** $(nowTime < DU.deadline)$;
5: **require** $(DU.lockTime < nowTime)$;
6: /*Access Behavior Audit*/
7: **if** $nowTime - DU.lastTime < interval$ **then**
8: $lockNum+ = 1$;
9: $DU.lockTime = nowTime + 2^{**}(lockNum - 1)$;
10: **throw** ("Illegal access.");
11: **else**
12: $DU.lastTime = nowTime$;
13: **end if**
14: /*Verify that the user attributes satisfy the policy*/
15: **require** $(\Omega_i == \Omega_i{}')$;
16: /*Keyword Search*/
17: **require** $(e(I_1, T_1)\, e(I_2, T_3) == e(T_2, I_3))$;
18: **return** CT, HD;

Algorithm 4 Search

To simplify the description, we take Fig. 3 as an example. Assuming $interval = 10$. When $nowTime - lastTime < 10$, it is judged as illegal access behavior and access is locked according

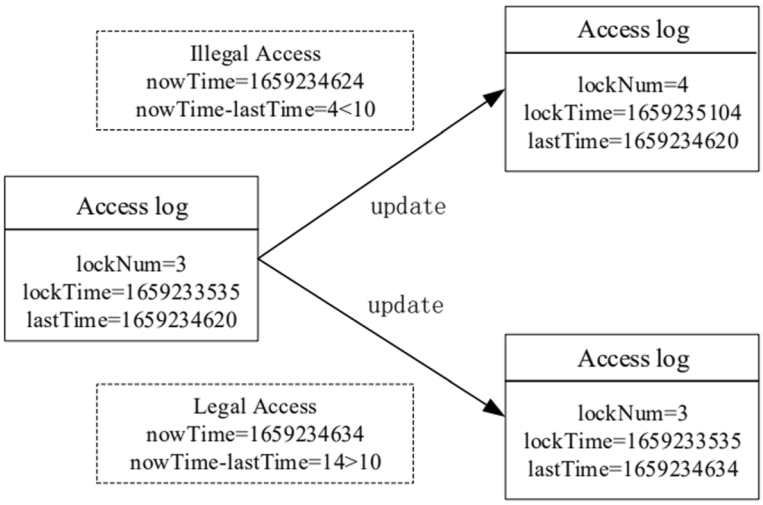

Fig. 3 Audit case for user access

to the penalty mechanism. Finally, the locking time and the number of locking times are updated. When $nowTime - lastTime \geq 10$, it is judged as legal access, and the time of the last successful access is updated to the current access time.

Analysis and evaluation

This section presents an analysis and evaluation of the proposed scheme from four aspects: security, functionality, communication and experimentation.

Security proof

Theorem 1 *The proposed scheme supports against chosen-plaintext attacks under the DBDH assumption.*

Proof

The challenger selects a bilinear mapping $e: G \times G \longrightarrow G_T$, *given a challenge tuple* (g, g^a, g^b, g^c, T), *where* $a, b, c \in Z_p, g \in G, T \in G_T$. *Then play the following game with the attacker:*

- **Initialization**: The attacker selects a target challenge policy P^* and sends it to challenger.
- **Parameter setting**: The challenger randomly selects $\alpha, \beta, x' \in Z_p$, set $\alpha = ab + x'$. For each attribute $a_i \in U$, it randomly selects $v_i \in Z_p$. Calculate the system public key:

$$PK = \left\{ \begin{array}{l} g, g^{ab+x'}, g^{\beta}, e(g,g)^{ab+x'}, \\ \{AK_i = g^{v_i} | \forall a_i \in U\} \end{array} \right\}. \tag{12}$$

The PK is sent to the attacker.

- **Phase 1**: The attacker selects an attribute set S and sends it to the challenger to initiate a private key request. The challenger chooses a random number $r' \in Z_p$ and lets $r = r' - b$. Then return SK to the attacker:

$$SK = \left\{ \begin{array}{l} S_1 = g^{ab+x'+\beta(r'-b)}, S_2 = g^{r'-b}, \\ \forall a_i \in S : S_i = g^{\frac{r'-b}{v_i}} \end{array} \right\}. \tag{13}$$

- **Challenge**: The attacker randomly selects two messages of equal length (M_0, M_1), a keyword W^*, and sends them to the challenger. The challenger randomly selects $\mu \in \{0,1\}$ and $k_1 \in Z_p$ to generate: $C_1' = g^{k_1}, C_y' = g^{v_i q_y(0)}$, where $a_i = att(y) \in Y$. After getting $CT_1^* = \left\{ C_1', C_y' \right\}$, the challenger performs the encryption algorithm to obtain the ciphertext:

$$CT^* = \left\{ \begin{array}{l} C = M_\mu \cdot T \cdot e(g,g)^{x'c}, C_0 = g^c, \\ C_1 = g^{\beta c + k_1}, C_2 = g^{H_2(M_\mu)}, C_y = C_y' \end{array} \right\}. \tag{14}$$

Then $_{\beta\sigma}$ randomly select $\sigma \in Z_p$ and compute: $I_1 = g^{\frac{\beta}{H_2(W^*)}}$, $I_2 = g^{(ab+x')\sigma}$, $I_3 = g^{\sigma}$. Return $\{CT^*, I_1, I_2, I_3\}$ to the attacker.

- **Phase 2**: Equivalent to Phase 1.
- **Guess**: The attacker guesses $\mu' \in \{0,1\}$. If $\mu' = \mu$, then $T = e(g,g)^{abc}$, and the challenger outputs 1. The attacker's advantage at this time is:

$$Pr\left[\Gamma\left(g, g^a, g^b, g^c, T = e(g,g)^{abc}\right) = 1\right] = \frac{1}{2} + \varepsilon. \tag{15}$$

If $\mu' \neq \mu$, then $T = R$, R is a random number in G_T, and the challenger outputs 0. The attacker's advantage at this time is:

$$Pr\left[\Gamma\left(g, g^a, g^b, g^c, T = R\right) = 1\right] = \frac{1}{2}. \quad (16)$$

In summary, the total advantage of an attacker in solving the DBDH problem is:

$$\frac{1}{2}Pr\left[\mu' = \mu\right] + \frac{1}{2}Pr\left[\mu' \neq \mu\right] - \frac{1}{2} = \frac{\varepsilon}{2}. \quad (17)$$

Since $\varepsilon/2 < \varepsilon$, the attacker's advantage in winning the game is negligible. Therefore, our scheme realizes the security under the chosen-plaintext attack.

Theorem 2 *The proposed scheme can prevent keyword guessing attacks in random oracles when given a one-way hash function H_2.*

Proof

If the attacker cannot correctly distinguish between $g^{x \cdot H_2(W_0)}$ and $g^{x \cdot H_2(W_1)}$ in the security game, this scheme can ensure the keyword security. The game process is as follows:

- **Initialization**: The challenger randomly selects $\alpha, \beta \in Z_p$, and a hash function: $H_2 : \{0,1\}^* \longrightarrow Z_p$. Then, execute the *Setup* algorithm and publish the *PK*.
- **Parameter setting**: The challenger randomly selects $r \in Z_p$ and calculates $S_1 = g^{\alpha+\beta r}$, $S_2 = g^r$.
- **Phase 1**: The attacker selects a keyword W^* and sends it to the challenger to apply for a search token. The challenger randomly selects $\pi \in Z_p$, calculates: $T_1 = g^{r\pi H_2(W^*)}$, $T_2 = g^{(\alpha+\beta r)\pi}$, $T_3 = g^{\pi}$, and returns $T^* = \{T_1, T_2, T_3\}$ to the attacker.
- **Challenge**: The attacker gives two keywords W_0, W_1. The challenger randomly selects a bit $\mu \in \{0,1\}$, calculates: $T_1 = g^{r\pi H_2(W_\mu)}$, $T_2 = g^{(\alpha+\beta r)\pi}$, $T_3 = g^{\pi}$, and sends them to the attacker.
- **Phase 2**: Similar to Phase 1 when the keywords W_0 and W_1 are not queried.
- **Guess**: The attacker guesses $\mu' \in \{0,1\}$. If $\mu' = \mu$, the attacker wins the game.

Assuming that the attacker can win the game with a non-negligible advantage ε in the guessing stage, this means that the attacker can solve the discrete logarithm problem, which is contrary to the fact. Moreover, since H_2 is a one-way hash function, attackers cannot obtain W_μ according to $H_2(W_\mu)$.

Theorem 3 *Our scheme supports policy hiding to ensure the security of user privacy attributes.*

Proof

In this scheme, attributes in the original access policy are hidden in the leaf nodes in the threshold access tree. It is difficult for the attacker to calculate the value k_1 of the root node. In addition, the data owner hashes the attributes in the access policy and then uses $e(H_1(att(i)), g^{\alpha})$ to hide the set of policy attributes. At the same time, if and only if authorized users can get from SK, and calculate $e(H_1(att(i))^{\alpha}, g)$. Since α is a private parameter and the hash function is unidirectional, thus the attacker cannot recover attributes and reconstruct the access policy. To sum up, our scheme can effectively hide the access policy and ensure the privacy of user attributes.

Theorem 4 *Our solution supports contract dynamic audit and strengthens user access authorization management.*

Proof

After the user initiates a query request to the smart contract, the contract will verify whether the user is within the specified access time limit. The access time limit of users with different attributes is not necessarily the same. Authorization decisions are made based on user access logs and real-time access behaviors. If the time interval of the user's continuous access is less than a certain threshold, it will be determined as illegal access, and the access will be locked according to the penalty mechanism. There is a positive correlation between the number of illegal access times and the lockout time. Meanwhile, the user's access log is updated with each access, and the access log cannot be changed due to the immutability of the blockchain. Finally, it verifies whether the user attributes meet the access policy according to the user access token, and performs the keyword-matching search. Therefore, this scheme can realize dynamic audit of user access and strengthen authorization management. To a certain extent, it helps to realize smart contract security.

Theorem 5 *Our scheme can reduce the cost of trust and ensure the storage security of ciphertext data.*

Proof

In this scheme, users get the metadata cipher text from the blockchain and no third party is involved in the whole process of authorization. The decentralized nature of the blockchain can create a fair and trustworthy transaction environment for two untrusted parties

and reduce the cost of trust in the system. Encrypted files are divided into multiple blocks and stored in IPFS, and metadata ciphertext is stored in the blockchain. Users do not need to worry about security risks and single-point failures caused by semi-honest third-party storage. In addition, even if the attacker obtains the ciphertext data through illegal means, it is still impossible to decrypt it without the corresponding attribute private key.

Functional analysis

In Table 2, we compare the functional characteristics of the proposed scheme and schemes [24, 25, 28, 29, 31, 32]. It can be seen that most schemes only support some functions. For example, schemes [28, 29, 32] support attribute revocation and keyword search, but not policy hiding. Schemes [24, 25, 31] support policy hiding, but not attribute revocation. In addition, only this scheme and scheme [32] support proxy decryption calculation, and only schemes [25, 32] apply blockchain technology. None of the comparison schemes support user behavior auditing. Taken together, our scheme is the most comprehensive and flexible, effectively meeting the needs of practical applications.

Communication analysis

From the communication point of view, we compare the storage cost of the proposed scheme with schemes [24, 28, 29], as shown in Table 3. $|G|, |G_T|$ represent the size of the elements in G and G_T, respectively. n_1 refers to the size of the attribute set in the ciphertext, n_2

Table 2 Functional comparison

Schemes	Fine grained	Policy hiding	Attribute revocation	Keyword search	Proxy decrypt	Blockchain	Dynamic audit
[24]	√	√	×	×	×	×	×
[25]	√	√	×	×	×	√	×
[28]	√	×	√	√	×	×	×
[29]	√	×	√	√	×	×	×
[31]	√	√	×	√	×	×	×
[32]	√	×	√	√	√	√	×
Ours	√	√	√	√	√	√	√

Table 3 Communication cost comparison

Schemes	[24]	[28]	[29]	Ours
Ciphertext size	$(2n_1+1)\|G\|+\|G_T\|$	$(2n_1+1)\|G\|+\|G_T\|$	$(2n_1+1)\|G\|+\|G_T\|$	$(n_1+3)\|G\|+\|G_T\|$
Index size	–	$(n_1+2)\|G\|+\|G_T\|$	–	$3\|G\|$
Private key size	$(2n_2+1)\|G\|$	$(3n_2+4)\|G\|$	$(2n_2+2)\|G\|$	$(2n_2+2)\|G\|$
Token size	–	$2\|G\|$	$(2n_2+1)\|G\|$	$3\|G\|+\|G_T\|$
Attribute revocation	–	$3n_3\|G\|$	$2n_3\|G\|$	$3n_3\|G\|$

Table 4 Computational cost comparison

Schemes	[24]	[28]	[29]	Ours
Encrypt	$(2n_1+2)E+n_1H$	$(4n_1+5)E+P+H$	$(2n_1+2)E+(n_1+1)H$	$n_1P+7E+(n_1+2)H$
KeyGen	$(2n_2+1)E+n_2H$	$(5n_2+8)E$	$(2n_2+2)E+n_2H$	$(2n_2+2)E+n_2H$
Token	–	$(n_2+2)E+H$	$(2n_2+1)E+H$	$3E+H+n_2P$
Search	–	n_2E+2P	n_2E+P	$3P$
Decrypt	n_2E+P	$(2n_2+1)P+n_2E$	–	$2E+H$
Revocation	–	$3n_3E$	$2n_3E$	$3n_3E$

refers to the size of the user attribute set, and n_3 refers to the size of the revocation attribute set. It can be seen from the table that the proposed scheme is better than other schemes in the storage cost of key ciphertext and keyword index. The attribute private key storage cost of this scheme is close to that of schemes [24, 28, 29], but this scheme supports policy hiding. In contrast, our scheme provides higher security. In addition, although the attribute revocation storage cost of this scheme is higher than that of scheme [29], this scheme is superior to scheme [29] in other aspects.

Experimental analysis

Table 4 gives a comparison of our scheme and schemes [24, 28, 29] in terms of computational cost. We mainly compare the computational cost of the six algorithms: *Encrypt*, *KeyGen*, *Token*, *Search*, *Decrypt* and *Revocation*. In the table, E represents the time of exponential operation, P represents the time of linear pairing, and H represents the time of hash computation. For a more intuitive presentation, we conducted simulation experiments on the operation of each algorithm. The experimental environment uses a host: Intel(R) Core(TM) i7-9700 CPU @ 3.00GHz, 32.0 GB RAM, Ubuntu 20.04.1. The target algorithm is implemented using Python and the bilinear pairwise encryption library PBC. Using the Truffle framework and Ganache to build a blockchain Ethereum development environment, the contract language uses Solidity. The experiment was repeated 100 times, and the average value was used as the evaluation standard.

As can be seen from Fig. 4, the encryption time of each scheme increases with the number of attributes. Since our scheme supports proxy encryption, the time consumption in the encryption phase is lower than other schemes. Figure 5 compares the computational cost of this scheme and other schemes during the key generation phase. It can be observed that the generation time of the attribute private key increases linearly with the increase in number of attributes. This is because the calculation of the user's attribute private key is related to the corresponding attributes, each of which is a part of the private key. Among them, scheme [28] has the highest computational cost. Figure 6 shows that the time cost of this scheme is the lowest during the token generation stage. Scheme [24] does not support keyword search, and schemes [28, 29] do not support policy hiding. Therefore, the advantages of this scheme are more obvious. The comparison of computational consumption in the search phase is shown in Fig. 7, the search time of the proposed scheme is maintained in a steady state, while the search cost of the schemes [28, 29] is positively correlated with the attribute base size. Figure 8 compares the time consumption

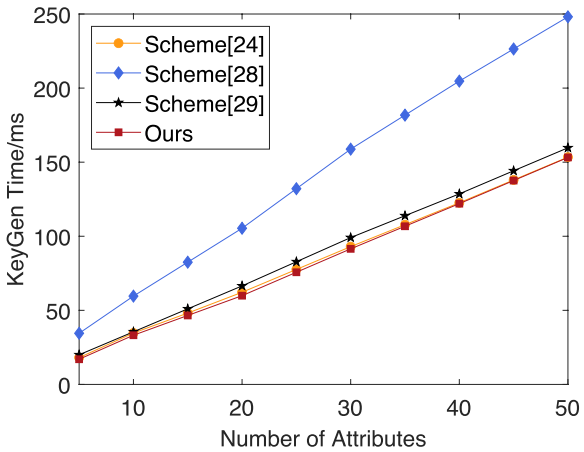

Fig. 5 Time consumption of KeyGen

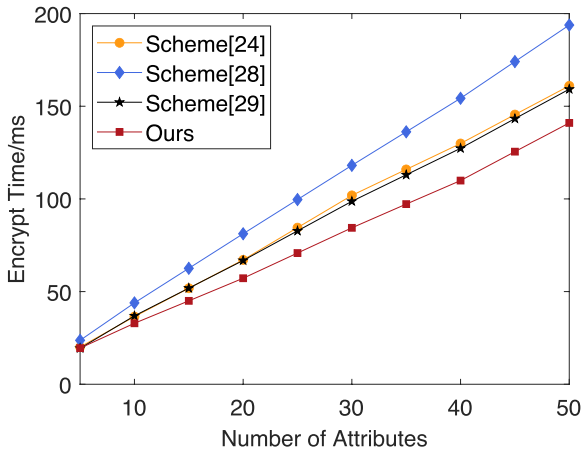

Fig. 4 Time consumption of Encrypt

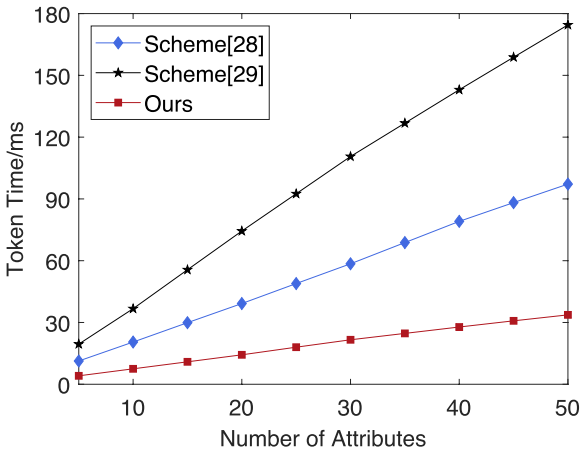

Fig. 6 Time consumption of Token

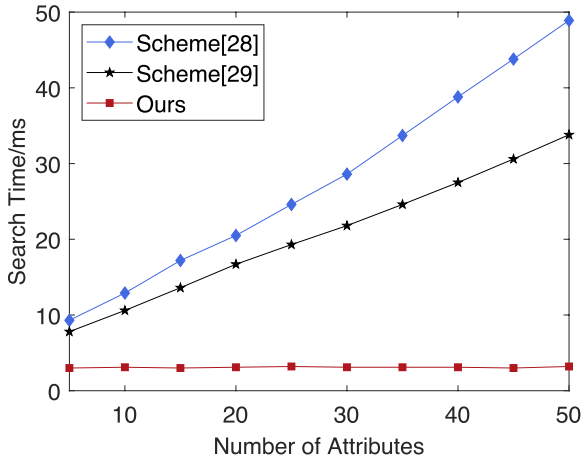

Fig. 7 Time consumption of Search

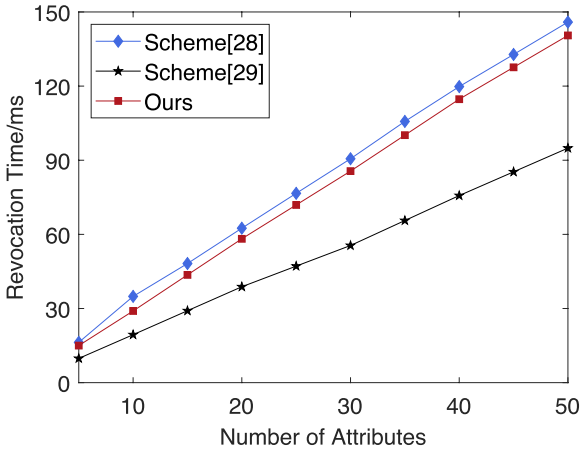

Fig. 9 Time consumption of Revocation

of each scheme in the decryption stage. Since this solution supports proxy decryption, the user decryption time is stable at about 1.5ms. Compared with schemes [24, 28], our scheme has the lowest decryption cost. According to Fig. 9, the time cost of our scheme is higher than that of the scheme [29] in the attribute revocation phase, but this does not affect the overall performance of this scheme.

Figure 10 illustrates the arbitration results of the Smart Contract for the authorization of persistent illegal access behavior by malicious users. The experiment simulated different access behaviors of ten accounts within the access period. Use positive and negative values to represent legal and illegal access behaviors, respectively. It can be observed that a user's access lock time increases exponentially with illegal behavior, which minimizes the impact of continuous illegal access or attacks on contract security.

Conclusion

In this paper, we propose a data security access control scheme based on blockchain and attribute-based searchable encryption in the cloud computing environment. This solution realizes fine-grained access and secure search of cloud data on the premise of supporting policy hiding and attribute revocation. At the same time, proxy encryption and decryption are introduced to reduce the computing cost of users. Combined with blockchain technology, it ensures the secure distribution of metadata ciphertext and keys, as well as a fair search for keywords. In addition, the smart contract is used to realize dynamic monitoring of user access behavior. Security analysis, performance comparison, communication analysis and computing analysis show that this scheme provides higher storage and computing performance while ensuring data security and user access fairness. It can be better applied to

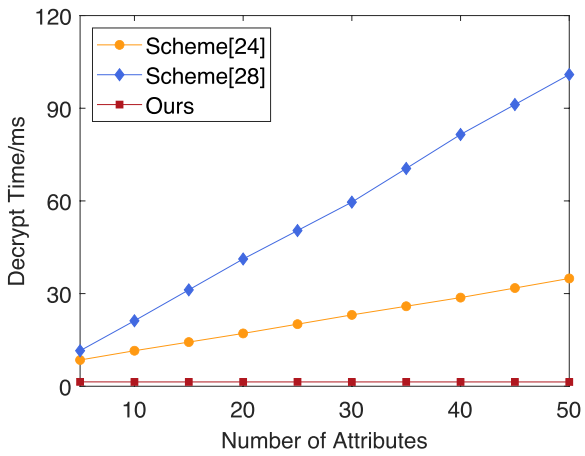

Fig. 8 Time consumption of Decrypt

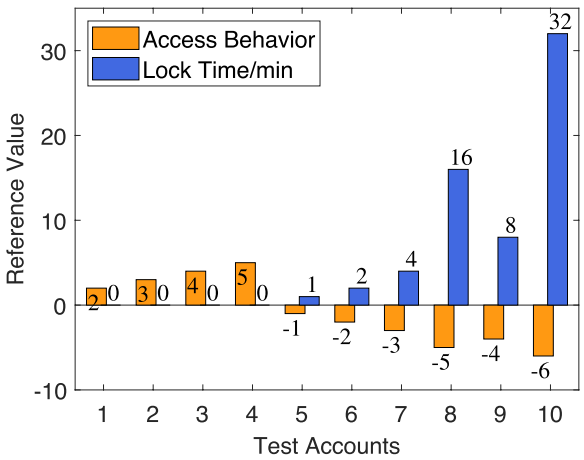

Fig. 10 Behavioral audit test

practical application scenarios such as smart grid and smart healthcare.

In future research, we will mainly focus on how to improve the efficiency of user access and multi-keyword search on the blockchain.

Authors' contributions

Liang Yan, Lina Ge, Zhe Wang, Guifen Zhang, Jingya Xu and Zheng Hu conceived and designed the study. Liang Yan completed the algorithm experiments and wrote the paper. Lina Ge guided the experimental design and data analysis. Zhe Wang participated in the revision of the paper. All authors reviewed and edited the manuscript. All authors read and approved the final manuscript.

Funding

This work was supported by the National Natural Science Foundation of China under Grant 61862007, and Guangxi Natural Science Foundation under Grant 2020GXNSFBA297103.

Availability of data and materials

The supporting data may be provided by corresponding author on request.

Declarations

Competing interests

The authors declare no competing interests.

Received: 15 October 2022 Accepted: 14 April 2023
Published online: 18 April 2023

References

1. Nayudu PP, Sekhar KR (2018) Cloud environment: A review on dynamic resource allocation schemes. Int J Appl Eng Res 13(6):4568–4575
2. Sandhu RS, Samarati P (1994) Access control: principle and practice. IEEE Commun Mag 32(9):40–48
3. Bethencourt J, Sahai A, Waters B (2007) Ciphertext-policy attribute-based encryption. In: 2007 IEEE symposium on security and privacy (SP'07); IEEE, CA. pp 321–334
4. Nayudu PP, Sekhar KR (2023) Dynamic time and location information in ciphertext-policy attribute-based encryption with multi-authorization. Intell Autom Soft Comput 35(3):3801-3813
5. Nayudu PP, Sekhar KR (2022) Accountable specific attribute-based encryption scheme for cloud access control. IntJ Syst Assur Eng Manag 1–10
6. Nayudu PP, Sekhar KR (2021) Enhancement of attribute - based encryption schemes through machine learning techniques: research challenges and opportunities. J Jilin University 40:1–18
7. Yuan Y, Wang FY et al (2016) Blockchain: the state of the art and future trends. Acta Autom Sin 42(4):481–494
8. Zhu N, Cai F, He J, Zhang Y, Li W, Li Z (2019) Management of access privileges for dynamic access control. Cluster Comput 22(4):8899–8917
9. Wang P, Yue Y, Sun W, Liu J (2019) An attribute-based distributed access control for blockchain-enabled iot. In: 2019 International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob). IEEE, Barcelona, pp 1–6
10. Kang J, Yu R, Huang X, Wu M, Maharjan S, Xie S, Zhang Y (2018) Blockchain for secure and efficient data sharing in vehicular edge computing and networks. IEEE Internet Things J 6(3):4660–4670
11. Liu B, Xiao L, Long J, Tang M, Hosam O (2020) Secure digital certificate-based data access control scheme in blockchain. IEEE Access 8:91751–91760
12. Yu J, Zhang H, Li S, Mao L, Ji P (2019) Data sharing model for internet of things based on blockchain. J Chin Mini-Micro Comput Syst 40(11):2324–2329
13. Al Breiki H, Al Qassem L, Salah K, Rehman MHU, Sevtinovic D (2019) Decentralized access control for iot data using blockchain and trusted oracles. In: 2019 IEEE International Conference on Industrial Internet (ICII). IEEE, Orlando, pp 248–257
14. Maesa DDF, Mori P, Ricci L (2019) A blockchain based approach for the definition of auditable access control systems. Comput Secur 84:93–119
15. Li R, Song T, Mei B, Li H, Cheng X, Sun L (2018) Blockchain for large-scale internet of things data storage and protection. IEEE Trans Serv Comput 12(5):762–771
16. Sandor VKA, Lin Y, Li X, Lin F, Zhang S (2019) Efficient decentralized multi-authority attribute based encryption for mobile cloud data storage. J Netw Comput Appl 129:25–36
17. Chen N, Li J, Zhang Y, Guo Y (2020) Efficient cp-abe scheme with shared decryption in cloud storage. IEEE Trans Comput 71(1):175–184
18. Fan K, Pan Q, Zhang K, Bai Y, Sun S, Li H, Yang Y (2020) A secure and verifiable data sharing scheme based on blockchain in vehicular social networks. IEEE Trans Veh Technol 69(6):5826–5835
19. Naz M, Al-zahrani FA, Khalid R, Javaid N, Qamar AM, Afzal MK, Shafiq M (2019) A secure data sharing platform using blockchain and interplanetary file system. Sustainability 11(24):7054
20. De SJ, Ruj S (2017) Efficient decentralized attribute based access control for mobile clouds. IEEE Trans Cloud Comput 8(1):124–137
21. Li H, Pei L, Liao D, Chen S, Zhang M, Xu D (2020) Fadb: A fine-grained access control scheme for vanet data based on blockchain. IEEE Access 8:85190–85203
22. Hur J (2013) Attribute-based secure data sharing with hidden policies in smart grid. IEEE Trans Parallel Distrib Syst 24(11):2171–2180
23. Cao L, Kang Y, Wu Q, Wu R, Guo X, Feng T (2020) Searchable encryption cloud storage with dynamic data update to support efficient policy hiding. China Commun 17(6):153–163
24. Huang C, Wei S, Fu A (2019) An efficient privacy-preserving attribute-based encryption with hidden policy for cloud storage. J Circ and Syst Comput 28(11):1950186
25. Gao S, Piao G, Zhu J, Ma X, Ma J (2020) Trustaccess: A trustworthy secure ciphertext-policy and attribute hiding access control scheme based on blockchain. IEEE Trans Veh Technol 69(6):5784–5798
26. Yang Y (2015) Attribute-based data retrieval with semantic keyword search for e-health cloud. J Cloud Comput Adv Syst Appl 4(1):1–6
27. Qiu S, Liu J, Shi Y, Zhang R (2017) Hidden policy ciphertext-policy attribute-based encryption with keyword search against keyword guessing attack. Sci China Inf Sci 60(5):1–12
28. Miao Y, Deng RH, Liu X, Choo KKR, Wu H, Li H (2019) Multi-authority attribute-based keyword search over encrypted cloud data. IEEE Trans Dependable Secure Comput 18(4):1667–1680
29. Yin H, Qin Z, Zhang J, Deng H, Li F, Li K (2020) A fine-grained authorized keyword secure search scheme with efficient search permission update in cloud computing. J Parallel Distrib Comput 135:56–69
30. Tu S, Waqas M, Huang F, Abbas G, Abbas ZH (2021) A revocable and outsourced multi-authority attribute-based encryption scheme in fog computing. Comput Netw 195:108196
31. Yin H, Li Y, Li F, Deng H, Zhang W, Li K (2022) An efficient and access policy-hiding keyword search and data sharing scheme in cloud-assisted iot. J Syst Archit 102533
32. Liu S, Yu J, Xiao Y, Wan Z, Wang S, Yan B (2020) Bc-sabe: Blockchain-aided searchable attribute-based encryption for cloud-iot. IEEE Internet Things J 7(9):7851–7867
33. Li W, Wu J, Cao J, Chen N, Zhang Q, Buyya R (2021) Blockchain-based trust management in cloud computing systems: a taxonomy, review and future directions. J Cloud Comput Adv Syst Appl 10(1):1–34
34. Onyema EM, Dalal S, Romero CAT, Seth B, Young P, Wajid MA (2022) Design of intrusion detection system based on cyborg intelligence for security of cloud network traffic of smart cities. J Cloud Comput Adv Syst Appl 11(1):1–20
35. Ma F, Fu Y, Ren M, Wang M, Jiang Y, Zhang K, Li H, Shi X (2019) Evm*: from offline detection to online reinforcement for ethereum virtual machine. In: 2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER): 24-27 February. IEEE, Harbin, pp 554–558
36. Kumar A, Abhishek K, Bhushan B, Chakraborty C (2021) Secure access control for manufacturing sector with application of ethereum blockchain. Peer Peer Netw Appl 14(5):3058–3074

37. Wohrer M, Zdun U (2018) Smart contracts: security patterns in the ethereum ecosystem and solidity. In: 2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE). IEEE, Campobasso, pp 2–8
38. Wang S, Yuan Y, Wang X, Li J, Qin R, Wang FY (2018) An overview of smart contract: architecture, applications, and future trends. In: 2018 IEEE Intelligent Vehicles Symposium (IV). IEEE, Changshu, pp 108–113
39. Zou W, Lo D, Kochhar PS, Le XBD, Xia X, Feng Y, Chen Z, Xu B (2019) Smart contract development: Challenges and opportunities. IEEE Trans Software Eng 47(10):2084–2106
40. Oliva GA, Hassan AE, Jiang ZMJ (2020) An exploratory study of smart contracts in the ethereum blockchain platform. Empir Softw Eng 25(3):1864–1904
41. Psaras Y, Dias D (2020) The interplanetary file system and the filecoin network. In: 2020 50th Annual IEEE-IFIP International Conference on Dependable Systems and Networks-Supplemental Volume (DSN-S). IEEE, Valencia, pp 80–80
42. Li F, Liu K, Zhang L, Huang S, Wu Q (2021) Ehrchain: a blockchain-based ehr system using attribute-based and homomorphic cryptosystem. IEEE Trans Serv Comput 15(5):2755-2765
43. Zheng Q, Li Y, Chen P, Dong X (2018) An innovative ipfs-based storage model for blockchain. In: 2018 IEEE/WIC/ACM international conference on web intelligence (WI): 03-06 December 2018. IEEE, Santiago, pp 704–708

Publisher's Note