Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 8350006, 15 pages
https://doi.org/10.1155/2022/8350006

Research Article

Novel Searchable Attribute-Based Encryption for the Internet of Things

Zhenhua Lu,[1] **Yuyan Guo**,[1] **Jiguo Li**,[2] **Weina Jia**,[3] **Liping Lv**,[3] **and Jie Shen**[4]

[1]*College of Computer Science and Technology, Huaibei Normal University, Huaibei, Anhui 235000, China*
[2]*College of Computer and Cyber Security, Fujian Normal University, Fuzhou, Fujian 350117, China*
[3]*College of Information Engineering, Zhengzhou Shengda University, Xinzheng, Henan 451100, China*
[4]*School of Mechanical Engineering, North China University of Water Resources and Electric Power, Zhengzhou, Henan 450000, China*

Correspondence should be addressed to Yuyan Guo; guoyuyan428@163.com

Received 16 January 2022; Accepted 8 March 2022; Published 11 April 2022

Academic Editor: Qingqi Pei

As the product of the third information technology revolution, the Internet of Things (IoT) has greatly altered our way of lifetime. Cloud storage has gradually become the best choice for data processing due to its scalability and flexibility. However, the cloud is not a completely trusted entity, such as tampering with user data or leaking personal privacy. Therefore, cloud storage usually adopts attribute-based encryption schemes to accomplish data confidentiality and fine-grained access control. However, applying the ABE scheme to the Internet of Things still faces many challenges, such as dynamic user revocation, data sharing, and excessive computational burden. In this paper, we propose a novel searchable attribute encryption system that replaces the traditional key generation center with consortium blockchain to generate and manage partial keys. In addition, our scheme can perform predecryption operations in the cloud, and users only need to spend a small amount of computational cost to achieve decryption operations. Security analysis proves that our scheme achieves security under both the chosen keyword attack and the chosen plaintext attack. Compared with other schemes, this scheme is more economical in terms of computing and storage.

1. Introduction

The Internet of Things (IoT) originated in the media field, and it is an important part of the new generation of information technology. It connects all items to the Internet through information sensing devices to achieve positioning, tracking, supervision, and intelligent identification [1]. Simply put, the IoT is the Internet that connects everything. In recent years, the IoT has gradually become digitized in the real world, reduced the dispersion of information, and integrated the digital information between objects. The IoT is widely used in the fields of transportation and logistics, industrial manufacturing, medical care, and smart environments [2–7].

Sahai and Waters first proposed the concept of attribute-based encryption (ABE) in 2005 [8]. Because the access structure needs to be more flexible to adapt to more application scenarios, Goyal et al. [9] and Bethencourt et al. [10], respectively, proposed the concepts of key policy ABE (KP-ABE) and ciphertext policy ABE (CP-ABE). In KP-ABE, the ciphertext is associated with a set of attributes, and the access policy is embedded in the key. In contrast, in CP-ABE, the ciphertext is associated with the access policy, and a set of attributes is embedded in the key [11–14].

Although IoT has now blossomed in various fields, there are still many problems in applying ABE to the IoT. Compared with the traditional Internet, the IoT lacks standardization. In addition, the IoT itself is a complex network system involving many application domains, which is difficult to manage [15]. This makes it challenging to achieve its security and privacy. Therefore, blockchain technology is widely used in IoT for its persistence, anonymity, and auditability [16].

In this article, we will present a new blockchain-aided searchable attribute-based encryption (BC-SABE) scheme. This scheme replaces a traditional centralized server using

a distributed consortium blockchain consisting of a predefined set of trusted consensus nodes. Our main contributions can be summarized as follows:

(i) We present a novel BC-SABE scheme. Our scheme uses a distributed consortium blockchain containing a set of credible consensus nodes to achieve the function of key generation. Pedersen secret sharing protocol [17] and reciprocal protocol [18] are used to generate all secret parameters, which also means that the master key is not needed. Our scheme can also support keyword search under cloud assistance. Users only need to provide user identity information and partial token information to the blockchain, and the cloud server will receive the complete token from the blockchain and search for it. Moreover, the scheme can realize predecryption in the cloud, which can greatly reduce the burden of users

(ii) In our scheme, the Pedersen secret sharing protocol enables the sharing of subsecrets between consensus nodes, and each consensus node can combine the subsecrets as the master secret. And the reciprocal protocol ensures that the key information is shared without a trusted party

(iii) In addition, we use blockchain technology to realize the dynamic revocation of users. Because consensus nodes in the blockchain can use time period tags and status tags to update the user revocation list, this also means that we can use the blockchain to update the user revocation list to achieve user revocation. And our scheme does not require the user to re-encrypt the ciphertext for user revocation

Our scheme can also be applied in simple medical scenarios. For example, hospital can register admission information for patient and store the admission information in the blockchain. Registered patients can enter their data and information into the cloud server. When searching for relevant information, he/she only needs to submit a partial token to blockchain, and then, blockchain can produce the complete token for him/her and send it to the cloud server for search operations. In addition, patients can access data information from the cloud, which will first generate a predecryption key for them, and the patient can fully decrypt the data with a simple calculation.

The remainder of this paper is organized as follows. In Section 2, we reviewed some related work, and then, we gave some preliminaries in Section 3, including the complexity assumptions, binary trees, and blockchain. The system model, system procedure, and security model are given in Section 4, and the detailed structure of the system is given in Section 5. We give the security proof of the scheme in Section 6 and compare the performance of the scheme in Section 7, and finally, we give a brief summary in Section 8.

2. Related Work

Since the concepts of ABE were introduced, many improved ABE schemes have been proposed, such as traceable ABE [19], anonymous ABE [20, 21], and hierarchical ABE [22–25]. However, the application of ABE to IoT is still an issue that needs to be discussed. The resources of the Internet of Things devices are limited, and most of the ABE algorithm encryption and decryption calculation costs are relatively large, so the outsourcing ABE scheme has been proposed [26–29]. In addition, IoT systems should be able to revoke malicious users and update legitimate users with new attributes. How to implement dynamic user revocation is also an issue. Liu et al. [30] proposed a direct revocation scheme; however, in this scheme, all data owners are required to maintain the revocation list. Recently, Cui et al. [31] proposed a server-aided revocable ABE scheme. This scheme outsources all the workload to the server at the time of user revocation, and each user stores only a fixed size private key.

However, this single authorization ABE scheme has limited security and cannot carry a large number of IoT devices. Many multiauthorization-based ABE schemes have been proposed by scholars [32–35]. Chase [32] first presented the concept of multiauthority attribute-based encryption (MA-ABE) in 2007. Recently, Belguith et al. [34] presented a policy-hidden outsourced MA-ABE scheme, which hides the access structure to protect user privacy. In [35], a new MA-CP-ABE scheme was presented by Sethi et al. This system decentralizes authority and can support white box traceability along with outsourcing decryption.

Recently, the widespread application of data sharing has deepened the academic research on searchable encryption schemes, and many searchable attribute-based encryption (SABE) schemes have also been proposed [36–44]. In [36], Miao et al. presented a multikeyword SABE scheme, which supports comparable attributes by using 0-encoding and 1-encoding. Xu et al. [37] proposed for the first time a decentralized attribute-based keyword search (ABKS) scheme for multikeyword search in cloud storage. In this scheme, data sharing and data searching can be achieved without a fully trusted central authority. Recently, a seed string searchable ABE (SSS-ABE) solution for sharing and querying encrypted data was proposed by Sun et al. [38]; data users can query the entire ciphertext by substring without presetting keywords.

Blockchain technology originated from Bitcoin; the concept of blockchain was first proposed by Satoshi Nakamoto in [45]. Blockchain is a distributed shared ledger and database. Compared with the previous centralized accounting model, blockchain can achieve decentralization, which means removing the trust intermediary, which also makes the transactions on blockchain more open and transparent. Recently, Wu et al. [46] used blockchain technology to propose a privacy-protected and traceable ABE scheme, using blockchain to achieve data integrity and nonrepudiation. In [47], Pournaghi et al. proposed a scheme to share medical data based on blockchain technology and attribute-based encryption (MedSBA), which utilizes blockchain to share medical data. Recently, Liu et al. [48] presented a blockchain-aided attribute-based searchable encryption scheme, and Zheng et al. [49] proposed a fair outsourcing decryption ABE scheme utilizing blockchain and sampling technology. In [50], Guo et al. used blockchain technology to propose an efficient and traceable ABE scheme with

dynamic access control, which implements dynamic access control and can flexibly update the access structure.

As of late, blockchain technology has been universally exploited in ABE scheme, because compared to the traditional server structure, blockchain has no central node, which allows no single institution or member to achieve control over global data, while any node stops working without affecting the overall operation of the system. In addition, blockchain is also superior to traditional key management center in ensuring the confidentiality of data. Therefore, we adopt blockchain technology to assure the security and robustness of the system.

3. Preliminaries

3.1. Composite Order Bilinear Groups and Complexity Assumptions. First of all, the concept of bilinear group is reviewed as follows. Let G and G_1 be the multiplicative groups of order $N = p_1p_2p_3$, g is a generator of G, and p_1, p_2, p_3 are three different prime. Then, $e : G \times G \longrightarrow G_1$ is a bilinear map, and it has these properties as follows:

(1) Bilinearity: For $\forall x, y \in Z_N$, $e(g^x, g^y) = e(g, g)^{xy}$

(2) Nondegeneracy: $e(g, g) \neq 1_{G_1}$

(3) Computability: There is an algorithm to calculate e efficiently

Now, we show the definition of the composite order bilinear groups. It is similar to bilinear groups except the order of the group is the product of two or more distinct prime numbers. That is to say, G is a composite order group; $G_{p_1}, G_{p_2}, G_{p_3}$ are its three subgroups of order p_1, p_2, p_3. For $\forall x \in G_{p_i}$ and $\forall y \in G_{p_j}$, if $x \neq y$, then $e(x, y) = 1$.

Decisional linear assumption. Given $(g_1^{\alpha}, g_2^{\beta}, g_1, g_2, g_3)$, $\alpha, \beta \in Z_N$, any probabilistic polynomial time (PPT) algorithm is difficult to distinguish $(g_1^{\alpha}, g_2^{\beta}, g_3^{\alpha+\beta}, g_1, g_2, g_3)$ from $(g_1^{\alpha}, g_2^{\beta}, g_1, g_2, g_3, A)$, where $g_1, g_2, g_3, A \in G$ and $a, b \in Z_p$ are randomly selected.

Decisional bilinear Diffie–Hellman problem (DBDH). Given $g^{\alpha}, g^{\beta}, g^{\gamma} \in G$ and $P \in G_1$, any PPT algorithm is difficult to distinguish $P = e(g, g)^{\alpha\beta\gamma}$ from $P = e(g, g)^{p}$, where $\alpha, \beta, \gamma, p \in Z_p^*$.

3.2. Access Structure and Linear Secret Sharing Scheme (LSSS)

Definition 1. Assume $O = \{attr_1, \cdots, attr_n\}$ is a set of attributes, and $\mathscr{A} \subset 2^O$ is a nonempty subset of 2^O, where 2^O represents the set constituted by all subsets of O; that is, $\mathscr{A}$ is a nonempty set constituted by some subsets of O. We call $\mathscr{A}$ is an access structure on O. If for any P, Q satisfies the condition $P \in \mathscr{A}$ and $P \subseteq Q$, namely $Q \in \mathscr{A}$, and then set $\mathscr{A} \subseteq O$ is monotonic. Authorized set refers to the set in $\mathscr{A}$; on the contrary, the unauthorized set is not in $\mathscr{A}$.

Definition 2. In the linear secret sharing matrix formed by the access policy, each row corresponds to an attribute value, that is, row vector and attribute value form a one-to-one mapping relationship. If the following two properties are satisfied, and then, a secret sharing scheme Σ on a set of $O = \{attr_1, \cdots, attr_n\}$ is called linear.

(1) The shared secret key for each attribute is a vector formed on Z_p

(2) In scheme Σ, there is an $n \times m$ secret sharing matrix $\boldsymbol{A}$, whose row label is $b(i)$, $i \in \{1, 2, \cdots, n\}$. Given a secret sharing column vector $u = (\mu, u_2, \cdots, u_m)$, where $\mu \in Z_p$ is the secret key to be shared, $u_2, \cdots, u_m$ is selected at random, $\mathbf{A}u$ represents the vector of n shared secret keys according to Σ. Shared $\gamma_i = (\mathbf{A}u)_i$, that is the inner product $\mathbf{A}u$ belongs to the property $b(i)$, where b is a function that maps $i \in \{1, 2, \cdots, n\}$ to $b(i)$

The LSSS matrix has an important feature, that is, linear reconstruction. Suppose Σ is a LSSS scheme representing access structure $\mathscr{A}$, $Q \in \mathscr{A}$ is an authorized set, and then, we can define $T \subset [n]$ as $T = \{i : b(i) \in Q\}$. If there has constant $\{\beta_i \in Z_p\}_{i \in T}$ that can be discovered in polynomial time such that $\{\gamma_i\}$ are valid shares of the secret key μ, then $\sum_{i \in T} \beta_i \gamma_i = \mu$. There is no such constant for any unauthorized set.

3.3. Binary Tree. First, there is a brief review of the definition of binary tree. Suppose UT represents a binary tree, in which there are L leaves corresponding to L users, the root node of BT is Rt. $path(\varphi)$ represents the set of all nodes on the path of leaf node φ from the root to φ. Note that φ and the root node are included here. φ_l, φ_r represents the left and right children of nonleaf nodes. The algorithm KUNodes is used to calculate the minimum set of nodes that need to release key updates, and only unrevoked users can decrypt the ciphertext within a period of time. That is, nodes in rl corresponding to time periods before or t do not have any ancestors in the set, and all other leaf nodes have exactly one ancestor in the set. Figure 1 gives the working principle of the KUNodes algorithm, where it first marks all ancestors of the revoked nodes as revoked and then outputs all unrevoked children of the revoked nodes. The Algorithm 1 is the formal definition of the KUNodes algorithm.

3.4. Shamir Secret Sharing [51]. The Shamir secret sharing scheme is based on Lagrange interpolation polynomials, which are described as follows:

(1) A trusted dealer D first randomly chooses a polynomial $g(x) = a_0 + a_1x + \cdots + a_{l-1}x^{l-1}$ with order l-1 such that $a_0 = s$, where $a_1, a_2, \cdots, a_{l-1}$ are in finite filed $F_p = GF(p)$. Then, D computes $s_1 = g(1), \ldots, s_m = g(m)$ and sends s_i to each shareholder p_i secretly

(2) More than l shareholder $p_r \subset p$ $(|p_r| \leq l)$ work together can reconstruct the secret using the Lagrange interpolating formula

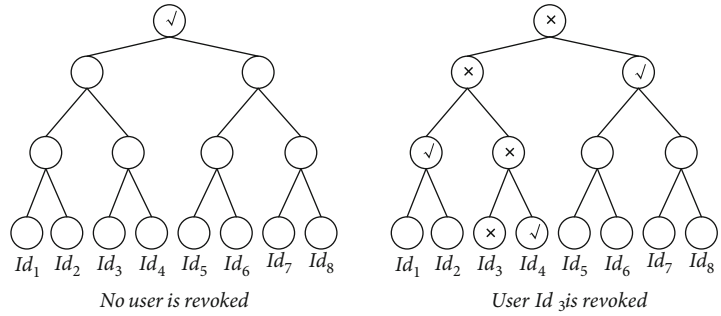


Figure 1: The KUNodes algorithm. This shows a pictorial depiction of the algorithm.

Inputs: a binary tree UT, a revocation list rl, a time period t, two empty sets P, Q.
$\forall(\varphi_i, t_i) \in rl$, if $t_i \le t$, then add $path(\varphi_i)$ to P.
$\forall p \in P$, if $p_l \notin P$, then add p_l to Q.
if $p_r \notin P$, add p_r to Q.
If $Q = \varnothing$, then add Rt to Q.
Return Q.

Algorithm 1: KUNodes.

3.5. Pedersen (l, m) Secret Sharing [17]. The Pedersen secret sharing scheme allows each dealer (shareholder) to randomly select a secret as a subsecret and can share the subsecret with other shareholders. Therefore, each shareholder can merger all the subsecrets as the master secret. The Pedersen secret sharing scheme is described as follows:

(1) Each shareholder $H_i (i \in [1, \cdots, m])$ randomly independently picks a subsecret S_i, and then, the master secret can be described as $S = \sum_{i=1}^{m} S_i$

(2) For each subsecret S_i, a l-1 polynomial $g(x)$ is randomly selected by shareholder H_i such that $S_i = g_i(0)$. After that, it calculates $s_{ij} = g_i(x_j)$,$(j = 1, 2, \cdots, m)$ for other shareholders by using Shamir's secret sharing. Finally, H_i sends each s_{ij} to other H_j secretly, and each shareholder H_i has m subshares s_{ij}

(3) Each shareholder H_i calculates its own master share $s_i = \sum_{j=1}^{m} s_{ji} = \sum_{j=1}^{m} g_j(x_i)$

(4) More than l shareholders $p_r \subset p$ $(|p_r| \le l)$ work together can reconstruct the secret by using the Lagrange interpolating formula

3.6. Reciprocal Protocol [18]. Suppose that shareholders H_i $(i \in [1, \cdots, m])$ share a secret μ using Pedersen (l, m) secret sharing protocol. The role of the reciprocal protocol is to get share μ^{-1} without disclosing relevant message about μ and μ^{-1}. The description of this protocol is as follows:

(1) Shareholders jointly run the Pedersen (l, m) secret sharing scheme to generate a (l, m) sharing of a random element $\alpha \in Z_q$. Denote all shares $\alpha_1, \alpha_2, \cdots, \alpha_m$ as $(\alpha_1, \alpha_2, \cdots, \alpha_m) \leftrightarrow^{(l,m)} \alpha$

(2) Shareholders jointly run the Pedersen (2l, m) secret sharing scheme to generate and retain a share of zero value β_i

(3) Shareholders need to pass the value $\mu_i\alpha_i + \beta_i$ and interpolating the corresponding $2l$ degree polynomial to reconstruct the value $\eta = \mu\alpha$

(4) Each shareholders sets $\delta_i = \eta^{-1}\alpha_i$ to calculate it share δ_i of μ^{-1}

3.7. Blockchain. On January 3, 2009, Satoshi Nakamoto generated the first Bitcoin block. A few days later, a second bitcoin block appeared to connect with the first block to form the chain, marking the birth of the blockchain. Because of its four main features, immutability, irreproducible uniqueness, smart contracts, and decentralized self-organization or community, blockchain is widely used in various fields.

In simple terms, a hash function (SHA-256) is used to form a blockchain. Each block contains a parent block hash, a timestamp, and a Merkle root. Where the parent block hash stores the hash value of the previous block header and is used to connect the previous block, the timestamp records the approximate time when the block was created, and the Merkle root is the Merkle tree root hash generated by the transaction list. There are three general types of blockchains: public blockchains, consortium blockchains, and private blockchains. Our system uses a consortium blockchain. Figure 2 illustrates the basic structure of a consortium blockchain.

In our consortium blockchain, consensus node nodes perform the consistency protocol to renovate the blockchain and reserve all nodes in the system with a consistent state. The consortium blockchain used in our system is similar to the scheme [48] in that firstly the consensus nodes can

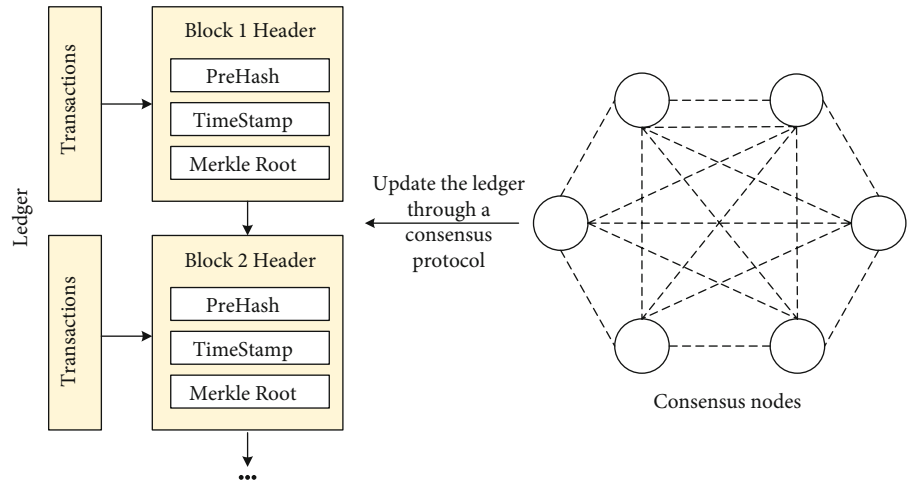


Figure 2: Consortium blockchain. Consensus node nodes perform the consistency protocol to renovate the blockchain.

initialize the system parameters using the Pedersen secret sharing scheme and the reciprocity protocol. Secondly, the consensus nodes manage the associated keys of the users. Updating the user revocation list requires the joint participation of all consensus nodes, which effectively improves the system security. Users can submit searches to the blockchain, and the cloud server is able to perform predecryption operations for the users.

4. System Definition

4.1. System Model. Our data management system includes the following four participants:

Data owner (DO): The data owner stores the generated index and encrypted data in the cloud, where the index is used for the cloud to perform search operations.

Data user (DU): The data user is able to store the generated partial token in the consortium blockchain and is able to get the predecrypted message from the cloud to fully decrypt it using their private key.

Blockchain (BC): The consortium blockchain in the system consists of a set of credible predefined consensus nodes, a data pool, and a distributed ledger. The blockchain is responsible for initializing the system, storing the users' public identity keys, and generating the users' public decryption keys, key update information, and predecryption keys. In addition, the blockchain is also responsible for generating the complete token and sending it to cloud.

Cloud server (CS): Cloud server can search and predecrypt for users, and putting predecryption operations in the cloud can effectively reduce the burden of users.

Figure 3 shows the system procedure.

4.2. System Procedure. Based on [48], the basic process of the scheme is defined as follows:

4.2.1. System Init. All consensus nodes run $Setup(1^k, \sigma) \longrightarrow GPK$ algorithm to get the GPK. Pedersen (l, m) secret sharing protocol and reciprocal protocol are used by all consensus nodes to jointly determine the master key, and the exact value of the master parameter is unknown.

4.2.2. User Registration and Revocation

(1) The data user runs the $IdKG(GPK, Id_U) \longrightarrow (sk_{Id_U}, pk_{Id_U})$ to get its identity key pair to join the system and sends pk_{Id_U} to BC. Then, the user public decryption key is generated by the consensus nodes using pk_{Id_U}. In the meantime, the user's predecryption key is also generated later using the public decryption key

(2) For user revocation, based on a time period t, a state mark sm, and the revocation list rl, consensus node runs the $Rv(Id_U, t, rl, sm) \longrightarrow rl$ to update rl whenever a user wishes to be revoked

4.2.3. Key Gen. In this step, consensus nodes generate three keys:

(1) Public decryption key generation: In this step, consensus nodes use the user identity Id_U, an access structure $(\mathscr{D}, b)$ to run $PubDecKG(GPK, Id_U, (\mathscr{D}, b), sm) \longrightarrow (PubDK_{Id_U}, sm)$ algorithm to generate the user public decryption key, which is used to verify whether the user has the attributes included in its attribute set

(2) Updated key generation: Consensus nodes run the $UpKG(GPK, t, rl, sm) \longrightarrow (U_t, sm)$ to get a key update message after rl is updated; it inputs a new time period t and a state mark sm, users who have not revoked will use it when generating a predecryption key

(3) Predecryption key generation: Consensus nodes use (U_t, sm) to run $PreDecKG(GPK, Id_U, (\mathscr{D}, b), pk_{Id_U},$

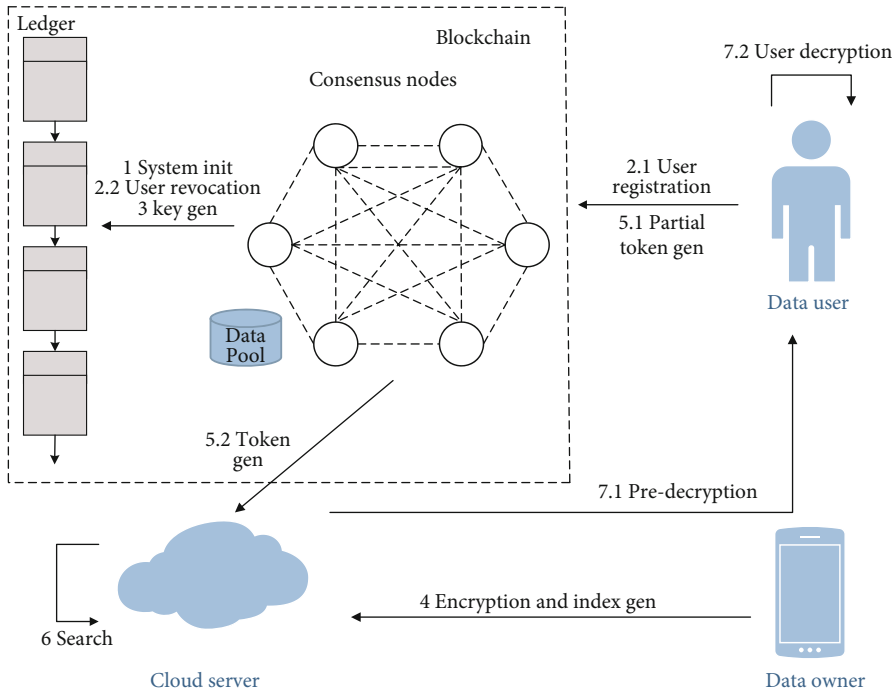


Figure 3: System procedures. The data management system contains four participants, and each participant operates as shown in the figure.

$U_t) \longrightarrow \mathrm{Pre}DK_{Id_U,t}$); it generates the predecryption key for user

4.2.4. Encryption. First, the data owner selects several keywords related to his/her data to generate a keyword set $\{w\}$, and then, he/she uses a symmetric algorithm with the key k_s to encrypt the data. Then, data owner runs the $Enc(GPK, \mathbf{S}, t, k_s) \longrightarrow CT$ to hide the symmetric encryption key and runs the $IdxG(GPK, \mathbf{S}, \{w\}) \longrightarrow Idx$ to generate an index set, where the same attribute set $\mathbf{S}$ is used. Finally, data owner sends CT and Idx to cloud.

4.2.5. Token Gen

(1) Patrial token generation: first, the data user runs the $PTokG(GPK, sk_{Id_U}, w) \longrightarrow \mathrm{Tok}'$ to generate the partial token. Then, data user sends $(\mathrm{Tok}', H(w))$ to BC

(2) Complete token generation: consensus nodes run $TokG(GPK, Id_U, (\mathscr{D}, b), \mathrm{Tok}') \longrightarrow Tok$ algorithm for data user after receiving Tok. Then, data user sends Tok to the cloud

4.2.6. Search. When data users need to search, the cloud runs the $Search(GPK, \mathbf{S}, Tok, Idx) \longrightarrow Addr/\perp$ algorithm to search.

4.2.7. Decryption

(1) Predecryption: The predecryption operation is done in the cloud, and the cloud takes as input the predecryption key sent to it by the blockchain, and it runs the $PreD(GPK, (\mathscr{D}, b), Id_U, \mathrm{Pre}DK_{Id_U,t}, CT, t) \longrightarrow CT'/\perp$ to convert the ciphertext of data users after receiving the requested key. In this phase, most of the calculation costs will be carried out in the cloud

(2) User decryption: Since a portion of the predecryption work has already been done in the cloud, data user runs the $Dec(GPK, sk_{Id_U}, CT') \longrightarrow k_s/\perp$ to generate the symmetric decryption key. After that, the complete decryption is done by running the symmetric algorithm

4.3. Security Model

4.3.1. Ciphertext Indistinguishability. Based on [48], we give a security definition of indistinguishability under chosen plaintext attacks (IND-CPA) for BC-SABE, defined by an indistinguishable game between adversary A and challenge B. Let U_m be the authority universe. Adversary A is defined as an (l,m) adversary that can compromise at most l-1 authority; Pedersen (l, m) secret sharing protocol and reciprocity protocol are applied in this security model.

Setup. A corrupted authority set U_ϕ is output by A, where $m_\phi \le l-1$. Then, B runs the $Setup(1^k, \sigma) \longrightarrow GPK$ to get the global public key GPK, a state mark sm and a revocation list rl. Note that rl is initially empty. It outputs $(GPK, rl, sm, \{a_i, r_i\}_{i \in U_\phi})$ to A.

Phase 1. First, B creates an empty list L, and A can perform the following queries adaptively:

(i) IdKey query: First of all, A issues a IdKey query on an identity Id_U, B returns (sk_{Id_U}, pk_{Id_U}) by running $IdKG(GPK, Id_U)$ algorithm, and then, it adds $(Id_U, sk_{Id_U}, pk_{Id_U})$ to list L. It returns sk_{Id_U} to A

(ii) PubDKey query: A issues a PubDKey Query on an identity Id_U and an access structure $(\mathscr{D}, b)$. If Id_U has not been issued to PubDKey query, then B returns (sk_{Id_U}, pk_{Id_U}) by running $IdKG(GPK, Id_U)$, runs $PubDecKG(GPK, Id_U, (\mathscr{D}, b), sm) \longrightarrow (PubDK_{Id_U}, sm)$ algorithm, and returns $(Id_U, PubDK_{Id_U}, sm)$ to A

(iii) Upkey query: Algorithm A submits a Upkey query on a time period t. B returns the U_t to A by running $UpKG(GPK, t, rl, sm)$

(iv) PreDKey query: Algorithm A submits a PreDKey query on $(Id_U, (\mathscr{D}, b), t)$. If Id_U has not been issued to PreDKey query, B runs the $IdKG(GPK, Id_U)$,$UpKG(GPK, t, rl, sm)$, $PubDecKG(GPK, Id_U, (\mathscr{D}, b), sm)$, and $PreDecKG(GPK, Id_U, (\mathscr{D}, b), pk_{Id_U}, U_t)$. Then, B returns the $PreDK_{Id_U,t}$ to A. Note that this oracle cannot be queried on a time period t before a Upkey query has been queried on t

(v) Revocation query: A submits a PreDKey Query on (Id_U, t), where t is not queried in UpKey query. B returns an updated revocation list rl to A by running $Rv(Id_U, t, rl, st)$

Challenge. A hands over two symmetric keys with same length SK_1^* and SK_2^*, an attribute set $\mathbf{S}^*$, and a time period t^* satisfying the following restrictions:

(i) If an identity Id_U^* has performed to the IdKey query, $\mathbf{S}^*$ of Id_U^* satisfies a query on $(Id_U^*, (\mathscr{D}^*, b^*))$ issued to the PubDKey query. Then, the revocation query must be queried on (Id_U^*, t^*) with $t = t^*$ or any t occurs before t^*, and the PreDKey query cannot be queried on (Id_U^*, t^*)

(ii) If Id_U^* with access structure $(\mathscr{D}^*, b^*)$ can be satisfied by $\mathbf{S}^*$ is not revoked before or at t^*, then Id_U^* has never been queried by the IdKey query

B picks random $\rho \in \{0, 1\}$ and runs $Enc(GPK, \mathbf{S}^*, t^*, SK_\rho^*) \longrightarrow CT^*$ to encrypt SK_ρ^*, and then, it returns CT^* to A.

Phase 2. A can adaptively perform the same five queries to B as in phase 1; the queries sent by A must also meet the above conditional restrictions.

Guess. A makes a guess ρ' for ρ; if $\rho' = \rho$, it wins.

The advantage of the adversary A in this game is described as $\Pr[\rho = \rho'] - 1/2$.

If the advantages of any (l, m) PPT adversary defined above are negligible, then a BC-SABE scheme is IND-CPA secure.

4.3.2. Index Indistinguishability. Based on [48], we give a security definition of indistinguishability under selective access structure and chosen keyword attacks (IND-sCKA) for BC-SABE, defined by an indistinguishable game between adversary A and challenge B.

Init. Let $\mathbf{S}^*$ be the challenge access structure defined by A.

Setup. B returns the global public parameter GPK to A by running $Setup(1^k, \sigma)$ algorithm.

Phase 1. A can adaptively submit the following query:

(i) IdKey query: A issues a IdKey query on an identity Id_U, B returns (sk_{Id_U}, pk_{Id_U}) by running $IdKG(GPK, Id_U)$, and then, it adds $(Id_U, sk_{Id_U}, pk_{Id_U})$ to list L and returns sk_{Id_U} to A

(ii) Token query: A issues a token query on Id_U, an access structure $\mathbf{S}$, and a keyword w. B runs $PTokG(GPK, sk_{Id_U}, w) \longrightarrow \mathrm{To}k'$ by using sk_{Id_U}. Then, B returns Tok to A by running the $TokG(GPK, Id_U, (\mathscr{D}, \rho), \mathrm{To}k')$ algorithm. Note that the challenge access structure does not contain the attribute set $\mathbf{S}$

(iii) Index query: A issues an index query on $\mathbf{S}$ and a keyword set $\{w\}$. Then, B returns an index set by running $IdxG(GPK, \mathbf{S}, \{w\})$

Challenge. A submits two keywords K_1^* and K_2^* of the same size. B picks random $\rho \in \{0, 1\}$, and then it returns the Ind^* to A by running the $IdxG(GPK, \mathbf{S}^*, w_\rho)$ algorithm with the challenge access structure $\mathbf{S}^*$.

Phase 2. A can perform IdKey query, token query, and index query to B; the queries sent by A must also meet the above conditional restrictions.

Guess. A makes a guess ρ' for ρ; if $\rho' = \rho$, it wins.

The advantage of the adversary A in this game is described as $\Pr[\rho = \rho'] - 1/2$.

If the advantages of any PPT adversary defined above are negligible, then a BC-SABE scheme is IND-sCKA secure.

5. Construction

5.1. System Init. $Cu = (Cu_1, Cu_2, \cdots, Cu_m)$ is a consensus node set which has m consensus nodes. First, consensus nodes exploit the Pedersen (l, m) secret sharing protocol [17] and the reciprocal protocol [18] to run the $Setup(1^k, \sigma) \longrightarrow GPK$ to generate the global public key GPK, the user revocation list rl, and the user tree UT, where $\sigma \in \{0, 1\}^{poly(k)}$ is a randomly public string. Let G be groups of a prime order p, g is a generator of G, and $e : G \times G \longrightarrow G_1$ is a bilinear map. Then, it chooses random $u_0, \cdots, u_\eta, h_0, \cdots, h_\tau \in G$. Let $P(x) = \prod_{j=0}^{\eta} u_j^{x_j}$ and $F(x) = \prod_{j=0}^{\tau} h_j^{x_j}$, $H : \{0, 1\}^* \longrightarrow Z_p^*$ be the collision-resisted hash function, and $D = (e, G, G_1, p, g, H)$ be the admissible bilinear group parameters. Then, Cu_i share two secret parameters $a, r \in G$ and using Pedersen (l, m) secret sharing scheme and reciprocal protocol to compute shares of r^{-1}. Based on its shares a_i, r_i, and j_i, each consensus node Cu_i computes and broadcasts g^{a_i}, g^{r_i}, and g^{j_i}; excess l consensus nodes cooperate to recreate $g^{r^{-1}} = \prod_{i=1}^{l} (g^{j_i})^{L(i)} = g^{\sum_{i=1}^{l} L(i) \cdot j_i}$; and each nodes spreads $g^{a_i/r}$. The public parameters of the system are also generated in this way: $g^r = \prod_{i=1}^{l} (g^{r_i})^{L(i)} = g^{\sum_{i=1}^{l} L(i) \cdot r_i}$, $e(g, g)^a = \prod_{i=1}^{l} (e(g, g)^{a})^{L(i)} = e(g, g)^{\sum_{i=1}^{l} L(i) \cdot a_i}$, $g^{\alpha/\gamma} = \prod_{i=1}^{l} (g^{a_i/r})^{L(i)} = g^{\sum_{i=1}^{l} L(i) \cdot a_i/r}$.Finally, it outputs global public key:

$$GPK = \left(D, P(x), F(x), u_0, \cdots, u_\eta, h_0, \cdots, h_\tau, e(g,g)^a, g^r, g^{a/r}\right). \tag{1}$$

5.2. User Registration and Revocation

5.2.1. User Registration. The IdKG algorithm inputs the *GPK* , the user identity Id_U, and UT; it picks random $g_3 \in G$ and $b_1, b_2 \in Z_p$; and then, it calculates $g_1 = g_3^{1/b_1}$, $g_2 = g_3^{1/b_2}$. The user runs this algorithm to generate the user public and private key pair $(pk_{Id_U}, sk_{Id_U}) = ((g_1, g_2, g_2), (b_1, b_2))$, and then the user sends pk_{Id_U} to BC. An undefined leaf node φ is selected by BC from the UT to storage Id_U and pk_{Id_U}.

5.2.2. User Revocation. The revoke algorithm is run by the consensus node to update the user revocation list *rl* whenever user want to be revoked. It inputs the user identity Id_U, a state mark *st*, a time period *t*, and *rl*; then, it will find all nodes y associated with the identity Id_U and put (y, t) into the *rl* list and outputs the revised *rl*.

5.3. Key Generation

5.3.1. Public Decryption Key Gen. Consensus nodes run the PubDecKG algorithm to generate the public decryption key $PubDK_{Id_U}$; it takes *GPK*, user identity Id_U, and a state mark *sm* and an access structure (D, b) as the input, where matrix D is an $n_D \times m_D$ with row label $b(i)$, $i \in \{1, 2, \cdots, l\}$. Then, it selects random the leaf node φ from the UT based on Id_U, and compute $path(\varphi)$. It selects random $v_{y,2}, \cdots, v_{y,m_D} \in Z_p$ and let $\vec{v} = (a, v_{y,2}, \cdots, v_{y,m_D})$, and then, it computes $v_{y,i} = D_i \cdot \vec{v}_y$ for $i \in [n_D]$, where D_i denotes the vector of the i-th row of matrix D. For each $y \in path(\varphi)$, it takes g_y and computes $g'_{y,i} = g^{v_{y,i}}/g_y$ and stores g_y in the node y. Then, share γ_1, γ_2, $\{\gamma_{y,j}\}_{j \in Id_U}$ among consensus nodes by exploiting the Pedersen (l, m) secret sharing scheme, and each consensus nodes computes and spreads $\{g^{\gamma_{y,j,i}}, F(b(i))^{\gamma_{y,j,i}}\}_{j \in Id_U}$ based on share $\{\gamma_{y,j,i}\}_{j \in Id_U}$. Then, excess l consensus nodes cooperate to compute the public decryption key as follows:

$$\begin{aligned} D_{y,1,j} &= g_3^{\gamma_1+\gamma_2} \cdot g'_{y,i} \cdot \prod_{i=1}^{l} \left(F(b(i))^{\gamma_{y,j,i}}\right)^{L(i)} \\ &= g_3^{\gamma_1+\gamma_2} \cdot g^{v_{y,j}}/g_y \cdot \prod_{i=1}^{l} \left(F(b(i))^{\gamma_{y,j,i}}\right)^{L(i)} \\ &= g_3^{\gamma_1+\gamma_2} \cdot g^{v_{y,j}}/g_y \cdot F(b(i))^{\sum_{i=1}^{l} \gamma_{y,j,i} \cdot L(i)}, \\ &= g_3^{\gamma_1+\gamma_2} \cdot g^{v_{y,j}}/g_y \cdot F(b(i))^{\gamma_{y,j}} \end{aligned} \tag{2}$$

$D_{y,2,j} = \prod_{i=1}^{l} (g^{\gamma_{y,j,i}})^{L(i)} = g^{\sum_{i=1}^{l} \gamma_{y,j,i} \cdot L(i)} = g^{r_{g,i}}$, $D_3 = g_1^{\gamma_1}$, $D_4 = g_2^{\gamma_2}$. Finally, BC stores PDK_{Id} in the ledger:

$$PDK_{Id} = \left\{ \left\{ y, \{D_{y,1,j}, D_{y,2,j}\}_{i \in [l]} \right\}_{y \in path(\varphi)}, D_3, D_4 \right\}. \tag{3}$$

5.3.2. Key Update Messages Gen. Consensus nodes run the UpKG algorithm to generate the update information U_t; it inputs the *GPK*, the revocation list *rl*, a time period *t*, the user tree UT, and a state mark *sm*. Share s_y among consensus nodes by exploiting the Pedersen (l, m) secret sharing scheme, and each consensus nodes computes and spreads $g^{s_{y,i}}, P(t)^{s_{y,i}}$ based on share $s_{y,i}$.

For all $y \in KUNodes(UT, rl, t)$, it takes g_y from the node y. Then, more than l consensus nodes cooperate to compute the update information as follows: $U_{y,1} = g_y \cdot \prod_{i=1}^{l} (P(t)^{s_{y,i}})^{L(i)} = g_y \cdot P(t)^{\sum_{i=1}^{l} s_{y,i} \cdot L(i)} = g_y \cdot P(t)^{s_y}$, $U_{y,2} = \prod_{i=1}^{l} (g^{s_{y,i}})^{L(i)} = g^{\sum_{i=1}^{l} s_{y,i} \cdot L(i)} = g^{s_y}$. BC stores U_t in the ledger:

$$U_t = \left\{ \left\{ y, U_{y,1}, U_{y,2} \right\}_{y \in KUNodes(UT, rl, t)} \right\}. \tag{4}$$

5.3.3. Predecryption Key Gen. Consensus nodes run the PreDecKG algorithm to generate the update information $PreDK_t$; it inputs the *GPK*, a time period *t*, user identity Id_U, an access structure (D, b), the user tree UT, a state mark *sm*, the revocation list *rl*, user public decryption key $PubDK_{Id}$, and a update information U_t. Then, let $I = Path(\theta)$ and $J = KUNodes(UT, rl, t)$, so we have $U_t = (\{y, U_{y,1}, U_{y,2}\}_{y \in J})$ and $PDK_{Id} = (\{y, \{D_{y,1,j}, D_{y,2,j}\}_{i \in [l]}\}_{y \in path(\varphi)}, D_3, D_4)$. If $I \cap J = \varnothing$, it returns $\perp$. Then, share s_y and $\{\gamma_{y,j}\}_{j \in Id_U}$ among consensus nodes by exploiting the Pedersen (l, m) secret sharing protocol. Based on share $s'_{y,i}$ and $\{\gamma'_{y,j,i}\}_{j \in Id_U}$, each consensus nodes computes and spreads $\{g^{\gamma'_{y,j,i}}, F(b(i))^{\gamma'_{y,j,i}}\}_{j \in Id_U}$, $g^{s'_{y,i}}, P(t)^{s'_{y,i}}$. Then, more than l consensus nodes cooperate to compute the predecryption key.

$$\begin{aligned} Tk_{1,j} &= D_{y,1,j} \cdot U_{y,1} \cdot \prod_{i=1}^{l} \left(F(b(i))^{\gamma'_{y,j,i}} \cdot P(t)^{s_{y,i}}\right)^{L(i)} \\ &= g_3^{\gamma_1+\gamma_2} \cdot g^{v_{y,j}}/g_y \cdot g_y \cdot P(t)^{s_x} \cdot \prod_{i=1}^{l} \left(F(b(i))^{\gamma'_{y,j,i}} \cdot P(t)^{s'_{y,i}}\right)^{L(i)} \\ &= g_3^{\gamma_1+\gamma_2} \cdot g^{v_{y,j}} \cdot P(t)^{s_y} \cdot F(b(i))^{\sum_{i=1}^{l} \gamma'_{y,j,i} \cdot L(i)} \cdot P(t)^{\sum_{i=1}^{l} s'_{y,i} \cdot L(i)} \\ &= g_3^{\gamma_1+\gamma_2} \cdot g^{v_{y,j}} \cdot F(b(i))^{\gamma_{y,j}+\gamma'_{y,j}} \cdot P(t)^{s_y+s_y'}, \end{aligned} \tag{5}$$

$Tk_{2,j} = D_{y,2,j} \cdot \prod_{i=1}^{l} (g^{\gamma'_{y,i}})^{L(i)} = g^{\gamma_y+\gamma_y'}$,
$Tk_3 = U_{y,2} \cdot \prod_{i=1}^{l} (g^{s'_{y,i}})^{L(i)} = U_{y,2} \cdot g^{\sum_{i=1}^{l} s'_{y,i} \cdot L(i)} = g^{s_y+s_y'}$. BC stores $PreDK_{Id_U,t}$ in the ledger.

$$\text{Pre}DK_{Id_U,t} = \left\{ \left\{ Tk_{1,j}, Tk_{2,j} \right\}_{j\in[l]}, Tk_3 \right\}. \quad (6)$$

5.4. Encryption

5.4.1. Data Encryption. Date owner runs the Enc algorithm to get the ciphertext CT. It inputs the GPK, an attribute set $\mathbf{S}$, and a symmetric key k_s and a time period t. Let $\mathbf{S} = \{\mathbf{S}_1, \cdots, \mathbf{S}_l\}$ and choose $\mu \in Z_p$. It calculates $C_0 = e(g,g)^{a\mu} \cdot k_s$, $C_1 = g^{\mu}$, $C_{2,i} = F(\mathbf{S}_i)^{\mu}$, and $C_3 = P(t)^{\mu}$ and outputs CT_{sym} and $CT = (\mathbf{S}, t, C_0, C_1, \{C_{2,i}\}_{i\in[l]}, C_3)$.

5.4.2. Index Generation. Date owners run the IdxG algorithm to generate the Idx. It inputs the GPK, the same attribute set S, and a keyword set $\{w_l\}_{l\in U_w}$, and then, it computes $Idx_{1,l} = e(g,g)^{a\mu\cdot H(w_l)}$, $Idx_2 = g^{r\mu}$, $Idx_{3,i} = g^{rv_i}$, and $Idx_{4,i} = F(b(i))^{v_i}$. Finally, it sent index set Idx along with CT_{sym} and CT to the cloud.

$$Ind = \left\{ \left\{ Idx_{1,l} \right\}_{l\in U_w}, Idx_2, \left\{ Idx_{3,i}, Idx_{4,i} \right\}_{i\in[n_D]}, \mathbf{S}^* \right\}. \quad (7)$$

5.5. Token Gen

5.5.1. Partial Token Gen. In this phase, data users run the PTokG algorithm to get the partial token. It inputs the GPK, user private key sk_{Id_U}, and a keyword w. Then, it randomly selects $q \in Z_p$; it computes the partial token $\text{To}k' = (g^{a/r})^{(b_1+b_2+q)}$ and the hash value $H(w)$. Finally, it sends $\text{To}k'$ and $H(w)$ to BC.

5.5.2. Complete Token Gen. Blockchain runs the TokG algorithm to get the complete token. It inputs the GPK, the user identity Id_U, the same access structure (D, b), and the partial token $\text{To}k'$. Share $\{\alpha_j\}_{j\in Id_U}$ among consensus nodes by exploiting Pedersen (l, m) secret sharing protocol. Based on share $\{\alpha_{j,i}\}_{j\in Id_U}$, each consensus nodes computes and spreads $\{g^{r\cdot\alpha_{j,i}}, F(b(i))^{a_{j,i}}\}_{j\in Id_U}$. Then, more than l consensus nodes cooperate to compute the complete token as follows:

$$\begin{aligned} Tok_{1,j} &= \text{To}k' \cdot \prod_{i=1}^{l} \left(F(b(i))^{\alpha_{j,i}}\right)^{L(i)} \\ &= \left(g^{a/r}\right)^{(b_1+b_2+r)} \cdot F(b(i))^{\sum_{i=1}^{l} \alpha_{j,i}\cdot L(i)} \\ &= \left(g^{a/r}\right)^{(b_1+b_2+r)} \cdot F(b(i))^{\alpha_j}, \end{aligned} \quad (8)$$

$Tok_{2,j} = \prod_{i=1}^{l} \left(g^{r\cdot\alpha_{j,i}}\right)^{L(i)} = g^{r\cdot\sum_{i=1}^{l}\alpha_{j,i}\cdot L(i)} = g^{r\cdot\alpha_j}$, $Tok_3 = \text{To}k' \cdot \left(g^{a/r}\right)^{H(w)} = \left(g^{a/r}\right)^{(b_1+b_2+H(w))}$. Finally, it sends the token Tok to the cloud:

$$Tok = \left\{ \left\{ Tok_{1,j}, Tok_{2,j} \right\}_{j\in Id_U}, Tok_3 \right\}. \quad (9)$$

5.6. Search. Cloud runs search algorithms to perform search operations; it takes the GPK, the same access structure (D, b), the complete token Tok, and the Ind as inputs. It yields the capacity address $Addr$ of related ciphertext if and only if the algorithm runs successfully; otherwise, the algorithm stops. First, it verifies whether the user's attribute set satisfies the access structure, and if not, it outputs a stop character $\perp$. If it is satisfied, let $T = \{i \in [n_D] \mid b(i) \in \mathbf{S}\}$, and then the algorithm can calculate a set $\{d_i \in Z_p\}_{i\in T}$ which makes $\sum_{i\in T} d_i D_i = (1, 0, \cdots, 0)^{m_D}$. Then, it verifies whether the equation described is valid.

$$Idx_{1,l} = \frac{e(Idx_2, Tok_3)}{\prod_{i\in T}\left(e\left(Tok_{1,b(i)}, Idx_{3,i}\right)/e\left(Tok_{3,b(i)}, Idx_{4,i}\right)\right)^{d_i}}. \quad (10)$$

If the formula holds, the stored address $Addr$ is the output.

5.7. Decryption

5.7.1. Predecryption. The predecryption operation is performed by the cloud, which runs the PreD algorithm to complete. It inputs the GPK, user identity Id_U, the predecryption key $\text{Pre}DK_{Id_U,t}$, a time period t, the same set $T = \{i \in [n_D] \mid b(i) \in \mathbf{S}\}$, and the ciphertext CT. If $\{v_i\}$ are valid shares of any secret μ from D, then the algorithm can compute a set $\{d_i \in Z_p\}_{i\in T}$ which makes $\sum_{i\in T} d_i v_i = \mu$. It computes

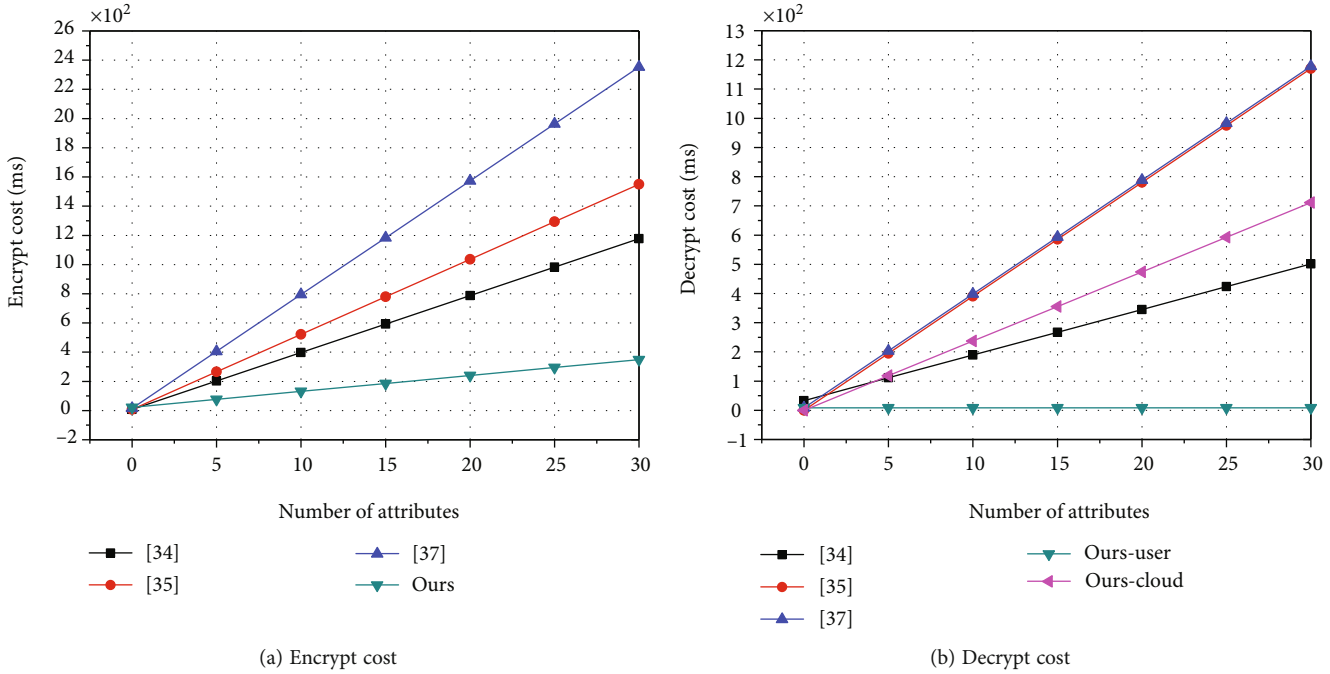
$$\begin{aligned} CT' &= \prod_{i\in T}\left(\frac{e\left(C_{2,i}, Tk_{2,b(i)}\right)e(C_3, Tk_3)}{e\left(Tk_{1,b(i)}, C_1\right)}\right)^{d_i} \\ &= \frac{1}{e\left(g_3^{\gamma_1+\gamma_2}, C_1\right)e(g, C_1)^a}. \end{aligned} \quad (11)$$

5.7.2. Decryption. The Dec algorithm inputs the GPK, private user key sk_{Id_U}, the predecryption ciphertext CT', and the symmetric decryption algorithm to generate the symmetric decryption key. This algorithm is run by the DU, and it can finish full decryption. The decryption is as follows. Using k_s to run symmetric algorithms can enable users to get plaintext, and users will not consume a lot of costs because the previous operations are already performed in the cloud.

$$\begin{aligned} k_s &= CT' \cdot e\left(D_3^{b_1} D_4^{b_2}, C_1\right) \cdot C_0 \\ &= \frac{e\left(g_1^{b_1\gamma_1} g_2^{b_2\gamma_2}, g^{\mu}\right) \cdot e(g,g)^{a\mu} \cdot k_s}{e\left(g_3^{\gamma_1+\gamma_2}, g^{\mu}\right)e(g, g^{\mu})^a} \\ &= \frac{e\left(g_3^{\gamma_1+\gamma_2}, g^{\mu}\right) \cdot e(g,g)^{a\mu} \cdot k_s}{e\left(g_3^{\gamma_1+\gamma_2}, g^{\mu}\right)e(g, g^{\mu})^a}. \end{aligned} \quad (12)$$

Table 1: Performance comparison. Five aspects were compared with the other four schemes.

Schemes	Encrypt cost	Index gen	Tok gen	Search	Decrypt cost
[34]	$Ep_T + (5N_e + 1)Ep$	No	No	No	$2N_T Pa + 3(EP_T + Ep)$
[35]	$(N_e + 1)Pa + 4N_e Ep$	No	No	No	$5N_T Pa + Ep_T$
[36]	$SymEnc$	$(2N_e + N_w + 3)Ep$	$(2N_t + 3)Ep$	$Ep_T + (2N_s + 4)Pa$	$SymDec$
[37]	$(3N_e + 2)Pa + 5N_e Ep$	No	No	No	$(5N_T + 1)Pa + 2Ep_T$
Ours	$Sym + (N_e + 2)Ep + Ep_T$	$(2N_e + 1)Ep + N_w Ep_T$	User:Ep BC:$2N_s Ep + Ep_T$	$N_T Ep_T + (2N_T + 1)Pa$	User:$2Ep_T + Sym + Pa$ Cloud:$2N_T Ep_T + (3N_T)Pa$.



(a) Encrypt cost

(b) Decrypt cost

Figure 4: Enc/Dec time cost comparison. It can be seen that our scheme is not very expensive in terms of encryption and decryption.

6. Security Proof

Theorem 4. *If Pedersen (l, m) secret sharing protocol is IND-CPA secure and the SR-ABE scheme presented by Cui et al. [31] is IND-CPA secure, then our BC-SABE scheme is IND-CPA secure.*

Proof: In contrast to the scheme in [31], distributed consortium blockchain is used in our scheme, replacing the centralized server in [31]. This allows the security of the whole system to be improved. Supposing that the adversary in our scheme can compromise at most l-1 authority, the reasons are as follows: Our PubDKey Oracle, the UpKey Oracle, and the PreDKey Oracle have excellent performance because it needs more than l authorities that are required to execute together. In addition, during the challenge phase, we defined related restrictions. Therefore, the proof of this scheme can be deduced from the security proof of the scheme [31] under the security of Pedersen (l, m) secret sharing protocol, and reciprocity protocol.

Theorem 5. *Under the DBDH assumption, The BC-SABE scheme is IND-sCKA secure.*

Proof: Assuming that there is a PPT adversary A who can win the exponential indistinguishability game with an advantage ε that cannot be ignored, then the challenge B is constructed to resolve the DBDH problem with an advantage $\varepsilon/2$ that cannot be ignored.

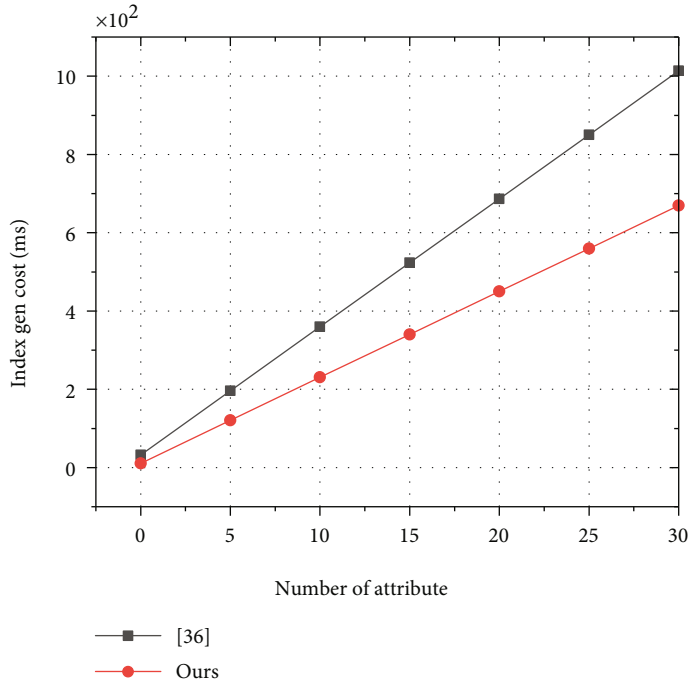
Init. Let $\mathbf{S}^*$ be the challenge access structure defined by A.

Setup. B returns the GPK to A by running the $Setup(1^k, \sigma)$ algorithm, the difference in GPK is $e(g, g)^a = e(g^\beta, g^\gamma) = e(g, g)^{\beta\gamma}$ and $r \in Z_p^*$, and other parameters are ignored here.

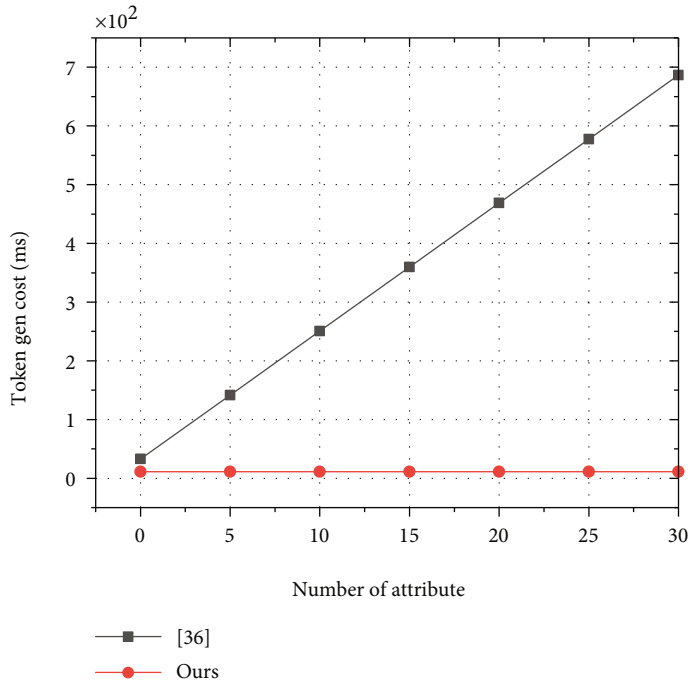
Phase 1. A can adaptively execute the following queries:

(i) IdKey query: A issues a IdKey query on an identity Id_U, B returns (sk_{Id_U}, pk_{Id_U}) by running $IdKG(GPK, Id_U)$, and then, it adds $(Id_U, sk_{Id_U}, pk_{Id_U})$ to list L and returns sk_{Id_U} to A

(ii) Token query: A issues a token query on Id_U, an access structure $\mathbf{S}$, and a keyword w. B runs $PTokG(GPK, sk_{Id_U}, w) \longrightarrow \mathrm{Tok}'$ by using sk_{Id_U}. Then, B

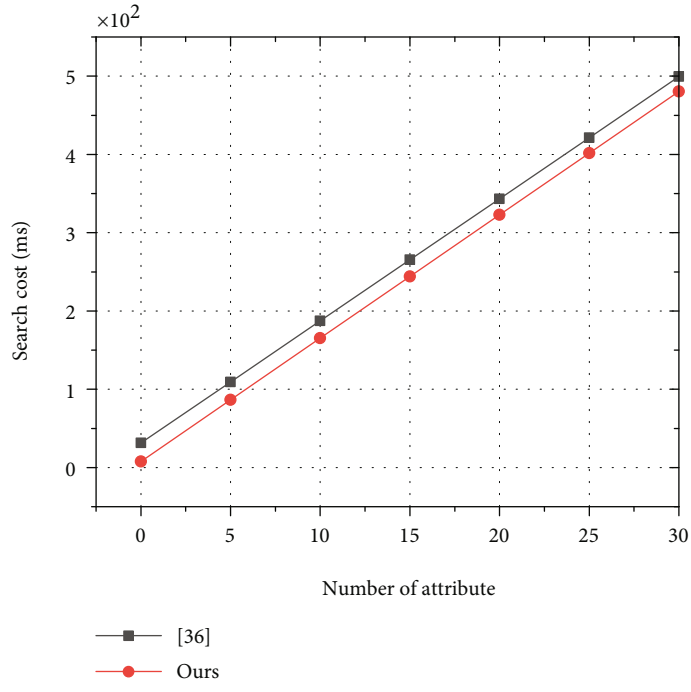


(a) Index gen cost



(b) Tok gen cost

Figure 5: Continued.



(c) Search cost

FIGURE 5: Other time cost comparison. It can be seen that our scheme is better in terms of index generation, token generation, and search compared to scheme [36].

returns Tok to A by running the $TokG(GPK, Id_U, (\mathscr{D}, \rho), \mathrm{Tok}')$ algorithm. The attribute set $\mathbf{S}$ does not meet challenge access structure

(iii) Index query: A issues an index query on a keyword set $\{w\}$ and $\mathbf{S}$. Then, B returns the index set by running $IdxG(GPK, \mathbf{S}, \{w\})$

Challenge. A submits two keywords w_1 and w_2 of the same size. B picks random $\rho \in \{0, 1\}$, and then, it returns the $Ind^* = (\{Idx^*_{1,l}\}_{l \in U_w}, Idx^*_2, \{Idx^*_{3,i}, Idx^*_{4,i}\}_{i \in [n_D]}, \mathbf{S}^*)$ to A by running the $IdxG(GPK, \mathbf{S}^*, w_\rho)$ algorithm with the challenge access structure $\mathbf{S}^*$, where $Idx^*_{1,w_\rho} = Z^{H(w_\rho)}$, $Idx^*_2 = g^{\alpha r}$, $Idx^*_{3,i} = g^{rv_i}$, and $Idx^*_{4,i} = F(b(i))^{v_i}$.

Two different situations require attention as follows:

(1) $P = e(g, g)^{\alpha\beta\gamma}$. Let $\mu = \alpha$ and $a = \beta\gamma$, and then, the index set obtained in this case is the real index. $Idx^*_{1,w_\rho} = e(g, g)^{a\mu \cdot H(w_\rho)}$, $Idx^*_2 = g^{\mu r}$, $Idx^*_{3,i} = g^{rv_i}$,$Idx^*_{4,i} = F(b(i))^{v_i}$

(2) $P = e(g, g)^p$. In this case, A cannot get information about ρ because of the randomness of p

Phase 2. A can perform IdKey query, token query, and index query to B. The queries sent by A must also meet the above conditional restrictions.

Guess. A performs a guess ρ' for ρ; if $\rho' = \rho$, it wins. If it is case one $P = e(g, g)^{\alpha\beta\gamma}$, then $\Pr[\rho' = \rho] = \varepsilon + 1/2$; if it is case two $P = e(g, g)^p$, then $\Pr[\rho' = \rho] = 1/2$. Finally, we can get that the probability that B can resolve DBDH assumption is

$$\left|\frac{1}{2}\Pr\left[\rho' = \rho\right] \mid P = e(g, g)^{\alpha\beta\gamma}\right| + \left|\frac{1}{2}\Pr\left[\rho' = \rho\right] \mid P = e(g, g)^{z}\right| = \left|\left[\frac{1}{2}\left(\frac{1}{2} + \varepsilon\right) + \frac{1}{2} \cdot \frac{1}{2}\right] - \frac{1}{2}\right| = \frac{\varepsilon}{2}. \tag{13}$$

$\varepsilon/2$ is negligible due to the difficulty of the BDDH problem; that is, it is negligible that A can break the advantage of our scheme; that is to say, our scheme is security.

7. Performance Comparison

The performance of other related scheme [34–37] is compared with this scheme in this section. Let Ep and Ep_T denote exponential operation in G and G_1; Pa denotes pairing operation. For convenience, N_T indicates the number of attributes in the decryption operation and search operation in the system, N_e indicates the number of attributes in the encryption operation in the system, N_s denotes the number of attributes in the token generation in the system, and N_w indicates the number of attributes in the index generation in the system. Let symmetric encryption and decryption operations expressed as Sym.

We use JPBC library version 2.0.0 for related experiments. The experiment was simulated on Windows system with an Intel(R) Core (TM) i5 CPU 3.20GHz and 8.00GB

RAM to approximate the actual operation. We have obtained the measured values of exponentiation and pairing operations. The operating times of Ep, Pa, and Ep_T are 10.9 ms, 7.8 ms, and 0.15 ms, respectively. Table 1 shows the comparison of our scheme with other schemes in terms of encryption cost, decryption cost, and other aspects.

Figure 4 shows the comparison of the cost of encryption and decryption between our scheme and the other three multiauthority attribute-based encryption schemes. It is not difficult to see that the encryption and decryption time has a linear relationship with the number of attributes. Our scheme shifts the decryption process to operate on the cloud server, which makes the user's computational cost effectively reduced.

Figure 5 compares our scheme with the scheme [36]. It is not difficult to see from the figure that our scheme is highly efficient in index generation, search, and token generation stages. Among them, in the token generation stage, our scheme transfers the work of token generation to the blockchain node, and users only need to generate part of the token.

Obviously, because most of the calculation and storage work in the scheme is handed over to cloud servers and blockchain nodes, this makes our scheme more efficient in all aspects, especially in user decryption and token generation. Although the performance of some algorithms will be affected by the throughput of the blockchain and other factors, the security of the scheme will not be affected.

8. Conclusion

In this essay, we have presented a new BC-SABE scheme that replaces the centralized key management server in [31] using a consortium blockchain. The consortium blockchain consist of a trusted set of consensus nodes and is responsible for jointly generating the relevant partial parameters. We can guarantee the confidentiality of data transmission using the Pedersen secret sharing protocol, which enables sharing of subsecrets among consensus nodes, and the reciprocity protocol ensures that key information is shared without a trusted party. The update of the user revocation list is also performed entirely by the blockchain without re-encrypting the ciphertext. In addition, we move the predecryption operations to be performed in the cloud, and users are able to fully decrypt them with only a small amount of computation. Performance analysis shows that this scheme is more efficient compared to other schemes.

Data Availability

We guarantee the confidentiality of data transmission.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Acknowledgments

This research was funded by the National Natural Science Foundation of China under Grant No. 61902140, No. 61972095, No. 62072104, and No. U21A20465; the Anhui Provincial Natural Science Foundation under Grant No. 1908085QF288; the Nature Science Foundation of Anhui Higher Education Institutions under Grant No. KJ2021A0527, No. KJ2019A0605, No. KJ2020A0034, and No. KJ2020A0032; the Natural Science Foundation of the Fujian Province under Grant No. 2020 J01159; and the Key Projects of Science and Technology of Henan Province under Grant No. 222102210043, No. 222102210173, and No. 222102210209.

References

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): a vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[2] J. Feng, L. Liu, and Q. Pei, "Min-max cost optimization for efficient hierarchical federated learning in wireless edge networks," *IEEE Transactions on Parallel and Distributed Systems*, 2021.

[3] Y. Chen, J. Li, C. Liu, J. Han, Y. Zhang, and P. Yi, "Efficient attribute based server-aided verification signature," *IEEE Transactions on Service Computing*, 2021.

[4] J. Li, Y. Chen, J. Han, C. Liu, Y. Zhang, and H. Wang, "Decentralized attribute-based server-aid signature in the internet of things," *IEEE Internet of Things Journal*, 2021.

[5] L. Liu, M. Zhao, and M. Yu, "Mobility-aware multi-hop task offloading for autonomous driving in vehicular edge computing and networks," *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[6] L. Liu, C. Chen, Q. Pei, S. Maharjan, and Y. Zhang, "Vehicular edge computing and networking: a survey," *Mobile Networks and Applications*, vol. 26, no. 3, pp. 1145–1168, 2021.

[7] P. Shi, H. Wang, S. Yang, C. Chen, and W. Yang, "Blockchain-based trusted data sharing among trusted stakeholders in IoT," *Software: Practice and Experience*, vol. 51, no. 10, pp. 2051–2064, 2021.

[8] A. Sahai and B. R. Waters, "Fuzzy identity-based encryption," *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, vol. 3494, pp. 457–473, 2005.

[9] 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, 2006.

[10] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *2017 IEEE Symposium on Security and Privacy*, pp. 321–334, Berkeley, CA, USA, 2007.

[11] J. Li, W. Yao, Y. Zhang, H. Qian, and J. Han, "Flexible and fine-grained attribute-based data storage in cloud computing," *IEEE Transactions on Services Computing*, vol. 10, no. 5, pp. 785–796, 2017.

[12] J. Li, W. Yao, J. Han, Y. Zhang, and J. Shen, "User collusion avoidance CP-ABE with efficient attribute revocation for cloud storage," *IEEE Systems Journal*, vol. 12, no. 2, pp. 1767–1777, 2018.

[13] J. Li, Q. Yu, Y. Zhang, and J. Shen, "Key-policy attribute-based encryption against continual auxiliary input leakage," *Information Sciences*, vol. 470, pp. 175–188, 2019.

[14] Q. Yu, J. Li, and S. Ji, "Fully secure ID-based signature scheme with continuous leakage-resilience," *Security and Communication Networks*, vol. 2022, 12 pages, 2022.

[15] P. K. Malik, R. Sharma, R. Singh et al., "Industrial internet of things and its applications in industry 4.0: state of the art," *Computer Communications*, vol. 166, pp. 125–139, 2021.

[16] L. Liu, J. Feng, Q. Pei et al., "Blockchain-enabled secure data sharing scheme in mobile-edge computing: an asynchronous advantage actor–critic learning approach," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2342–2353, 2020.

[17] T. P. Pedersen, "A threshold cryptosystem without a trusted party," *Workshop on the Theory and Application of Cryptographic Techniques*, vol. 547, pp. 522–526, 1991.

[18] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, "Robust threshold DSS signatures," *International Conference on the Theory and Applications of Cryptographic Techniques*, vol. 1070, pp. 354–371, 1996.

[19] K. Zhang, Y. Li, Y. Song, L. Lu, T. Zhang, and Q. Jiang, "A traceable and revocable multiauthority attribute-based encryption scheme with fast access," *Security and Communication Networks*, vol. 2020, 14 pages, 2020.

[20] X. Gao and L. Zhang, "Efficient anonymous ciphertext-policy attribute-based encryption for general structures supporting leakage-resilience," *International Journal of Network Security*, vol. 22, no. 5, pp. 763–774, 2020.

[21] H. Nasiraee and M. Ashouri-Talouki, "Anonymous decentralized attribute-based access control for cloud-assisted IoT," *Future Generation Computer Systems*, vol. 110, pp. 45–56, 2020.

[22] M. Ali, J. Mohajeri, M. R. Sadeghi, and X. Liu, "A fully distributed hierarchical attribute-based encryption scheme," *Theoretical Computer Science*, vol. 815, pp. 25–46, 2020.

[23] J. Li, Q. Yu, and Y. Zhang, "Hierarchical attribute based encryption with continuous leakage-resilience," *Information Sciences*, vol. 484, pp. 113–134, 2019.

[24] J. Li, N. Chen, and Y. Zhang, "Extended file hierarchy access control scheme with attribute based encryption in cloud computing," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 2, pp. 983–993, 2021.

[25] N. Chen, J. Li, Y. Zhang, and Y. Guo, "Efficient CP-ABE scheme with shared decryption in cloud storage," *IEEE Transactions on Computers*, vol. 71, no. 1, pp. 175–184, 2022.

[26] M. Green, S. Hohenberger, and B. Waters, "Outsourcing the decryption of abe ciphertexts," in *Proceedings of the 20th USENIX conference on Security*, pp. 1–16, San Francisco, CA, 2011.

[27] H. El Gafif and A. Toumanari, "Efficient ciphertext-policy attribute-based encryption constructions with outsourced encryption and decryption," *Security and Communication Networks*, vol. 2021, 17 pages, 2021.

[28] C. Feng, K. Yu, M. Aloqaily, M. Alazab, Z. Lv, and S. Mumtaz, "Attribute-based encryption with parallel outsourced decryption for edge intelligent IoV," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 11, pp. 13784–13795, 2020.

[29] J. Li, Y. Wang, Y. Zhang, and J. Han, "Full verifiability for outsourced decryption in attribute based encryption," *IEEE Transactions on Services Computing*, vol. 13, no. 3, pp. 478–487, 2020.

[30] J. K. Liu, T. H. Yuen, P. Zhang, and K. Liang, "Time-based direct revocable ciphertext-policy attribute-based encryption with short revocation list," *International Conference on Applied Cryptography and Network Security*, vol. 10892, pp. 516–534, 2018.

[31] H. Cui, T. Hon Yuen, R. H. Deng, and G. Wang, "Server-aided revocable attribute-based encryption for cloud computing services," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 14, article e5680, 2020.

[32] M. Chase, "Multi-authority attribute-based encryption," in *Theory of cryptography conference*, pp. 515–534, Springer, Berlin, Heidelberg, 2007.

[33] H. Qian, J. Li, Y. Zhang, and J. Han, "Privacy preserving personal health record using multi-authority attribute-based encryption with revocation," *International Journal of Information Security*, vol. 14, no. 6, pp. 487–497, 2015.

[34] S. Belguith, N. Kaaniche, M. Laurent, A. Jemai, and R. Attia, "Phoabe: securely outsourcing multi-authority attribute based encryption with policy hidden for cloud assisted ioT," *Computer Networks*, vol. 133, pp. 141–156, 2018.

[35] K. Sethi, A. Pradhan, and P. Bera, "Practical traceable multi-authority CP-ABE with outsourcing decryption and access policy updation," *Journal of Information Security and Applications*, vol. 51, article 102435, 2020.

[36] Y. Miao, J. Ma, and X. Liu, "Practical attribute-based multi-keyword search scheme in mobile crowdsourcing," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 3008–3018, 2017.

[37] Q. Xu, C. Tan, W. Zhu, Y. Xiao, Z. Fan, and F. Cheng, "Decentralized attribute-based conjunctive keyword search scheme with online/offline encryption and outsource decryption for cloud computing," *Future Generation Computer Systems*, vol. 97, pp. 306–326, 2019.

[38] X. Sun, H. Wang, X. Fu et al., "Substring-searchable attribute-based encryption and its application for IoT devices," *Digital Communications and Networks*, vol. 7, no. 2, pp. 277–283, 2021.

[39] J. Li, X. Lin, Y. Zhang, and J. Han, "KSF-OABE: outsourced attribute-based encryption with keyword search function for cloud storage," *IEEE Transactions on Services Computing*, vol. 10, no. 5, pp. 715–725, 2017.

[40] J. Li, Y. Zhang, J. Ning, X. Huang, G. S. Poh, and D. Wang, "Attribute based encryption with privacy protection and accountability for CloudIoT," in *IEEE Transactions on Cloud Computing*, 2020.

[41] Y. Lu, J. Li, and Y. Zhang, "Secure channel free certificate-based searchable encryption withstanding outside and inside keyword guessing attacks," *IEEE Transactions on Services Computing*, vol. 14, no. 6, pp. 2041–2054, 2021.

[42] Y. Lu, J. Li, and F. Wang, "Pairing-free certificate-based searchable encryption supporting privacy-preserving keyword search function for IIoTs," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 4, pp. 2696–2706, 2021.

[43] Y. Lu, J. Li, and Y. Zhang, "Privacy-preserving and pairing-free multi-recipient certificateless encryption with keyword search for cloud-assisted IIoTs," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 2553–2562, 2020.

[44] Y. Lu, J. Li, and F. Wang, "Lightweight public key authenticated encryption with keyword search against adaptively-chosen-targets adversaries," *IEEE Transactions on Mobile Computing*, 2021.

[45] S. Nakamoto, *Bitcoin: A peer-to-peer electronic cash system*, Decentralized Business Review, 2008.

[46] A. Wu, Y. Zhang, X. Zheng, R. Guo, Q. Zhao, and D. Zheng, "Efficient and privacy-preserving traceable attribute-based encryption in blockchain," *Annals of Telecommunications*, vol. 74, no. 7-8, pp. 401–411, 2019.

[47] S. M. Pournaghi, M. Bayat, and Y. Farjami, "MedSBA: a novel and secure scheme to share medical data based on blockchain technology and attribute-based encryption," *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, no. 11, pp. 4613–4641, 2020.

[48] S. Liu, J. Yu, Y. Xiao, Z. Wan, S. Wang, and B. Yan, "bc-sabe: blockchain-aided searchable attribute-based encryption for cloud-IoT," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 7851–7867, 2020.

[49] H. Zheng, J. Shao, and G. Wei, "Attribute-based encryption with outsourced decryption in blockchain," *Peer-to-Peer Networking and Applications*, vol. 13, no. 5, pp. 1643–1655, 2020.

[50] L. Guo, X. Yang, and W. C. Yau, "TABE-DAC: efficient traceable attribute-based encryption scheme with dynamic access control based on blockchain," *IEEE Access*, vol. 9, pp. 8479–8490, 2021.

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