Hindawi
Security and Communication Networks
Volume 2021, Article ID 4132597, 16 pages
https://doi.org/10.1155/2021/4132597

Research Article

A Blockchain-Based CP-ABE Scheme with Partially Hidden Access Structures

Yang Ba,[1] Xuexian Hu ,[1] Yue Chen,[1] Zenghang Hao,[1] Xuewei Li,[2] and Xincheng Yan[3]

[1]*State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou 450001, China*
[2]*Kunming Audit Center, Kunming 650001, China*
[3]*State Key Laboratory of Space Medicine Fundamentals and Application, Beijing 100094, China*

Correspondence should be addressed to Xuexian Hu; xuexian_hu@hotmail.com

Received 27 August 2021; Accepted 12 October 2021; Published 8 November 2021

Academic Editor: Qi Jiang

Data sharing has become a key technology to break down data silos in the big data era. Ciphertext-policy attribute-based encryption (CP-ABE) is widely used in secure data-sharing schemes to realize flexible and fine-grained access control. However, in traditional CP-ABE schemes, the access structure is directly shared along with the ciphertext, potentially leading to users' private information leakage. Outsourcing data to a centralized third party can easily result in privacy leakage and single-point bottlenecks, and the lack of transparency in data storage and sharing casts doubts whether users' data are safe. To address these issues, we propose a blockchain-based CP-ABE scheme with partially hidden access structures (BCP-ABE-PHAS) to achieve fine-grained access control while ensuring user privacy. First, we propose an efficient CP-ABE scheme with partially hidden access structures, where the ciphertext size is constant. To assist data decryption, we design a garbled Bloom filter to help users quickly locate the position of wildcards in the access structure. Then, to improve storage efficiency and system scalability, we propose a data storage scheme that combines blockchain technology and the interplanetary file system, ensuring data integrity. Finally, we employ smart contracts for a transparent data storage and sharing process without third-party participation. Security analysis and performance evaluation show that the proposed BCP-ABE-PHAS scheme can preserve policy privacy with efficient storage and low computational overhead.

1. Introduction

Cloud computing promotes the aggregation of storage and computational resources and has a tremendous market value. However, when data owners outsource data to cloud services, they lose control of their data, and their private information is at risk of leakage [1]. Recently, data security incidents have occurred frequently, and such events undermine users' confidence in data security and raise concerns regarding cloud storage.

In 2005, Sahai and Waters [2] proposed attribute-based encryption (ABE) to achieve fine-grained access control. The ABE scheme is mainly categorized into ciphertext-policy ABE (CP-ABE) [3] and key-policy ABE (KP-ABE) [4]. In the KP-ABE scheme, the secret key and ciphertext are associated with the access structure (or access policy) and attribute set, respectively. In this case, the ciphertext can only be decrypted when the attribute set satisfies the access policy. Contrarily, in the CP-ABE scheme, the ciphertext and secret key are associated with the access policy and attribute set, respectively.

The CP-ABE scheme features fine-grained access control and one-to-many secure data sharing. However, in the traditional CP-ABE scheme, the access policy is directly shared along with the ciphertext. Consequently, anyone can get this access policy while obtaining the ciphertext; however, the access policy may contain the user's sensitive information.

Consider a scenario in which a patient with a social security number (SSN) 123-456-789 wants to outsource his (or her) health data to the cloud and establish an access policy, as shown in Figure 1(a). This patient designs an

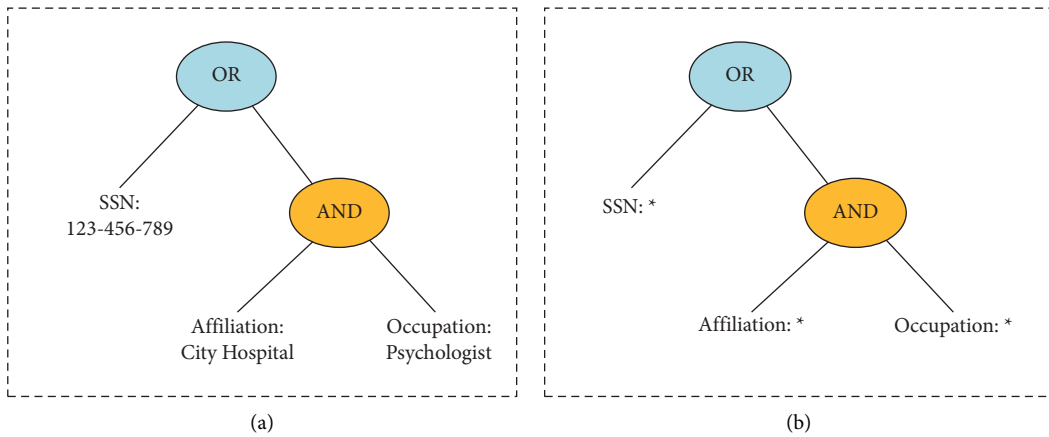

Figure 1: (a) Access structure. (b) Partially hidden access structure.

access policy based on which only this patient or the psychologist at the city hospital can access the data. If the patient uses the traditional CP-ABE scheme to send the encrypted data and access policy to the cloud, anyone with access to this cloud can obtain the patient's access policy. Thus, the data security of this patient, who is suffering from psychological problems, is undoubtedly threatened.

The most effective way to protect a user's access policy information is to hide the attribute information. Policy hiding involves fully and partially hiding. In the CP-ABE scheme, fully hidden access policies imply that no attribute information in the access policy is revealed, and partially hidden access policies imply that only sensitive attribute values are hidden. As shown in Figure 1(b), the partially hidden access policy is expressed as (SSN: $*$ OR (Affiliation: $*$ AND Occupation: $*$)), where the attribute values that may expose a user's information are hidden. A tradeoff is obtained between the efficiency of the CP-ABE scheme and the fully hidden access structure using a partially hidden access structure embedded in the CP-ABE scheme to reduce computational costs [5].

Additionally, centralized storage architectures are vulnerable to various network attacks such as single point of attack, man-in-the-middle attack, and distributed denial-of-service attack [6, 7]. Owing to such attacks, data owners may lose control of their data. Because blockchain technology is transparent, decentralized, and unforgeable, blockchain-based data storage and sharing schemes have been proposed to resist such attacks. Blockchain is an append-only distributed database, so large-scale data can quickly bloat the blockchain and make it expensive and inefficient to scale. To alleviate the storage pressure of the blockchain, we propose a storage scheme that combines blockchain technology and the interplanetary file system (IPFS) [8].

Therefore, we propose a blockchain-based CP-ABE scheme with a partially hidden access structure (BCP-ABE-PHAS) to realize secure data storage and sharing. Our main contributions are summarized as follows:

(1) We propose a CP-ABE scheme with partially hidden access structures to achieve fine-grained access control and ensure user privacy. Moreover, to assist data decryption, we design a garbled Bloom filter (GBF) to locate the position of wildcards in the access policy.

(2) To ensure data integrity and improve system scalability, we adopt a storage scheme that combines blockchain technology and the IPFS, in which the real ciphertext is stored in the IPFS, and meanwhile, the access policy is stored on the blockchain.

(3) We employ smart contracts to achieve automated and trusted access control, where the entire data storage and sharing process is transparent without third-party participation.

(4) Security analysis and performance evaluation show that the proposed scheme can achieve effective privacy preservation without incurring considerable overhead.

The remainder of this paper is organized as follows. In Section 2, we introduce the related work. In Section 3, preliminaries are described. We then present the system architecture and security model in Section 4, followed by the detailed construction of the proposed scheme in Section 5. The security analysis and the performance evaluation are performed in Section 6, and the conclusions are presented in Section 7.

2. Related Work

Bethencourt et al. [3] proposed the first CP-ABE scheme. This scheme allows data owners to specify a fine-grained access policy for their data to realize secure data sharing. However, an access policy may contain a user's sensitive information which is attached to the ciphertext as a plaintext, causing privacy leakage [9].

To address this problem, some schemes that hide the access policy have been proposed. For example, Nishide et al. [10] proposed a CP-ABE scheme with hidden access policies. They proposed two schemes in which only attribute values are hidden using AND gates on multivalued attributes with wildcards. Based on this scheme [10], Li et al. [11] implemented user accountability while hiding the access policy.

Phuong et al. [12] proposed two CP-ABE schemes with a hidden access policy. In this case, the access structure employs AND gates on positive and negative attributes with wildcards, and the ciphertext length is constant. Although these schemes are secure and efficient, AND-based access policies are limited in terms of expressiveness.

Thus, to facilitate a more expressive access policy, Lai et al. [13] proposed a partially hidden CP-ABE scheme that supports linear secret-sharing scheme-based access policy. Based on this scheme [13], Zhang et al. [14] proposed a scheme that can support a large attribute universe. However, these schemes are built using composite-order bilinear groups; thus, their efficiency is low. Katz et al. [15] first proposed the inner-product predicate encryption. However, the "superpolynomial blowup" problem makes the CP-ABE schemes that use the attribute-hiding IPE to construct a fully hidden access policy very inefficient [16]. Hur [17] proposed a CP-ABE scheme that can support any monotonous access policy. In this case, the access policy is hidden by attribute remapping, and most decryption operations are delegated to the cloud storage center to considerably reduce the requester's computational overhead.

Because blockchain technology is decentralized, tamperproof, and transparent, it is widely used in secure data sharing and access control schemes. Based on inner-product predicate encryption [15], Gao et al. [18] proposed a trustworthy secure CP-ABE scheme with a fully hidden access policy based on blockchain technology. This scheme combines inner-product encryption and homomorphic encryption to hide access policies and uses smart contracts to store the generated proof on the blockchain permanently. Zhang et al. [19] proposed an access control for the Internet of Things (IoT) based on the smart contract which consists of the judge contract, access control contract, and register contract to achieve intelligent and efficient access control. Additionally, Xu et al. [20] proposed a blockchain-based smart healthcare system for large-scale health data privacy preservation. This system uses digital envelope technology to verify the confidentiality of information; however, it can only support one-to-one secure transmission, which does not satisfy the requirements of users who simultaneously employ multiple third parties to provide services. In the IoT environment, Xu et al. [21] and Novo [22] adopted the blockchain technology to realize secure data sharing and access control; however, these schemes do not satisfy large-scale storage and privacy protection requirements.

3. Preliminaries

In this section, we introduce some basic knowledge associated with our BCP-ABE-PHAS.

3.1. Bilinear Map. Let $\mathbb{G}$ and $\mathbb{G}_T$ be multiplicative cyclic groups of prime order q. A bilinear mapping is a function $e: \mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G}_T$ which has the following properties:

(1) Bilinearity: $\forall u, v \in \mathbb{G}$ and $\forall a, b \in \mathbb{Z}_q^*$, there exists $e(u^a, v^b) = e(u, v)^{ab}$

(2) Nondegeneracy: there exists $g \in \mathbb{G}$ such that $e(g, g) \neq 1$

(3) Computability: $\forall u, v \in \mathbb{G}$, $e(u, v)$ can be effectively computed

3.2. Blockchain. Blockchain is an append-only data structure in a peer-to-peer network environment, where data blocks are connected chronologically in a chain and the data in a blockchain are assured to be tamperproof, unforgeable, and traceable using cryptography [23]. As shown in Figure 2, a block comprises the block header and block body. The block header consists of four components: (1) PreBkHash, which is the digest of the previous block; (2) TS, which is the timestamp of the block creation; (3) nonce, which is the consensus proof computed by miners and guarantees the consensus of the block; (4) Merkle root, which is the root hash of the Merkle hash tree. The block body stores transaction details.

The concept of smart contracts was first proposed by Szabo [24]. A smart contract is a program that contains code (its function) and data (its state). Smart contracts are used in Ethereum blockchain [25].The contract address is usually given when a contract is deployed to the blockchain. Contract address is the address to a collection of codes on the blockchain that executes functions. These functions of a contract address are executed when a transaction is made to the contract address. Once a smart contract is deployed in the network, it can run as programmed without human intervention.

3.3. Bloom Filter. The Bloom filter is a space-efficient probabilistic data structure used to determine whether an element is contained in a specific set [26]. The Bloom filter is an m-bit array that can represent a set S of maximum n elements. The Bloom filter has k independent hash functions $H = (h_1, \ldots, h_k)$, where $h_i: \{0,1\}^* \mapsto [1, m]$ and $1 \leq i \leq k$ indicates that the value generated by the hash function is uniformly distributed in $[1, m]$. Herein, a Bloom filter with parameters (m, n, k, H) is represented as $(m, n, k, H)0BF$, a Bloom filter encoding the set S is represented as BF_S, and the value at index i in BF_S is represented as $BF_S[i]$.

First, all bits in the Bloom filter are set to 0. As shown in Figure 3, when we add the element x in the set $S = \{x, y\}$ to the Bloom filter, we set $BF_S[h_i(x)] = 1$ for $1 \leq i \leq 3$. When we verify the existence of an element y in set S, if $BF_S[h_i(y)] = 0$ exists for $1 \leq i \leq 3$, this proves that $y \notin S$; otherwise, $y \in S$ with a high probability.

The Bloom filter yields false positives; in other words, it yields an element that does not belong to the set S, but the corresponding position values are all 1. As shown in Figure 3, the element z does not belong to the set $\{x, y\}$; however, $BF_S[h_i(z)] = 1$ for $1 \leq i \leq 3$. According to Bose et al. [27], the false positive probability is negligible if we select the optimal k and m values.

3.4. Secret Sharing. Secret sharing technology is an important aspect of cryptography research. For example, Shamir [28] proposed a (k, n)-threshold secret-sharing scheme. The

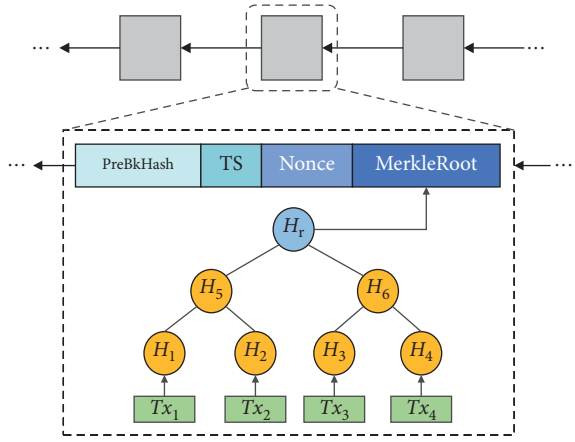

Figure 2: Blockchain structure.

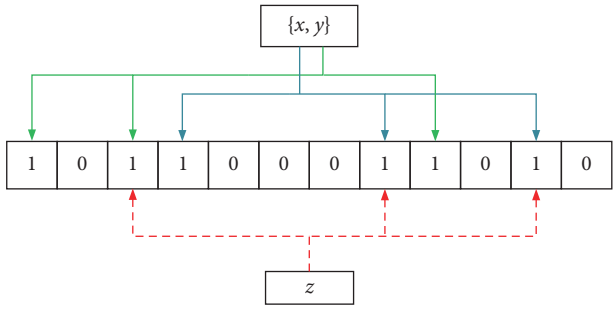

Figure 3: Bloom filter.

basic concept of this scheme is that the secret s to be shared is divided and distributed to n participants. The secret can be recovered in the case of minimum k participants; otherwise, the secret cannot be recovered. When $k = n$, secret sharing can be obtained using the $\oplus$ (XOR) operation. Randomly generate $n-1$ bit strings $r_1, \ldots, r_{n-1}$ with the same length as the secret s, and calculate $r_n = r_1 \oplus \ldots \oplus r_{n-1} \oplus s$; each of r_i is a part of the secret s. Finally, the secret s can be obtained by computing $r_1 \oplus \ldots \oplus r_n$.

3.5. Attribute Vector. As shown in Figure 4, we define two attribute vectors $\overline{W} = (\overline{w}_1, \ldots, \overline{w}_L) \in \Sigma_*^L$ and $W = (w_1, \ldots, w_L) \in \Sigma^L$, where $\Sigma \subset \mathbb{Z}_q^*$ and $\Sigma_* = \Sigma \cup \{*\}$. Here, $\overline{W}$ contains the wildcard $*$, and $J = \{j_1, \ldots, j_n\} \subset \{1, \ldots, L\}$ represents the set of wildcard positions in $\overline{W}$.

The decryption algorithm discussed in this paper employs the following polynomial identity, where w_i is the attribute value at position i in the attribute vector.

$$\sum_{i=1}^{L} \prod_{j \in J} (i-j) w_i = \sum_{i=1, i \notin J}^{L} \prod_{j \in J} (i-j) w_i. \tag{1}$$

We use Viète's formulas [29] to construct the polynomial $\prod_{j \in J}(i-j) = x^n + a_{n-1}x^{n-1} + \cdots + a_0$ in equation (1), and the coefficients are calculated as follows:

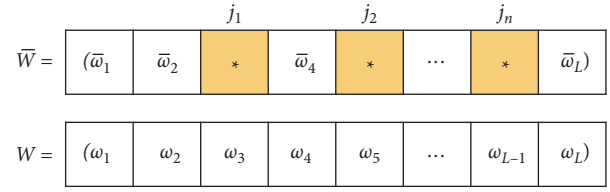

Figure 4: Attribute vector.

$$a_{n-k} = (-1)^{i-n} \prod_{1 \le i_1 < i_2 < \ldots < i_k \le n} j_{i_1} j_{i_2} \ldots j_{i_k}, 0 \le k \le n, \tag{2}$$

where $n = |J|$. Here, if J is clear, we can calculate the polynomial coefficients a_i. For example, when $J = \{j_1, j_2, j_3\}$, we can construct polynomial $(x - j_1)(x - j_2)(x - j_3)$ and calculate the coefficients:

$$\begin{aligned} a_0 &= -j_1 j_2 j_3, \\ a_1 &= j_1 j_2 + j_1 j_3 + j_2 j_3, \\ a_2 &= -(j_1 + j_2 + j_3), \\ a_3 &= 1. \end{aligned} \tag{3}$$

3.6. Decision Linear Assumption. Let $\mathbb{G}$ be a bilinear group of prime order q with a generator g. For any probabilistic polynomial-time (PPT) adversary $\mathcal{A}$, its advantage $Adv_{\mathcal{A}}(\lambda)$ in solving the decision linear (DLIN) problem [30] in $\mathbb{G}$ is

$$\begin{aligned} \mathrm{Adv}_{\mathcal{A}}(\lambda) = \Big| &\Pr\left[\mathcal{A}\left(g, g^a, g^b, g^{ac}, g^d, g^{b(c+d)} = 1\right)\right] \\ &- \Pr\left[\mathcal{A}\left(g, g^a, g^b, g^{ac}, g^d, g^r = 1\right)\right]\Big|, \end{aligned} \tag{4}$$

where the probability is taken over all possible choices of $a, b, c, d, r \in \mathbb{Z}_q^*$. We say that the DLIN assumption holds in $\mathbb{G}$ if there exists a negligible function $\varepsilon(\lambda)$ such that $Adv_{\mathcal{A}}(\lambda) < \epsilon$ for any PPT algorithm $\mathcal{A}$.

4. System Architecture and Security Model

4.1. System Architecture. The system architecture of the proposed scheme is shown in Figure 5. As illustrated, the system architecture involves five entities, i.e., attribute authority (AA), IPFS, data owner (DO), data user (DU), and blockchain.

AA: the AA manages all attributes in the system and assigns attributes to users. It is also responsible for generating public parameters and issuing secret keys based on the users' attributes. In this paper, the AA is fully trusted.

IPFS: the IPFS is a distributed file storage system based on content addressing. Note that there is no central server node in the IPFS; thus, it can avoid the risk of a single point of failure. The IPFS uses an encryption algorithm to calculate the hash value *hashipfs* of a file, and this *hashipfs* is used as the file's address. This approach reduces the repeated storage of files and ensures the integrity of files.

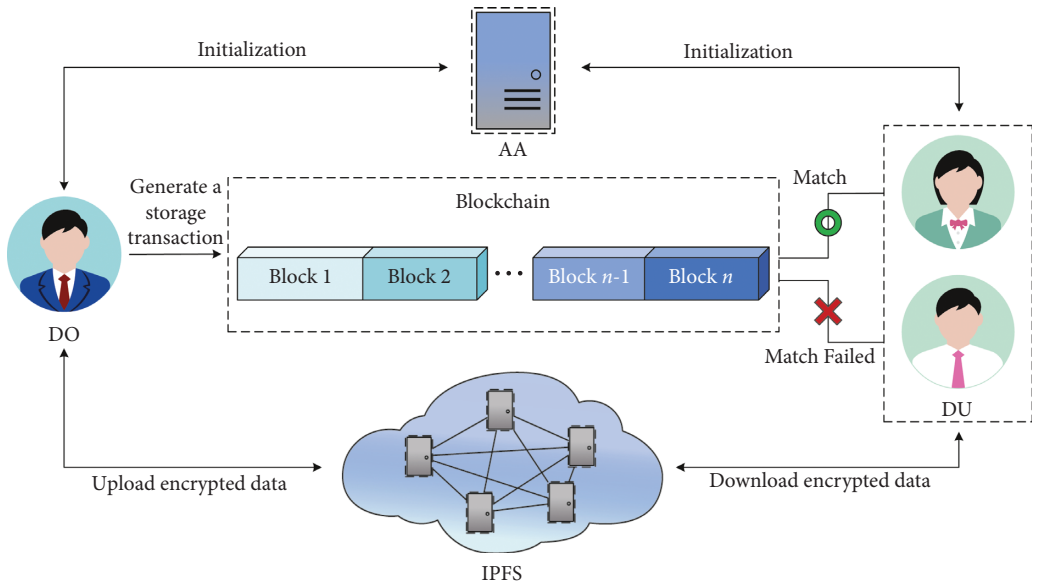

Figure 5: System architecture.

DO: the DO selects the *file* to be shared and creates a corresponding access policy. First, the DO encrypts the *file* using the symmetric key *aeskey* and stores the ciphertext *encfile* in the IPFS. Then, the proposed CP-ABE scheme is used to encrypt *aeskey* and generate the ciphertext CT. Finally, *hashipfs* and CT are stored on the blockchain using a smart contract.

DU: the DU sends a request to the AA, and the AA generates a secret key based on the attribute set of the DU. The DU obtains CT stored on the blockchain using the smart contract and decrypts CT based on its secret key. Here, if the attribute set satisfies the access structure set by the DO, then the DU can obtain the *file* from the IPFS using *aeskey* and *hashipfs*.

Blockchain: the blockchain is an append-only distributed database, where data are stored permanently and are tamperproof. To ensure secure data sharing and fine-grained access control, the DO only stores *hashipfs* and CT on the blockchain using smart contracts.

4.2. The Definition of the BCP-ABE-PHAS Scheme. Here, we present the definition of our scheme. This scheme mainly involves the following four algorithms:

(i) $Setup(1^\lambda)$: the Setup algorithm is executed by the AA. This algorithm takes security parameter 1^λ as the input and outputs the public parameters PK and master secret key MSK.

(ii) $KeyGen(PK, MSK, W)$: the KeyGen algorithm is executed by the AA. Here, PK, MSK, and W of the DU are taken as inputs, and the secret key SK_W associated with W is the output.

(iii) $Encrypt(PK, M, (\overline{W}, J))$: the Encrypt algorithm is executed by the DO. This algorithm comprises the BuildGBF and GenCT functions.

$BuildGBF(\overline{W}, J)$: this function takes an access policy $\overline{W}$ and the wildcard position set J as inputs and outputs GBF

$GenCT(PK, M, (\overline{W}, J))$: this function takes PK, a message M, an access policy $\overline{W}$, and the wildcard position set J as inputs and outputs ciphertext CT

(iv) $Decrypt(PK, SK_W, GBF, CT)$: the Decrypt algorithm is executed by the DU and comprises the QueryGBF and Dec functions.

$QueryGBF(W, GBF)$: this function takes an attribute vector W of the DU as the input and queries GBF to obtain the wildcard position set J

$Dec(PK, SK_W, CT, J)$: this function takes PK, SK_W, J, and CT as inputs and outputs M

Here, let $(PK, MSK) \leftarrow Setup(1^\lambda)$, $SK_W \leftarrow KeyGen(PK, MSK, W)$, and $(CT, GBF) \leftarrow Encrypt(PK, M, (\overline{W}, J))$. For correctness, we require the following conditions to hold:

(1) If the attribute vector W of the DU satisfies the access policy $\overline{W}$, then $M \leftarrow Decrypt(PK, SK_W, GBF, CT)$

(2) Otherwise, $Decrypt(PK, SK_W, GBF, CT)$ outputs a random message

4.3. Security Model

Definition 1. A CP-ABE scheme with the hidden access policy is semantically secure in the selective model if for all PPT adversaries $\mathcal{A}$,

$$\left|\Pr\left[Exp_{\mathcal{A}}(\lambda) = 1\right] - \frac{1}{2}\right| < \varepsilon(\lambda), \tag{5}$$

for some negligible function $\epsilon(\lambda)$. Based on [29], the security game $Exp_{\mathcal{A}}(\lambda)$ is described as follows:

Init: $\mathscr{A}$ selects two different challenge attribute vectors $\overline{W}_0, \overline{W}_1 \in \Sigma_*^L$, for at least one $\overline{w}_i \neq *$.

Setup: the challenger $\mathscr{B}$ runs $\text{Setup}(1^\lambda)$ algorithm, which outputs PK and MSK. It sends PK to $\mathscr{A}$ and keeps MSK to itself.

Query phase 1: $\mathscr{A}$ adaptively issues key queries for the attribute vector $W \in \Sigma^L$, under the restriction that $w_i \neq \overline{w}_{0i}$ and $w_i \neq \overline{w}_{1i}$. $\mathscr{B}$ runs $KeyGen(PK, MSK, W)$ algorithm to obtain SK_W and sends SK_W to $\mathscr{A}$.

Challenge: $\mathscr{A}$ submits two messages M_0, M_1 ($|M_0| = |M_1|$) and sends them to $\mathscr{B}$. Given $\overline{W}_0$ and $\overline{W}_1$, $\mathscr{B}$ randomly selects $\beta \in \{0, 1\}$ and encrypts M_β under $\overline{W}_\beta$. Finally, $\mathscr{B}$ sends CT_β to $\mathscr{A}$.

Query phase 2: query phase 2 is the same as query phase 1.

Guess: finally, $\mathscr{A}$ outputs its guess $\beta' \in \{0, 1\}$ for β. If $\beta' = \beta$, then return 1; else, return 0.

5. Construction of the BCP-ABE-PHAS

5.1. Setup Phase. In this paper, we use $U = \{att_1, \ldots, att_L\}$ to represent the attribute universe in the system. Here, $V_i = \{v_{i,1}, \ldots, v_{i,n_i}\}$ is the set of possible values of the i^{th} category attribute, where $n_i = |V_i|$. Thus, the user's attribute vector is $W = (w_1, \ldots, w_L)$, where $w_i \in V_i, 1 \leq i \leq L$. The access structure of the proposed scheme is $\overline{W} = (\overline{w}_1, \ldots, \overline{w}_L) = \wedge_{i \in I_W} \overline{w}_i$, where $I_W = \{i | 1 \leq i \leq L, \overline{w}_i \neq * \}$. If $\overline{w}_i = w_i$ or $\overline{w}_i = *$, we use $W \vDash \overline{W}$ to denote that the user's attribute vector W satisfies the access policy $\overline{W}$; otherwise, we use $W \nvDash \overline{W}$ to denote that the user's attribute vector W does not satisfy the access policy $\overline{W}$. Here, the wildcard $*$ in the access structure means "do not care." The upper bound of the wildcard in the access structure is defined as N, where $N \ll L$.

The setup phase is run by the AA. Here, $\mathbb{G}$ and $\mathbb{G}_T$ are the multiplicative cyclic groups of a large prime order q, g is a generator of $\mathbb{G}$, and e: $\mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G}_T$ is a bilinear map. The AA randomly chooses $\alpha, t_1, t_2, (x_1, \ldots, x_N) \in \mathbb{Z}_q$ and $V_0, U_1, \ldots, U_L \in \mathbb{G}$ and sets $\Omega_1 = e(g, V_0)^{\alpha t_1}$ and $\Omega_2 = e(g, V_0)^{\alpha t_2}$. Let $V_j = V_0^{x_j}$ for $j = 1, \ldots, N$. The AA also generates k independent hash functions $H = (h_1, \ldots, h_k)$. Therefore, the public parameters are expressed as follows:

$$PK = (e, \mathbb{G}, \mathbb{G}_T, g, q, \Omega_1, \Omega_2, g^\alpha, V_0, (x_1, \ldots, x_N), H, (U_1, \ldots, U_L)). \tag{6}$$

Additionally, the master secret key is expressed as follows:

$$MSK = (\alpha, t_1, t_2, (V_1, \ldots, V_N)). \tag{7}$$

5.2. Data Encryption Phase. The encryption phase is executed by the DO and involves three main parts, which are described in the following section.

5.2.1. IPFS Storage. The DO selects the *file* to be shared, generates the symmetric key *aeskey* using the Advanced Encryption Standard (AES), and encrypts the *file* using *aeskey* to generate the ciphertext *encfile*.

To relieve the pressure on blockchain storage, the proposed scheme stores *encfile* in the IPFS using the *ipfs add encfile* command, and the IPFS returns unique hash value *ipfshash* to retrieve *encfile*. Note that anyone can obtain the ciphertext *encfile* stored in the IPFS using the *ipfs get ipfshash* command.

5.2.2. Hidden Access Policy. The blockchain is public, all participants can obtain the data on the blockchain, so we need to hide the attribute information of the access policy. The access policy developed by the DO is $\overline{W} = (\overline{w}_1, \ldots, \overline{w}_L)$. Assume that the access policy $\overline{W}$ contains $n \leq N$ wildcards that occur at positions $J = \{j_1, \ldots j_n\}$.

When data are decrypted, determining the position of the wildcard symbols is essential; however, directly sending the set J may reveal the user's private information. Thus, to solve this problem, we adopt an efficient positioning algorithm based on the GBF. The GBF is a combination of a Bloom filter and secret sharing technology. Differing from traditional Bloom filters that use a bit array, the GBF uses an array of λ bits. The GBF can verify whether an attribute exists in the specified set and locate the position index of the attribute to realize the hidden set J and protect the user's private information. In addition to the probability of hash function collisions, the probability of string matching must be verified. Therefore, the false positive probability of the GBF is less than that of the traditional Bloom filter.

When the DO adds an element att_j, $j \in J$, to the GBF, the algorithm first uses the (k, k)-secret-sharing scheme to randomly generate $k - 1$ λ-bit strings $r_{1,j}, \ldots, r_{k-1,j}$ and sets $r_{k,j} = r_{1,j} \oplus \ldots \oplus r_{k-1,j} \oplus j$. Then, it hashes att_j with k independent hash functions $H = (h_1, \ldots, h_k)$ and obtains $h_1(att_j), \ldots, h_k(att_j)$, where $h_i(att_j)$ is uniformly distributed in $[1, m]$. Finally, generated $r_{i,j}$ is stored in the GBF based on the position index generated by $h_i(att_j)$.

When elements are further added to the GBF, if a certain position is already occupied by previously added elements, we reuse the share already stored in the GBF. As shown in Figure 6, when we add j_2 to the GBF, the hash value of $h_d(att_{j_2})$ is the same as the hash value of $h_i(att_{j_1})$. If we modify r_{i,j_1}, the previously added element j_1 cannot be restored; thus, we set $r_{d,j_2} = r_{i,j_1}$. The construction of the GBF is presented in Algorithm 1.

The DO constructs a GBF to hide the set J of wildcard positions based on Algorithm 1 and then uses Viète's

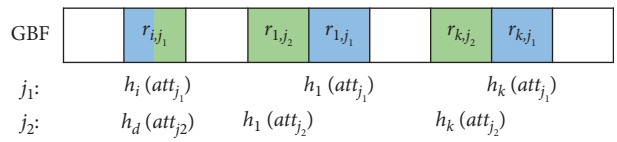

Figure 6: The garbled Bloom filter.

Input: set J, security parameter λ, m, and k hash functions $H = \{h_1, \ldots, h_k\}$
Output: garbled Bloom filter GBF

```
(1) GBF = new m-element array of bit strings;
(2) for i = 0 to m − 1 do
(3)    GBF[i] = NULL;//initialize the GBF with "NULL"
(4) end for
(5) for each x ∈ J do
(6)    emptySolt = −1, finalShare = x;
(7)    for i = 0 to k − 1 do
(8)       j = h_{i+1}(att_x);//get the index of the position
(9)       if GBF[j] == NULL then
(10)         if emptySolt == −1 then
(11)            emptySolt = j;
(12)         else
(13)            GBF[j]←{0,1}^λ;//get a new share
(14)            finalShare = finalShare ⊕ GBF[j];
(15)         end if
(16)      else
(17)         finalShare = finalShare ⊕ GBF[j];//reuse an existing share
(18)      end if
(19)   end for
(20)   GBF[emptySolt] = finalShare;//store the last share
(21) end for
(22) for i = 0 to m − 1 do
(23)   if GBF[i] == NULL then
(24)      GBF[i]←{0,1}^λ;//fill the position with random strings
(25)   end if
(26) end for
```

Algorithm 1: Build GBF.

formulas to compute $\{a_i\}_{1 \le i \le n}$. Here, $m = (\sum_{k=0}^{n} x_k a_k)^{-1}$, where $x_0 = 1$. It randomly chooses $r_1, r_2 \in \mathbb{Z}_q^*$. The DO then creates CT as

$$\begin{aligned} C_0 &= M\Omega_1^{r_1}\Omega_2^{r_2}, \\ C_1 &= g^{\alpha m r_1}, \\ C_2 &= g^{m r_2}, \\ C_3 &= \left(V_0 \prod_{i=1}^{L} U_i \prod_{k=1}^{n} \overline{w}_i^{(i-j_k)m}\right)^{r_1+r_2}, \end{aligned} \tag{8}$$

where M is $aeskey$. Therefore, the ciphertext is $CT = (C_0, C_1, C_2, C_3, GBF)$.

5.2.3. Blockchain Storage. A blockchain is an append-only distributed database that stores data on the blockchain permanently, which ensures that the data can be tamper-proof but increases the storage pressure on the blockchain. Therefore, in this paper, the ciphertext *encfile* is stored in the IPFS, and only *ipfshash* and CT are stored on the blockchain. To achieve secure data sharing and fine-grained access control, we employ smart contracts to ensure that the data storage and sharing process is open and transparent without third-party participation. Here, the public and private keys of the DO in the blockchain are represented by BPK_{DO} and BSK_{DO}, respectively.

Generally, the DO is the creator of the access control contract (ACC) who wants to share data with DUs. The ACC provides application binary interfaces (ABIs) to manage and implement access control. The ABIs of the ACC are presented in Table 1.

The DO creates and deploys the ACC and then obtains the contract address and ABIs. Furthermore, the DO sends a transaction to execute the *uploadfile* ABI of the ACC to upload the data on the blockchain (Algorithm 2).

5.3. Key Generation Phase. The key generation phase is executed by the AA. In this phase, the secret key is generated for the DU, the attribute vector of which is $W = (w_1, \ldots, w_L) \in \Sigma^L$. The AA randomly chooses $s \in \mathbb{Z}_q$ and sets $s_1 = t_1 + s$ and $s_2 = t_2 + s$. Then, the following algorithm is executed:

Table 1: ABIs of the ACC.

ABI	Permissions	Description
uploadfile()	Contract creator	The ABI uploads data on the blockchain
getfile()	Public	The ABI obtains the data stored on the blockchain
kill()	Contract creator	The ABI performs the *selfdestruct* operation to delete the ACC

Input: CT, *ipfshash*, *time*, BSK_{DO}, ACC address *addr*, and *uploadfile* ABI
Output: storage transaction $Tx_{storage}$
(1) Compute the message digest $MD = H(time, ipfshash, CT)$;
(2) Generate the signature $sign = sign_{BSK_{DO}}(MD)$;
(3) Generate a storage transaction $Tx_{storage} = \{time, ipfshash, CT, sign\}$;
(4) Send $Tx_{storage}$ according to *addr* and *uploadfile* ABI;
(5) **return** $Tx_{storage}$;

Algorithm 2: Generate a storage transaction.

$$
\begin{aligned}
K &= g^{\alpha s},\\
K_1 &= \left\{K_{1,0}, K_{1,1}, \ldots, K_{1,N}\right\}\\
&= \left\{V_0^{s_1}\prod_{i=1}^{L}U_i^{sw_i}, V_1^{s_1}\prod_{i=1}^{L}U_i^{siw_i}, \ldots, V_N^{s_1}\prod_{i=1}^{L}U_i^{si^Nw_i}\right\},\\
K_1' &= \left\{K_{1,0}', K_{1,1}', \ldots, K_{1,N}'\right\}\\
&= \left\{V_0^{\alpha s_2}\prod_{i=1}^{L}U_i^{\alpha sw_i}, V_1^{\alpha s_2}\prod_{i=1}^{L}U_i^{\alpha siw_i}, \ldots, V_N^{\alpha s_2}\prod_{i=1}^{L}U_i^{\alpha si^Nw_i}\right\}.
\end{aligned}
\tag{9}
$$

Therefore, the secret key of the DU is $SK = (K, K_1, K_1')$.

5.4. Data Decryption Phase. Data decryption is performed by the DU and mainly comprises the following three phases.

5.4.1. Obtain Data on the Blockchain. The DU sends a transaction to execute the *getfile* ABI of the ACC to obtain the data stored on the blockchain (Algorithm 3).

Therefore, the DU obtains *ipfshash* and CT stored on the blockchain.

5.4.2. QueryGBF. The DU obtains the data stored on the blockchain, where $CT = (C_0, C_1, C_2, C_3, GBF)$. As observed from the data encryption phase, obtaining J is the key to decrypting CT. Here, the DU obtains J according to Algorithm 4.

First, we determine whether the hash value of the attribute exists in the GBF. If the corresponding position of the attribute in the GBF is 0, the attribute must not be in J. When all GBF positions corresponding to the k hash values of the attribute are not empty, the DU must calculate the position index of the wildcard in the access policy using $\oplus$; if the calculated value is the same as the position index corresponding to the attribute vector of the DU, the attribute is present in J; otherwise, this attribute is not in J.

5.4.3. Data Decryption Phase. In this phase, the DU obtains the set of wildcard positions J and then uses Viète's formulas to compute $\{a_i\}_{1\le i\le n}$, where $m = (\sum_{k=0}^{n} x_k a_k)^{-1}$. The ciphertext can only be decrypted when the attributes of the DU can satisfy the access policy. Then, M can be computed as follows:

$$
M = C_0 \frac{e(K, C_3)}{e\left(C_1, \prod_{k=0}^{n} K_{1,k}^{a_k}\right) e\left(C_2, \prod_{k=0}^{n}\prod_{k=0}^{n}\left(K_{1,K}'\right)^{a_k}\right)}.
\tag{10}
$$

When the DU successfully decrypts the data, it obtains *aeskey*. Then, the DU obtains *encfile* stored in the IPFS using *ipfshash*. Subsequently, *aeskey* is used to decrypt *encfile* to obtain the *file* shared by the DO.

6. Security Analysis and Performance Evaluation

6.1. Correctness. In this section, we verify the correctness of the proposed scheme. When we use a decryption key that satisfies the given access policy, the Decrypt algorithm indeed returns the correct message.

$$
\begin{aligned}
e(K, C_3) &= e\left(g^{\alpha s}, \left(V_0\prod_{i=1}^{L}U_i^{\prod_{k=1}^{n}(i-j_k)\overline{w}_i/\sum_{m=0}^{n}x_m a_m}\right)^{r_1+r_2}\right)\\
&= e(g, V_0)^{\alpha s(r_1+r)}\prod_{i=1}^{L}e(g, U_i)^{\alpha s\overline{w}_i(r_1+r_2)\sum_{k=1}^{n}(i-j_k)/\sum_{m=0}^{n}x_m a_m},
\end{aligned}
\tag{11}
$$

Input: ACC address *addr*, *getfile* ABI, and BPK_{DO}
Output: verified result
(1) Obtain the data according to *addr* and *getfile* ABI;
(2) Compute the message digest $MD' = H(time, ipfshash, CT)$;
(3) Verify the signature $MD = Verify_{BPK_{DO}}(sign)$;
(4) **if** $MD' == MD$ **then**
(5) **return** True;
(6) **else**
(7) **return** False;
(8) **end if**

Algorithm 3: Obtain the data stored on the blockchain.

Input: garbled Bloom filter GBF, n, the number of attributes L, security parameter λ, m, and k hash functions $H = \{h_1, \ldots, h_k\}$
Output: set J
(1) J = new set of length n;
(2) **for** $x = 1$ to L **do**
(3) $recovered = \{0\}^{\lambda}$;
(4) **for** $i = 1$ to k **do**
(5) $j = h_i(att_x)$;
(6) **if** $GBF[j] == NULL$ **then**
(7) break;
(8) **else**
(9) $recovered = recovered \oplus GBF[j]$;
(10) **end if**
(11) **end for**
(12) **if** $recovered == x$ **then**
(13) $J.add(x)$;
(14) **end if**
(15) **end for**

Algorithm 4: QueryGBF.

where for equation (11), we use that $\sum_{k=0}^{n} i^k a_k = \prod_{k=1}^{n} (i - j_k)$,

$$
\begin{aligned}
e\left(C_1, \prod_{k=0}^{n} K_{1,k}^{a_k}\right) &= e\left(g^{\alpha r_1 / \sum_{m=0}^{n} x_m a_m}, \prod_{k=0}^{n}\left(V_k^{s_1} \prod_{i=1}^{L} U_i^{si^k w_i}\right)^{a_k}\right) \\
&= e(g, V_0)^{\frac{\alpha r_1 s_1 \prod_{k=0}^{n} x_k a_k}{\sum_{m=0}^{n} x_m a_m}} \prod_{i=1}^{L} e(g, U_i)^{\alpha s r_1 w_i \sum_{k=0}^{n} i^k a_k / \sum_{m=0}^{n} x_m a_m} \\
&= \Omega_1^{r_1} e(g, V_0)^{\alpha s r_1} \prod_{i=1}^{L} e(g, U_i)^{\alpha s r_1 w_i \sum_{k=0}^{n} i^k a_k / \sum_{m=0}^{n} x_m a_m} \\
e\left(C_2, \prod_{k=0}^{n} \left(K'_{1,k}\right)^{a_k}\right) &= e\left(g^{\frac{r_2}{\sum_{m=0}^{n} x_m a_m}}, \prod_{k=0}^{n}\left(V_k^{\alpha s_2} \prod_{i=1}^{L} U_i^{\alpha s i^k w_i}\right)^{a_k}\right) \\
&= e(g, V_0)^{\frac{\alpha r_2 s_2 \prod_{k=0}^{n} x_k a_k}{\sum_{m=0}^{n} x_m a_m}} \prod_{i=1}^{L} e(g, U_i)^{\alpha s r_2 w_i \sum_{k=0}^{n} i^k a_k / \sum_{m=0}^{n} x_m a_m} \\
&= \Omega_2^{r_2} e(g, V_0)^{\alpha s r_2} \prod_{i=1}^{L} e(g, U_i)^{\alpha s r_2 w_i \sum_{k=0}^{n} i^k a_k / \sum_{m=0}^{n} x_m a_m}.
\end{aligned}
\tag{12}
$$

Then, we have

$$e\left(C_1, \prod_{k=0}^{n} K_{1,k}^{a_k}\right) e\left(C_2, \prod_{k=0}^{n} \left(K_{1,k}'\right)^{a_k}\right) = \Omega_1^{r_1} \Omega_2^{r_2} e(g, V_0)^{\alpha s (r_1 + r_2)} \prod_{i=1}^{L} e(g, U_i)^{\frac{\alpha s w_i (r_1 + r_2) \sum_{k=0}^{n} i^k a_k}{\sum_{m=0}^{n} x_m a_m}}. \tag{13}$$

If the secret key of the DU is valid, then $w_i = \overline{w}_i$, $i \notin \{j_1, \ldots, j_n\}$. Thus,

$$C_0 \frac{e(K, C_3)}{e\left(C_1, \prod_{k=0}^{n} K_{1,k}^{a_k}\right) \cdot e\left(C_2, \prod_{k=0}^{n} \left(K_{1,k}'\right)^{a_k}\right)} = \frac{M \Omega_1^{r_1} \Omega_2^{r_2}}{\Omega_1^{r_1} \Omega_2^{r_2}} = M. \tag{14}$$

6.2. Security Analysis

Theorem 1. *The proposed BCP-ABE-PHAS scheme is semantically secure in the selective model assuming that the DLIN assumption holds in group* $\mathbb{G}$.

Proof. Assume there exists a PPT adversary $\mathscr{A}$ that can break the selective semantic security. We then build an algorithm $\mathscr{B}$ that uses $\mathscr{A}$ to solve the DLIN problem in $\mathbb{G}$.

Here, the challenger selects a bilinear group $\mathbb{G}$ of prime order q and a generator $g \in \mathbb{G}$, as well as the group $\mathbb{G}_T$ and a bilinear map e: $\mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G}_T$. Then, the challenger randomly chooses five values $a, b, c, d, r \in \mathbb{Z}_q^*$ and computes $Z_0 = g^{b(c+d)}$ and $Z_1 = g^r$. The challenger randomly chooses $\beta \in \{0, 1\}$ and sends the tuple $(g, g^a, g^b, g^{ac}, g^d, Z_\beta)$ to $\mathscr{B}$. Note that the goal of $\mathscr{B}$ is to guess β with a probability greater than 1/2. To generate a guess, $\mathscr{B}$ interacts with $\mathscr{A}$ in the following selective semantic security experiment.

Init: $\mathscr{A}$ chooses two challenge attribute vectors $\overline{W}_0 \in \Sigma_*^L$ and $\overline{W}_1 \in \Sigma_*^L$. Here, the wildcard position sets are denoted as J_0 and J_1, respectively. Then, $\mathscr{B}$ randomly chooses $\gamma = \{0, 1\}$, where $\overline{W}_\gamma = (\overline{w}_1, \ldots, \overline{w}_L)$.

Setup: $\mathscr{B}$ chooses $N \le L$, which is the upper bound number of wildcards. Then, $\mathscr{B}$ chooses $v_0, u_1, \ldots, u_L \in \mathbb{Z}_q$ uniformly at random and sets the following:

$$\begin{aligned} x_j &= \frac{\sum_{i=1}^{L} i^{ju_i}}{\sum_{i=1}^{L} u_i} \quad \text{for } j = 0, \ldots, N, \\ V_j &= \left(g^b\right)^{x_j v_0} g^{-\sum_{i=1}^{L} i^{ju_i}} \quad \text{for } j = 0, \ldots, N, \\ U_i &= g^{\frac{u_i}{\overline{w}_i}} \quad \text{for } i = 1, \ldots, L. \end{aligned} \tag{15}$$

Here, $\mathscr{B}$ randomly chooses $\sigma_1, \sigma_2, \sigma_3 \in \mathbb{Z}_q$ and computes $\Omega_1 = e(g^a, V_0)^{\sigma_1 - \sigma_2}$ and $\Omega_2 = e(g^{\sigma_3} \cdot (g^a)^{-\sigma_2}, V_0)$.

The public key is expressed as follows:

$$PK = \left(e, \mathbb{G}, \mathbb{G}_T, g, q, \Omega_1, \Omega_2, g^\alpha, V_0, (x_1, \ldots, x_N), H, (U_1, \ldots, U_L)\right). \tag{16}$$

Additionally, the master secret key is expressed as follows:

$$MSK = \left(\alpha = a, t_1 = \sigma_1 - \sigma_2, t_2 = \frac{\sigma_3}{a} - \sigma_2, (V_1, \ldots, V_N)\right). \tag{17}$$

Query phase 1: in this phase, $\mathscr{B}$ will respond to the key query of $\mathscr{A}$. Each time, $\mathscr{A}$ will commit an attribute vector $W = (w_1, \ldots, w_L)$ and set $s = \sigma_2$, $s_1 = t_1 + \sigma_2$, and $s_2 = \sigma_2 + \sigma_3/a - \sigma_2 = \sigma_3/a$. Then, $\mathscr{B}$ responds by computing

$$\begin{aligned}
K &= g^{a\sigma_2},\\
K_1 &= \left\{K_{1,0}, K_{1,1}, \ldots, K_{1,N}\right\}\\
&= \left\{ g^{\left(bv_0 - \sum_{i=1}^{L} u_i\right)\sigma_1} \prod_{i=1}^{L} g^{\sigma_2 u_i/\overline{w}_i} w_i, g^{\left(bv_0 - \sum_{i=1}^{L} u_i\right)x_1\sigma_1} \prod_{i=1}^{L} g^{\sigma_2 i u_i/\overline{w}_i} w_i \right.\\
&\quad \left., \ldots, g^{\left(bv_0 - \sum_{i=1}^{L} u_i\right)x_N\sigma_1} \prod_{i=1}^{L} g^{\sigma_2 i^N u_i/\overline{w}_i} w_i \right\}.\\
K_1' &= \left\{K_{1,0}', K_{1,1}', \ldots, K_{1,N}'\right\}\\
&= \left\{ g^{\left(bv_0 - \sum_{i=1}^{L} u_i\right)\sigma_3} \prod_{i=1}^{L} (g^a)^{\sigma_2 u_i/\overline{w}_i} w_i, g^{\left(bv_0 - \sum_{i=1}^{L} u_i\right)x_1\sigma_3} \prod_{i=1}^{L} (g^a)^{\sigma_2 i u_i/\overline{w}_i} w_i \right.\\
&\quad \left., \ldots, g^{\left(bv_0 - \sum_{i=1}^{L} u_i\right)x_N\sigma_3} \prod_{i=1}^{L} (g^a) \prod_{i=1}^{L} (g^a)^{\sigma_2 i^N u_i/\overline{w}_i w_i} \right\}.
\end{aligned} \tag{18}$$

Finally, $\mathscr{B}$ sends $SK = (K, K_1, K_1')$ to $\mathscr{A}$.

Challenge: when query phase 1 is over, $\mathscr{A}$ sends two messages $M_0, M_1 \in \mathbb{G}_T$, $(|M_0| = |M_1|)$, to $\mathscr{B}$. Then, $\mathscr{B}$ randomly chooses and outputs its guess and selects a message M_γ to encrypt under $\overline{W}_\gamma$. Then, $\mathscr{B}$ creates

$$\begin{aligned}
C_0 &= M_\gamma \cdot e\left(g^{ac}, g^{bv_0}\right)^{\sigma_1 - \sigma_2} \cdot e\left(g^{ac}, g\right)^{\sum_{i=1}^{L} u_i (\sigma_1 - \sigma_2)}\\
&\quad \cdot e\left(g^d, g^b\right)^{v_0\sigma_3} \cdot e\left(g^d, g^{-a\sigma_2}\right)^{-\sum_{i=1}^{L} u_i} \cdot e\left(g^d, g^{\sigma_3}\right)^{-\sum_{i=1}^{L} u_i},\\
C_1 &= g^{ac/\sum_{m=0}^{n} x_m a_m},\\
C_2 &= g^{d/\sum_{m=0}^{n} x_m a_m},\\
C_3 &= Z_\beta^{v_0},
\end{aligned} \tag{19}$$

$\mathscr{B}$ sends the challenge ciphertext $CT = (C_0, C_1, C_2, C_3, GBF)$ to $\mathscr{A}$.

Query phase 2: query phase 2 is the same as query phase 1.

Guess: $\mathscr{A}$ outputs its guess $\gamma' \in \{0, 1\}$ for γ.

Finally, if $\gamma' = \gamma$, $\mathscr{B}$ outputs 1; otherwise, $\mathscr{B}$ outputs 0.

In the following, we analyze the probability of success for $\mathscr{B}$. Here, if $\beta = 0$, then $\mathscr{B}$ will behave correctly as a

challenger to $\mathcal{A}$. Furthermore, $\mathcal{A}$ will have the probability of $1/2 + \epsilon$ of guessing γ. If $\beta = 1$, $\mathcal{A}$ will have the probability of $1/2$ of guessing γ.

To conclude this proof, we obtain the following:

$$\begin{aligned}
&\left|\Pr\left[\mathcal{B}\left(g, g^a, g^b, g^{ac}, g^d, g^{b(c+d)}\right) = 1\right] - \Pr\left[\mathcal{B}\left(g, g^a, g^b, g^{ac}, g^d, g^r\right) = 1\right]\right| \\
&\geq \left|\Pr[\beta = 0 \wedge \gamma' = \gamma]\right| - \left|\Pr[\beta = 1 \wedge \gamma' = \gamma]\right| \\
&= \left|\frac{1}{2}\Pr[\gamma' = \gamma \mid \beta = 0] - \frac{1}{2}\Pr[\gamma' = \gamma \mid \beta = 1]\right| \\
&= \frac{1}{2}\left|\Pr[Exp_{\mathcal{A}}(\lambda) = 1] - \frac{1}{2}\right| \\
&\geq \frac{1}{2}\varepsilon,
\end{aligned} \tag{20}$$

which is nonnegligible, thereby contradicting the DLIN assumption. □

6.3. Performance Evaluation

6.3.1. Data Storage Efficiency. We conduct an experiment to verify the efficiency of two storage schemes, i.e., cloud-based server storage and IPFS-based distributed storage. Here, we set up a local server and an IPFS cluster in the same local area network. The cluster comprises five local devices, all of which are Ubuntu 18.04 systems with an Intel Core i5 CPU at 2.4 GHz with 4 GB RAM. We compare the performance of the two storage schemes based on the time required for uploading and downloading files. Figure 7 shows the transmission time required by the two schemes. Note that all experimental results are the average of 30 trials. As shown in Figure 7, the upload and download time of files in the IPFS-based scheme are less than those of the files in the cloud-based scheme. Moreover, the time required for uploading and downloading the files in the cloud-based scheme exhibits a faster growth trend than the IPFS-based scheme when the file size increased. Therefore, the IPFS-based scheme can improve storage efficiency and system scalability.

6.3.2. GBF Efficiency. We conduct another experiment to verify the storage and query efficiency of the GBF. Here, we use double hashing technology, and the k hash functions of the GBF are constructed using the 128-bit SpookyHash and MurmurHash. The length of the GBF is set to $m = 1024$, and $\lambda = 8$ in this experiment. Furthermore, the number of attributes in the access policy is 5–35, the number of wildcards is 2–14, and the number of hash functions is $k = 6, 8$, and 10. Note that all reported experimental results are the average values obtained over 30 trials. As shown in Figure 8, for $k = 8$, the time required to add 10 wildcards to the GBF is approximately 4.5 ms, and the query time is approximately 3.5 ms. Therefore, introducing the GBF does not increase the computational overhead of the system.

6.3.3. Performance Evaluation. We also analyze the performance of our scheme and five existing CP-ABE schemes with AND gates in terms of the ciphertext size, decryption consumption, whether access policies are hidden, and so on. The results are presented in Table 2, where p represents the pairing operation, e represents the exponentiation operation, l represents the number of attributes in the access structure, m represents the number of possible values for an attribute, and n represents the number of wildcards in the access structure. From Table 2, among all schemes that support wildcards and the hidden access policy, our scheme exhibits the smallest ciphertext size. Furthermore, the ciphertext size of our scheme is constant. Note that the decryption consumption of our scheme is related to the number of wildcards n, where $n \ll l$; thus, our scheme has the advantage in terms of decryption consumption.

To evaluate the actual performance of our scheme, we compare it to schemes proposed in [10, 31]. Here, we implement our scheme on a desktop PC (3.4 GHz Intel Core i7 CPU with 16 GB RAM) based on Ubuntu 18.04 LTS and Java Pairing-Based Cryptography Library (JPBC) 2.0.0. This implementation uses a 160-bit elliptic curve group based on the supersingular curve $y^2 = x^3 + x$ over a 512-bit finite field. The number of attributes in the access policy is 5–35, and the number of wildcards is 2–14. To ensure accuracy in our experiments, all reported experimental results are the averages obtained over 30 trials.

Figure 9(a) shows the size of the public parameters in the setup phase. The size of public parameters in all schemes increases linearly with the increase in the number of attributes. Nishide et al. [10] defined all possible values for each attribute in the public parameter; thus, the size of the public parameters is larger than other schemes. Figure 9(b) presents the execution time of the key generation phase. In our scheme, the key is generated

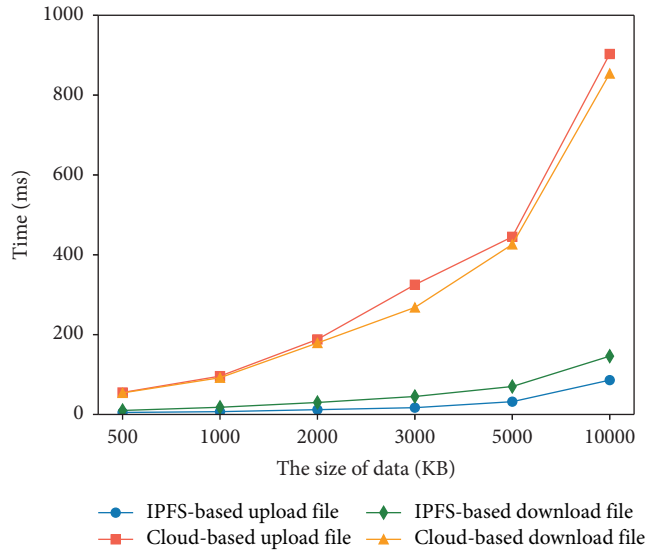

Figure 7: Transmission time of the two schemes.

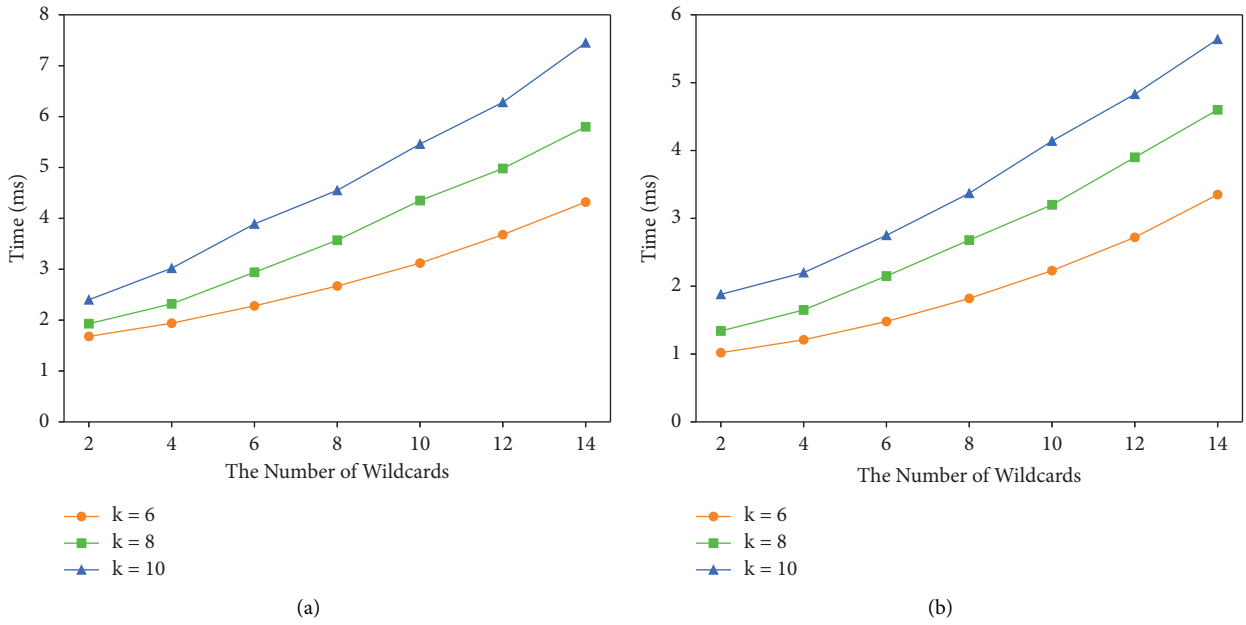

Figure 8: Computational time for (a) BuildGBF and (b) QueryGBF.

by the AA with high computational power; therefore, although the time required in our scheme is greater than other schemes, it will not affect the efficiency of our scheme. Figure 9(c) presents the execution time of the encryption operation. Here, the number of exponentiation operations in the encryption algorithm is related to the attribute; thus, the required time increases with the increase in the number of attributes. The scheme in [10] exhibits more exponentiation operations in the encryption phase than our scheme and the scheme in [31]; thus, the required time is greater than the other two schemes. Figure 9(d) shows the execution time of the decryption operation. Note that the existing scheme [31] does not hide the access policy; thus, its decryption time is fixed. The decryption time of our scheme is less than that of the scheme in [10].

Table 2: Performance comparison of CP-ABE schemes.

Scheme	Group order	Access structures	Ciphertext size	Decryption cost	Wildcard	Hidden policy
[10]	Prime	AND gates on multivalued attributes	$\lvert\mathbb{G}_T\rvert + (2ml+1)\lvert\mathbb{G}\rvert$	$(3l+1)p$	√	√
[12]	Prime	AND gates on ±	$\lvert\mathbb{G}_T\rvert + (4n+2)\lvert\mathbb{G}\rvert$	$(4n+2)p$	√	√
[31]	Prime	AND gates on multivalued attributes	$\lvert\mathbb{G}_T\rvert + 2\lvert\mathbb{G}\rvert$	$2p$	×	×
[32]	Composite	AND gates on multivalued attributes	$\lvert\mathbb{G}_T\rvert + (2ml+2)\lvert\mathbb{G}\rvert$	$(l+1)p$	√	√
[33]	Prime	AND gates on ±	$\lvert\mathbb{G}_T\rvert + (l+1)\lvert\mathbb{G}\rvert$	$(l+1)p$	√	×
Ours	Prime	AND gates on multivalued attributes	$\lvert\mathbb{G}_T\rvert + 3\lvert\mathbb{G}\rvert$	$3p+2ne$	√	√

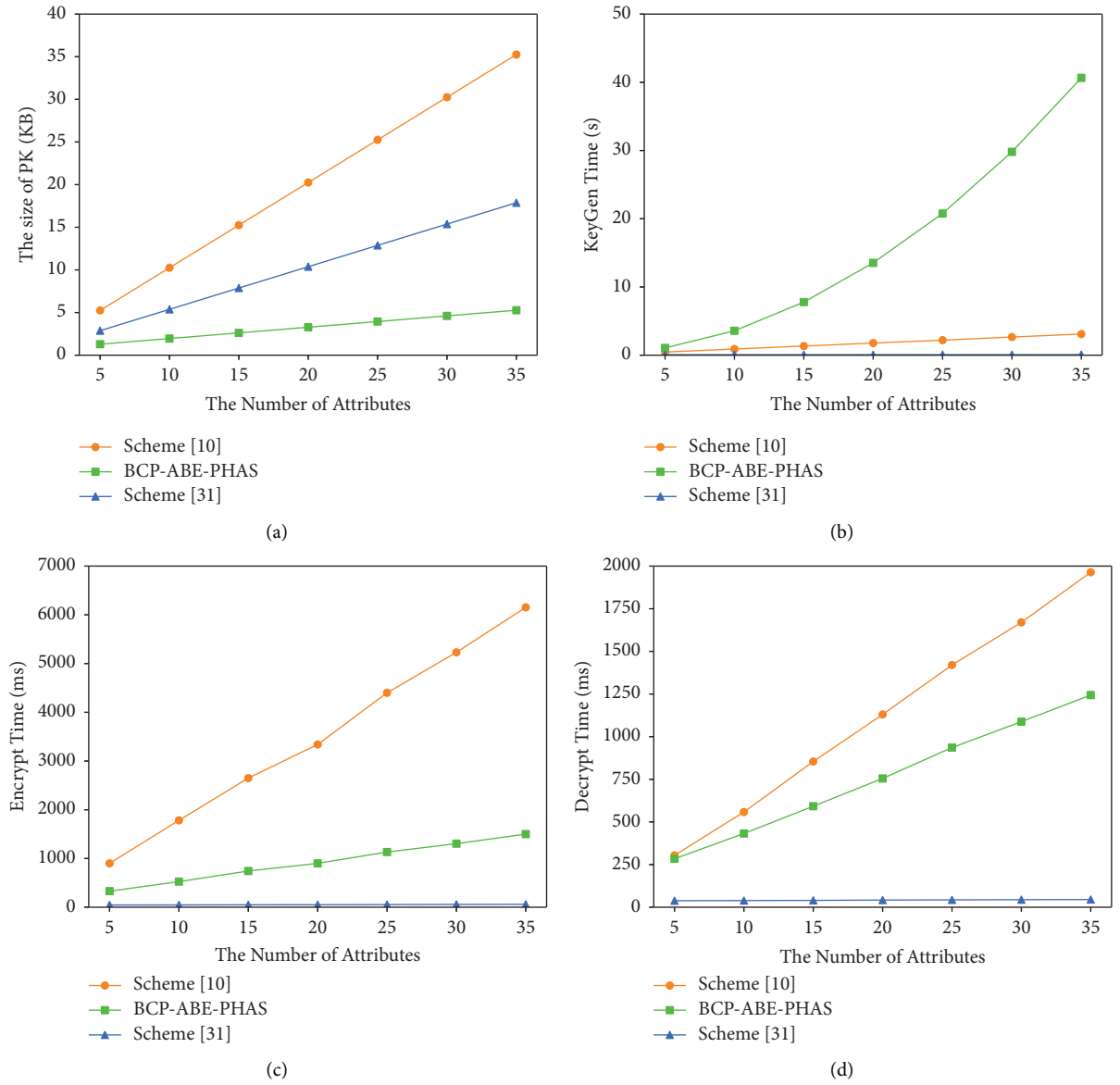

Figure 9: Performance analysis and comparisons. (a) Public key. (b) Key generation. (c) Encryption data. (d) Decryption data.

In summary, our scheme has advantages in terms of data encryption and decryption when implementing the hidden access policy.

7. Conclusion

In this paper, we propose a BCP-ABE-PHAS scheme to achieve trustworthy access while ensuring user privacy. Traditional centralized storage architectures are vulnerable to various network attacks, e.g., single point of attack, man-in-the-middle attack, and distributed denial-of-service attack. Therefore, we adopt a data storage scheme that combines blockchain technology and the IPFS; this approach relieves the storage pressure on the blockchain and guarantees data integrity. The experimental results demonstrate that our scheme is efficient and maintains a constant ciphertext size. Furthermore, to assist data decryption, we design a GBF to help users quickly locate the position of wildcards in the access policy. The proposed scheme uses smart contracts to guarantee that the entire data storage and sharing process is transparent, dynamic, and automated. The results of security analysis and performance evaluations demonstrate that our scheme is secure and efficient.

Data Availability

As part of the data in the paper is confidential, the code cannot be published for the time being. If data needed, send the corresponding author an email.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant Nos. 62172433, 62172434, 61862011, and 61872449).

References

[1] R. Lu, H. Zhu, X. Liu, J. K. Liu, and J. Shao, "Toward efficient and privacy-preserving computing in big data era," *IEEE Network*, vol. 28, no. 4, pp. 46–50, 2014.

[2] A. Sahai and B. Waters, "Fuzzy identity-based encryption, lecture notes in computer science," in *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 457–473, Springer, Aarhus, Denmark, May 2005.

[3] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 321–334, IEEE, CA, USA, May 2007.

[4] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, pp. 89–98, Alexandria, VA, USA, November 3 2006.

[5] H. Cui, R. H. Deng, J. Lai, X. Yi, and S. Nepal, "An efficient and expressive ciphertext-policy attribute-based encryption scheme with partially hidden access structures, revisited," *Computer Networks*, vol. 133, pp. 157–165, 2018.

[6] Z. Li, D. Wang, and E. Morais, "Quantum-safe round-optimal password authentication for mobile devices," *IEEE Transactions on Dependable and Secure Computing*, p. 1, 2020.

[7] S. Qiu, D. Wang, G. Xu, and S. Kumari, "Practical and provably secure three-factor authentication protocol based on extended chaotic-maps for mobile lightweight devices," *IEEE Transactions on Dependable and Secure Computing*, p. 1, 2020.

[8] N. Nizamuddin, H. R. Hasan, and K. Salah, "IPFS-blockchain-based authenticity of online publications," in *Proceedings of the International Conference on Blockchain*, pp. 199–212, Springer, Xi'an, China, December 22 2018.

[9] C. Wang, D. Wang, G. Xu, and D. He, "Efficient privacy-preserving user authentication scheme with forward secrecy for industry 4.0," *Science China Information Sciences*, vol. 65, no. 1, pp. 1–15, 2021.

[10] T. Nishide, K. Yoneyama, and K. Ohta, "Attribute-based encryption with partially hidden encryptor-specified access structures," in *Proceedings of the International Conference on Applied Cryptography and Network Security*, pp. 111–129, Springer, New York, NY, USA, June 2008.

[11] J. Li, K. Ren, B. Zhu, and Z. Wan, "Privacy-aware attribute-based encryption with user accountability," in *Proceedings of the International Conference on Information Security*, pp. 347–362, Springer, Pisa, Italy, September 2009.

[12] T. V. X. Phuong, G. Yang, and W. Susilo, "Hidden ciphertext policy attribute-based encryption under standard assumptions," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 1, pp. 35–45, 2015.

[13] J. Lai, R. H. Deng, and Y. Li, "Expressive CP-ABE with partially hidden access structures," in *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*, pp. 18-19, ACM, Seoul, Republic of Korea, May 2 2012.

[14] Y. Zhang, D. Zheng, and R. H. Deng, "Security and privacy in smart health: efficient policy-hiding attribute-based access control," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 2130–2145, 2018.

[15] J. Katz, A. Sahai, and B. Waters, "Predicate encryption supporting disjunctions, polynomial equations, and inner products," in *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 146–162, Springer, Istanbul, Turkey, April 2008.

[16] A. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters, "Fully secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption," in *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 62–91, Springer, French Riviera, May 30 2010.

[17] J. Hur, "Attribute-based secure data sharing with hidden policies in smart grid," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 11, pp. 2171–2180, 2013.

[18] S. Gao, G. Piao, J. Zhu, X. Ma, and J. Ma, "TrustAccess: a trustworthy secure ciphertext-policy and attribute hiding access control scheme based on blockchain," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 6, pp. 5784–5798, 2020.

[19] Y. Zhang, S. Kasahara, Y. Shen, X. Jiang, and J. Wan, "Smart contract-based access control for the Internet of Things," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1594–1605, 2018.

[20] J. Xu, K. Xue, S. Li et al., "Healthchain: a blockchain-based privacy preserving scheme for large-scale health data," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8770–8781, 2019.

[21] R. Xu, Y. Chen, E. Blasch, G. Chen, and C. A. C. Blend, "A blockchain-enabled decentralized capability-based access control for IoTs," in *Proceedings of the 2018 IEEE International Conference on Internet of Things*, pp. 1027–1034, Halifax NS Canada, Auguest 3 2018.

[22] O. Novo, "Blockchain meets IoT: an architecture for scalable access management in IoT," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1184–1195, 2018.

[23] S. Nakamoto, "Bitcoin: a peer-to-peer electronic cash system," 2008, https://bitcoin.org/bitcoin.pdf.

[24] N. Szabo, "Smart contracts," 1994, http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html.

[25] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," 2014, https://www.gavwood.com/paper.pdf.

[26] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[27] P. Bose, H. Guo, E. Kranakis et al., "On the false-positive rate of Bloom filters," *Information Processing Letters*, vol. 108, no. 4, pp. 210–213, 2008.

[28] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612-613, 1979.

[29] S. Sedghi, P. van Liesdonk, S. Nikova, P. Hartel, and W. Jonker, "Searching keywords with wildcards on encrypted data," in *Proceedings of the International Conference on Security and Cryptography for Networks*, pp. 138–153, Springer, Amalfi, Italy, September 16 2020.

[30] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," in *Proceedings of the Annual International Cryptology Conference*, pp. 41–55, Springer, Santa Barbara, California, USA, August 22 2002.

[31] K. Emura, A. Miyaji, A. Nomura, K. Omote, and M. Soshi, "A ciphertext-policy attribute-based encryption scheme with constant ciphertext length," in *Proceedings of the International Conference on Information Security Practice and Experience*, pp. 13–23, Springer, Xi'an, China, April 2009.

[32] J. Lai, R. H. Deng, and Y. Li, "Fully secure cipertext-policy hiding CP-ABE," in *Proceedings of the International Conference on Information Security Practice and Experience*, pp. 24–39, Springer, Guangzhou, China, May 2011.

[33] L. Cheung and C. Newport, "Provably secure ciphertext policy ABE," in *Proceedings of the 14th ACM Conference on Computer and Communications Security*, pp. 456–465, ACM, NY, USA, October 31 2007.