Information sharing has become an important application in modern supply chain management systems with business technology development. Because traditional supply chain information systems have problems such as easy data tampering, low information transparency, and interaction delays, blockchain has been taken consideration into supply chain information sharing research. Furthermore, blockchain technology is expected to provide decentralized supply chain information sharing solutions to enhance security, availability, and transparency. However, with the in-depth study of the application of blockchain technology in supply chain information sharing, people have found that the data stored publicly in the blockchain are still threatened by privacy leakage. In addition, due to the openness and accessibility of the blockchain, the lack of fine-grained access control is also apparent. In order to improve the security of data, we propose a novel privacy-preserving multiauthority attribute-based access control scheme for secure blockchain-based information sharing in a supply chain. In this scheme, blockchain stores encrypted supply chain information on distributed nodes. Multiple attribute authorities manage different attributes of users to achieve fine-grained access control and flexible authorization. Even if some attribute authorities fail, the user’s private key will not be leaked. In user secret key generation, we adopt an anonymous key generation protocol to realize the secure distribution of user keys by the attribute authorities. Furthermore, in order to meet the protection of communication privacy between blockchain nodes, properties of policy hiding and identity hiding are considered. Finally, we design experiments to analyze the performance of our scheme, including secret key sizes and running time of encryption and decryption.

1. Introduction

A supply chain is a huge and complex network that tightly integrates suppliers, carriers, manufacturers, and consumers. In the supply chain, information sharing has played an important role in resource utilization, demand analysis, and production management for upstream suppliers and downstream manufacturers in each industry. In addition, as consumers pay more attention to raw materials, product processing, and logistics, the research on the transparency and traceability of supply chain information has become a research hotspot in the field of the supply chain. However, a supply chain is often characterized by high levels of globalization, a large number of participating corporations, and complex product processing links, which contain the confidential and critical private information of corporations and consumers. Therefore, a traditional centralized supply chain system is vulnerable to facing a single point of failure, which makes the entire network and organizations involved in potential danger (e.g., hacking or corruption) [1]. In
addition, since centralized organizations introduce third parties for management, there is also a problem of inefficient access and limited control of information. Therefore, finding decentralized solutions to improve the security, availability, and transparency of information sharing in the supply chain is very crucial.

Blockchain is a novel deneutralized computing model, which is expected to provide decentralized solutions for information sharing in a supply chain system. It was firstly described in the context of Bitcoin [2], which Satoshi Nakamoto designed. Blockchain integrates distributed data storage, independent peering transactions, consensus mechanisms, programmable smart contracts, dynamic encryption algorithms, etc. It enables multiparty transactions in a distributed environment. It also has the characteristics of traceable information sources and is difficult to tamper with transaction data [3]. With the development of blockchain technology, it is increasingly applied in many fields such as medical care [4], banking system [5], and Internet of Things (IoT) applications [6]. In recent years, scholars have paid more attention to the application of blockchain in supply chain information sharing. Abidi et al. [7] constructed a new privacy preservation model in supply chain networks based on blockchain technology, where blockchain plays a promising role for secure information sharing. Venkatesh et al. [8] developed a system architecture that integrates the use of blockchain, IoT, and big data analytics to allow suppliers to efficiently and effectively monitor supply chain social sustainability.

However, with the application of technologies such as deep learning and data mining, the blockchain is still facing the threat of network attacks such as DDoS attacks [9], DNS attacks [10], and deanonymity analysis [11], which hinders the availability of services to users and threatens blockchain node security. Therefore, the data in the blockchain system are not completely secure because it transfers data through address identification similar to e-mail addresses, which may cause user privacy issues. Furthermore, although the address identification is not directly related to the real-world user identity, the blockchain data are completely open and transparent, leading to privacy leaks. So, it is indispensable to optimize the access control scheme for the traditional blockchain-based supply chain information sharing systems.

Attribute-based encryption (ABE) [12] is proposed as an encryption system that inherits the advantages of identity encryption, in which the secret key of a user and the ciphertext are dependent on attributes (e.g., phone number or e-mail address). It can provide fine-grained access control for an information sharing system. Due to the excellent performance of ABE in controlling and managing access to data, it has been considered to be one of the key technologies that can continuously improve the security and privacy performance of a blockchain system in recent years [13]. In this paper, we propose a blockchain-based supply chain information sharing access control scheme based on the privacy-preserving multiauthority ciphertext-policy ABE. This scheme also introduces the user attribute revocation mechanism to enhance flexibility and security.

In our scheme, the main contributions are summarized as follows:

(1) A novel multiauthorities attribute-based encryption scheme has been proposed to solve the problems of privacy leakage and incomplete anonymity in blockchain-based information sharing systems of the supply chain.

(2) We take properties of attribute revocable, policy hiding, identity hiding, and secret key anonymous generation into account to improve the security and flexibility of our scheme.

(3) An anonymous key generation protocol is proposed to solve the problem of secure key issuance between users and attribute authorities.

The rest of the article is structured as follows: Section 2 reviews the relevant issues of the research topic. Section 3 introduces the mathematical preliminaries involved in the paper. Section 4 explains the access control system model and security model. Section 5 describes our work in detail. Section 6 analyzes the correctness and security of the scheme we proposed. Section 7 analyzes the security of the anonymous key generation protocol. The results of the comparison and experiments will be shown in Section 8. Finally, Section 9 summarizes the work of this article and looks forward to future work.

2. Related Works

In this section, we review some existing works associated with the blockchain-based access control scheme and the applications of ABE and MA-ABE on it.

2.1. Blockchain-Based Access Control. The leakage of sensitive data in the block can pose a great threat to the security of the blockchain and its applications. Therefore, the research on the data access control scheme of the blockchain has become a hot topic in blockchain security. Huang et al. [14] created a new type of blockchain system with a credit-based workload consensus mechanism (Proof of Work), and researchers also designed a method for regulating sensor data access. The data authentication management method realized the data privacy protection of the Industrial Internet of Things and the efficiency improvement of the blockchain under the Internet of Things. Khan et al. [15] proposed a role-based fine-grained access control model for blockchain resource management. This model can monitor a resource continuously during the operation and update the attributes accordingly. Tan et al. [16] presented a general access control framework, which provided a unified and feasible way for users to achieve decentralized, lightweight, and fine-grained access control of the blockchain-based green Internet of Things. However, these studies mainly focused on optimizing the blockchain consensus mechanism and smart contracts, that is, more improvements to the blockchain security technology itself, without taking into account the encryption algorithm cracking, key leakage, and privacy leaks in the blockchain.
2.2. ABE for Blockchain-Based Access Control. Due to its wide application scenarios, attribute-based encryption has become a very popular research direction in cryptography. The emergence of ABE provides another solution to meet the needs of secure access control in blockchain applications. Xu et al. [17] created a distributed attribute-based keyword search (ABKS) scheme, where ABKS is a keyword search mechanism that introduces public key encryption using keyword search into the context of the attribute cryptosystem. The scheme allows data owners have the flexibility to share data content with groups of users that meet access control conditions. Yu et al. [18] provided an update-oriented access control scheme for blockchain-based IoT systems, which is compatible with the ABE technique. In the scheme, historical on-chain data can only be accessible to new members and inaccessible to the revoked members. Qin et al. [19] proposed an access control scheme with lightweight decryption based on ABE and blockchain technologies, where a user credibility incentive mechanism is designed to calculate the user’s credibility according to the user’s access behavior and gives a reputation score to adjust the endorsement protocol dynamically. These studies enhance the blockchain’s ability to protect data and user privacy in its applications and enrich the research on the ABE-based blockchain information sharing access control scheme. However, in traditional attribute-based cryptography, a trusted entity known as a Key Generation Center (KGC) is responsible for generating users’ private keys, which contradicts the decentralized blockchain.

2.3. MA-ABE for Blockchain-Based Access Control. The multi-authority ABE was first proposed in [20]. This solution is based on the fact that different types of transactions are handled exclusively by different organizations in reality, and it is improved on the standard ABE scheme. MA-ABE and its variants have been applied in researching and applying blockchain access control security. Guo et al. [21] constructed a multi-authority attribute-based encryption and signature scheme and introduced pseudorandom functions to achieve data sharing and collision avoidance. Qin et al. [22] proposed a blockchain-based multi-authority access control called BMAC for secure data sharing in the cloud, where the Shamir secret-sharing scheme and a permission blockchain were introduced. In this scheme, multiple authorities jointly managed each attribute to avoid a single point of failure. Banerjee et al. [23] offered a new blockchain-envisioned fine-grained user access control scheme for data security and scalability in IoT environment, which supported multiple attribute authorities and constant size key and ciphertext. Hei et al. [24] designed a novel MA-ABE scheme based on blockchain, which ensured that a relevant user could obtain the final decryption attribute key only after all attribute authorities had publicly issued their keys to the blockchain. Sethi et al. [25] presented a practical decentralized multi-authority traceable and efficiently revocable attribute-based cryptosystem with outsourcing decryption advantage. Rouhani et al. [26] proposed a distributed attribute-based access control (ABAC) system based on blockchain to provide trusted auditing of access attempts and presented a level of transparency that both access requesters and resource owners can benefit from this system. In this paper, we have considered four cryptographic properties to improve the security of MA-ABE for the blockchain-based access control scheme. Furthermore, the current multi-authority attribute-based encryption schemes are mainly based on prime order and composite-order bilinear groups. Prime order bilinear groups are better than composite-order bilinear groups in terms of computational efficiency. However, the number of bilinear groups satisfying prime order is relatively small, and the structure of bilinear groups is single, which cannot meet the practical application needs. Therefore, we constructed a multi-authority attribute-based encryption scheme based on composite-order bilinear groups.

3. Preliminaries

3.1. Composite-Order Bilinear Group. The concept of the composite-order bilinear group was firstly proposed by Boneh et al. in [27].

Let \( p_1, p_2, p_3 \) be distinct primes and \( G \) and \( G_T \) be cyclic groups of order \( N = p_1 p_2 p_3 \). The map \( e: G \times G \rightarrow G_T \) is a bilinear map if it meets the following three properties:

1. Bilinearity: For all \( a, b \in \mathbb{Z}_N \) and \( u, v \in G \), we have \( e(u^a, v^b) = e(u, v)^{ab} \).
2. Nondegeneracy: For all \( g \in G \) such that \( e(g, g) \neq 1 \) hold.
3. Computability: For all \( u, v \in G \), \( e(u, v) \) is efficiently computable.

Let \( G \) be a composite-order bilinear group and \( G_T \) be the subgroups of order \( p_i \) in \( G \), we can get that \( \forall g_i \in G_{p_i} \) and \( \forall g_j \in G_{p_j} \), then \( e(g_i, g_j) \) is an identity element in \( G_T \), which has been explained in [28]. The orthogonal property of subgroups will be used in dual-system encryption to achieve semifunctionality.

3.2. Assumption. There will be four assumptions illustrated in the proof of the security of our scheme. For assumptions 1–4, we refer to the assumptions 1–4 from Lewko and Waters’s scheme [29], where \( g \in \mathcal{G} \) means randomly selecting \( g \) in \( \mathcal{G} \).

3.2.1. Assumption 1. For a given group generator \( \mathcal{G} \), we define the following distribution:

\[
\begin{align*}
(N &= p_1 p_2 p_3, G, G_T, e, A) \\
T_1 &\sim G, \\
T_2 &\sim G_{p_i}, \\
D &= (G, g_1). 
\end{align*}
\]

We define \( A_{D, T_1} = \Pr[A(D, T_1)] - \Pr[A(D, T_2)] \) is the advantage of algorithm \( \mathcal{A} \) in breaking Assumption 1. The assumption is true if no algorithm exists that can
respectively distinguish $T_1$ and $T_2$ in $G$ and $G_{P_i}$ in polynomial time.

3.2.4. Assumption 4. The assumption is true if no algorithm exists that can, respectively, distinguish $f$ in breaking Assumption 4.

3.2.2. Assumption 2. For a given group generator $\mathcal{G}$, we define the following distribution:

\[ (N = p_1 p_2 p_3, G, G_T, e) \overset{R}{\rightarrow} \mathcal{G} g_1, \]

\[ X_1 \overset{R}{\leftarrow} G_{P_i}, \]

\[ X_2 \overset{R}{\leftarrow} G_{P_i}, \]

\[ g_3 \overset{R}{\leftarrow} G_{P_i}, \]

\[ D = (G, g_1, g_3, X_1 X_2), \]

\[ T_1 \overset{R}{\leftarrow} G_{P_i}, T_2 \overset{R}{\leftarrow} G_{P_i}. \]

We define $A dv 2_{A, \mathcal{G}} = |Pr[A(D, T_1)] - Pr[A(D, T_2)]|$ is the advantage of algorithm $A$ in breaking Assumption 2. The assumption is true if no algorithm exists that can, respectively, distinguish $T_1$ and $T_2$ in $G_{P_i}$ and $G_{P_i}$ in polynomial time.

3.2.3. Assumption 3. For a given group generator $\mathcal{G}$, we define the following distribution:

\[ (N = p_1 p_2 p_3, G, G_T, e) \overset{R}{\rightarrow} \mathcal{G} g_1, \]

\[ X_1 \overset{R}{\leftarrow} G_{P_i}, \]

\[ Y_1 \overset{R}{\leftarrow} G_{P_i}, \]

\[ X_3 \overset{R}{\leftarrow} G_{P_i}, \]

\[ Y_3 \overset{R}{\leftarrow} G_{P_i}, \]

\[ D = (G, g_1, X_1 X_3, Y_2 Y_3), \]

\[ T_1 \overset{R}{\leftarrow} G_{P_i}, T_2 \overset{R}{\leftarrow} G_{P_i}. \]

We define $A dv 3_{A, \mathcal{G}} = |Pr[A(D, T_1)] - Pr[A(D, T_2)]|$ is the advantage of algorithm $A$ in breaking Assumption 3. The assumption is true if no algorithm exists that can, respectively, distinguish $T_1$ and $T_2$ in $G_{P_i}$ and $G_{P_i}$ in polynomial time.

3.2.4. Assumption 4. For a given group generator $\mathcal{G}$, we define the following distribution:

\[ (N = p_1 p_2 p_3, G, G_T, e) \overset{R}{\rightarrow} \mathcal{G} g_1, \]

\[ g_1 \overset{R}{\leftarrow} G_{P_i}, \]

\[ g_2 \overset{R}{\leftarrow} G_{P_i}, \]

\[ g_3 \overset{R}{\leftarrow} G_{P_i}, \]

\[ a, b, c, d \overset{R}{\leftarrow} Z_N, \]

\[ D = (G, g_1, g_2, g_3, g_a b c, g_1 g_2 g_3, g_1 g_2 g_3), \]

\[ T_1 = \epsilon(g_1, g_1)^{abc}, \]

\[ T_2 \overset{R}{\leftarrow} G_T. \]

We define $A dv 4_{A, \mathcal{G}} = |Pr[A(D, T_1)] - Pr[A(D, T_2)]|$ is the advantage of algorithm $A$ in breaking Assumption 4. The assumption is true if no algorithm exists that can, respectively, distinguish $T_1$ and $T_2$ in polynomial time.

3.3. Access Structure. Let $P = \{p_1, p_2, \ldots, p_n\}$ be the collection of participants. A collection $\mathcal{A} \subseteq 2^P$ is monotone, if any arbitrary $B$ and $C$ can get that $B \in \mathcal{A}$ and $B \subseteq C$, and then $C \in \mathcal{A}$ holds. An access structure (monotone access structure) is a collection (monotone access structure) $\mathcal{A} \in 2^P/\emptyset$. The sets in $\mathcal{A}$ are authorized sets, besides the sets not in $\mathcal{A}$ are unauthorized sets.

3.3.1. Linear Secret-Sharing Scheme (LSSS) [30]. Linear secret-sharing scheme is an effective method to build an access structure. Let $P$ be the collection of participants; $A$ be a matrix with $m$ rows and $r$ columns. The map $\rho: \{1, 2, \ldots, m\} \rightarrow P$ pairs each row with each participant by labeling. A linear secret-sharing scheme II: $(A, \rho)$ over the set $P$ and $Z_N^*$ for access structure $\mathcal{A}$, if $\pi$ satisfies the following two polynomial-time algorithms:

1. Share $(A, \rho)$: the algorithm takes the shared secret $s \in Z_N^*$ as input and randomly chooses a group of elements $v_2, v_3, \ldots, v_n \in Z_N^*$ to generate a column vector $v = (s, v_2, v_3, \ldots, v_n)^T$. Then, it outputs $\lambda = Av$. We define that $A_i$ is the $i$-th row of matrix $A$ so that $\lambda_{\rho(i)} = A_i v$ is the element belonging to participant $\rho(i)$.

2. Recover $(S)$: the algorithm sets a set $S$ of participants as input. Then, we define another set $I = i | P(i) \in S$. If $S \in \mathcal{A}$, there will exist a group of constants $\omega_i | i \in I$, which can satisfy that $\sum_{i \in I} \omega_i \cdot \lambda_{\rho(i)} = s$. So, we can recover the secret. However, if $S \notin \mathcal{A}$, we cannot recover the right secret.
3.4. Commitment Scheme. The commitment scheme we used is based on Pedersen [31], which uses the discrete logarithm assumption. A commitment scheme is composed of three following algorithms:

1. **Setup** (1) → params: this algorithm takes a security parameter $\lambda^1$ as input and outputs the set of public parameters.

2. **Commit** (params, $M$) → (com, decom): this algorithm takes the public parameter set params and a message $M$ as input and outputs a commitment com and a decommitment decom, where decom can decommit com to $M$.

3. **Decommit** (params, $M$, com, decom) → $\{0, 1\}$: this algorithm takes the public parameter set params, a message $M$, a commitment com, and a decommitment decom as input and outputs 1 if decom can decommit com to $M$; otherwise, the algorithm outputs 0.

A commitment scheme must provide two properties of hiding and binding. The hiding property needs the message $M$ to keep unreleased until the user releases it. Other property only needs the decom to decommit the commitment com to $M$.

3.5. Zero-Knowledge Proof. The zero-knowledge proof scheme we used is based on [32]. By PoK$ \mathcal{(a, b, c)}$: $y = g^a h^b \land y = \overline{g^a h^b}$, we define a zero-knowledge proof of knowledge of $a$, $b$, and $c$, where $y = g^a h^b$ and $\overline{y} = \overline{g^a h^b}$ respectively hold by the group $G = \langle g \rangle = \langle h \rangle$ and $\overline{G} = \langle \overline{g} \rangle = \langle \overline{h} \rangle$. Commonly, the values in the parenthesis represent the knowledge that needs to be proven, while the rest of the values are known by the verifier. There is an extractor that can be used to rewind the knowledge from the successful prover.

4. System and Security Model

4.1. System Model. In this paper, we propose a MA-ABE access control scheme for a blockchain-based supply chain information sharing system. The system model is shown in Figure 1. Our system model considers five main participants: supply chain, data owner, blockchain, data users, and attribute authorities. The definition and functions of each participant are as follows:

**Supply chain:** the supply chain is the source of all data. We assume the source data in a supply chain is correctly uploaded by suppliers, who record the data in materials production, product processing, logistics transportation, warehouse storage, goods purchasing, etc. Furthermore, because of the limited size of blockchain blocks, the supply chain data involved in this article only contains small-capacity data such as transaction records, cargo numbers, and logistics processes. For large-capacity data such as videos and images, we will not discuss in this paper, but we will do more researches on it in the future.

**Data Owner:** the owner encrypts his/her goods’ supply chain information and sends encrypted data to the blockchain with the access policy. An access policy can be defined as follows: (Attribute 1: (Researcher identified by Authority 1)) OR (Attribute 2: (Analyst identified by Authority 2)) AND (Attribute 3: (User identified by Authority 3)), so only the users with Attributes 1 and 2 or the users with Attributes 1 and 2 can correctly decrypt owner’s ciphertext. Furthermore, the owner can update the encrypted data in a new block.

**Attribute Authorities:** each attribute authority can generate and distribute attribute public keys based on data users’ attributes within the scope of the attributes it manages, such as phone numbers, identities, and affiliation. When data users’ attributes changed, the attribute authority has to change user attribute secret keys. In addition, attribute authority can create a block. It has to package the data owner’s encrypted data in the block and add it to the blockchain.

**Blockchain:** blockchain is the distributed environment for storing data owner’s encrypted data in storage nodes and making it public to users who can connect to the blockchain system.

**Data Users:** data users get their attribute secret keys from the attribute authorities and decrypt data owner’s encrypted data. However, only the users with the correct attribute secret keys can decrypt the ciphertext properly.

4.2. Security Model. We let $\mathcal{A}$ be an adversary and $\mathcal{C}$ be a challenger, then the game between $\mathcal{A}$ and $\mathcal{C}$ will carry out as follows:

**GlobalSetup:** $\mathcal{C}$ executes this algorithm and sends global parameter set $g$ to $\mathcal{A}$. Then, $\mathcal{A}$ returns a group of corrupted authorities $AA \subseteq AA$ and gives the challenge access structure $\mathcal{A}^* = (A^*, \rho^*)$, where $AA$ is the set of all authorities.

**AuthoritySetup:** challenger $\mathcal{C}$ obtains authority authorities’ public and private keys and then sends the public keys to the adversary.

**Key Query Phase 1:** adversary $\mathcal{A}$ queries the secret key by submitting a pair $(\text{attr}_i, GID)$ to challenger $\mathcal{C}$, where $\text{attr}_i$ belongs to good authorities. Then, $\mathcal{C}$ responds by providing the relative secret keys. However, $\mathcal{A}$ cannot make queries on any attribute set which satisfies the access structure $\mathcal{A}_i$. Besides, the request of $GID$ cannot belong to the corrupted authorities.

**Challenge:** adversary $\mathcal{A}$ transmits two messages with the same length, $M_0$ and $M_1$, and the access structure $\mathcal{A}^* = (A^*, \rho^*)$ to challenger $\mathcal{C}$. It is worth noting that every attributes of the corrupted authority set $AU$ cannot satisfy $\mathcal{A}^*$. $\mathcal{C}$ randomly selects a bit $b \in \{0, 1\}$ and returns the encrypted data $CT_b$ of message $M_b$ under the access structure $\mathcal{A}^*$ to $\mathcal{A}$.
Key Query Phase 2: adversary $\mathcal{A}$ makes more key queries as long as the attribute sets do not satisfy the challenge access structure $A^*$. 

Guess: adversary $\mathcal{A}$ tries to guess whether the message $M'_b$ meets the encrypted data $CT_b$. If $b' = b$, the advantage of adversary $\mathcal{A}$ to win this game is calculated as follows:

$$\text{Adv}(\mathcal{A}) = \left| \Pr[b' = b] - \frac{1}{2} \right|.$$  \hspace{1cm} (5)

5. Our Works

5.1. System Scheme. Our system scheme’s workflow is shown in Figure 2. We divide the system workflow into five stages: Setup, Key Generation, Encrypt, Decrypt, and Update. The details are as follows.

(1) Setup: multiple attribute authorities generate the system’s public parameters through the $\text{GlobalSetup}$ algorithm and use the consensus algorithm to negotiate and reach an agreement. After that, each attribute authority executes the $\text{AuthoritySetup}$ algorithm to create its public key and private key. At the same time, attribute tags are assigned to users according to their attributes, such as phone number, identification, and affiliation.

(2) Key generation: the attribute authority uses the $\text{KeyGen}$ algorithm to generate the user’s public and private key according to the user’s attributes.

(3) Encrypt: the data owner uses the $\text{Encrypt}$ algorithm to encrypt the access structure and data.

(4) Block generation: when the encrypted data need to be uploaded to the blockchain, the data owner broadcasts ciphertext to the blockchain network. When attribute authorities have collected enough ciphertexts, use a hash function to calculate the hash value of each ciphertext and construct a Merkle tree structure. Then, the attribute authority puts all ciphertexts into the block body and adds them into the blockchain along with the block. The structure of the block is shown as Figure 3. Finally, storage nodes store the blockchain data. It is noted that the size of the block is related to the number and size of the ciphertext stored in the block body.

(5) Decrypt: the data user requests the data owner’s encrypted data from the blockchain node and decrypts it through the $\text{Decrypt}$ algorithm.

(6) Revoke and update: the attribute authority center can use the $\text{KeyUpdate}$ algorithm to update the user key flexibly according to the user’s attribute change. In addition, users can also update the encrypted data using the $\text{CTUpdate}$ algorithm and pack it in a new block.
5.2. Algorithm Construction. Like other schemes [33, 34], although our scheme is based on the composite-order bilinear group construction, the whole system is limited to subgroups \( G_{p_1} \). Subgroups \( G_{p_2} \) and \( G_{p_3} \) are only used in the security proof process to construct the semifunctional key and the semifunctional ciphertexts. Here, the specific seven main algorithms of the cryptography involved in our scheme are as follows:

GlobalSetup \((1^\lambda) \rightarrow gp\).

(1) Let \( \phi \) be an algorithm to generate composite-order bilinear groups. \( \phi \) takes the security parameter \( 1^\lambda \) as input and randomly selects number \( N = p_1p_2p_3 \) as the order of the group \( G \), where \( g_1 \) is a generator of \( G_{p_1} \). Then, set \( h = g_1^\mu \), where \( \mu \in \mathbb{Z}_N^* \) is randomly chosen. We set \( H: \{0, 1\}^* \rightarrow \mathbb{Z}_N^* \) and \( H^*: \{0, 1\}^* \rightarrow G \) as two strong collision resistance hash functions for mapping random numbers to values in \( \mathbb{Z}_N^* \) and \( G \). Let \( e: G \times G \rightarrow \mathbb{G}_T \) be a bilinear map.

(2) The algorithm generates a user’s unique global indicator \( GI \) in the system and uses the hash function \( H \) to get \( u = H(GI \, D) \in \mathbb{Z}_N^* \). Then, user public key will be represented as \( PK_u = \{g_U^\mu\} \).
(3) We define the set of N attribute authorities as \( AA = \{ AA_1, AA_2, \ldots, AA_N \} \), where each attribute authority \( AA_i \) manages its attribute set \( AA_i \).

(4) We set

\[
gp = (e, g_1, N, G, G_p, H, H^t, AA, AA, PK_u).
\]

\[
\text{AuthoritySetup} (gp) \rightarrow [PK_i, SK_i].
\]

Each attribute authority \( AA_i \) runs this algorithm. For attribute \( a_i \) belonging to \( AA_i, AA_i \) chooses two random exponents \( \alpha, \beta \in Z_N^* \). Then, \( AA_i \) randomly selects a exponent \( y_i \) for itself and gets its public key \( PK_i \) and secret key \( SK_i \) as follows:

\[
SK_i = \{ (a_i, \beta_i) \}_{\alpha, \beta, AA_i},
\]

\[
PK_i = \{ (e(g_1, g_1)^{\alpha_i}, g_1^{\beta_i}) \}_{\alpha, \beta, AA_i, g_1^x}.
\]

\[
\text{Encrypt} (gp, M, PK_i, AA) \rightarrow ct.
\]

(1) Data owner executes the Encrypt algorithm and selects a LSSS-based access policy \( \AA = (A, \rho) \), in which for each attribute \( a_i \) involved, the owner sets a random number \( t \in Z_N^* \) and replaces \( a_i \) as \( \vartheta_j = e(H_t(\alpha_j), g_1^{\rho^t}) \) in the LSSS matrix \( (A, \rho) \).

(2) The algorithm firstly chooses a secret \( s \in Z_N^* \) as the sharing secret. Then, it generates a random column vector \( \nu = (s, v_2, v_3, \ldots, v_m) \in Z_N^m \) with \( s \) as the first element, where \( \forall i \in [2, m], v_i \in Z_N \).

(3) We set \( \lambda_x = A_x \nu \), where \( A_x \) is the \( x \)-th row of matrix \( A \). Then, the algorithm generates another random column vector \( \omega = (0, \omega_2, \omega_3, \ldots, \omega_m) \in Z_N^m \) with \( 0 \) as the first element, where \( \omega_x = A_x \omega \). For each row of the matrix \( A \), the algorithm randomly selects \( r_x \in Z_N \). At least, the encrypted data of the message \( M \in Z_N \) can be defined as follows:

\[
C_0 = Me(g_1, g_1)^s,
\]

\[
C_1 = g_1^{\alpha_1 C_{1,x}} e(g_1, g_1)^{\lambda_1 \omega_1 r_x},
\]

\[
C_{2,x} = g_1^{r_x},
\]

\[
C_{3,x} = g_1^{\omega_x r_x}.
\]

Here, \( ct = \{ C_0, C_1, [C_{1,x}, C_{2,x}, C_{3,x}] \} \).

\[
\text{KeyGen} (gp, [PK_i, SP_i], AA, u, PK_u) \rightarrow K_{u,j}.
\]

Attribute authorities use the anonymous key distribution protocol to generate the attribute secret keys to users. For each attribute \( a_i \in AA_u \cap AA_j \), where \( AA_u \) is user \( u \)'s attributes set which is verified by authorities. Then, the user attribute secret keys \( K_{u,j} \) can be computed as follows:

\[
K_1 = g_1^{\alpha_1 h^{\beta_1 u v_j}},
\]

\[
K_2 = h^{\lambda_1 u v_j},
\]

\[
K_3 = H'(\alpha_j)^{v_j}.
\]

5.3 Anonymous Key Generation Protocol

Algorithm 1 shows the anonymous key generation protocol built in this paper. The details of this algorithm are shown as follows:

(1) The data user \( u \) selects \( \rho, a_1, a_2 \in Z_N^* \) randomly and calculates \( P = g_1^{\rho}, Q = h^{\rho}, P' = g_1^{\rho}, Q' = h^{\rho} \). Then, \( u \) sends \( (P, Q, P', Q') \) to authority \( AA_u \).

Here, \( K_3 \) is used to hide the attribute \( a_j \), and \( K_{u,j} = \{ K_1, K_2, K_3 \} \).

\[
\text{Decrypt} (gp, ct, K_{u,j}) \rightarrow M.
\]

Each data user has to query the owner's encrypted data (ciphertext) from blockchain and uses the Decrypt algorithm to get the original message \( M \). Furthermore, data user can replace his/her attribute \( a_j \in AA_u \) by calculating \( \vartheta_j = e(K_3, C_j) \) in the access policy \( \AA \). Then, the user gains a set \( X = \{ x: \rho(x) \} \) from \( \AA \). Finally, the user chooses a constant \( c_\varnothing \in Z_N^* \), which meets \( \sum_{x \in \varnothing} C_x = (1, 0, \ldots, 0) \). The Decrypt algorithm is run to decrypt the ciphertext \( ct \). If the data user has the secret keys of each \( A_{x,i} \), he/she will decrypt the ciphertext \( ct \) as follows:

\[
M = \prod_x (C_{1,x} e(K_{2,x}, C_{3,x})/e(K_1, C_{2,x}))^c_x.
\]

KeyUpdate (\( gp, a_j, PK_i, SK_i, PK_u \) \( \rightarrow K'_{u,j} \). Assume that \( a_j \) is an attribute revoked and updated by the authority \( AA_u \). Then, \( AA_u \) must regenerate a corresponding new user attribute secret key for \( a_j \) through the following steps:

(1) Authority \( AA_u \) regenerates two new attribute random exponents \( \alpha_j, \beta_j \in Z_N^* \).

(2) For users who have the attribute \( a_j \), \( AA_i \) calculates \( UKa_j = \alpha - \alpha_j, UKB_j = \beta - \beta_j \).

(3) In the end, user \( u \) computes

\[
K_1' = \frac{K_1}{g_1^{UKa_j} K_2'},
\]

\[
K_2' = K_2.
\]

CUpdate (\( ct, AA, UKa_j, UKB_j \) \( \rightarrow ct' \). If the access policy \( (A, \rho) \) involves the revoked and updated attribute \( a_j \), i.e., \( \rho(x) = j \), the data owner will update his/her ciphertext stored in a new block in blockchain. The ciphertext can be updated as follows:

\[
C_{1,x}' = \frac{C_{1,x}}{e(g_1, g_1)^{UKa_j}},
\]

\[
C_{2,x}' = C_{2,x},
\]

\[
C_{3,x}' = C_{3,x}.
\]
(2) AA2 chooses $c \in Z_N^*$ randomly and returns it to $u$.

(3) $u$ computes $a'_1 = a_1 - c/p, a'_2 = a_2 - cu$, and sends $a'_1$ and $a'_2$ to $AA_i$.

(4) $AA_i$ verifies $P' = P^h g_1^{a'_1}$ and $Q' = Q^h g_1^{a'_2}$. If they are verified rightly, $AA_i$ will continue. If not, $AA_i$ will abort.

(5) $AA_i$ computes $\eta = (\gamma_i + u)p$ by two-party secure computation (2PC Protocol).

(6) $AA_i$ generates $b_1, b_2, b_3 \in Z_N^*$ randomly. Then, it calculates $K'_1 = P^n h_2^{\beta_i}, K'_2 = Q^n h_2^{\beta_i}, K''_1 = P^n h_3^{\beta_i}, K''_2 = Q^n h_3^{\beta_i}$ and sends $(K'_1, K'_2, K''_1, K''_2)$ to $u$.

(7) $u$ returns a random number $t \in Z_N^*$ to $AA_i$.

$$X = C_{1x} e(K_2, C_{3x}) = e(g_1, g_1)^{\gamma_i} e(g_1, g_1)^{\omega_i / \gamma_i} u^{\omega_i},$$

$$Y = e(K_1, C_{2x}) = e(g_1, g_1)^{\gamma_i} e(g_1, g_1)^{\omega_i / \gamma_i} h_3^a.$$ 

So, we can compute

$$\Pi_a \left( \frac{X}{Y} \right) c_x = e(g_1, g_1)^{\gamma_i} e(g_1, g_1)^{\omega_i / \gamma_i} u^{\omega_i} = e(g_1, g_1)^{\gamma_i} e(g_1, g_1)^{\omega_i / \gamma_i} u^{\omega_i} = e(g_1, g_1)^{\gamma_i} e(g_1, g_1)^{\omega_i / \gamma_i} u^{\omega_i}.$$ 

(12)

(13)

6.2. Correction of Update

(1) $K'_1 g_1^K K'_2 h_2^{\beta_i} / h_3^{\beta_i} = K_1$. 

(2) $C_{1x} e(g_1, g_1)^{UK_{K}(a_i)} Y_x e(g_1, g_1)^{UK_{K}(a_i)} / e(g_1, g_1)^{UK_{K}(a_i)} e(g_1, g_1)^{UK_{K}(a_i)} = C_{1x}$. 

(3) $C_{2x} / C_{3x} = g_1^{\omega_i / \gamma_i} g_1^{\omega_i / \gamma_i} = C_{3x}$. 

6.3. Security of Our Scheme. In this paper, the dual-system encryption [35] is applied to prove the security of our scheme. We refer to the proof method in [29, 36, 37]. In the dual-system encryption technology, secret keys and ciphertexts are divided into two forms: normal and semifunctional. It is noteworthy that a normal key can decrypt normal ciphertexts and semifunctional ciphertexts, while a semifunctional key can only decrypt normal ciphertexts, but it cannot decrypt semifunctional ciphertexts. In addition, the semifunctional key and ciphertext only exist in the security certification, and they will not appear in the actual system application. To more accurately describe the semifunctional ciphertext and the semifunctional key, we select two fixed random values $z_j, t_j \in Z_N^*$ for each attribute $a_j$, and these two random numbers are the same for the semifunctional ciphertexts and the semifunctional keys, and they will not be changed by different users.

6.3.1. Semifunctional Ciphertext. To generate the semifunctional ciphertext, we randomly choose $g_2 \in G_2, g_3 \in G_3, u_2 \in Z_N^*, u_3 \in Z_N^*$. Then, we set $\delta_x = A_x u_3, \sigma_x = A_x u_3$ for each row $A_x$ in the access policy matrix $A$. In addition, we set $B$ as the subset of related rows, which is marked by corrupted attribute authorities in $A$, and we set $\bar{B}$ as the subset of rows marked by good attribute authorities in $A$. Three exponents $a_x, b_x, c_x \in Z_N^*$ are chosen randomly.

At first, we set the original ciphertext as follows:

$$C_0 = Me(g_1, g_1)^{\gamma_i},$$

$$C_1 = g_1^{C_{1x}} e(g_1, g_1)^{\omega_i / \gamma_i} C_{2x} = g_1^{C_{1x}},$$

$$C_{2x} = g_1^{\omega_i / \gamma_i}.$$ 

For each $x$, if $A_x \in B$, the semifunctional ciphertext will be computed as follows:

(14)
(1) User $u$ has his/her global ID $u = H(GI_D)$.
(2) Authority $AA_i$ has this identity $y_i \in Z_N^*$ and $u$’s attribute $a_j$’s parameters $a_j, b_j \in Z_N^*$.
(3) $u$ randomly chooses $\rho \in Z_N^*$.
(4) $u$ computes $P = g_1^\rho, Q = h^\rho, \text{Pok}[(u, \rho); \text{PNQ}]$ and sends them to $AA_i$.
(5) $u$ should proof he/she knows $(u, \rho)$ in zero-knowledge.
(6) if the proof is successful then
(7) $AA_i$ continues.
(8) else
(9) $AA_i$ aborts.
(10) end if

Algorithm 1: Anonymous key generation protocol.

\[ C_0 = C'_0, \]
\[ C_1 = C'_1, \]
\[ C_{1,x} = C'_{1,x}, \]
\[ C_{2,x} = C'_{2,x}, \]
\[ C_{3,x} = C'_{3,x} g_2^{\beta_i} g_3^{\sigma_c}. \]

For each $x$, if $A_x \in \mathcal{B}$, the semifunctional ciphertext will be computed as follows:
\[ C_0 = C'_0, \]
\[ C_1 = C'_1, \]
\[ C_{1,x} = C'_{1,x}, \]
\[ C_{2,x} = C'_{2,x} g_2^{\delta_i + \sigma_{c(x)}} g_3^{\beta_i + \delta_{c(x)}}. \]
\[ C_{3,x} = C'_{3,x} g_2^{\delta_i + \sigma_{c(x)}} g_3^{\beta_i + \sigma_{c(x)}}. \]

6.3.2. Semifunctional Key. There are two kinds of semifunctional keys, namely, type-1 and type-2. We set $H(u) = h^{1/\nu_{\gamma_1}}$ as the value in the subgroup $G_{p_j}$ of $G$ and select an exponent $c \in Z_N$ at random.

The type-1 semifunctional key is calculated as follows:
\[ K_2 = H(u)g_2^c, \]
\[ K_1 = K'_1 g_3^{\frac{c}{2}}. \]

The type-2 semifunctional key is calculated as follows:
\[ K_2 = H(u)g_2^c, \]
\[ K_1 = K'_1 g_3^{\frac{c}{2}}. \]

Here, $K'_1 = g_1^\rho h^{\delta_i/\nu_{\gamma_1}}$ is the normal key.

When we apply a type-1 semifunctional key to decrypt a semifunctional ciphertext, the extra part $e(g_2, g_3)^{\sigma_c}$ will prevent the ciphertext from being directly decrypted unless $\delta_i = 0$. Similarly, when we apply a type-2 semifunctional key to decrypt a semifunctional ciphertext, the extra part $e(g_2, g_3)^{\delta_i}$ will prevent the ciphertext from being directly decrypted unless $\sigma_c = 0$.

6.3.3. Attack Games. The games we used in this paper are defined as follows:

Game_{real}: this game is a real game, where the ciphertext and user secret key are normal.

Game_{0}: comparing with Game_{real}, the challenging ciphertext is a semifunctional.

Game_{j,1}: comparing with Game_{0}, the first $j - 1$ keys obtained are type-2 semifunctional keys, while the $j$-th key is the type-1 semifunctional key, and other keys are normal keys. In addition, the challenging ciphertext is the semifunctional ciphertext. If $q$ is the number of times that the adversary \( \mathcal{A} \) queries for the key, the range of $j$ will be $1$ to $q$.

Game_{j,2}: comparing with Game_{0}, the first $j$ keys obtained are type-2 semifunctional keys, while other keys are normal. Moreover, the challenging ciphertext is the semifunctional ciphertext, and all keys in Game_{q,2} are type-2 semifunctional keys.

Game_{final}: in this game, all keys are type-2 semifunctional keys, and the ciphertext is a semifunctional ciphertext of a random message. We can get that the adversary \( \mathcal{A} \) has no advantage in winning the game.

6.4. Lemma 1 and Its Proof

**Lemma 1.** If there is a polynomial-time algorithm \( \mathcal{A} \) that can distinguish Game_{real} and Game_{0} at the nonnegligible advantage $\varepsilon$, another polynomial-time algorithm $\mathcal{B}$ will be constructed to break Assumption 1 with the advantage $\varepsilon$. 

6.4.1. Proof

GlobalSetup: challenger $\mathcal{C}$ runs the GlobalSetup algorithm. Simulator $\mathcal{S}$ receives the global parameter $g_{p} = (e, g_{1}, n, p_{1}, p_{2}, G, G_{p}, H, H')$ from the challenger $\mathcal{C}$ and simulates Game$_{real}$ or Game$_{0}$ between the adversary $\mathcal{A}$ and $\mathcal{S}$. $\mathcal{A}$ specifies a set of corrupted attribute authorities $A_{\mathcal{A}} \subset A\mathcal{A}$, where $A\mathcal{A}$ represents the set of all attribute authorities in the system. Then, $\mathcal{A}$ develops an access structure $\mathcal{A} = (A, \rho)$ that he/she needs to challenge.

AuthoritySetup: $\mathcal{B}$ executes the AuthoritySetup algorithm. For the corrupted attribute authorities, $\mathcal{B}$ sends their public and private keys to $\mathcal{A}$, and for good attribute authorities, $\mathcal{B}$ only sends its public key to $\mathcal{A}$.

Key Query Phase 1: when $\mathcal{A}$ queries the key of $(a_{j}, GI\ D)$, $\mathcal{B}$ runs the KeyGen algorithm to generate the related keys.

Challenge: $\mathcal{A}$ submits two messages $M_{0}$ and $M_{1}$ with the same length to $\mathcal{B}$. $\mathcal{B}$ randomly chooses a bit $b \in \{0, 1\}$ and sends the confidential message $M_{b}$ under the access structure $\mathcal{A}$ to $\mathcal{A}$.

$\mathcal{B}$ selects a random shared number $s \in Z_{N}$ and sets $C_{0} = Me(g_{1}, g_{1})^{s}$. $\mathcal{B}$ also selects two random column vectors $v = (s, v_{2}, \ldots, v_{m}) \in Z_{N}^{m}$ and $\varphi = (0, \varphi_{2}, \varphi_{3}, \ldots, \varphi_{m}) \in Z_{N}^{m}$. Then, $\mathcal{B}$ selects $A_{x} = A_{x}v, \omega_{x} = A_{x}\varphi$, where $A_{x}$ is the $x$-th row in the matrix $A$.

$\mathcal{A}$ provides $\mathcal{B}$ with the random parameters $a_{j}, b_{1}$ for the attribute $a_{j}$ contained in the access structure $(A, \rho)$, which belongs to the corrupted attribute authorities. Here, $\mathcal{B}$ represents a subset of the relevant rows of attributes with good attribute authorities in $A$, and $\mathcal{B}$ represents a subset of the relevant rows with corrupted attribute authorities in $A$. For each $A_{x}$ in $\mathcal{B}$, $\mathcal{B}$ randomly selects $r_{x} \in Z_{N}$ and relatively selects $r_{x}' \in Z_{N}$ for each $A_{x}$ in $\mathcal{B}$, where $r_{x} = r_{x}'$. We set the $G_{p}$ part of $T$ as $g_{1}^{0}$, the $G_{p}$ part as $g_{1}^{3}$, and the $G_{p}$ part as $g_{1}^{3}$, where $r, c, d \in Z_{N}$ are selected randomly.

For $A_{x} \in B$, the ciphertexts are computed as follows:

$$C_{1, x} = e(g_{1}, g_{1})^{b_{1}A_{x}v}, C_{2, x} = g_{1}^{r_{x}}, C_{3, x}$$

(19)

For $A_{x} \in \overline{B}$, the ciphertexts are computed as follows:

$$C_{1, x} = e(g_{1}, g_{1})^{b_{1}A_{x}v}, C_{2, x} = T^{r_{x}'}, C_{3, x}$$

(20)

A ciphertext is generated in two cases: $A_{x} \in B$ and $A_{x} \in \overline{B}$. If $T \in G_{p}$, it will be a normal ciphertext. When $T \in G$, it is a semifunctional ciphertext. Due to $T \in G_{p}$ and $T \in G$ are indistinguishable, the game Game$_{real}$ and Game$_{0}$ cannot be distinguished. The proof is shown as follows:

(1) When $T \in G_{p}$, the $G_{p}$ part of $T$ is $g_{1}^{0}$. For each $A_{x} \in \overline{B}$, $T^{r_{x}} = g_{1}^{r_{x}}$, because the first random value of the vector $\varphi$ is $0$, we cipher the ciphertext $C_{3, x}$ as a normal ciphertext. For each $A_{x} \in \overline{B}$, $C_{2, x}$ and $C_{3, x}$ are normal ciphertexts, because of $T^{r_{x}} = g_{1}^{r_{x}}$ and $r_{x} = r_{x}'$. Finally, we can get that when $T \in G_{p, 1}$, all ciphertexts are normal.

When $T \in G$, we can know that $T = g_{1}^{0}g_{1}^{j}$. If $A_{x} \in B$, $T^{r_{x}} = g_{1}^{r_{x}}$, $g_{1}^{r_{x}}$ in the ciphertext $C_{3, x}$. In the semifunctional ciphertext, $\delta_{x} = (A_{x}, \varphi \mod(p_{3}), \sigma_{x} = (A_{x}, \varphi \mod(p_{3}))$. So $C_{3, x}$ can be seen as a semifunctional ciphertext.

According to the remainder theorem, if $A_{x} \in B$, $(\varphi \mod(p_{1}), \varphi \mod(p_{2}), \varphi \mod(p_{3}))$ will be uncorrelated for a random value $\alpha$. We can set exponents

$$a_{x} = (c_{x}^{e}) \mod(p_{2}), b_{x} = (d_{x}^{e}) \mod(p_{3}), z_{x} = \beta_{x}(a_{x}, b_{x} \mod(p_{3})) \mod(p_{3})$$

because $a_{x}, b_{x}, z_{x} \mod(p_{3})$ and $g_{x}$ are all distributed randomly, $C_{2, x}$ and $C_{3, x}$ are semifunctional ciphertexts.

(2) In a semifunctional ciphertext, we can get $\delta_{x} = A_{x}u_{2}$ and $\sigma_{x} = A_{x}u_{4}$, where $u_{2}$ and $u_{4}$ are random vectors. In this game, the first element of $\varphi$ is $0$, and adversary $\mathcal{A}$ cannot distinguish whether the value is random or not.

When $A_{x} \in B$, according to the monotone span program [36], we can know that $\varphi = (0, \varphi_{2}, \ldots, \varphi_{m})$ and $\varphi' = (\varphi_{2}, \varphi_{3}, \ldots, \varphi_{m})$ can generate the same secret-sharing value, which means adversary $\mathcal{A}$ cannot figure out whether the first element is $0$ or another $\varphi_{1}$.

When $A_{x} \in \overline{B}$, $z_{x} \mod(p_{3})$ is random in $\delta_{x} + A_{x}z_{p}(x)$. If $(a_{x} \mod(p_{2}) \neq 0$, the exponent $\delta_{x}$ of $g_{x}$ will be represented by $z_{p}(x)$. So, the secret-sharing value can be hidden, and the shared value $\delta_{x}$ is appropriately distributed from the perspective of adversary $\mathcal{A}$. The exponent $\sigma_{x}$ of $g_{x}$ is still appropriately distributed for the same reason.

Therefore, when $T \in G_{p, 1}$, ciphertexts are normal and simulator $\mathcal{B}$ can simulate the Game$_{real}$. When $T \in G$, ciphertexts are semifunctional and simulator $\mathcal{B}$ can simulate the Game$_{real}$. $\mathcal{B}$ can use $\mathcal{A}$ to break Assumption 1 with advantage $\epsilon$.

6.5. Lemma 2 and Its Proof

Lemma 2. If there is a polynomial-time algorithm $\mathcal{A}$ that can distinguish Game$_{j-1, 2}$ and Game$_{j, 1}$ at the nonnegligible advantage $\epsilon$, then another polynomial-time algorithm $B$ can be constructed to break Assumption 2 with the advantage $\epsilon$.

6.5.1. Proof. The proof of Lemma 2 is similar to that of Lemma 1, but challenger $\mathcal{C}$ needs to send the public parameter $(g_{1}, g_{3}, X_{1}, X_{2}, T)$ to simulator $\mathcal{B}$.

GlobalSetup and AuthoritySetup: the same as in Lemma 1. According to the difference of $T$, $\mathcal{A}$ and $\mathcal{B}$ will simulate Game$_{j-1, 2}$ and Game$_{j, 1}$. For each attribute $a_{i}$ belonging to a good attribute authority $AA_{i}$, $\mathcal{B}$
generates its random parameters \(a_i, \beta_i \in \mathbb{Z}_N\) and sends the public key to \(\mathcal{A}\).

**Key Query Phase 1:** assume that \(\text{GID}_k\) indicates the identity of the \(k\)-th user queried by adversary \(\mathcal{A}\), which means \(\mathcal{A}\) performs a key query \((a_i, \text{GID}_k)\). Then, \(\mathcal{B}\) needs to perform the following feedback:

When \(k > j\), \(K'_1 = H(u) = g_1^{\gamma_k}a_i\), \(K'_1 = g_1^{\phi_i}g_1^{\beta_j}\), where \(K'_1\) is normal.

When \(k < j\), \(K'_1 = H(u) = (g_1g_3)^{\gamma_k}a_i\), \(K'_1 = g_1^{\phi_i}\), where \(K'_1\) is semifunctional of type-2.

\(c = 1/\gamma_j + u, t_i = \beta_i\).

When \(k = j\), if \(T \in G_p\), let \(T = g_1^{\beta_j}\), then \(K'_1 = H(u) = T^{1/\gamma_k}a_i\) and \(K'_1 = T^{1/\gamma_k}g_1^{\phi_i}\), where \(K'_1\) is normal; if \(T \in G_p\), let \(T = g_1^{\beta_j}\), then \(K'_1 = T^{1/\gamma_k}a_i\) and \(K'_1 = T^{1/\gamma_k}g_1^{\phi_i}\), where \(K'_1\) is semifunctional.

\(\mathcal{C}\) submits two messages \(M_0, M_1\) with the same length to \(\mathcal{B}\). \(\mathcal{B}\) randomly chooses a bit \(b \in \{0, 1\}\) and sends the confidential message \(M_b\) under the access structure \(\mathcal{A}\) to \(\mathcal{A}\). \(\mathcal{B}\) selects a random shared number \(s \in \mathbb{Z}_N\) and sets \(C_0 = M_e(g_1, g_1); \mathcal{B}\) selects three random column vectors \(v = (v_1, v_2, v_3, \ldots, v_m) \in \mathbb{Z}_N^m\), \(\varphi = (0, \varphi_2, \ldots, \varphi_m) \in \mathbb{Z}_N^m\) and \(u = (u_1, u_2, \ldots, u_m) \in \mathbb{Z}_N^m\). Then, \(\mathcal{B}\) sets \(\lambda_i = A_i, v, \omega_i = A_i, \varphi\), where \(A_i\) is the \(x\)-th row in the matrix \(A\). The definitions of \(B\) and \(\mathcal{B}\) are the same as in the proof of Lemma 1.

If \(A_x \in B\), \(\mathcal{B}\) randomly selects \(r_x \in \mathbb{Z}_N\) for each \(A_x \in B\). Then, ciphertexts can be computed as follows:

\[
\begin{align*}
C_{1,x} &= e(g_1, g_1)^{\lambda_i}e(g_1, g_1)^{\omega_i(r_x)}, \\
C_{2,x} &= C_{2,x} = g_1^s, \\
C_{3,x} &= (X_{1,x}, X_{3})^\omega g_1^{\omega_i(r_x)} y_x^{\gamma_i} = C_{3,x} g_2 u^s x g_3^{y_i}.
\end{align*}
\]

If \(A_x \in B\), \(\mathcal{B}\) randomly selects \(r_x, \omega_x, \phi'_x \in \mathbb{Z}_N\) for each \(A_x \in B\), where \(r_x = r_x, \omega_x = r_x\), then ciphertexts are as follows:

\[
\begin{align*}
C_{1,x} &= e(g_1, g_1)^{\lambda_i}e(g_1, g_1)^{\omega_i(r_x)}, \\
C_{2,x} &= (X_{1,x}, X_{3})^{\omega_i} g_3^{\omega_i(r_x)} y_x^{\gamma_i} = g_1^{x} X_{1,x} g_1^{y_i}, \\
C_{3,x} &= C_{3,x} g_2 (X_{1,x})^{\omega_i} (X_{3})^{\omega_i} g_3^{\omega_i(r_x)} y_x^{\gamma_i} = C_{3,x} g_2 u^s x g_3^{y_i}.
\end{align*}
\]

**Lemma 3.** If there is a polynomial-time algorithm \(\mathcal{A}\) that can distinguish \(\text{Game}_{j,1}\) and \(\text{Game}_{j,2}\) at the nonnegligible advantage \(\epsilon\), another polynomial-time algorithm \(\mathcal{B}\) will be constructed to break Assumption 3 with the advantage \(\epsilon\).

**Proof.**

The proof of Lemma 3 is similar to that of Lemma 3, but challenger \(\mathcal{C}\) needs to send the public parameter \((N, g_1, X_1, X_3, Y_2, Y_3, T)\) to simulator \(\mathcal{B}\).

**6.6.3. GlobalSetup and AuthoritySetup:** the same as in Lemma 2. According to the difference of \(T\), \(\mathcal{A}\) and \(\mathcal{B}\) will simulate \(\text{Game}_{j,1}\) and \(\text{Game}_{j,2}\). For each attribute \(a_i\), \(\mathcal{B}\) belonging to a good attribute authority \(AA_i\), \(\mathcal{B}\) generates its random parameters \(a_i, \beta_i \in \mathbb{Z}_N\) and sends the public key to \(\mathcal{A}\).

**Key Query Phase 1:** assume that \(\text{GID}_k\) indicates the identity of the \(k\)-th user queried by adversary \(\mathcal{A}\), which means \(\mathcal{A}\) performs a key query \((a_i, \text{GID}_k)\). Then, \(\mathcal{B}\) needs to perform the following feedback:

When \(k > j\), \(K'_1 = H(u) = g_1^{\gamma_k}a_i\), \(K'_1 = g_1^{\phi_i}\), where \(K'_1\) is normal.

When \(k < j\), \(K'_1 = H(u) = (X_{1,x} X_{3})^{\gamma_k}a_i\), \(K'_1 = g_1^{\phi_i}\), where \(K'_1\) is semifunctional.

\(r_x\) and \(r_x\) are uncorrelated. Therefore, we can set \(a_x = (r_x)^{\gamma_k}a_i\), \(b_x = (\varphi_i)^{\gamma_k}\) mod \(p_3\), \(C_{1,x} = C_{1,x} g_2 u^s x g_3^{y_i}\), where we can set \(z_{p_2} = (\beta_{p_2})^{\gamma_k}a_i\), \(t_{p_3} = (\beta_{p_3})^{\gamma_k}\) mod \(p_3\), \(\delta_x = (\omega_i)^{\gamma_k}\) mod \(p_3\). So, ciphertexts are semifunctional.

**(2)** The proof method of the second aspect is the same as Lemma 1 too.

In dual-system encryption, both the simulator \(\mathcal{B}\) and the adversary \(\mathcal{A}\) are required to be unable to determine whether the key is a semifunctional key. Even though the secret-sharing value of \(\delta_x\) is hidden from \(\mathcal{A}\), the secret-sharing value needs to be set to 0. By setting the secret-sharing value of \(\delta_x\) to 0, the ciphertext generated by \(\mathcal{B}\) is a nominal semifunctional ciphertext. In this way, if \(\mathcal{B}\) wants to test whether the \(j\)-th key is a semifunctional key or a normal key, it will generate challenge ciphertexts that this key can decrypt. However, the ciphertexts are all nominal semifunctional ciphertexts, and users can successfully decrypt them. In other words, \(\mathcal{B}\) cannot tell whether the \(j\)-th key is a semifunctional key of type-1 or a normal key.

Therefore, when \(T \in G_p\), the keys are normal and simulator \(\mathcal{B}\) can simulate the Game\(_{j,1,2}\). When \(T \in G_p\), keys are semifunctional keys of type-1 and simulator \(\mathcal{B}\) can simulate the Game\(_{j,1,2}\). \(\mathcal{B}\) can use \(\mathcal{A}\) to break Assumption 2 with advantage \(\epsilon\).
calculated as follows:

$$g_3^{\beta_i/\gamma_{iyu}} \text{, where } K'_1 \text{ is semifunctional of type-2, and}$$

$$c = 1/y_{iu} + \lambda, t_i = \beta_i.$$

When \( k = j \), if \( T \in G_{p;\beta} \), let \( T = g_1^{\beta_i} g_2^{\beta_j} \), then \( K'_2 = H(u) = g_1^{\beta_i g_2^{\beta_i/\gamma_{iyu}}} \) and \( K'_1 = T g_1^{\beta_i g_2^{\beta_j/\gamma_{iyu}}} \), where \( K'_1 \) is the semifunctional key of type-1, where \( c = \theta_j/y_{iu} + u, z_i = \beta_i \); if \( T \in G_{p;\beta} \), let \( T = g_1^{\beta_i} g_3^{\beta_j} \), then \( K'_2 = T g_1^{\beta_i g_3^{\beta_j/\gamma_{iyu}}} \), and \( K'_1 = g_1^{\beta_i g_3^{\beta_j/\gamma_{iyu}}} \). When \( x \) is semifunctional, the access structure \( A \) can be simulated to break Assumption 3 with advantage \( \varepsilon \).

6.7.4. Its Proof

Lemma 4. If there is a polynomial-time algorithm \( \mathcal{A} \) that can distinguish Game1 and Game2 at the nonnegligible advantage \( \varepsilon \), another polynomial-time algorithm \( \mathcal{B} \) will be constructed to break Assumption 4 with the advantage \( \varepsilon \).

Proof

The proof of Lemma 4 is similar to that of Lemma 3, but challenger \( \mathcal{C} \) needs to send the public parameter \((N, g_1, g_2, g_3, g_1^b, g_2^b, g_3^b, T)\) to simulator \( \mathcal{B} \).

GlobalSetup and AuthoritySetup: the same as in Lemma 2. According to the difference of \( T \), \( \mathcal{A} \) and \( \mathcal{B} \) will simulate Game1 and Game2 at the first elements of them, respectively. According to the nature of the complexity assumption [34], we can find a vector with 1 as the first element, which can be orthogonal to all the rows in \( \mathcal{B} \).

The solution is that \( \mathcal{B} \) randomly selects \( v_1 \) and \( v_2 \), where 1 and 0 are the first elements of them, respectively. According to the nature of the complexity assumption [34], we can find a vector with 1 as the first element, which can be orthogonal to all the rows in \( \mathcal{B} \).

If \( A_{x} \in B, \mathcal{B} \) will randomly choose \( r_x, a_x \in Z_N \), where \( r_x = r_x, a_x \in Z_N \), where \( r_x = r_x, a_x \in Z_N \). Then, the ciphertexts can be calculated as follows:

$$C_{1x} = e(g_1, g_1)\lambda x e(g_1, g_1)^{\nu_{iyu}} R_x, C_{2x} = C_{2x},$$

$$C_{3x} = g_1^{\alpha_i} g_2^{\beta_j/\gamma_{iyu}} (Y_2, Y_2)^{\nu_{iyu}}. \tag{23}$$

(1) If \( A_{x} \in B, C_{1x} = g_1^{\alpha_x} g_2^{\beta_j/\gamma_{iyu}} (Y_2, Y_2)^{\nu_{iyu}} = C_{3x} \lambda x g_3^{\nu_{iyu}}, \) the exponents of \( g_3 \) and \( g_2 \) will be random, because \( (\nu_{iyu}) \mod (p_3) \) and \( (\alpha_x) \mod (p_3) \) are uncorrelated, where \( \delta_x \equiv (\nu_{iyu}) \mod (p_3), \sigma_x \equiv (\alpha_x) \mod (p_3). \) So, ciphertexts are semifunctional.

If \( A_x \in B, C_{1x} = (Y_2, Y_2)^{\nu_{iyu}} g_2^{\beta_j/\gamma_{iyu}} g_3^{\nu_{iyu}}, \) the exponents of \( g_2 \) and \( g_3 \) will be computed as \( a_x \equiv (tr_x) \mod (p_3), b_x \equiv (pr_x) \mod (p_3), \)

If \( A_x \in B, C_{2x} = g_1^{\alpha_x} g_2^{\beta_j/\gamma_{iyu}} (Y_2, Y_2)^{\nu_{iyu}} g_3^{\nu_{iyu}} = C_{3x} \lambda x g_3^{\nu_{iyu}}, \) the exponents of \( g_2 \) and \( g_3 \) will be computed as \( x \equiv (tr_x) \mod (p_3), b_x \equiv (pr_x) \mod (p_3), \) so, ciphertexts are semifunctional.

(2) When \( k = j \), the key is a semifunctional key. So, whether it is a type-1 semifunctional key or a type-2 one, the ciphertexts cannot be decrypted, which means that the key is indistinguishable.

Therefore, when \( G_{p;\beta} \) the keys are normal and simulator \( \mathcal{B} \) can simulate the Game1. When \( T \in G_{p;\beta} \), keys are semifunctional keys of type-1 and simulator \( \mathcal{B} \) can simulate the Game1. \( \mathcal{B} \) can use \( \mathcal{A} \) to break Assumption 3 with advantage \( \varepsilon \).
\[ C_{1,x} = e(g_1, g_1^{r_x}) e(g_1, g_1^{r_x})^{-\eta_x} a_v e(g_1, g_1)^{\lambda x + r_x r_y} \]  

(26)

\[ \mathcal{B} \text{ randomly generates two vectors } \varphi = \{0, \varphi_2, \ldots, \varphi_{\rho(x)}\} \in Z_N^\rho \text{ and a number } \varphi_x \in Z_N, \text{ where } \omega_x = \alpha x \varphi_x. \text{ We define } C_{2,x} \text{ as follows:} \]

\[ C_{2,x} = (g_1^{r_x} g_3^{r_x})^{-\eta_x} g_1^{\rho x} g_3^{\rho x} g_1^{\eta_x + r_x \beta \rho x} g_3^{r_x}, \]  

(27)

where \( \eta_x \text{ and } \varphi_{\rho(x)} \) are uncorrelated under \( \text{mod}(p_2) \) and \( \text{mod}(p_3) \). So, we can get that \( \delta_x = (\eta_x) \text{ mod}(p_3) \), \( \varphi_x = (\varphi_{\rho(x)}) \text{ mod}(p_2) \), \( \alpha_x = (\varphi_x) \text{ mod}(p_3) \), and \( \eta_x = (\eta_x) \text{ mod}(p_2) \). Therefore, ciphertexts are semifunctional.

If \( \alpha x \not\perp \beta \rho x \), \( \omega_x = \alpha x \varphi_x \), ciphertexts will be semifunctional ciphertexts, namely, \( \omega_x \) is an attribute.

If \( \alpha x \perp \beta \rho x \), \( \omega_x \) is a random, ciphertexts are semifunctional.

Because \( \eta_x \text{ mod}(p_2) \) and \( \eta_x \text{ mod}(p_3) \) are uncorrelated, these ciphertexts are normal.

If \( s = abc \), ciphertexts will be semifunctional ciphertexts, which are properly distributed. If \( T = e(g_1, g_1)^{abc} \) is the semifunctional encryption of the message \( M_0 \), the game \( \mathcal{B} \) will simulate Gamefinal. In addition, if \( T \) is random, \( \mathcal{B} \) will simulate Gamefinal. Therefore, \( \mathcal{B} \) can use \( \delta \) to break Assumption 4 with advantage \( \epsilon \).

So, in the final attack game of Lemma 4, when the challenger \( \mathcal{E} \) encrypts a random message, the adversary’s attack advantage is negligible. As a result, according to the defined Assumptions 1–4, we can prove that real attack games and the above attack games are indistinguishable. Thus, it proves that the adversary’s advantage in winning the real attack game is negligible.

### 7. Security of the Anonymous Key Generation Protocol

The security proof of the anonymous key generation protocol is divided into two parts, namely, leak-freeness and selective-failure blindness, which are used for legitimate and malicious attribute authorities.

#### 7.1. Leak-freeness

We assume that a malicious user \( U \) interacts with a legitimate attribute authority \( AA_i \) and runs this anonymous key generation protocol in a real security game. In an ideal security game, there is a simulator \( \delta \) that can execute the KeyGen algorithm with a trusted party. Nevertheless, there is no distinguisher \( \mathcal{D} \) that can distinguish an ideal game from a real one. The interaction between \( U \) and \( \mathcal{D} \) can be simulated by the simulator \( \delta \), and \( \mathcal{D} \) can perform the following steps.

(1) \( \delta \) sends the public keys of \( AA_i \) to \( U \).

(2) \( U \) sends \( (P, Q) \) to \( \delta \), then uses zero-knowledge proof to prove he/she does own \( (u, \rho) \).

(3) \( \delta \) sends the element \( \omega \) to the trusted attribute authority \( AA_i \), then \( AA_i \) generates user attribute secret keys \( K_1 \) and \( K_2 \) to \( \delta \).

(4) \( \delta \) computes \( K_1' = K_1^{\omega} \) and \( K_2' = K_2^{\omega} \). Then, \( \delta \) returns \( K_1 ' \) and \( K_2 ' \) to \( U \).

If \( K_1 ' \) and \( K_2 ' \) are correct keys generated from a trusted attribute authority \( AA_i \) in the ideal game, \( K_1 ' \) and \( K_2 ' \) will be considered to be the right keys of \( AA_i \) in the real game. Hence, \( \mathcal{D} \) cannot distinguish whether the game is real or ideal.

#### 7.2. Selective-Failure Blindness

We assume that a malicious attribute authority \( AA_i \) generates its public key \( PK_i \) and two global user identifiers \( u_0 \) and \( u_1 \). \( AA_i \) produces a random bit \( b \in \{0, 1\} \) and then it accesses two oracles in a black-box manner, namely, \( O'(g, u_b, \tilde{u}_b PK_j) \) and \( O'(g, u_{1-b}, \tilde{u}_{1-b} PK_j) \), which are adopted to play the role of legitimate users. \( O \) and \( AA_i \) execute the anonymous key generation protocol and then \( O \) can get two kinds of keys \( SK_{u_0} \) and \( SK_{u_1} \). If there are some errors during the execution of the anonymous key generation protocol, the returned key will be considered meaningless and return \( SK_{u_b} = \bot \).

If \( SK_{u_0} \neq \bot \) and \( SK_{u_1} \neq \bot \), \( AA_i \) outputs \( (SK_{u_0}, SK_{u_1}) \); \( SK_{u_0} = \bot \) and \( SK_{u_1} \neq \bot \), \( AA_i \) outputs \( (\bot, SK_{u_1}) \); \( SK_{u_0} \neq \bot \) and \( SK_{u_1} = \bot \), \( AA_i \) outputs \( (SK_{u_0}, \bot) \); \( SK_{u_0} = \bot \) and \( SK_{u_1} = \bot \), \( AA_i \) outputs \( (\bot, \bot) \).

In the anonymous key generation protocol, \( O \) provides \( (P, Q) \) to \( AA_i \) and computes \( PoK(\mu, \rho) \) with \( P = g_1^{\rho} \) and \( Q = h_\mu \).

Then, \( AA_i \) performs two oracles, which are regarded in-calculation distinguished. Otherwise, the indistinguishability of the zero-knowledge proof and the hidden policy of the commitment scheme will be broken. If \( AA_i \) can use any of the calculation strategies to output the key of the first oracle \( (K_1', K_2', K_1'' \text{ or } K_2'') \), it will predict the key \( SK_{u_b} \) without interacting with the two oracles.

(1) \( AA_i \) verifies \( PoK_1((\alpha, \beta, \eta)) \); \( K_1' = P^{\alpha} h_\beta^{\eta} \land K_2' = Q^{\eta} \). If the proof is a fault, \( AA_i \) will output \( SK_{u_b} = \bot \).

(2) \( AA_i \) generates a pair of different keys \( (K_1', K_2') \) for the second oracle and verifies \( PoK_1((\alpha, \beta, \eta)) = Q^{\eta} \).
\[ K_i' = \text{P}^x \cdot \text{h}^{\beta/\pi \cdot K_2'} = Q^{1/\eta}. \] If the proof is a fault, \( AA_i \) will output \( SK_{u_i,b} = \perp \).

(3) If the verification fails, that is, if \( SK_{u_i} = \perp \) and \( SK_{u_i,b} \neq \perp \), \( AA_i \) will output \( \perp, SK_{u_i,b} \). If \( SK_{u_i} \neq \perp \) and \( SK_{u_i,b} = \perp \), \( AA_i \) will output \( (SK_{u_i,b}, \perp) \), and if \( SK_{u_i} = \perp \) and \( SK_{u_i,b} = \perp \), \( AA_i \) will output \( (\perp, \perp) \).

(4) If the verification is successful, \( AA_i \) will execute the anonymous key generation protocol by itself using the inputs \((u_i, A_{u_i}, (u_{1-b}, A_{u_{1-b}}))\) and the output \((SK_{u_i}, SK_{u_{1-b}})\).

When \( AA_i \) is verified like a trusted \( O \), the two predictions will have the same distribution. So, \( AA_i \) can predict the output of the two predictions and has the same advantages in distinguishing \( O(gp, u_i, A_{u_i}, PK_i) \) and \( O(gp, u_{1-b}, A_{u_{1-b}}, PK_i) \) as the same final output. Therefore, the advantage of \( AA_i \) in distinguishing two predictions comes from the received \((K_i', K_j', K_i^*, K_j^*)\) and authentication \( PoK[(\alpha, \beta, \eta)]: K_i^* = \text{P}^x \cdot \text{h}^{\beta/\pi \cdot K_2^*} = Q^{1/\eta} \). However, because of the witness indistinguishability of zero-knowledge proof and the strategic hiddenness of the commitment mechanism, we can infer that the advantage of \( AA_i \) in distinguishing between the two predictions is negligible.

8. Performance Analysis

In this section, we have the comparisons between our scheme with other schemes in Qin et al. [22], Liang et al. [37], and Malluhi et al. [38] from the view of property and efficiency to reflect the feasibility as well as the practicality of our work.

8.1. Property Analysis. In Table 1, we can find comparisons in scheme properties, such as attribute revocable, policy hiding, user identity hiding, and anonymous key generation, which reflects the comprehensiveness of our scheme in those compared schemes. Considering the actual application scenarios, more comprehensive features are of greater significance for protecting system data security and user privacy.

8.2. Theoretical Analysis. We compare our scheme with other schemes for blockchain-based access control in terms of attribute authority (AA)’s secret key size, user secret key size, ciphertext size, and decryption overhead. We define that \( N \) is the total number of attributes. \( K \) is the number of attribute authorities. \( A \) is the access structure. \(|Z_{\alpha}^*| = 60\text{byte}, |G| = 124\text{byte}, |G_1| = 124\text{byte}| \) are the lengths of the group \( Z_{\alpha}^* \), \( G \), and \( G_1 \) in the pairing group. \([A_{\alpha}] \) and \([AA_i]\) are the attribute number of user \( \alpha \) and authority \( i \). \( P \) is the cost of performing a bilinear operation. \( E \) is the cost of performing an exponential operation. It is worth noting that there is not the concept of attribute authority in [38], but a special key user can realize the function of generating secret keys for other users in the whole system. When comparing the schemes in this paper, we considered the special key users as attribute authorities.

As shown in Table 2, the size of the attribute authority secret key of our work is smaller than the other works. For the size of user secret, our work is the same as [37] and [38]. When \( 2K \) is smaller than \([A_{\alpha}]\), it will also be smaller than [22]. In terms of ciphertext, our work is slightly higher than [22, 38]. Because the exponential operation cost is much higher than the bilinear operation cost in the paired group, our work is most efficient when decrypting. In conclusion, our work has advantages in storage, communication, and computing.

8.3. Experiment Analysis. We use a 64bit Windows 10 laptop with an eight-core 2.40 GHz Intel Core(TM) i5-1135G7 processor and 16 GB memory for experiments. Moreover, we implemented the above scheme in Java by using Intellij IDEA 2019.3. The JPBC (Java Pairing-Based Cryptography) Library [39] and a Type-A1 curve \( y^2 = x^3 + \text{x} \) with 160 bits are respectively used to deal with the pairing computations and provide test parameters.

In order to compare the actual running time cost of the Encryption and Decryption phase under different access control structures, we designed two kinds of experiments. In Experiment 1, the number of total attributes increases while the number of access attributes is fixed. It simulates the impact on the running cost of data owners and data users when attribute authorities continuously add new attributes. In Experiment 2, the number of access attributes increases while the number of total attributes is fixed. It simulates the impact on the running cost of data owners and data users when the data owner changes the access structure. In each experiment, we simulate five attribute authorities that manage the same number of different attributes. An access structure specifies that each data user needs a given number of attributes to decrypt the ciphertext properly. We repeated each group of experiments 5 times and computed the average running time of the encryption and decryption phase. Note that each message consists of 124 random integers in these experiments. The running time is shown in Table 3. It should be noted that, according to the actual model of each scheme, we used all attribute keys to encrypt the ciphertext in the process of encryption and stored the access structure in the ciphertext. Moreover, we simulated a legal user only with access attributes to decrypt the ciphertext. Therefore, the encryption algorithm generally needs more attributes, which makes the running time cost of the encryption algorithm much higher.

For a more intuitive comparison, the comparison charts in running time of encryption, decryption, and total are shown in Figures 4 and 5. For the running time cost in encryption, our scheme is more efficient than the other scheme in [22, 38] but slightly higher than [37]. Meanwhile, the running time cost of our work has obvious advantages compared with other schemes in the decryption phase. Our scheme has obvious advantages compared with the schemes in [22, 38] for the total running time comparison. It shows that our scheme is more suitable for application in the supply chain information sharing system where many data users need to access user data frequently.
Table 1: Scheme property comparison.

| Scheme          | Attribute updatable/revocable | Policy hiding | User identity | Anonymous key generation |
|-----------------|-------------------------------|--------------|---------------|--------------------------|
| Ours            | ✓                             | ✓            | ✓             | ✓                        |
| Qin et al. [22] | ×                             | ×            | ✓             | ✓                        |
| Liang et al. [37]| ×                             | ✓            | ✓             | ✓                        |
| Malluhi et al. [38]| ✓                           | ✓            | ✓             | ✓                        |

Table 2: Scheme performance comparison.

| Scheme          | Attribute authority secret key size | User secret size | Ciphertext size | Decryption overhead |
|-----------------|-------------------------------------|------------------|----------------|---------------------|
| Ours            | $(2\lceil A^* + 1 \rceil | Z_q^*)$ | $(\lceil A^* + 2K \rceil | G)$ | $(N + 1) | G_T | (3N + 1) | G)$ | $3NP + NE$ |
| Qin et al. [22] | $(2\lceil A^* + 2 \rceil | Z_q^*)$ | $2\lceil A^* | G)$ | $(N + 1) | G_T | (2N + 1) | G)$ | $2NP + 3NE$ |
| Liang et al. [37]| $(3\lceil A^* + 1 \rceil | Z_q^*)$ | $(\lceil A^* + 2K \rceil | G)$ | $(N + 1) | G_T | (3N + 1) | G)$ | $4NP + NE$ |
| Malluhi et al. [38]| $(\lceil A^* | G)$ | $(\lceil A^* + 2K | G)$ | $| G_T | | G(| A + 1)$ | $(2N + 4P + 2NE$ |

Table 3: Experiment running time comparison.

| Access structure | Size | Ours | Liang et al. [37] | Malluhi et al. [38] | Qin et al. [22] |
|------------------|------|------|-------------------|---------------------|-----------------|
|                  |      | Encrypt | Decrypt | Encrypt | Decrypt | Encrypt | Decrypt | Encrypt | Decrypt |
| **Comparison based on access structure with a fixed number of access attributes (ms)** |
| Total = 15       | 937.6 | 277 | 1311.4 | 487.2 | 846.6 | 478 | 1022 | 369.4 |
| Total = 19       | 1159.6 | 305.6 | 1529.8 | 488 | 964.2 | 480.8 | 1195.6 | 411 |
| Total = 23       | 1434.8 | 315.2 | 2110.2 | 517.4 | 1256.2 | 475.6 | 1692.4 | 421.6 |
| Total = 27       | 1726.2 | 328 | 2315 | 507.8 | 1361.4 | 460.2 | 1893.2 | 394.2 |
| Access = 10      | 1934 | 332.6 | 2755 | 518.6 | 1662.4 | 477.8 | 2107.6 | 393.8 |
| Total = 35       | 2098.4 | 312 | 2993 | 483 | 1649.6 | 445.2 | 2356.8 | 386 |
| Access = 10      | 2299 | 300 | 3173.6 | 458.4 | 1769.2 | 432.4 | 2435.2 | 371.6 |
| Total = 43       | 2535.4 | 314.6 | 3420.6 | 473 | 1857 | 432 | 2673 | 370.8 |
| Total = 47       | 2773.8 | 312.8 | 3932.8 | 482.6 | 2227.2 | 458 | 3372 | 396.8 |
| Access = 10      | 2788.9 | 307.6 | 4015.4 | 460.6 | 2340.4 | 425 | 3342.8 | 371.8 |
| Total = 55       | 2924.4 | 308 | 4350.6 | 454 | 2470.8 | 431.2 | 3390.8 | 361.4 |

| **Comparison based on access structure with a fixed number of total attributes (ms)** |
| Access = 5       | 1840.2 | 159 | 2793.2 | 330.4 | 1887.4 | 335 | 2021 | 369.4 |
| Access = 7       | 1874.6 | 226.4 | 2820 | 425.2 | 1846.2 | 412.8 | 2112.2 | 347.2 |
| Access = 9       | 1864.4 | 276.4 | 2787 | 403.4 | 1729.8 | 446.6 | 2143.2 | 341.2 |
| Access = 11      | 2000 | 351.4 | 2647.2 | 522 | 1676.6 | 409.4 | 2062.8 | 409.4 |
| Access = 13      | 1963.2 | 441.2 | 2575.2 | 604.8 | 1699 | 556.2 | 2087.8 | 487.8 |
| Access = 15      | 1835.8 | 455.8 | 2691.4 | 705.6 | 1644.8 | 667.8 | 2120.4 | 576.8 |
| Access = 17      | 1909.6 | 494.6 | 2666.8 | 809.8 | 1636.2 | 761.4 | 1988.4 | 621.6 |
| Access = 19      | 1844 | 576.8 | 2558.2 | 977 | 1657.4 | 830.2 | 1992 | 702.8 |
| Access = 21      | 1904.8 | 656.6 | 2608 | 980.6 | 1682.6 | 896 | 2060.8 | 764.8 |
| Access = 23      | 1883.4 | 720.2 | 2608 | 1088.4 | 1672.4 | 996.4 | 2024.6 | 870 |
| Access = 25      | 1834.2 | 753.4 | 2552.4 | 1093.2 | 1681.8 | 1027.8 | 1972.6 | 905 |

Figure 4: Comparison of the running time in experiment 1. (a) Encryption. (b) Decryption. (c) Total.
In summary, our scheme has its advantages in the aspects of security, storage, and efficiency compared with the schemes in [22, 37] and [38].

9. Conclusion and Future Works

This paper aims to provide an effective access control scheme for a blockchain-based supply chain information sharing system to protect data privacy. Therefore, we propose a new privacy-preserving multiauthORITY attribute-based access control scheme. Our scheme ensures the confidentiality of blockchain data and provides fine-grained access control for data sharing. Taking into account the actual needs of the system, the scheme also supports user attribute key updates and ciphertext updates to cope with the dynamic change of user attribute authority. Furthermore, the scheme adopts LSSS as the access structure, which improves computational efficiency. In addition, the scheme is proven to be fully secure under the assumption of dual-system encryption. Finally, we prove that our work is feasible and effective through analysis and experimental comparison.

In order to test the safety and efficiency of our plan, we are working on implementing our scheme under a real blockchain-based supply chain information sharing system. In addition, our future research focuses on quantum attacks and lightweight encryption algorithms.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

Our work was supported by the National Natural Science Foundation of China, under Grant No. 61972208, and Jiangsu Postgraduate Research and Innovation Plan, under Grant Nos. KYCX20_0761 and KYCX21_0794.

References

[1] S. A. Abeyratne and R. P. Monfared, "Blockchain ready manufacturing supply chain using distributed ledger," International Journal of Renewable Energy Technology, vol. 5, no. 9, pp. 1–10, 2016.
[2] S. Nakamoto, “Bitcoin: a peer-to-peer electronic cash system,” Decentralized Business Review, vol. 2160, 2008.
[3] Y. Lu, “The blockchain: state-of-the-art and research challenges,” Journal of Industrial Information Integration, vol. 15, pp. 80–90, 2019, https://www.sciencedirect.com/science/article/pii/S2452414X19300019.
[4] A. A. Abdellatif, L. Samara, A. Mohamed, A. Erbad, C. F. Chiasserini, and M. Guizani, “Medge-chain: leveraging edge computing and blockchain for efficient medical data exchange,” IEEE Internet of Things Journal, vol. 8, no. 21, Article ID 15762, 2021.
[5] M. Sumathi and S. Sangeetha, “Blockchain based sensitive attribute storage and access monitoring in banking system,” International Journal of Cloud Applications and Computing, vol. 10, no. 2, pp. 77–92, 2020.
[6] L. Tseng, L. Wong, S. Otoum, M. Aloqaily, and J. Ben Othman, "Blockchain for managing heterogeneous internet of things: a perspective architecture," IEEE Network, vol. 34, no. 1, pp. 16–23, 2020.
[7] M. H. Abidi, H. Alkhalefah, U. Umer, and M. K. Mohammed, “Blockchain-based secure information sharing for supply chain management: optimization assisted data sanitization process,” International Journal of Intelligent Systems, vol. 36, no. 1, pp. 260–290, 2021, https://onlinelibrary.wiley.com/doi/abs/10.1002/int.22299.
[8] V. G. Venkatesh, K. Kang, B. Wang, R. Y. Zhong, and A. Zhang, “System architecture for blockchain based transparency of supply chain social sustainability,” Robotics and Computer-Integrated Manufacturing, vol. 63, 2020 https://www.sciencedirect.com/science/article/pii/S0736584519301589, Article ID 101896.
[9] G. I. Shidaganti, A. S. Inamdar, S. V. Rai, and A. M. Rajeve, “Scef: a model for prevention of ddos attacks from the cloud,” International Journal of Cloud Applications and Computing, vol. 10, no. 3, pp. 67–80, 2020.
[10] H. Berger, A. Z. Dvir, and M. Geva, "A wrinkle in time: a case study in dns poisoning," International Journal of Information Security, vol. 20, no. 3, pp. 313–329, 2021.
[11] C. Li, X. Jiang, X. Dai, and H. Jin, “Amvchain: authority management mechanism on blockchain-based voting systems,” Peer-to-peer Networking and Applications, vol. 14, pp. 1–12, 2021.

[12] S. Amit and B. Waters, “Fuzzy identity-based encryption,” in Advances in Cryptology – EUROCRYPT 2005, R. Cramer, Ed., pp. 457–473, Springer, Berlin, Germany, 2005.

[13] R. Zhang, R. Xue, and L. Liu, “Security and privacy on blockchain,” ACM Computing Surveys, vol. 52, no. 3, pp. 1–34, 2019.

[14] J. Huang, L. Kong, and G. Chen, “Towards secure industrial iot: blockchain system with credit-based consensus mechanism,” IEEE Transactions on Industrial Informatics, vol. 15, no. 6, pp. 3680–3689, 2019.

[15] M. Y. Khan, M. F. Zuhairi, T. Ali, T. Alghamdi, and J. A. Marmolejo-Saucedo, “An extended access control model for permissioned blockchain frameworks,” Wireless Networks, vol. 26, no. 7, pp. 4943–4954, 2020.

[16] L. Tan, N. Shi, K. Yu, M. Alobaidy, and Y. Jararweh, “A blockchain-empowered access control framework for smart devices in green internet of things,” ACM Transactions on Internet Technology, vol. 21, no. 3, 2021.

[17] Q. Xu, C. Tan, W. Zhu, and Y. Xiao, “Decentralized attribute-based conjunctive keyword search scheme with online/offline encryption and outsourcing decryption for cloud computing,” Future Generation Computer Systems, vol. 97, pp. 306–326, 2019, https://www.sciencedirect.com/science/article/pii/S0167739X18329613.

[18] G. Yu, X. Zha, X. Wang et al., “Enabling attribute revocation for fine-grained access control in blockchain-iot systems,” IEEE Transactions on Engineering Management, vol. 67, no. 4, pp. 1213–1230, 2020.

[19] X. Qin, Y. Huang, Z. Yang, and L. Xing, “Lbac: a lightweight blockchain-based access control scheme for the internet of things,” Information Sciences, vol. 554, pp. 222–235, 2021, https://www.sciencedirect.com/science/article/pii/S0020025521012019.

[20] M. Chase, “Multi-authority attribute based encryption,” in Theory of Cryptography, S. P. Vadhan, Ed., pp. 515–534, Springer, Berlin, Germany, 2007.

[21] R. Guo, H. Shi, Q. Zhao, and Z. Dong, “Secure attribute-based signature scheme with multiple authorities for blockchain in electronic health records systems,” IEEE Access, vol. 6, Article ID 11676, 2018.

[22] X. Qin, Y. Huang, Z. Yang, and L. Xing, “A blockchain-based access control scheme with multiple attribute authorities for secure cloud data sharing,” Journal of Systems Architecture, vol. 112, 2021 https://www.sciencedirect.com/science/article/pii/S1383762120301405, Article ID 101854.

[23] S. Banerjee, B. Bera, A. K. Das, S. Chattopadhyay, and M. K. Khan, “Private blockchain-envisioned multi-authority cp-abe-based user access control scheme in iiot,” Computer Communications, vol. 169, pp. 99–113, 2021, https://www.sciencedirect.com/science/article/pii/S014036642100044X.

[24] Y. Hei, J. Liu, H. Feng, D. Li, Y. Liu, and Q. Wu, “Making ma-abe fully accountable: a blockchain-based approach for secure digital right management,” Computer Networks, vol. 191, 2021 https://www.sciencedirect.com/science/article/pii/S138912862100389, Article ID 108029.

[25] K. Sethi, A. Pradhan, and P. Bera, “Pmter-abe: a practical multi-authority cp-abe with traceability, revocation and outsourcing decryption for secure access control in cloud systems,” Cluster Computing, vol. 24, no. 2, pp. 1525–1550, 2021.

[26] S. Rouhani, R. Belchior, R. S. Cruz, and R. Deters, “Distributed attribute-based access control system using permissioned blockchain,” World Wide Web, vol. 24, no. 5, pp. 1617–1644, 2021.

[27] D. Boneh, E.-J. Goh, and K. Nissim, “Evaluating 2-dnf formulas on ciphertexts,” in Theory of Cryptography, Joe Kilian, Ed., Springer, Berlin, Germany, 2005.

[28] L. Allison and B. Waters, “New techniques for dual system encryption and fully secure hibe with short ciphertexts,” in Theory of Cryptography, D. Micciancio, Ed., Springer, Berlin, Germany, 2010.

[29] L. Allison and B. Waters, “Decentralizing attribute-based encryption,” in Advances in Cryptology – EUROCRYPT 2011, K. G. Paterson, Ed., Springer, Berlin, Germany, 2011.

[30] A. Beimel, Secure schemes for secret sharing and key distribution, phd thesis, israel institute of technology technion, Haifa, Israel, 1996.

[31] T. P. Pedersen, “Non-interactive and information-theoretic secure verifiable secret sharing,” in Advances in Cryptology — CRYPTO ’91, J. Feigenbaum, Ed., Springer, Berlin, Germany, 1992.

[32] C. Jan and M. Stadler, “Efficient group signature schemes for large groups,” in Advances in Cryptology — CRYPTO ’97, B. S. Kaliski, Ed., Springer, Berlin, Germany, 1997.

[33] J. Li, Q. Yu, and Y. Zhang, “Hierarchical attribute based encryption with continuous leakage-resilience,” Information Sciences, vol. 484, pp. 113–134, 2019, https://www.sciencedirect.com/science/article/pii/S0020025519300684.

[34] J. Zhang and M. Wu, “Efficient cloud-based private set intersection protocol with hidden access attribute and integrity verification,” Security and Communication Networks, vol. 2021, Article ID 3963037, 13 pages, 2021.

[35] B. Waters, “Dual system encryption: realizing fully secure ibe and hibe under simple assumptions,” in Advances in Cryptology – CRYPTO 2009, S. Halevi, Ed., Springer, Berlin, Germany, 2009.

[36] Y. Yang, X. Chen, H. Chen, and X. Du, “Improving privacy and security in decentralizing multi-authority attribute-based encryption in cloud computing,” IEEE Access, vol. 6, Article ID 18009, 2018.

[37] P. Liang, L. Zhang, L. Kang, and J. Ren, “Privacy-preserving decentralized abe for secure sharing of personal health records in cloud storage,” Journal of Information Security and Applications, vol. 47, pp. 258–266, 2019, https://www.sciencedirect.com/science/article/pii/S2214212618307749.

[38] J. Feigenbaum, V. Shkaya, V. D. Tran, and V. C. Trinh, “Decentralized ciphertext-policy attribute-based encryption schemes for lightweight devices,” Computer Communications, vol. 145, pp. 113–125, 2019, https://www.sciencedirect.com/science/article/pii/S014036641930324X.

[39] A. De Caro and V. Ioovino, “Jpbc: Java pairing based cryptograph,” in Proceedings of the IEEE Symposium on Computers and Communications (ISCC), Kerkarya, Greece, June 2011.