Blockchain-Based Ciphertext Policy-Hiding Access Control Scheme

Ruizhong Du[1,2] and Tianhe Zhang[1(✉)]

[1] School of Cyber Security and Computer, Hebei University, Baoding 071002, China
ztian73817@gmail.com

[2] Hebei Provincial Key Laboratory of High Credibility Information System, Baoding 071000, China

Abstract. Ciphertext policy attribute encryption(CP-ABE) can realize one-to-many fine-grained access control, which can effectively solve the security problem of shared data. However, most existing CP-ABE protocols exhibit a single point of failure and access policy privacy disclosure problems. To resolve these issues, a blockchain-based ciphertext policy-hiding access control scheme is proposed. First, we propose a vector generation algorithm using vector compression techniques that can improve the efficiency of our scheme. Then, the privacy policy is protected by combining CP-ABE and inner product encryption. We then solve the user attribute revocation problem and single point of failure in the cloud-based access control models using ethereum. Finally, security analysis, the theoretical results, and experimental results show that the proposed solution is secure and efficient.

Keywords: Fine-grained access control · Blockchain · Policy-hiding · Inner product encryption

1 Introduction

As an emerging data interaction mode, cloud computing has dramatically transformed people's way of life, more individual and organizational users store their data online and share them remotely. People can access and obtain data as they wish. However, data stored in the cloud may contain private and confidential information. Thus, an attack or lack of monitoring may cause major accidents, such as data tampering or privacy leakage [4]. Data encryption is considered to be an effective method to achieve data security.

CP-ABE [5] can achieve fine-grained access control of user outsourced data and has higher flexibility and practicability, which has attracted extensive attention in industry. In CP-ABE, the ciphertext is related to the access policy, and the decryption key is related to the user's attributes. The trusted authority generates encryption-related parameters for the data owner to encrypt and sends the decryption key to the user. If the user attributes match the access policy, the user can decrypt the ciphertext to obtain the plaintext.

F. Li et al. (Eds.): SecureComm 2022, LNICST 462, pp. 272–289, 2023.
https://doi.org/10.1007/978-3-031-25538-0_15

In many current CP-ABE schemes, the data owner typically uploads encrypted data to a cloud server. The reliability of data access control thus depends to a large extent on the cloud server [6,25], and there may be a single point of failure. The blockchain can solve this problem well and improve the trustworthiness of the entire system. However, due to blockchain transparency, deploying access policies directly on the blockchain may reveal users' private data. For example, in a distributed access control system, Alice stores her electronic medical record access control policy in a blockchain. If the policy states that it can only be accessed by a cardiologist, Based on this access policy, an attacker can then infer that Alice may have a heart attack without obtaining her medical records. That is, an attacker could infer Alice's condition by knowing the contents of the access policy without accessing the contents of the medical records. Therefore, preventing malicious users from obtaining private information from access policies is a critical issue [21].

Currently, existing policy-hiding CP-ABE schemes have two forms: full policy-hiding [23] and partial policy-hiding [26]. In CP-ABE schemes with full policy-hiding, the entire attribute (name and value) is protected. Generally, full policy-hiding can be implemented by converting the access policy and user attributes into two vectors. Only when the result of multiplying the transformed two vectors is zero, the attribute set can satisfy the corresponding access policy. In this way, sensitive information is hidden by representing the access policy as a vector. Partial policy-hiding cannot protect all attributes, only attribute values, but not attribute names. Although full policy-hiding is not as efficient as partial policy-hiding, it can provide better privacy protection. For privacy-sensitive systems, any leakage of sensitive information can seriously threaten the data owner. Therefore, it is necessary to protect the personal information of data owners by implementing full policy hiding.

To resolve these issues, this study proposes a blockchain-based ciphertext policy-hiding scheme that achieves distributed fine-grained access control and protects the privacy of access policy. The main research contents of this study are as follows:

- We propose an efficient vector generation algorithm using vector compression techniques. Using ABE and inner product operation, we design a CP-ABE scheme with full policy hiding that can avoid leaking privacy-sensitive attribute information. Also, the proposed scheme performs constant bilinear pairing operations during decryption.
- We use the interplanetary file system to store encrypted information and store the ciphertext hash address using smart contracts. The storage overhead of the blockchain is reduced while achieving distributed and trustworthy access control. Revocation contract protects the private key from abuse.
- The security is demonstrated based on its application to difficult problems, and the performance is analyzed and explained via simulations, which verify the effectiveness of the scheme.

2 Related Works

2.1 Blockchain-Based Access Control Scheme

Saini et al. [20] built a distributed access system to achieve the goal of being patient-centric and accessible to medical records, which stored encrypted data on cloud servers, reducing the storage overhead of blockchain, but its scheme has scalability and performance problems. Zhang et al. [30] combine ethereum and traditional access control schemes and proposes a scheme to realize distributed access control through smart contract interaction. Their framework includes a policy management contract, two attribute management contracts, and an access contract for executing access control. Due to the openness and transparency of the blockchain, this scheme has the risk of attribute leakage. Ding et al. [9] proposed a scheme for the Internet of Things and used blockchain technology to simplify the access control process. The attributes of IoT devices are uploaded to the blockchain through transactions, and data access is allowed when the attributes meet the conditions of the access policy. Wang et al. [22] proposed a fine-grained access control framework based on blockchain. However, this scheme has the problems of access policy privacy leakage and poor scalability. Gao et al. [11] proposed an attribute hiding scheme based on blockchain and inner product encryption. The performance of this scheme degrades with increasing attributes in the system. Also, the scheme is constructed based on composite order bilinear groups (COBG), and the computational cost is relatively high.

Thus, these solutions primarily concentrates on enhancing the reliability and automation of the system, and there are some problems in terms of efficiency, scalability and user privacy.

2.2 Traditional Encryption Scheme

Phuong et al. [18] proposed an algorithm that uses Viete's formula and inner product encryption to achieve full policy hiding, but this scheme is less efficient. Subsequently, a series of policy hiding schemes [7,14] based on this algorithm appeared, despite some performance issues. Gan et al. [10] proposed a partial policy hiding scheme based on prime order bilinear groups, and added a decryption test algorithm in the system, which reduced the decryption overhead within a certain range, but the efficiency of the scheme was still low. Lai et al. [15] proposed a scheme to protect policy privacy, which only supports partial policy hiding. In addition, the decryption efficiency is low because the larger the access policy, the more the number of bilinear pairings. Based on [15], Zhang et al. [29] built a novel privacy protection scheme. Although this scheme can protect users' attribute privacy, its decryption overhead is still large. Liu et al. [17] built a multiauthorization outsourced scheme to solve the problem of complex decryption and key leakage, which achieves full policy hiding and has traceability. However, this scheme exhibits both attribute and user revocations. Hao et al. [12] proposed an efficient full policy-hiding scheme using Bloom filters, but their scheme cannot defend against attribute guessing attacks [31]. Zhang et al. [28] designed

a scheme to resist attribute guessing attacks based on the linear secret sharing scheme, but its decryption algorithm is inefficient, and resource consumption on IoT devices is large. Hu et al. [13] constructed a highly efficient policy hiding scheme to protect privacy, but their study only supports partial policy hiding and cannot perform attribute revocation. Xiong et al. [24] achieved partial policy hiding using the linear secret sharing scheme. However, its scheme has many exponentiation operations and pairing processes during the execution of the system, which results in low efficiency. There are also some policy hiding schemes based on COBG [16,19,27], whose efficiency is low.

Thus, traditional encryption schemes have many disadvantages, such as inefficiency, privacy leakage and attribute revocation. So we propose a blockchain-based ciphertext policy-hiding access control scheme to avoid these problems. While protecting user privacy, this scheme overcomes the problems of attribute revocation and inefficiency.

3 Preliminaries

3.1 Bilinear Operation

We let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be three cyclic groups of prime order p, where $\mathbb{G}_1 \neq \mathbb{G}_2$, and there are no efficiently computable homomorphisms between $\mathbb{G}_1$ and $\mathbb{G}_2$. e: $\mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is considered to be an asymmetric bilinear operation if

- For all $g \in \mathbb{G}_1, h \in \mathbb{G}_2$, and $a, b \in \mathbb{Z}_p^*$, $e\left(g^a, h^b\right) = e(g,h)^{ab}$.
- $e(g,h) \neq 1$.
- $e(g,h)$ for all $g \in \mathbb{G}_1$ and $h \in \mathbb{G}_2$ can be calculated efficiently.

3.2 Complexity Assumption

Asymmetric Decisional Bilinear Diffie-Hellman Problem (DBDH). There are the following two distributions, where $g \in \mathbb{G}_1, h \in \mathbb{G}_2, a, b, c \in \mathbb{Z}_p^*$ and $T \in \mathbb{G}_T$ are all randomly generated.

$$-\mathcal{P}_A := \left(g, g^a, g^c, h, h^a, h^b, e(g,h)^{abc}\right) \in \mathbb{G}_1^3 \times \mathbb{G}_2^3 \times \mathbb{G}_T$$

$$-\mathcal{R}_A := \left(g, g^a, g^c, h, h^a, h^b, T\right) \in \mathbb{G}_1^3 \times \mathbb{G}_2^3 \times \mathbb{G}_T$$

$Adv_{\mathcal{A}}^{DBDH}$ is the advantage of algorithm $\mathcal{A}$ to distinguish the following two distributions. Specifically as follows:

$$Adv_{\mathcal{A}}^{DBDH} = |\Pr[\mathcal{A}(D) = 1] - \Pr[\mathcal{A}(R) = 1]|,$$

D and R are selected from $\mathcal{P}_A$ and $\mathcal{R}_A$, respectively.

An algorithm $\mathcal{B}$ has advantage $Adv_{\mathcal{A}}^{DBDH} = \epsilon$ in solving the $DBDH$ problem if:

$$\begin{aligned} &|\Pr\left[\mathcal{B}\left(g, g^a, g^c, h, h^a, h^b, e(g,h)^{abc}\right) = 0\right] \\ &\quad - \Pr\left[\mathcal{B}\left(g, g^a, g^c, h, h^a, h^b, T\right) = 0\right]| \geq \epsilon \end{aligned}$$

where the probability is above the random choice of generators $g \in \mathbb{G}_1$ and $h \in \mathbb{G}_2$, exponents $a, b, c \in \mathbb{Z}_p^*, T \in \mathbb{G}_T$ and the random bits used by $\mathcal{B}$.

Definition 1. If for any PPT algorithms $\mathcal{A}$, the function $Adv_{\mathcal{A}}^{DBDH}(\lambda)$ is a negligible function of λ, we consider the $DBDH$ to hold for $\mathcal{G}$, where $\mathcal{G}$ is a bilinear group generator.

To justify the security of the access policy, another assumption needs to be introduced.

$\mathcal{P}$-Asymmetric Decisional Bilinear Diffie-Hellman Problem ($\mathcal{P}$-DBDH)

The following two distributions are similar to those in the $DBDH$ assumption, with the difference that $T \in \mathbb{G}_1$ instead of $T \in \mathbb{G}_T$.

$$-\mathcal{D}_N := \left(g, g^a, g^{ab}, g^c, h, h^a, h^b, g^{abc}\right) \in \mathbb{G}_1^4 \times \mathbb{G}_2^3 \times \mathbb{G}_1$$

$$-\mathcal{D}_R := \left(g, g^a, g^{ab}, g^c, h, h^a, h^b, T\right) \in \mathbb{G}_1^4 \times \mathbb{G}_2^3 \times \mathbb{G}_1$$

$Adv_{\mathcal{A}}^{\mathcal{P}-DBDH}$ is the advantage of algorithm $\mathcal{A}$ to distinguish the following two distributions. Specifically as follows:

$$Adv_{\mathcal{A}}^{\mathcal{P}-DBDH} = |\Pr[\mathcal{A}(N) = 1] - \Pr[\mathcal{A}(P) = 1]|$$

N and P are selected from $\mathcal{D}_N$ and $\mathcal{D}_R$, respectively.

An algorithm $\mathcal{B}$ has advantage $Adv_{\mathcal{A}}^{\mathcal{P}-DBDH} = \epsilon_{\mathcal{P}}$ in solving the $\mathcal{P} - DBDH$ problem if:

$$\begin{aligned} &|\Pr\left[\mathcal{B}\left(g, g^a, g^{ab}, g^c, h, h^a, h^b, g^{abc}\right) = 0\right] \\ &- \Pr\left[\mathcal{B}\left(g, g^a, g^{ab}, g^c, h, h^a, h^b, T\right) = 0\right]| \geq \epsilon_{\mathcal{P}} \end{aligned}$$

where the probability is above the random choice of generators $g \in \mathbb{G}_1$ and $h \in \mathbb{G}_2$, exponents $a, b, c \in \mathbb{Z}_p^*, T \in \mathbb{G}_1$ and the random bits used by $\mathcal{B}$.

Definition 2. If for any PPT algorithms $\mathcal{A}$, the function $Adv_{\mathcal{A}}^{\mathcal{P}-DBDH}(\lambda)$ is a negligible function of λ, we consider the $\mathcal{P} - DBDH$ to hold for $\mathcal{G}$, where $\mathcal{G}$ is a bilinear group generator.

3.3 Access Structure

We let $\boldsymbol{U} = \{U_1, U_2, \cdots, U_L\}$ be attributes, $\mathrm{U}_k \in \{\text{“+”}, \text{“−”}\}$; let $\boldsymbol{P} = \{P_1, P_2, \cdots, P_L\}$ be an access policy, $\mathrm{P}_k \in \{\text{“+”}, \text{“−”}, \text{“*”}\}$, where $k \in \{1, 2, \cdots, L\}$. The wildcard “$*$” means that both “$+$” and “$-$” are accepted. For example:

We assume that $\boldsymbol{U} = \{U_1 = AI, U_2 = CE, U_3 = Faulty, U_4 = Student\}$, where “$AI$” and “$CE$” represent artificial intelligence and communications engineering, respectively. Anna is an AI faculty member, and Bell is a CE student. P_1 can be matched by all CE faculties without being in the AI, and P_2 can be matched by all AI faculty members and students, excluding those in the CE. User attributes and access policies are shown in Table 1.

Table 1. User attributes and access policies

Attribute	A_1	A_2	A_3	A_4
Description	AI	CE	Faculty	Student
Anna	+	−	+	−
Bell	−	+	−	+
P_1	−	+	+	−
p_2	+	−	*	*

3.4 Viete's Formulas

There are two vectors $p = (p_1, p_2, \cdots p_l)$ and $u = (u_1, u_2, \cdots u_l)$. Vector p contains positive signs, negative signs and wildcards, and vector u only contains positive signs and negative signs. The set of positions $I = \{i_1, \ldots, i_n\} \subseteq \{1, 2, \cdots l\}$ represents the positions of the wildcards in the vector p.

If there is a one-to-one correspondence between the attributes of the user and the attributes contained in the access policy, then $((p_i = u_i) \vee (p_i = *))$, $i \in [1, l]$, converted into mathematical form as shown in (1):

$$\sum_{i=1, i\in I}^{l} p_i \prod_{k_w \in I} (i - k_w) = \sum_{i=1}^{L} u_i \prod_{k_w \in I} (i - k_w) \tag{1}$$

We let $\prod_{k_w \in I} (i - k_w) = \sum_{j=1}^{n} a_j i^j$. So Eq. 2 can be obtained as follows:

$$\sum_{i=1, i\in I}^{l} p_i \prod_{k_w \in I} (i - k_w) = \sum_{j=0}^{n} a_j \sum_{i=1}^{L} u_i i^j \tag{2}$$

Then we choose a random number B_i to hide the information in Eq. (2) and put p_i, u_i as the exponents of B_i. Thus, we can obtain the following formula:

$$\prod_{i=1, i\notin I}^{L} B_i^{p_i \prod_{k=I}^{(i-k)}} = \prod_{j=0}^{n} \left(\prod_{i=1}^{l} B_i^{u_i i^j} \right)^{a_j} \tag{3}$$

According to Viete's formulas, the coefficients a_j in Eq. (2) can be represented by k_w:

$$a_{n-j} = (-1)^j \sum_{1 \le i_1 \le i_2 \le \ldots < i_j \le n} k_i k_{i_2} \cdots k_{i_j}, 0 \le j \le n = |I| \tag{4}$$

For example, if we have $I = \{3, 4\}$, then we can get $(x - 3)(x - 4)$, and $a_2 = 1, a_1 = -(3 + 4) = -7, a_0 = 3 * 4 = 12$.

4 System Overview

4.1 System Model

The system model built in this study is illustrated in Fig. 1. The model includes five entities: the data owner (DO), data user (DU), consensus nodes (CN),

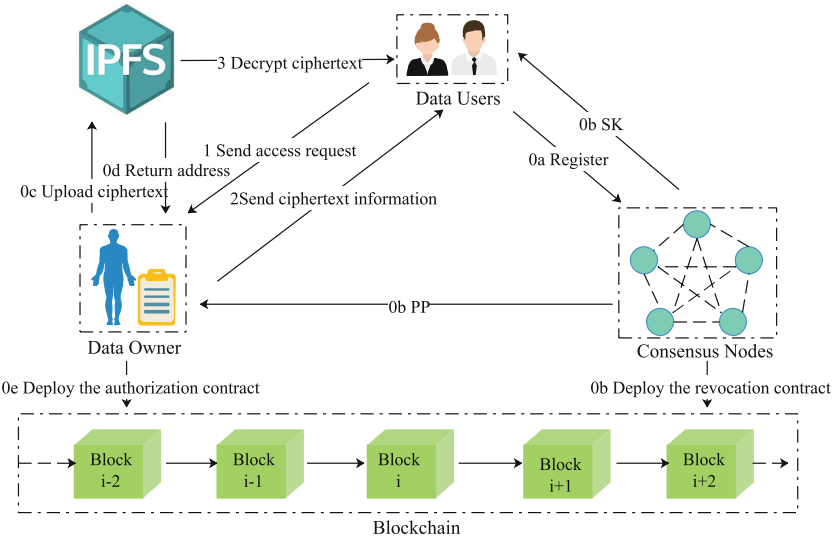

Fig. 1. System model

blockchain (BC) and interplanetary file system (IPFS). We suppose that communication between any two of the five entities is secure In this system, the access control workflow can be divided into a preparation phase (0a-0e) and an execution phase (1–3). The preparation phase primarily involves the distribution of keys, the deployment of contracts and the upload of ciphertext information. The execution phase mainly judges the access request.

Preparation Phase: First, the DU registers, and CN generate relevant parameters and deploy the revocation contract. DO encrypts the message M on the basis of the public key and access policy and stores the ciphertext on IPFS. After that, DO deploys the authorization contract and stores the ciphertext related information on the blockchain through transactions.

Execution Phase: First, the DU generates access information according to their own needs and send them to DO. After verifying the identity of the DU, the DO sends the ciphertext-related information to the DU. Du decrypts the ciphertext after performing the integrity check on the ciphertext.

DO: A DO specifies with which attributes users can access their own data and encrypt the data, after which the encrypted data are uploaded to the IPFS, and its hash address is kept on Ethereum through the authorization contract.

DU: A DU generates access information according to their own needs and send them to DO, obtains the ciphertext address after passing the verification of the authorization contract, and then decrypts the ciphertext.

CN: We displace the trusted authority in the traditional encryption system with a set of pre-defined trusted consensus nodes, which are responsible for generating system parameters and deploying revocation contracts.

BC: A BC includes the authorization contract and the revocation contract. The revocation contract is responsible for revoking retired or resigned users, and the authorization contract is responsible for the release of ciphertext information and the judgment of access requests.

IPFS: A IPFS is a distributed system that stores ciphertexts and returns hashed addresses.

4.2 Security Model

We now define selective IND-CPA security for CP-ABE with hidden access policy. The solution in this paper needs to implement IND-CPA security and access policy security. This paper proves IND-CPA security by using $DBDH$ assumption, and proves the security of access policy by $\mathcal{P} - DBDH$ assumption. We define the following security game Γ_w for our scheme.

- **Initialization**. Adversary $\mathcal{A}$ outputs two challenge vectors p_0 and p_1 to challenger $\mathcal{B}$.
- **Setup**. $\mathcal{B}$ runs the Setup (λ, n) algorithm and gives a public key PK to $\mathcal{A}$.
- **Phase 1**: $\mathcal{A}$ may adaptively make a polynomial number of queries to create a secret key for the attribute u subject to the restrictions that $\langle p_0, u\rangle \neq 0$ and $\langle p_1, u\rangle \neq 0$.$\mathcal{B}$ generates a secret key and inform $\mathcal{A}$ of it.
- **Challenge**: $\mathcal{A}$ outputs challenge messages M_0, and M_1. $\mathcal{B}$ chooses a random bit w. $\mathcal{A}$ given $C_T \leftarrow$ Encrypt (PP, M_w, p_w).
- **Phase 2**: Same as Phase 1.
- **Guess**: $\mathcal{A}$ outputs a bit $w' \in \{0, 1\}$ for $\mathcal{B}$ and wins the game if $w' = w$. Thus, we define the advantage $\mathcal{A}$ as:

$$Adv_{\mathcal{A}}{}^{IND-sPH-CPA}(\lambda) = \left|\Pr\left[w' = w\right] - \frac{1}{2}\right|$$

We say that the scheme is selective IND-CPA secure and policy-hiding if $Adv_{\mathcal{A}}{}^{IND-sPH-CPA}(\lambda)$ is negligible.

4.3 Attribute Vector and Policy Vector Generation Algorithms

A policy vector and attribute vector generation algorithm is proposed in [18], but the algorithm must generate three vectors: (one policy vector and two attribute vectors), markedly increasing the computational cost of this scheme. Based on [18], We propose an algorithm using vector fusion technology, which only needs to generate an attribute vector and a policy vector, and the algorithm is also applicable to other wildcard-based access structures. Since our vector generation algorithm generates fewer vectors, fewer policy matches are performed during the system execution phase, so it is more efficient. Specifically, this algorithm can be described as follows:

Algorithm 1. Vector Generation Algorithm

Input: An attribute set $\boldsymbol{U} = \{U_1, U_2, \cdots, U_L\}$ and a policy set $\boldsymbol{P} = \{P_1, P_2, \cdots, P_L\}$.

Output: A policy vector and an attribute vector.

(1): Positive, negative and wildcard symbols in an access structure are first separated into three position sets J, K and I;

(2): while $k_w \in I$ do

(3): Expand $\prod_{k_w \in I}(i - k_w) = \sum_{j=0}^{n} a_j i^j$ to derive coefficients a_j

(4): for $k_w \in I$ and $i \in J$ do

(5): Compute $\prod_J = + \sum_{i \in J} \prod_{k_w \in I}(i - k_w)$

(6): for $k_w \in I$ and $i \in K$ do

(7): Compute $\prod_K = + \sum_{i \in K} \prod_{k_w \in I}(i - k_w)$

(8): Compute $\Pi = \Pi_J + \Pi_K$

(9): end for

(10): Positive and negative symbols in a user attribute set are also separated into two position sets J' and K';

(11): for $i = 1$ to l and $i \in J'$ do

(12): Compute $u_j = + \sum_{i \in J'} i^j$

(13): for $i = 1$ to l and $i \in K'$ do

(14): Compute $u'_j = + \sum_{i \in K'} i^j$;

(15): Compute $u = u_j + u'_j$

(16): Return policy vector $\overrightarrow{p} = (a_0, a_1, \cdots, a_n, 0_{n+1}, \cdots, 0_l, \Pi)$, att-ribute vector $\overrightarrow{u} = (u_0, u_1, \cdots, u_l, -1)$

For instance, as shown in Table 1, $p_2 = (+, -, *, *)$ for the set of wildcard positions $I = \{3, 4\}$, the set of positive positions $J = \{1\}$ and the set of negative positions $K = \{2\}$. Based on Viete's formulas, we can obtain:

$$a_2 = 1; a_1 = -7, a_0 = 12$$

Therefore, the policy vector can be obtained as shown in Table 2.

Table 2. Policy vector

	0	1	2	3
p_2	12	-7	1	$(1-3)(1-4) + (2-3)(2-4)$

Andy's attribute set $u_{Andy} = (+, -, +, -)$. The set of positive positions $J' = \{1, 3\}$ and the set of negative positions $K' = \{2, 4\}$. Therefore, the attribute vector can be obtained as shown in Table 3.

Then, we can calculate $p_2 \cdot u_{Andy} = 12 \times 4 - 7 \times 10 + 1 \times 30 - 8 = 0$. Therefore, Andy's attributes satisfy the access policy p_2.

4.4 Smart Contract Design

In our system, smart contracts mainly include authorization contracts and revocation contracts, as follows:

Table 3. Attribute vector

	0	1	2
+	$1^0 + 3^0$	$1^1 + 3^1$	$1^2 + 3^2$
−	$2^0 + 4^0$	$2^1 + 4^1$	$2^2 + 4^2$
u_{Andy}	4	10	30

(1) Revocation Contract: This contract is deployed by CN and is responsible for managing users who have left or retired in the system. It mainly includes the following functions:

updateAssert(): This function is responsible for updating the revocation list, and putting resigned or retired users into the revocation list.

getAssert(): This function returns true or false to determine whether the user has been revoked.

(2) Authorization Contract: This contract is deployed by DO and is responsible for storing ciphertext-related information. When DU sends an access request, it calls the revocation contract to check the identity of DU, so as to determine whether to send ciphertext-related information. It mainly includes the following functions:

initStorage(): This function is responsible for storing ciphertext related information.

judge(): In the access control phase, this function calls the getAssert() function in the revocation contract, and decides whether to send the ciphertext information according to the result.

4.5 Our Construction

Our solution consists of the following six procedures:

Setup(λ, n): The algorithm generates relevant parameters for the system and is executed by the CN. Given a security parameter λ and n attributes, CN pick random values $(z, a_0, a_1, \ldots, a_n) \in \left(\mathbb{Z}_p^*\right)^{n+2}$ and set them as follows:

$$g_1 = g^{a_1}, \ldots, g_n = g^{a_n}, g_0 = g^z, \quad h_1 = h^{a_1}, \ldots, h_n = h^{a_n}, h_0 = h^{a_0}$$

The MSK and the PP are given by:

$$\begin{aligned} &\text{MSK} = (h, h_0, h_1, \cdots, h_n), \\ &PP = (g, g_0, g_1, \cdots, g_n, e(g, h_0)) \end{aligned}$$

KeyGen(PP, MSK, u): The algorithm generates decryption key for DU and is done by CN, which randomly chooses $R \in \mathbb{Z}_p^*$, and based on public parameters, master key and attribute vector u. The CN then generate $sk_u = (sk_1, sk_2, u)$ as:

$$sk_1 = h_0 \prod_{i=1}^{n} h_i^{u_i R}, sk_2 = h^R$$

Encrypt(PP, p, M): The algorithm is performed by the DO. Relying on PP and access policy p, the DO selects a random $s \in \mathbb{Z}_p^*$ and generates the encrypted message $C_T = (c_0, c_1, c_{2,1}, \cdots, c_{2,n})$ as below:

$$c_0 = M \cdot e\left(g, h_0\right)^s, c_1 = g^s, c_{2,i} = g_0^{p_i s} g_i^s \ for\ 1 \leq i \leq n.$$

On-blockchain: This algorithm is also performed by the DO, which stores ciphertext-related information on the blockchain through the authorization contract:

$$Tx = (id_T, storeAddress, sign)$$

where id_T is the identifier; and $storeAddress$ is the hash address of the ciphertext. Because IPFS is based on content addressing, it is also the ciphertext integrity check code. After obtaining the ciphertext, the user can use it to perform an integrity check first and then decrypt it. $sign$ is the digital signature. DU can judge whether the encrypted data is what they want based on the digital signature.

Decrypt($sk_{\boldsymbol{u}}$, CT): This algorithm is implemented by the DU, which obtains ciphertext-related information after passing the verification of the authorization contract. The DU first verifies the digital signature of the DO; obtains the ciphertext and verifies its integrity; and finally recovers message M if the user attributes meet the attributes specified by the data owner. We can obtain the plaintext information M by the following calculation:

$$M = e\left(c_1, sk_1\right)^{-1} \cdot e\left(\prod_{i=1}^{n} \left(c_{2,i}\right)^{u_i}, sk_2\right) \cdot c_0 = M \cdot e(g,h)^{zsR\langle p,u\rangle}$$

If $\langle p, u\rangle = 0$, then we get M.

Revocation: We now assume that the DU_1 no longer has attribute u_1. Because DU_1 has asked for and obtained sk_{u_1}, it is able to obtain message M encrypted under access policy p_1.

To solve this problem, CN map user addresses to true and false through the mapping type in Solidity. Specifically, as follows:

$$mapping(address => bool)\ public\ identify$$

if DU_1 has the attribute u_1.

$$identify[userAddress] => true$$

otherwise:

$$identify[userAddress] => false$$

When the DU_1 accesses the message, the authorization contract can determine whether to send ciphertext information through the key value corresponding to the user's address.

5 Security Proof

5.1 Security Analysis of Blockchain Operations

Trustability: Many current schemes introduce a central entity to generate relevant parameters in the system, but when this central entity fails, the entire system will be paralyzed. We ensure the trust of access control by storing ciphertext addresses on the blockchain. During the access authorization stage, each DU is verified by the authorization contract, achieving trusted authorization with less human intervention.
Privacy: Most of the existing methods directly store the access policy on the blockchain, from which attackers can easily obtain DO information. To protect privacy, it is quite important to protect the access structure. The proposed scheme can hide access policies through vector representations. Data are encrypted and stored on an IPFS, and its hash address is kept on Ethereum, which ensures that malicious users cannot obtain sensitive data and also decreases the cost of saving data in Ethereum.
Integrity: We store the ciphertext hash address on the blockchain through smart contracts, which improves system scalability. However, after the DO saves the ciphertext address on Ethereum, it is still possible to change the ciphertext content. In order to ensure that DU can obtain the latest information, we save the ciphertext hash value on the blockchain through a smart contract for users to check the integrity of the ciphertext content. Also, storing the ciphertext hash address on the blockchain avoids a situation where the DO refuses to provide their personal data, ensuring that any unrevoked user will have access to the.
Traceability: Because our scheme is based on blockchain, any authorization information will be recorded as an immutable access transaction, making the proposed scheme traceable.

5.2 Security Analysis of Scheme

Theorem 1. *If the DBDH assumption and the $\mathcal{P}-DBDH$ assumption hold, then any PPT attacker cannot selectively get plaintext information and access policy information.*

The detailed security proof of Theorem 1 is omitted to conserve space. Please contact the author to obtain it.

6 Comparisons and Performance Analysis

6.1 Implementation Details

In order to verify the practical feasibility of the proposed method, we finished the proposed scheme in JAVA using the JPBC Library 2.0.0 [8]. Experiments were implemented on a computer with 8 GB RAM and an Intel i5-9400 2.90 GHz CPU. We used Type-A pairing, which is constructed by elliptic curve $y^2 = x^3 + x$. We

utilised Remix-Solidity IDE [2] to edit and compile the authorization contract and the revocation contract, and then used Ropsten [3] as the test network. After we have written the smart contract on the Remix-Solidity IDE, we will deploy the smart contract on the Ropsten test network through Metamask [1], Ropsten and Ethereum have the same consensus mechanism.

6.2 Comparison of Functional Characteristics

In this part, we compare our scheme with the proposed existing methods, which are shown in Table 4. We primarily compare the access revocation, type of hiding and other performance features. Comparative results illustrate that our scheme has certain advantages in different properties. As shown in Table 4, the scheme in this paper and the method in reference [7,11,14] achieve full policy hiding, whereas other schemes [10,13,15,24,28,29] only achieve partial policy hiding. Also, in these schemes that achieve full policy hiding, only the proposed scheme and the scheme [11] are based on blockchain. However, scheme [11] is constructed based on COBG, and in the plaintext recovery stage, the more attributes of the user, the lower the decryption efficiency. Additionally, only the scheme in this paper and the method in reference [7,14] are wildcard constructions. Among these wildcard-based schemes, only our method is implemented using asymmetric pairing and supports access revocation.

Table 4. Properties comparison

Scheme	Type of hiding	Asymmetric pairing	Wildcard	Blockchain	Access revocation
[15]	Partially hidden	✓	×	×	×
[29]	Partially hidden	✓	×	×	×
[24]	Partially hidden	×	×	×	×
[13]	Partially hidden	×	×	×	✓
[11]	Fully hidden	×	×	✓	×
[14]	Fully hidden	×	✓	×	×
[7]	Fully hidden	×	✓	×	×
[10]	Partially hidden	×	×	×	×
[28]	Partially hidden	×	×	×	✓
Our scheme	Fully hidden	✓	✓	✓	✓

6.3 Deployment Cost and Operating Cost

Some energy must be expended to deploy smart contracts on Ethereum and run some functions in the smart contract. Ethereum uses a unit called “gas” to describe how much energy it takes to perform an operation, such as deploying a smart contract or run some functions. The more complex the task, the more gas it consumes.

Table 5 shows the gas paid for some operations. We can see that deploying the contract costs relatively more, while executing some ABIs cost relatively little. Please note that since the Ropsten test network is used in the scheme, the gas values obtained are only experimental values and cannot represent the real situation. In a real system, we can reduce gas values by using a consensus mechanism with high consensus efficiency.

Table 5. Gas cost for some operations

Blockchain operation	Gas
Deploy the authorization contract	286352
Deploy the revocation contract	288549
Access revocation	29394
Upload ciphertext information	25984

Table 6. Notations for comparison

Notation	Description
$\|G_i\|$	Length of element in G_1, G_2 and G_T
$\|Z_p\|$	Length of element in Z_p
E_i	Exponentiation in G_1, G_2 and G_T
P	Pairing computation
l	The number of rows of the LSSS
s	The number of attributes of DU
c	The number of the attribute categories
ω	The number of wildcards
W	All minimal rowsets
k	The size of W

6.4 Theoretical Results

We now consider comparing the storage and computational costs of some of the studies in Table 4 that are more relevant to the content of this paper with the solution in this paper. Table 7 is the storage cost, and Table 8 is the computation cost. Related notations are described in Table 6. We compare the PK, CT and SK sizes of these schemes in Table 7. Table 7 shows that the PK length in [28] is the shortest; compared to [7], the proposed protocol achieves a shorter PK. The storage costs of [7] and the proposed method depend on the number of wildcards in the policy made by DO, which typically requires a shorter storage overhead than [10,13,28]. Obviously, the storage overhead of the three aspects of the scheme in this paper is smaller than [7]. Therefore, the storage cost required by the proposed scheme is minimal.

In Table 8, encryption time, secret key generation time and decryption time in schemes [10,13,28] are all related to the number of user attributes. The more attributes, the greater the computational overhead. The computational costs of [7] and the scheme of this paper are only related to the number of wildcards. In real systems, the number of wildcards will generally be significantly less than the number of user attributes, especially in attribute-sensitive systems. During decryption, pairing operations in these schemes increase with the number of attributes or wildcards, resulting in a long decryption time and a large computational cost for the decryption algorithms. The number of pairing calculations for the scheme in this paper is certain in the decryption phase and is thus more efficient. As a result, compared with other schemes in the table, the scheme in this paper has the highest efficiency.

Table 7. Storage cost

Scheme	Size of PK	Size of CT	Size of SK
[13]	$7\|G_1\| + \|G_T\|$	$(5l+2)\|G_1\| + 2\|G_T\|$	$(4s+2)\|G_1\|$
[10]	$5\|G_1\| + \|G_T\|$	$(s+2)\|G_1\|$	$(15l+10)\|G_1\| + 2\|G_T\|$
[28]	$4\|G_1\| + \|G_T\|$	$(c+3)\|G_1\|$	$(4l+2)\|G_1\| + 2\|G_T\|$
[7]	$(\omega+5)\|G_1\| + \|G_T\|$	$(\omega+4)\|G_1\| + \|G_T\|$	$(\omega+4)\|G_1\|$
Ours	$(\omega+2)\|G_1\| + \|G_T\|$	$(\omega+1)\|G_1\| + \|G_T\|$	$2\|G_1\| + \omega\|Z_p\|$

Table 8. Computational cost

Scheme	Computation cost of Enc	Computation cost of SK	Computation cost of Dec
[13]	$(8l+3)E_1 + 2E_T$	$(7c+3)E_1$	$2kE_T + (3k+1)P$
[10]	$(30s+15)E_1 + 2E_T$	$(2s+3)E_1$	$10sE_1 + sE_T + (10s+15)P$
[28]	$(6l+2)E_1 + 2E_T$	$(c+3)E_1$	$(k+l)E_T + 2(k+l+1)P$
[7]	$(\omega+5)E_1 + E_T$	$(\omega+4)E_1$	$(\omega+4)P$
Ours	$(2\omega+1)E_1 + E_T$	$(\omega+1)E_1$	$\omega E_1 + 2P$

6.5 Experimental Results

Because the proposed scheme uses blockchain, its efficiency is restricted by the blockchain. In the proposed scheme, we store the ciphertext address on Ethereum to reduce the cost of storing the entire ciphertext on the blockchain and to improve system scalability. The efficiency of computing on the blockchain depends on the consensus algorithm it uses. The blockchain with high consensus algorithm efficiency will execute faster. Thus, we primarily focus on the efficiency of encryption algorithms and decryption algorithms, only comparing the metrics of our scheme with those of the schemes proposed in [7,10,28]. To facilitate

display on the coordinate graph, we use attributes to describe the abscissa and evaluate the worst-case efficiency of the proposed scheme. Comparative results are shown in Figs. 2 and 3. The experimental data is the average of 20 runs on the computer.

In the encryption stage, the comparison result of the calculation cost between our study and other schemes is shown in Fig. 2. Obviously, the encryption time of the proposed scheme and other schemes is related to the number of attributes. The more attributes, the longer the encryption time. In terms of encryption efficiency, the encryption efficiency of the proposed scheme is much higher than that of scheme [10,28] and slightly lower than that of scheme [7], but the encryption time cost of our study and scheme [7] is not much different, and the growth rate of encryption time is basically the same.

It can be seen from the Fig. 3 that although the decryption time of each scheme increases with the increase of the number of user attributes, compared with the schemes in [7,10,28], the proposed scheme has a lower growth rate and the efficiency of this scheme is also the highest. Especially when the number of attributes in the system is large, the user decryption experience in the proposed scheme is the best. Due to the large computational cost of pairing operation, and the pairing operation in this paper is fixed, the scheme in this paper has high efficiency in the decryption stage.

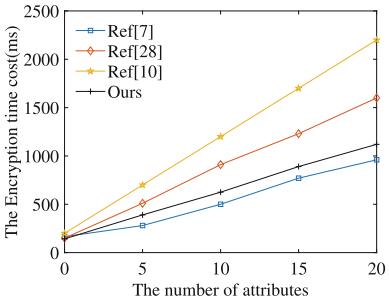

Fig. 2. Ciphertext generation stage

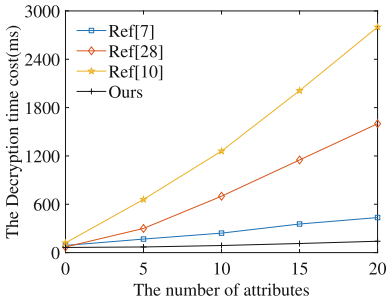

Fig. 3. Decryption phase

7 Conclusions

In this study, we built a blockchain-based ciphertext policy hiding access control scheme, which achieves distributed and reliable access control, and ensures user privacy and security. In addition, we use the revocation function by maintaining a revocation list through smart contracts, which protects users' private keys from abuse. Theoretical analysis and procedural data show that the our study is safe and effective. Because the efficiencies of operations using blockchain primarily depend on the consensus algorithm, we plan to consider using Ethereum 2.0 to implement the proposed scheme in future work to improve system throughput.

References

1. Metamask (2022). https://metamask.io
2. Remix ide for Ethereum smart contract programming (2022). https://remix.ethereum.org
3. Ropsten Testnet Explorer (2022). https://ropsten.etherscan.io
4. Bertrand, Y., Boudaoud, K., Riveill, M.: What do you think about your company's leaks? A survey on end-users perception toward data leakage mechanisms. Front. Big Data, 38 (2020). https://doi.org/10.3389/fdata.2020.568257
5. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: 2007 IEEE Symposium on Security and Privacy (SP 2007), pp. 321–334. IEEE (2007)
6. Butun, I., Österberg, P.: A review of distributed access control for blockchain systems towards securing the internet of things. IEEE Access **9**, 5428–5441 (2020)
7. Chen, Y., et al.: Efficient attribute-based data sharing scheme with hidden access structures. Comput. J. **62**(12), 1748–1760 (2019)
8. De Caro, A., Iovino, V.: jPBC: Java pairing based cryptography. In: 2011 IEEE Symposium on Computers and Communications (ISCC), pp. 850–855. IEEE (2011)
9. Ding, S., Cao, J., Li, C., Fan, K., Li, H.: A novel attribute-based access control scheme using blockchain for IoT. IEEE Access **7**, 38431–38441 (2019)
10. Gan, T., Liao, Y., Liang, Y., Zhou, Z., Zhang, G.: Partial policy hiding attribute-based encryption in vehicular fog computing (2021)
11. Gao, S., Piao, G., Zhu, J., Ma, X., Ma, J.: TrustAccess: a trustworthy secure ciphertext-policy and attribute hiding access control scheme based on blockchain. IEEE Trans. Veh. Technol. **69**(6), 5784–5798 (2020)
12. Hao, J., Huang, C., Ni, J., Rong, H., Xian, M., Shen, X.S.: Fine-grained data access control with attribute-hiding policy for cloud-based IoT. Comput. Netw. **153**, 1–10 (2019)
13. Hu, G., Zhang, L., Mu, Y., Gao, X.: An expressive "test-decrypt-verify" attribute-based encryption scheme with hidden policy for smart medical cloud. IEEE Syst. J. **15**(1), 365–376 (2020)
14. Jin, C., Feng, X., Shen, Q.: Fully secure hidden ciphertext policy attribute-based encryption with short ciphertext size. In: Proceedings of the 6th International Conference on Communication and Network Security, pp. 91–98 (2016)
15. Lai, J., Deng, R.H., Li, Y.: Expressive CP-ABE with partially hidden access structures. In: Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security, pp. 18–19 (2012)
16. Li, Q., Zhang, Y., Zhang, T., Huang, H., He, Y., Xiong, J.: HTAC: fine-grained policy-hiding and traceable access control in mHealth. IEEE Access **8**, 123430–123439 (2020)
17. Liu, S., Yu, J., Hu, C., Li, M.: Traceable multiauthority attribute-based encryption with outsourced decryption and hidden policy for CIoT. Wirel. Commun. Mob. Comput. **2021**, 16 (2021)
18. Phuong, T.V.X., Yang, G., Susilo, W.: Hidden ciphertext policy attribute-based encryption under standard assumptions. IEEE Trans. Inf. Forensics Secur. **11**(1), 35–45 (2015)
19. Rana, S., Mishra, D.: Efficient and secure attribute based access control architecture for smart healthcare. J. Med. Syst. **44**(5), 1–11 (2020)
20. Saini, A., Zhu, Q., Singh, N., Xiang, Y., Gao, L., Zhang, Y.: A smart-contract-based access control framework for cloud smart healthcare system. IEEE Internet Things J. **8**(7), 5914–5925 (2020)

21. Shafeeq, S., Alam, M., Khan, A.: Privacy aware decentralized access control system. Futur. Gener. Comput. Syst. **101**, 420–433 (2019)
22. Wang, S., Zhang, Y., Zhang, Y.: A blockchain-based framework for data sharing with fine-grained access control in decentralized storage systems. IEEE Access **6**, 38437–38450 (2018)
23. Xiong, H., Yang, M., Yao, T., Chen, J., Kumari, S.: Efficient unbounded fully attribute hiding inner product encryption in cloud-aided WBANs. IEEE Syst. J. **16**, 5424–5432 (2021)
24. Xiong, H., Zhao, Y., Peng, L., Zhang, H., Yeh, K.H.: Partially policy-hidden attribute-based broadcast encryption with secure delegation in edge computing. Futur. Gener. Comput. Syst. **97**, 453–461 (2019)
25. Xu, G., Li, H., Dai, Y., Yang, K., Lin, X.: Enabling efficient and geometric range query with access control over encrypted spatial data. IEEE Trans. Inf. Forensics Secur. **14**(4), 870–885 (2018)
26. Yan, X., He, G., Yu, J., Tang, Y., Zhao, M.: Offline/online outsourced attribute-based encryption with partial policy hidden for the internet of things. J. Sens. **2020** (2020)
27. Zeng, P., Zhang, Z., Lu, R., Choo, K.K.R.: Efficient policy-hiding and large universe attribute-based encryption with public traceability for internet of medical things. IEEE Internet Things J. **8**(13), 10963–10972 (2021)
28. Zhang, W., Zhang, Z., Xiong, H., Qin, Z.: PHAS-HEKR-CP-ABE: partially policy-hidden CP-ABE with highly efficient key revocation in cloud data sharing system. J. Ambient Intell. Human. Comput. **13**, 1–15 (2021)
29. Zhang, Y., Zheng, D., Deng, R.H.: Security and privacy in smart health: efficient policy-hiding attribute-based access control. IEEE Internet Things J. **5**(3), 2130–2145 (2018)
30. Zhang, Y., Yutaka, M., Sasabe, M., Kasahara, S.: Attribute-based access control for smart cities: a smart-contract-driven framework. IEEE Internet Things J. **8**(8), 6372–6384 (2020)
31. Zhang, Z., Zhang, W., Qin, Z.: A partially hidden policy CP-ABE scheme against attribute values guessing attacks with online privacy-protective decryption testing in IoT assisted cloud computing. Futur. Gener. Comput. Syst. **123**, 181–195 (2021)