Hindawi
Security and Communication Networks
Volume 2022, Article ID 4978802, 14 pages
https://doi.org/10.1155/2022/4978802

Research Article

A Practical and Efficient Blockchain-Assisted Attribute-Based Encryption Scheme for Access Control and Data Sharing

Linjian Hong,[1] **Kai Zhang**,[2] **Junqing Gong**,[1] **and Haifeng Qian**[3]

[1]*East China Normal University, Shanghai, China*
[2]*Shanghai University of Electric Power, Shanghai, China*
[3]*CCCC Intelligence Transportation Co., Ltd., Tianjin, China*

Correspondence should be addressed to Junqing Gong; jqgong@sei.ecnu.edu.cn

Received 25 January 2022; Accepted 8 September 2022; Published 30 September 2022

Academic Editor: Helena Rifà-Pous

Attribute-based encryption (ABE) is a powerful encryption scheme with flexible access control over encrypted data that has been widely adopted in cloud computing scenarios to facilitate data sharing. However, despite convenience and efficiency provided by data sharing based on cloud, it is commonly vulnerable to issues like key abuse (namely, illegal key sharing by user or key distribution by authority) and key escrow (namely, illegal decryption by ABE authority). Hence, exploring a more secure ABE scheme that can be key abuse and key escrow resistant is crucial. Furthermore, data modification that happens in cloud storage and outsourced computation is also a challenge for the cloud-based data sharing schemes. To handle the above issues, in this paper, we propose a secure and efficient data sharing scheme based on attribute-based encryption (ABE) and blockchain equipped with InterPlanetary File System (IPFS). In particular, we show that the large-universe ABE with outsourced decryption (LU-ABE-OD) scheme proposed by Ning et al. is vulnerable to key escrow attack, which is not secure enough in the data sharing scenario. Therefore, based on their basic proposal, we construct an improved multi-authority LU-ABE-OD scheme to encrypt personal data, which are stored in the IPFS system while blockchain is applied to store the hash value returned by IPFS and be responsible for the outsourced decryption. As a result, our scheme greatly reduces the decryption overheads of the user while risks of key abuse and key escrow can be settled. Meanwhile, the introduction of IPFS significantly reduces the storage burden on chain without data tampering problem. Through theoretical analysis and experimental simulation, we prove the feasibility, security, and efficiency of our scheme.

1. Introduction

With the advent of the era of big data, the storage, processing, and sharing of massive data has become a bottleneck. To solve this issue, cloud storage has been considered to be the most efficient, convenient, and economical method for many users. However, the trust issue in terms of data privacy between the cloud and users is brought up, and thus encrypting sensitive data before they are uploaded is of great significance. Furthermore, apart from data privacy, particular scenario like data sharing always requires flexible access of encrypted data, which leads to the necessity of a fine-grained access control encryption scheme.

Attribute-based encryption (ABE), a cryptographic primitive proposed by Sahai and Waters [1] that guarantees only those who possess certain attributes can access the targeted encrypted data, is proved to be a positive solution. According to Goyal et al. [2], ABE schemes can be generally divided into two patterns: ciphertext-policy ABE (CP-ABE) and key-policy ABE (KP-ABE). In a CP-ABE scheme, the access policy is embedded into ciphertexts and user's private key is associated with an attribute collection, which is widely used in traditional cloud-based data sharing scenarios. However, practical applications based on basic ABE schemes suffer from the restriction of small-universe setting (SU-ABE) [3], in which the attribute universe must be determined when generating public parameters and no attribute can be added then. Hence, large-universe ABE (LU-ABE) [3] was proposed, realizing the unbounded attribute universe so that new attributes can be added arbitrarily. Meanwhile, key

abuse and key escrow issues remain challenges to be faced. Specifically, the former risk can be caused by malicious users illegally sharing their keys (which can also be maliciously attained by attacker, namely, key leakage) and attribute authority (AA) illegally distributing keys to unauthorized users while the latter risk mainly caused by curious AA who masters all users' private keys and intends to violate their privacy. For key abuse problem, traceable ABE schemes [4–6] were proposed to track the owner of leaked or abused key to relieve subsequent impact. For key escrow issue, multi-authority ABE (MA-ABE) schemes [7–9] were considered to be promising since the powerful decryption ability of a single center has been dispersed and none of the centers can illegally intrude on users' privacy.

Apart from all the above, traditional ABE schemes also suffer from the limitation in efficiency. Furthermore, data encrypted by ABE will lead to huge decryption costs. To solve these issues especially for resource-limited users, Green et al. [13] proposed the outsourcing decryption scheme of ABE (ABE-OD), which greatly reduces local decryption overheads by outsourcing most of decryption tasks to the cloud. Later, based on Waters's CP-ABE scheme, Lai et al. [14] realized the verification of ABE-OD in the standard model to handle the malicious computation in ABE-OD scenarios. Along the previous constructions, schemes [15–19] have been proposed to promote efficiency and reduce costs. Nonetheless, the introduction of cloud in the above leads to the centralization of system, which will bring up problems of data tampering.

Blockchain, with its powerful decentralized mechanism, has attracted a growing number of researchers and provides feasible approaches to address the impact of cloud since data stored on chain are tamper resistant and transparent. Efficient blockchain-based CP-ABE schemes for specific data sharing scenarios like electronic health records (EHRs) and Internet of Things (IoT) [20–26] have received widespread attention. However, in most mentioned cases, the decryption of ciphertexts is still outsourced to cloud while blockchain is only used to record access control data in which storage burden on chain raises another problem. Fortunately, benefiting from peer-to-peer distributed storage systems such as InterPlanetary File System (IPFS) [27] and distributed hash table (DHT) [28], the challenge of on-chain data storage can be well solved through stored off chain. Schemes in [12, 29, 30] adopting such distributed storage system instead of on-chain or cloud storage are proved secure and efficient. Furthermore, according to the above proposals, blockchain can also be considered as an effective technique that takes over decryption tasks as the role of cloud, thus making it significant to explore both secure and efficient blockchain-based data sharing scheme.

1.1. Motivation. In traditional data sharing scenarios, cloud computing and cloud storage are usually adopted to process data. However, despite great convenience brought by the introduction of the third party, it unavoidably leads to trust issues. Most schemes assume that the third party (i.e., the cloud) is semi-trusted or even trusted, which means users can only confirm whether the result of outsourced computation is correct and the stored data are tampered. Nonetheless, it cannot prevent the underlying threat of data modification by malicious cloud in practice. To address the issue of data tampering, a promising solution is to introduce blockchain or shared database for storage instead of the cloud. However, in terms of huge computational costs of ABE decryption in data sharing scenarios, we should also consider the improvement of decryption efficiency especially for the users who are resource-limited. Blockchain equipped with smart contracts or chaincode, which can perform specific computing task, provides a feasible solution. Among the previous mentioned blockchain-based data sharing schemes, proposals in [20–26] ignored the huge data storage on chain. Wang et al. [29] proposed an improved scheme by IPFS, but the user has to take huge computational costs, while Wang et al. [12] put forward an ABE-OD scheme assisted by blockchain to reduce users' costs, but it ignored the security in practice. Pham et al. [30] considered both security and efficiency, but their scheme lacks concrete instantiation and security analysis. Therefore, it is still necessary to explore a secure and efficient blockchain-based access control and data sharing scheme, which is the focus of this article.

1.2. Contributions. In this paper, we mainly propose a data sharing system based on blockchain and ABE which can achieve both high security and efficiency for data users. In particular, we present a blockchain-based large-universe multi-authority CP-ABE with outsourced decryption scheme (LU-MAABE-OD) based on basic outsourced CP-ABE scheme proposed by Ning et al. [10] and traceable multi-authority CP-ABE scheme proposed by Zhang et al. [11]. Our contributions are as follows:

(i) We construct a user-friendly data sharing scheme that combines large-universe CP-ABE and blockchain to promote flexibility and efficiency. The constant size of public parameters makes the system more scalable, and it will be practical and convenient for users to join without the limitation on attribute universe. Moreover, partial decryption assisted by smart contracts or chaincode in blockchain instead of the cloud in traditional scenarios can further reduce the computation overhead of the user side since the verification is not necessary.

(ii) Our proposed scheme is multi-authority based and key escrow and key abuse resistant. The user's decryption secret key is produced by the collaboration of two semi-trusted parties, i.e., key generation center (KGC) and attribute authority (AA), which is also blinded by the public key owned by user, so that none of participants can forge the key alone. Besides, in our scheme, the key generated is a transformation key which can just partially recover the ciphertexts and can be public, and thus none of the third party can recover the ciphertexts (i.e., the encapsulated key). Furthermore, due to the

sophisticated design of key sanity and owner check mechanism before the decryption of blockchain, the public transformation key can only be used by its owner in our system, thus avoiding malicious computation burden on chain.

(iii) We give formal security analysis and concrete construction which proves that our proposal is feasible and secure. Besides, experimental simulation and comparison indicate that our scheme is efficient, especially for resource-constrained users.

Table 1 shows a functional comparison between our proposal and several representative related schemes.

1.3. Organization. We organize the rest of our paper as follows. In Sections 2 and 3, we briefly review the related work and preliminaries involved in our scheme, respectively. Then, in Section 4, the overview of our system model, system assumptions, formal description, and security model will be displayed. In Section 5, we will describe our concrete scheme and security analysis in detail. Next, we evaluate the system in Section 6 and finally draw a conclusion in Section 7.

2. Related Work

2.1. Attribute-Based Encryption. Attribute-based encryption (ABE), first proposed by Sahai and Waters [1], was a sophisticated cryptography scheme that could support flexible and fine-grained access control, thus commonly being used in data sharing scenarios. However, the huge computational overhead has been the bottleneck, thus attracting numerous relevant research studies [31–33] on promoting the security, expressiveness, and efficiency of ABE. Among them, a promising method to reduce costs of user is to outsource partial decryption to a proxy (i.e., cloud), which was formalized in [13] as ABE with outsourcing decryption (ABE-OD). In such a setting, users only need to expose a transformation key to cloud while the final decryption key is kept secretly. To solve the invalid decryption in ABE-OD scenarios, Lai et al. [14] put forward a verifiable ABE-OD scheme while sacrificing the size of ciphertexts and operation complexity compared with [31], followed by schemes [16–19] proposed to achieve higher efficiency and expressiveness. However, all of the above schemes restrict the final decryption and verification to individual who owns the final decryption key, thus leading to inflexible data sharing and access since data users must obtain the final decryption key from the data owner. Besides, the limitation on attribute universe restricts their practical scenarios because no additional attribute can be added when the public parameters are set up, which was defined as small-universe ABE (SU-ABE) [3]. For better scalability, large-universe ABE (LU-ABE) [3] was put forward so that new attributes can be added arbitrarily. Nonetheless, key abuse and key escrow remain two obstacles. To achieve higher security, traceable ABE schemes [4–6] that could deal with key abuse issue were proved effective by embedding user's id in the key. In 2017, Ning et al. [10] presented an efficient key abuse resistant LU-ABE-OD scheme while it depends on a single authority that is considered to be honest, thus leading to key escrow problem. To remove the impact caused by centralization of single authority, multi-authority ABE schemes [7–9] which could deal with key escrow problem were proposed. Since the key is jointly generated by multiple AA, any AA cannot forge the key or decrypt underlying data. Later, Zhang et al. [11] proposed a multi-authority LU-ABE scheme which is resistant to key escrow and key abuse attack. In their scheme, the key was comprised by the collaboration of two authorities (i.e., KGC and AA) and traceable mechanism was integrated to follow the abused key. However, the user decryption cost has not been taken into consideration so that it cannot be well performed particularly for resource-limited users. Furthermore, although all of the abovementioned ABE schemes can achieve higher efficiency or security in data sharing scenarios, most of them ignore the impact of malicious cloud services while data integrity is of great significance especially for data owners. In a word, research on more secure data sharing scheme is worthy of exploration.

2.2. Blockchain-Based Data Sharing Scheme. The development of blockchain technology contributes to addressing issues in data sharing caused by traditional cloud storage system due to its transparency and tamper resistance. For better combination with practical applications, schemes in [20–23] proposed blockchain-based EHRs sharing models and schemes in [24–26] put forward blockchain-based sharing models in scenario of IoT (Internet of Things). Despite the sophisticated design of ABE schemes that can support traceable, searchable, and outsourced decryption, key leakage resistance, etc., none of them concentrates on the storage burden on chain caused by huge ABE ciphertext storage overheads. To reduce overheads on chain, distributed storage systems like IPFS and DHT have been widely concerned. Based on IPFS and smart contracts, Wang et al. [29] put forward a data sharing model in which personal data management is controlled by data owner himself. Despite the promotion of privacy, data owner has to bear huge computation costs of key generation and distribution while data user has to take on the whole decryption cost, too. Similarly, Wang et al. [12] put forward a personal privacy data protection scheme based on blockchain and DHT, combing proxy reencryption scheme to outsource partial decryption task to blockchain and achieving access transparency. Nonetheless, data owner still undertakes large amounts of computation tasks and the proposed ABE scheme is actually poor in security. Later, a decentralized storage system combining IPFS, ABE, and blockchain was proposed by Pham et al. [30] without concrete construction and experimental analysis that can prove the feasibility. Therefore, exploring a more secure and efficient data sharing scheme in decentralized storage system remains an issue.

3. Preliminaries

In this section, we briefly present the involved basic knowledge related to our system.

3.1. Bilinear Pairings. Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ denote three multiplicative cyclic groups of the same prime order p. g_1 and g_2 are the generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively.

Table 1: A comparison between our scheme and related schemes.

Scheme	Large universe	Multi-authority	Outsourced decryption	Blockchain	Access structure
NCD + 17 [10]	✓	×	✓	×	LSSS
ZZP + 20 [11]	✓	✓	×	×	LSSS
WCY21 [12]	×	×	✓	✓	Tree
Our scheme	✓	✓	✓	✓	LSSS

e: $\mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ is a bilinear map with properties as follows:

(i) Bilinearity: for all $a, b \in \mathbb{Z}_p$, $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$.

(ii) Non-degeneracy: $e(g_1, g_2) \neq 1$.

(iii) Computability: e should be efficiently computable.

3.2. Access Structure

Definition 1 (access structure [34]). Let $\mathcal{U}$ be the attribute universe. A collection $\mathbb{A} \subseteq 2^{\mathcal{U}}$ is monotone while $\forall \mathcal{X}, \mathcal{Y}$ satisfy that if $\mathcal{X} \in \mathbb{A}$ and $\mathcal{X} \subseteq \mathcal{Y}$ then $\mathcal{Y} \in \mathbb{A}$. An access structure (respectively, monotone access structure) is a collection (resp. monotone collection) $\mathbb{A}$ of non-empty subsets of $\mathcal{U}$, i.e., $\mathbb{A} \subseteq 2^{\mathcal{U}}/\{\emptyset\}$. The sets in $\mathbb{A}$ are called the authorized sets while the sets not in $\mathbb{A}$ are called the unauthorized ones. In this paper, the role of the parties is delegated by attributes. Thus, $\mathbb{A}$ is a collection of authorized attributes and we only discuss the monotone access structure.

3.3. Linear Secret Sharing Schemes (LSSS)

Definition 2 (linear secret sharing schemes [34]): Let $\mathcal{U}$ denote the attribute and p be a prime. A secret sharing scheme Π over a set of parties on $\mathcal{U}$ is called linear (over $\mathbb{Z}_p$) if it satisfies

(1) The shares of secret $s \in \mathbb{Z}_p$ for each attribute form a vector over $\mathbb{Z}_p$.

(2) For each access structure $\mathbb{A}$ on $\mathcal{U}$, there exists a share-generating matrix M with l rows and n columns for Π. And for $i = 1, \ldots, l$, a function ρ labels i^{th} row of M with attribute $\rho(i)$ from $\mathbb{S}$. During the generation of the shares s, we consider the column vector $\overrightarrow{v} = (s, r_2, \ldots, r_n)$, where $r_2, \ldots, r_n \in \mathbb{Z}_p$ are randomly chosen. Then the vector of l shares of s according to Π is equal to $M\overrightarrow{v} \in \mathbb{Z}_p^{l \times 1}$. The share $(M\overrightarrow{v})_j$ where $j \in [l]$ "belongs" to attribute $\rho(j)$.

According to [34], each linear secret sharing scheme satisfying the above definition enjoys the linear reconstruction property, defined as follows: suppose that Π is an LSSS for an access control $\mathbb{A}$, and $S^* \in \mathbb{A}$ is an authorized set. Let $I \subset \{1, 2, \ldots, l\}$ be defined as $I = \{i \in [l] \wedge \rho(i) \in S^*\}$. For any valid shares $\{\lambda = (M\overrightarrow{v})_i\}_{i \in I}$ of a secret s according to Π, there exist constants $\{\omega_i \in \mathbb{Z}_p\}_{i \in I}$ such that $\sum_{i \in I} \omega_i \lambda_i = s$.

3.4. Blockchain and IPFS.

First proposed in [35], blockchain is known as the underlying concept of Bitcoin and can simply be considered as a distributed ledger in the peer-to-peer network, in which all transactions can be recorded on chain. As each transaction in blockchain is public, traceable, and tamper resistant, it gradually plays an important role in various industrial scenarios (i.e., medical, financial, educational, and so on) and blockchain-based system Ethereum [36] and Hyperledger Fabric [37] have achieved prominent achievement. Most significantly, the introduction of blockchain solves the trust problem in traditional centralized systems due to the arrival of smart contract [38] (in Ethereum, or called chaincode in Hyperledger Fabric), which is a series of codes that can be executed and recorded automatically when running conditions are triggered. Once executed successfully, the result cannot be tampered. Nonetheless, blockchain has to undertake the issue of massive data storage on chain, and a promising method is to transfer the data off chain. IPFS, a peer-to-peer distributed file system proposed by Benet [27], provides a promising solution. Unlike traditional Internet HTTP protocol which needs to search the domain address, IPFS is designed to search the content address so that any peer can be a server and can also request the file in content-addressed block, thus contributing to cheaper, faster, and more secure network services. With the assistance of IPFS, the hash value of data can be stored on chain instead while the data can be stored in IPFS. So far, the combination of blockchain and IPFS system has attracted attention from researchers in different fields.

3.5. Review of Ning Et Al.'s Basic Scheme [10].

Our blockchain-based outsourced CP-ABE scheme is related to that of basic outsourced CP-ABE scheme [10] proposed by Ning et al., so we now revisit it. According to their construction, the system is composed of four entities, namely, cloud, authority (AA), data owner (DO), and data user (DU), in which cloud is semi-honest (who will obey the protocol but try to collect private information of users as possible) while AA is honest (who is totally trusted by other entities). Their construction can be briefly depicted as follows:

(i) $\mathbf{Setup}(1^\lambda) \longrightarrow (pp, msk)$: with the input of a security parameter λ, this algorithm (run by AA) produces a bilinear group $D = (\mathbb{G}, \mathbb{G}_T, p, e)$ and randomly chooses elements $g, h, u, v, w \in \mathbb{G}$ and $\alpha \in \mathbb{Z}_p$. It outputs the system public parameters $pp = (D, g, h, u, v, w, e(g, g)^\alpha)$ and system master secret key $msk = \alpha$.

(ii) $\mathbf{Setup}_c(pp) \longrightarrow (pk_c, sk_c)$: with the input of public parameters pp, this algorithm (run by cloud) randomly chooses $s_c \in \mathbb{Z}_p$ and generates the public key $pk_c = (P_c = g^{s_c})$ and secret key $sk_c = s_c$ of cloud.

(iii) $\mathbf{Setup}_u(pp) \longrightarrow (pk_u, sk_u)$: with the input of public parameters pp, this algorithm (run by DU) randomly chooses $s_u \in \mathbb{Z}_p$ and generates the public key $pk_u = (P_u = g^{s_u})$ and secret key $sk_u = s_u$ of user.

(iv) $\mathbf{KeyGen}(pp, pk_c, pk_u, msk, \mathcal{S}) \longrightarrow (sk_{\mathcal{S}})$: with the input of public parameters pp, public key of cloud pk_c, public key of user pk_u, the system master secret key msk and attribute set of user $\mathcal{S}$., this algorithm (run by AA) randomly chooses $\beta, r, \{r_\tau\}_{\tau\in[k]} \in \mathbb{Z}_p$ and then computes $K_0 = P_u^{\alpha} P_c^{\beta} w^r, K_1 = g^{\beta}$, $K_2 = g^r, \left\{K_{\tau,3} = g^{r_\tau}, K_{\tau,4} = (u^{A_\tau} h)^{r_\tau} v^{-r}\right\}_{\tau\in[k]}$. It finally outputs the secret key $sk_{\mathcal{S}} = (\mathcal{S}, K_0, K_1, K_2, \left\{K_{\tau,3}, K_{\tau,4}\right\}_{\tau\in[k]})$.

(v) $\mathbf{KeyGen}_{\text{out}}(sk_{\mathcal{S}}) \longrightarrow tk_{\mathcal{S}}$: the algorithm (run by DU) simply sets $tk_{\mathcal{S}} = sk_{\mathcal{S}}$ and returns the transformation key $tk_{\mathcal{S}}$.

(vi) $\mathbf{Encrypt}(pp, \mathbb{A}) \longrightarrow (ct_{\mathbb{A}}, K_{DU})$: with the input of public parameters pp and an LSSS access structure $\mathbb{A} = (M, \rho)$, this algorithm (run by DU) first randomly chooses $s \in \mathbb{Z}_p$ and $r_2, \ldots, r_n \in \mathbb{Z}_p$ to construct a column vector $\overrightarrow{v} = (s, r_2, \ldots, r_n)$ and computes the vector of shares $\overrightarrow{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_l)^T = M\overrightarrow{v}$. It also randomly chooses $r_i \in \mathbb{Z}_p$ for each M_i and $\{t_\tau\}_{\tau\in[l]} \in \mathbb{Z}_p$. It then computes $C_0 = g^s, \left\{C_{\tau,1} = w^{\lambda_\tau} v^{t_\tau}, C_{\tau,2} = (u^{\rho(\tau)} h)^{-t_\tau}, C_{\tau,3} = g^{t_\tau}\right\}_{\tau\in[l]}$, in which the encapsulated key is $K_{DU} = e(g,g)^{\alpha s}$ and ciphertext is $ct_{\mathbb{A}} = ((M,\rho), C_0, \left\{C_{\tau,1}, C_{\tau,2}, C_{\tau,3}\right\}_{\tau\in[l]})$.

(vii) $\mathbf{Decrypt}_{out}(pp, ct_{\mathbb{A}}, tk_{\mathcal{S}}, sk_c) \longrightarrow \text{pct}/\perp$: with the input of public parameters pp, the ciphertext $ct_{\mathbb{A}}$, the transformation key $tk_{\mathcal{S}}$ and secret key of cloud sk_c, this algorithm (run by cloud) calculates a share to attributes in $\mathcal{S}$ and computes the constants $\left\{\omega_i \in \mathbb{Z}_p\right\}_{i\in I}$ with the set of rows $I = \{i\colon \rho(i) \in \mathcal{S}\}$ in M such that $\sum_{i\in I}\omega_i M_i = (1, 0, \ldots, 0)$. If $\mathcal{S}$ does not satisfy the access structure $\mathbb{A}$, it outputs $\perp$. Otherwise, it calculates

$$P = \prod_{i\in I}\left(e\left(C_{\tau,1}, K_2 e\right)\left(C_{\tau,2}, K_{\tau,3}\right)e\left(C_{\tau,3}, K_{\tau,4}\right)\right)^{\omega_i},$$
$$C' = \frac{e(C_0, K_0)}{e\left(C_0, (K_1)^{sk_c}\right)\cdot P} = e(g, P_u)^{\alpha s}. \tag{1}$$

And it outputs the partially decrypted ciphertext $\text{pct} = (\mathbb{A}, C')$.

(viii) $\mathbf{Decrypt}_u(pct, sk_u) \longrightarrow key$: with the input of a partially decrypted ciphertext pct and the user secret key sk_u, the algorithm (run by DU) calculates and outputs

$$\text{key} = (C')^{sk_u^{-1}} = \left(e(g, P_u)^{\alpha s}\right)^{s_u^{-1}} = e(g,g)^{\alpha s}. \tag{2}$$

As is proved in [10], the above construction is secure under the proposed security model. Nonetheless, it cannot be practical in terms of the honest AA. In fact, if we consider AA as a semi-honest party, the proposed scheme will no longer be secure under key escrow attack.

(ix) With the master secret key α, semi-honest AA privately calculates g^{α}. With the given ciphertext $ct_{\mathbb{A}}$, semi-honest AA withdraws $C_0 = g^s$.

(x) AA easily computes $\text{key} = e(g^{\alpha}, g^s) = e(g,g)^{\alpha s}$ which is actually the encapsulated key $K_{DU} = e(g,g)^{\alpha s}$, thus making the system insecure.

To sum up, scheme in [10] is insecure in the security model comprised of semi-honest AA, and thus solving key escrow issue caused by such single-authority system and exploring scheme with higher security is necessary.

4. Our Construction

In this section, we mainly propose our construction of system model, system assumptions, formal description, and security model.

4.1. System Model. In our blockchain-based data sharing model, key/data encapsulation mechanism (KEM/DEM) [39] is applied in which our LU-MAABE-OD scheme is involved to generate the symmetric session key for encrypting data. As depicted in Figure 1, our data sharing model mainly consists of five entities, namely, key generation center (KGC), attribute authority (AA), data user (DU), data owner (DO), and blockchain (BC) equipped with smart contracts (SCs) or chaincode (chaincode is preferred in our system) and IPFS.

(i) KGC: the key generation center who participates in the setup of public parameter and key generation of DO and DU.

(ii) AA: the attribute authority who participates in the setup of public parameter and key generation of DO and DU.

(iii) DO: the owner who expects to share some data with those who satisfy specific access structure, so he/she needs to determine the access policy and the symmetric session key which is encrypted by KEM ciphertexts and further used to encrypt shared data; then he/she generates DEM ciphertexts which will be uploaded to IPFS together with KEM ciphertexts. After uploading, DO only needs to save the IPFS address of KEM/DEM ciphertexts to BC.

(iv) DU: the user who owns certain access structure and wants to access data. DU is uniquely labelled by his/her public identity ID in the whole system. Besides, DU possesses his/her own public and secret key pair which contributes to the generation of private transformation key together with his/her attribute set and ID with the help of KGC and AA. The transformation key can be

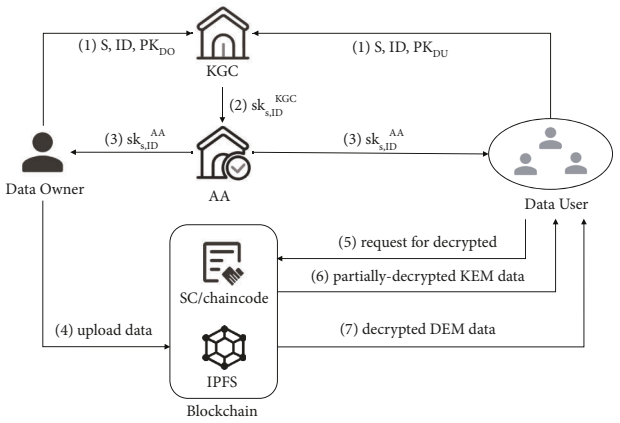

Figure 1: System overview of our scheme.

public while the secret key is always kept privately and used to make final decryption.

(v) BC: the data storage center which is maintained by a number of recording nodes, which are called miners in public blockchain like Ethereum and authorized nodes in permissioned blockchain like Hyperledger. In our scheme, BC with SC/chaincode can also undertake the task of partially decrypting KEM ciphertexts. To relieve the burden on chain, all the KEM/DEM ciphertexts are stored in IPFS while the addresses of IPFS are stored on chain. When needed, BC will retrieve the corresponding data through the IPFS address.

The workflow can be mainly described as follows:

(i) *Initialization*: in the phase of system public parameter generation, both KGC and AA will participate and keep their master keys secretly. Any user (i.e., U_i) can randomly choose his/her private key and generate his/her public key according to system public parameter. It should be noted that no user can use a duplicate public key.

(ii) *Authentication*: KGC and AA audit the authentication information ID together with attribute set $\mathcal{S}$, which can be denoted as a pair $(\mathcal{S}, ID)$ submitted by U_i and collectively generate his/her transformation key $tk_{\mathcal{S},ID}$.

(1) U_i forwards his/her key pair to KGC. After audited, an intermediate key $sk_{\mathcal{S},ID}^{KGC}$ will be produced and sent to AA.
(2) After receiving$sk_{\mathcal{S},ID}^{KGC}$, AA generates the final decryption key $sk_{\mathcal{S},ID}^{AA}$ and sends it to U_i.
(3) U_i simply keeps $sk_{\mathcal{S},ID}^{AA}$ as the transformation key $tk_{\mathcal{S},ID}$.

(iii) *DataEncryption*: with the help of KEM/DEM mechanism, DO encrypts data with the generated encapsulated key which is encrypted by KEM ciphertexts to produce DEM ciphertexts, and then KEM and DEM ciphertexts will be uploaded to IPFS. If finished, corresponding IPFS addresses should be saved on chain for searching.

(iv) *DataDecryption*: to obtain the encapsulated key for decryption, DU can decrypt KEM ciphertexts with the help of chaincode according to the following steps.

(1) DU can invoke the chaincode which is preencoded and deployed by one system maintainer of blockchain for helping decryption. If executed, the chaincode will check whether $tk_{\mathcal{S},ID}$ is used by the key owner. It first checks whether ID of DU satisfies $\mathcal{H}(ID) = L$, in which L is the embedded value in $tk_{\mathcal{S},ID}$ that records the hash of key owner's ID. If it fails, decryption will be refused. Otherwise, it will then do key sanity check to validate whether $tk_{\mathcal{S},ID}$ is just forged by changing ID. If it fails, decryption will also be refused. It should be noted that such checking is designed to avoid heavy burden caused by invalid decryption of malicious users.
(2) If the key is confirmed to be used by its owner, KEM ciphertexts from IPFS will be decrypted and partially decrypted ciphertext pct will be sent back to DU through chaincode.
(3) DU finally decrypts pct with his/her private key and achieves the encapsulated key that can be further used in the decryption of DEM ciphertexts stored in IPFS.

We briefly take the medical data sharing scenario as an example for the better understanding of our protocol. We choose Hyperledger as the blockchain in which recording nodes are composed of some permissioned regulators and the preencoded decryption chaincode is deployed by one of them, and the system initiators are KGC and AA. Any user can generate their own public-secret key pair according to the system public parameter and then achieve the transformation key generated by the cooperation of KGC and AA according to his/her unique label ID and access structure in the system. According to the specific access policy, DO who wants to share medical data can generate KEM ciphertexts that can decrypt the encapsulation key, which can be further used to encrypt the medical data to DEM ciphertexts, and then KEM/DEM ciphertexts are uploaded to IPFS. DU who wants to access medical data but lacks computing power can decrypt KEM ciphertexts by invoking the deployed chaincode with the transformation key. During the execution of the chaincode, it will first check whether the key is used by its owner. If not, the execution of the chaincode will fail. Otherwise, the chaincode will output the decrypted result and DU only needs to decrypt the encapsulation key with his/her secret key by performing one exponential operation and finally decrypts the DEM ciphertexts from IPFS with the encapsulation key to achieve the underlying medical data. It is easy to find that the application of blockchain can not only solve the problem of tamper-resistant data storage but also help the decryption of resource-limited users with chaincode.

4.2. System Assumptions. In this paper, we suppose all the entities are driven by their interests and not entirely trusted.

(i) KGC: it is a semi-honest entity who may be curious about the data owned by DO and the secret key of DU, but it will honestly execute the agreed protocol. Besides, it is noted that KGC will never collude with AA.

(ii) AA: it is also a semi-honest entity who may be curious about the data owned by DO and the secret key of DU, but it will honestly execute the agreed protocol. Besides, it is noted that AA will never collude with KGC.

(iii) DO: DO may be curious about the data owned by other DOs and he/she will always keep his/her secret key in the system.

(iv) DU: DU may be curious about the data that he/she cannot access and he/she will always keep his/her secret key in the system.

(v) BC: in terms of the consensus protocol of blockchain, we consider this entity as a semi-honest one, which means the recording nodes will honestly obey the specific protocol and run chaincode, but they may also be curious about the underlying data. Besides, they are restricted not to collude with KGC and AA at the same time.

4.3. Definition. According to the description above, a formal definition of our proposed LU-MAABE-OD scheme consists of ten probabilistic polynomial time (PPT) algorithms as follows.

(1) $\mathbf{Setup}_{KGC}(1^{\lambda}) \longrightarrow (pp_{KGC}, \mathrm{msk}_{\mathrm{KGC}})$: with a security parameter λ, this algorithm (run by KGC) produces KGC public parameter pp_{KGC} and KGC master secret key msk_{KGC}.

(2) $\mathbf{Setup}_{AA}(pp_{KGC}) \longrightarrow (pp_{AA}, msk_{AA}, pp)$: with KGC public parameter pp_{KGC}, this algorithm (run by AA) produces AA public parameter pp_{AA} and AA master secret key msk_{AA}. And it finally outputs system public parameter which is denoted by the union $pp = pp_{KGC} \cup pp_{AA}$.

(3) $\mathbf{Setup}_{u}(pp) \longrightarrow (pk_u, sk_u)$: with public parameter pp, the output of this algorithm (run by user) is the public key pk_u and master secret key sk_u of user.

(4) $\mathbf{KeyGen}_{KGC}(pp, msk_{KGC}, pk_u, \mathcal{S}, ID) \longrightarrow (sk^{KGC}_{\mathcal{S},ID})$: with public parameter pp, KGC master key msk_{KGC}, user public key pk_u together with his/her attribute set $\mathcal{S}$ and id label ID, this algorithm (run by KGC) produces the partial decryption key $sk^{KGC}_{\mathcal{S},ID}$.

(5) $\mathbf{KeyGen}_{AA}(pp, msk_{AA}, sk^{KGC}_{\mathcal{S},ID}, \mathcal{S}, ID) \longrightarrow (sk^{AA}_{\mathcal{S},ID})$: with public parameter pp, AA master key msk_{AA}, partial decryption key $sk^{KGC}_{\mathcal{S},ID}$, the user's attribute set $\mathcal{S}$ and id label ID, this algorithm (run by AA) produces the final decryption key $sk^{AA}_{\mathcal{S},ID}$.

(6) $\mathbf{KeyGen}_{u}(sk^{AA}_{\mathcal{S},ID}) \longrightarrow (tk_{\mathcal{S},ID})$: with final decryption key $sk^{AA}_{\mathcal{S},ID}$, the output of this algorithm (run by user) is the user transformation key $tk_{\mathcal{S},ID}$.

(7) $\mathbf{Encrypt}(pp, \mathbb{A}) \longrightarrow (K_{DU}, ct_{\mathbb{A}})$: with public parameter pp and the LSSS access structure $\mathbb{A}$ chosen by DO, this algorithm (run by user) produces the encapsulated key K_{DU} which is further used for generation of DEM ciphertext and outputs KEM ciphertext $ct_{\mathbb{A}}$.

(8) $\mathbf{KeySanityCheck}(pp, tk_{\mathcal{S},ID}) \longrightarrow 1/0$: with public parameter pp and user transformation key $tk_{\mathcal{S},ID}$, this algorithm (run by BC) checks whether the key is well-formed and used by its owner. If it passes, this algorithm will output 1. Otherwise, it outputs 0.

(9) $\mathbf{Decrypt}_{BC}(pp, ct_{\mathbb{A}}, tk_{\mathcal{S},ID}) \longrightarrow pct/\perp$: with public parameter pp, user transformation key $tk_{\mathcal{S},ID}$, and KEM ciphertext $ct_{\mathbb{A}}$, this algorithm (run by chaincode invoked by DU) produces the transformed ciphertext pct if the $tk_{\mathcal{S},ID}$ passes **KeySanityCheck** and access structure $\mathbb{A}$ embedded in it satisfies the given access policy. Otherwise, it outputs $\perp$.

(10) $\mathbf{Decrypt}_{DU}(pct, sk_u) \longrightarrow key$: with the transformed ciphertext pct together with user master secret key sk_u, this algorithm (run by user) finally produces the encapsulated key key for decrypting DEM ciphertext.

4.4. Security Model. Let $\Sigma = (Setup_{KGC}, \mathbf{Setup}_{AA}, \mathbf{Setup}_{u}, \mathbf{KeyGen}_{KGC}, \mathbf{KeyGen}_{AA}, \mathbf{KeyGen}_{u}, \mathbf{Encrypt}, \mathbf{KeySanityCheck}, \mathbf{Decrypt}_{BC}, \mathbf{Decrypt}_{DU})$ denote our LU-MAABE-OD scheme. To concretely define the security notion of Σ, specific security requirements should be satisfied which can be described as the following security games.

Definition 3 (RCCA-secure ABE with outsourcing [13]). A CP-ABE scheme with outsourcing is RCCA-secure if any PPT adversary $\mathcal{A}$ can win the RCCA game defined as follows with at most negligible advantage.

(i) *CPA security*: it is defined that a scheme is CPA-secure, namely, secure against chosen-plaintext attacks if $\mathcal{A}$ cannot make decryption queries.

(ii) *Selective security*: it is defined that a scheme can achieve selective security if an **Init** stage is added before **Setup** where $\mathcal{A}$ commits to the challenge $\mathbb{A}^*$.

RCCA-secure game for malicious DU which is similar to scheme [10]: in our LU-MAABE-OD scheme, the security model of RCCA security game can be denoted by $Game^{RCCA}_{\Sigma,\mathcal{A}}$, in which the adversary $\mathcal{A}$ interacts with the challenger $\mathcal{C}$.

(1) **Setup**: $\mathcal{C}$ executes algorithm $\mathbf{Setup}_{KGC}$ and $\mathbf{Setup}_{AA}$, then pp is forwarded to $\mathcal{A}$.

(2) **QueryPhase-1**: $\mathcal{C}$ initializes an integer counter $j = 0$, an empty table T, and a set D. $\mathcal{C}$ answers the following queries from $\mathcal{A}$:

(i) **Create(S)**: $\mathscr{C}$ sets $j := j + 1$ and then executes $\mathbf{Setup}_u$ to achieve a user master secret key sk_u and public key pk_u. It then runs $\mathbf{KeyGen}_{KGC}$ on attribute-identity pairs $(\mathcal{S}, ID)$ to fetch the intermediate decryption key $sk_{\mathcal{S},ID}^{KGC}$. It also executes $\mathbf{KeyGen}_{AA}$ on $sk_{\mathcal{S},ID}$ to fetch the final decryption key $sk_{\mathcal{S},ID}^{AA}$ (can also be denoted by $sk_{\mathcal{S},ID}$). Afterwards, $\mathbf{KeyGen}_u$ on $sk_{\mathcal{S},ID}$ is run to fetch the user transformation key $tk_{\mathcal{S},ID}$. Eventually, the entry $(j, (\mathcal{S}, ID), sk_{\mathcal{S},ID}, tk_{\mathcal{S},ID}, pk_u, sk_u)$ is stored in T.

(ii) **Corrupt.SK(i)**: $\mathscr{C}$ checks whether the *i-th* entry $(i, (\mathcal{S}, ID), sk_{\mathcal{S},ID}, pk_u, sk_u)$ is in T. If such an entry exists, $D := D \cup \{(\mathcal{S}, ID)\}$ and $(sk_{\mathcal{S},ID}, pk_u, sk_u)$ is returned to $\mathscr{A}$, or $\perp$ is output.

(iii) **Corrupt.TK(i)**: $\mathscr{C}$ checks whether the *i-th* entry $(i, (\mathcal{S}, ID), tk_{\mathcal{S},ID})$ is in T. If such an entry exists, $tk_{\mathcal{S},ID}$ will be returned to $\mathscr{A}$, or $\perp$ is output.

(iv) **Decrypt(i,ct)**: $\mathscr{C}$ checks whether the *i-th* entry $(i, (\mathcal{S}, ID), sk_{\mathcal{S},ID})$ is in T. If there is such an entry, it then returns the decryption on ct, or it will output $\perp$.

(3) **Challenge**: a challenge value $\mathbb{A}^*$ is submitted by $\mathscr{A}$, which has the restriction that $\mathcal{S}$ does not satisfy $\mathbb{A}^*$. $\mathscr{C}$ then executes **Encrypt** to achieve ciphertexts $(ct_{\mathbb{A}}^*, K_{DU}^*)$, and then a random bit $b \in \{0, 1\}$ is selected. If $b = 0$, $(ct_{\mathbb{A}}^*, K_{DU}^*)$ is returned, or a random key K_R^* is chosen in the encapsulated key space and it returns $(ct_{\mathbb{A}}^*, K_R^*)$ if $b = 1$.

(4) **QueryPhase-2**: the same as **QueryPhase-1** except it should satisfy the restrictions that neither can $\mathscr{A}$ issue a **Corrupt** query nor $\mathscr{A}$ can make a **Decrypt** query on $(ct_{\mathbb{A}}^*, K_{DU}^*)$.

(5) **Guess**: $\mathscr{A}$ outputs a guess $b \in \{0, 1\}$ of b. It is defined that $\mathscr{A}$ wins the game $Game_{\Sigma,\mathscr{A}}^{RCCA}$ if $b' = b$.

Definition 4. The LU-MAABE-OD is RCCA-secure if any PPT adversary can win the above game $Game_{\Sigma,\mathscr{A}}^{RCCA}$ with at most a negligible advantage.

Key Sanity Check Game. In our LU-MAABE-OD scheme, the security model of key sanity check game is similar to scheme [11] and can be denoted by $Game_{\Sigma,\mathscr{A}}^{KSC}$, in which the adversary $\mathscr{A}$ interacts with the challenger $\mathscr{C}$.

(1) $\mathscr{C}$ invokes $\mathscr{A}$ to run algorithm $\mathbf{Setup}_{KGC}$ and $\mathbf{Setup}_{AA}$ to produce the system parameter pp, the master secret key of KGC msk_{KGC}, and AA msk_{AA}.

(2) $\mathscr{A}$ returns pp, a ciphertext ct together with two distinct transformation keys $tk_{\mathcal{S},ID}^{(1)}$ and $tk_{\mathcal{S},ID}^{(2)}$ which are associated with the same attribute-identity label $(\mathcal{S}, ID)$.

(3) It is defined that $\mathscr{A}$ wins $Game_{\Sigma,\mathscr{A}}^{KSC}$ if the following requirements are satisfied:

(i) $\mathrm{Check}(pp, tk_{\mathcal{S},ID}^{(i)}) = 1, i = 1, 2$, which indicates that both of transformation keys can pass the key sanity check.

(ii) $\mathrm{Dec}(pp, ct, tk_{\mathcal{S},ID}^{(i)}) \neq \perp, i = 1, 2$, which indicates that both of transformation keys can be used to decrypt the ciphertext.

(iii) $\mathrm{Dec}(pp, ct, tk_{\mathcal{S},ID}^{(1)}) \neq \mathrm{Dec}(pp, ct, tk_{\mathcal{S},ID}^{(2)})$, which indicates that the results of decryption by two transformation keys are different.

Definition 5. LU-MAABE-OD is a scheme that can make key sanity check if any PPT adversary $\mathscr{A}$ can win the above game $Game_{\Sigma,\mathscr{A}}^{KSC}$ with at most a negligible advantage.

5. LU-MAABE-OD Construction

5.1. Construction. Our LU-MAABE-OD scheme is based on the idea in [10, 11], and the detailed construction is described in Figure 2.

5.2. Correctness. The correctness of our LU-MAABE-OD scheme is proved as described in Figure 3.

5.3. Security Analysis

Theorem 1. *Assume that the CP-ABE scheme in [33] is selectively CPA-secure, and the proposed LU-MAABE-OD scheme is selectively CPA-secure with respect to Definition 4.*

Proof. We simply denote the LU-MAABE-OD system proposed in Section 1 by $\Sigma = (\mathbf{Setup}_{KGC}, \mathbf{Setup}_{AA}, \mathbf{Setup}_u, \mathbf{KeyGen}_{KGC}, \mathbf{KeyGen}_{AA}, \mathbf{KeyGen}_u, \mathbf{Encrypt}, \mathbf{KeySanityCheck}, \mathbf{Decrypt}_{BC}, \mathbf{Decrypt}_{DU})$ and the CP-ABE system in [33] by $\Sigma' = (\mathbf{Setup}', \mathbf{KeyGen}', \mathbf{Encrypt}', \mathbf{Decrypt}')$. To prove the security of Σ, we reduce the selective security of it to Σ', which indicates that if there is an adversary $\mathscr{A}$ who can selectively break Σ with a non-negligible advantage ϵ, it can also be used to break Σ'. We build a PPT adversary $\mathscr{A}$ with a challenge access structure (M^*, ρ^*) that can selectively break Σ with a non-negligible advantage ϵ and a PPT simulator algorithm $\mathscr{B}$ that can selectively break Σ' with a non-negligible advantage ϵ. In the system, challenger of Σ' is denoted by $\mathscr{C}$.

(1) **Init**: $\mathscr{B}$ receives a challenge access structure (M^*, ρ^*) selected by $\mathscr{A}$ and $\mathscr{B}$ sends it to $\mathscr{C}$.

(2) **Setup**: after obtaining (M^*, ρ^*), $\mathscr{C}$ generates public parameter $pp = (D, g, h, u, v, w, e(g, g)^{\alpha})$ in which $D = (p, \mathbb{G}, \mathbb{G}_T, e)$ and then sends it to $\mathscr{B}$. Once received, $\mathscr{B}$ chooses $\alpha', \beta \in \mathbb{Z}_p$ at random and sets $pp_{KGC} = (D, h, u, v, w, g^{\beta}, e(g, g)^{\alpha'})$, $msk_{KGC} = \{\alpha', \beta\}$, $pp_{AA} = e(g, g)^{\alpha}$, $msk_{AA} = \alpha/\alpha'$.

(3) **QueryPhase-1**: $\mathscr{B}$ initializes an integer counter $j = 0$, an empty table T, and then answers the queries from $\mathscr{A}$ as follows:

(i) **Create(S)**: $\mathscr{A}$ with an attribute-identity set $(\mathcal{S}, ID)$ issues the decryption key query. When

- **Setup**$_{KGC}(1^\lambda) \to (pp_{KGC}, msk_{KGC})$: With the input of a security parameter λ, KGC runs this algorithm to generate a bilinear mapping $D = (p, \mathbb{G}, \mathbb{G}_T, e)$ and sets the attribute universe $\mathcal{U} = \mathbb{Z}_p$. It then uniformly picks $g, h, u, v, w \in \mathbb{G}$ and $\alpha', \beta \in \mathbb{Z}_p$ at random. The master secret key of KGC $msk_{KGC} = \{\alpha', \beta\}$ is kept and the public parameter of KGC is output as $pp_{KGC} = (D, g, h, u, v, w, g^\beta, e(g,g)^{\alpha'})$.
- **Setup**$_{AA}(pp_{KGC}) \to (pp_{AA}, msk_{AA}, pp)$: With the input of KGC public parameter pp_{KGC} shared by KGC, AA randomly selects $\mu \in \mathbb{Z}_p$ as its master secret key $msk_{AA} = \mu$, and generates the corresponding public parameter $pp_{AA} = e(g,g)^{\alpha' \cdot \mu}$. It finally outputs the system public parameter $pp = (D, g, h, u, v, w, g^\beta, e(g,g)^{\alpha' \cdot \mu}, \mathcal{H})$ in which $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_p$ is a secure hash function.
- **Setup**$_u(pp) \to (pk_u, sk_u)$: With the input of public parameter pp, any user can randomly choose $d_u \in \mathbb{Z}_p$ as his/her master secret key $sk_u = d_u$ and keep it. Then, the user's public key will be published as $P_u = (pk_u = g^{d_u})$.
- **KeyGen**$_{KGC}(pp, msk_{KGC}, pk_u, \mathcal{S}, ID) \to (sk^{KGC}_{\mathcal{S},ID})$: The KGC key generation algorithm will randomly choose $r', \{r'_\tau\}_{\tau \in [k]} \in \mathbb{Z}_p$ and then compute $K'_0 = P_u^{\frac{\alpha'}{\beta + \mathcal{H}(ID)}} w^{r'}, K'_1 = g^{r'}, K'_2 = g^{r'\beta}, K'_3 = e(g, P_u)^{\alpha'}, L = \mathcal{H}(ID)$, and for every $\tau \in [k]$
$$K'_{\tau,1} = g^{r'_\tau},\ K'_{\tau,2} = (u^{A_\tau} h)^{r'_\tau} v^{-r'(\beta + \mathcal{H}(ID))}$$
And this algorithm will output secret key $sk^{KGC}_{\mathcal{S},ID} = (\mathcal{S}, L, K'_0, K'_1, K'_2, K'_3, \{K'_{\tau,1}, K'_{\tau,2}\}_{\tau \in [k]})$.
- **KeyGen**$_{AA}(pp, msk_{AA}, sk^{KGC}_{\mathcal{S},ID}, \mathcal{S}, ID) \to (sk^{AA}_{\mathcal{S},ID})$: The AA key generation algorithm computes $K_0 = (K'_0)^\mu = P_u^{\frac{\alpha'\mu}{\beta + \mathcal{H}(ID)}} w^{r'\mu}, K_1 = (K'_1)^\mu = g^{r'\mu}, K_2 = (K'_2)^\mu = g^{r'\beta\mu}, K_3 = (K'_3)^\mu = e(g, P_u)^{\alpha'\mu}, K_{\tau,1} = (K'_{\tau,1})^\mu = g^{r'_\tau \mu}, K_{\tau,2} = (K'_{\tau,2})^\mu = (u^{A_\tau} h)^{r'_\tau \mu} v^{-r'\mu(\beta + \mathcal{H}(ID))}$. And secret key is $sk^{AA}_{\mathcal{S},ID} = (\mathcal{S}, L, K_0, K_1, K_2, K_3, \{K_{\tau,1}, K_{\tau,2}\}_{\tau \in [k]})$.
- **KeyGen**$_u(sk^{AA}_{\mathcal{S},ID}) \to (tk_{\mathcal{S},ID})$: The algorithm simply returns $tk_{\mathcal{S},ID} = sk^{AA}_{\mathcal{S},ID}$ as the transformation key.
- **Encrypt**$(pp, \mathbb{A}) \to (ct_{\mathbb{A}}, K_{DU})$: With the input of public parameter pp and an LSSS access structure $\mathbb{A} = (M, \rho)$, where M is an $l \times n$ matrix and ρ is a mapping from $[l]$ to the universe $\mathcal{U}$. The encryption algorithm run by data owner first randomly selects $s \in \mathbb{Z}_p$ and $r_2, ..., r_n \in \mathbb{Z}_p$ to construct a random column vector $\vec{v} = (s, r_2, ..., r_n)$. The vector of shares can be calculated as $\vec{\lambda} = (\lambda_1, \lambda_2, ..., \lambda_l)^T = M\vec{v}$. It also randomly chooses $r_i \in \mathbb{Z}_p$ for each M_i and $\{t_\tau\}_{\tau \in [l]} \in \mathbb{Z}_p$. Then, it computes $C_0 = g^s, C_1 = g^{s\beta}$, and for every $\tau \in [l]$
$$C_{\tau,1} = w^{\lambda_\tau} v^{t_\tau},\ C_{\tau,2} = (u^{\rho(\tau)} h)^{-t_\tau},\ C_{\tau,3} = g^{t_\tau}$$
This encryption algorithm will output ciphertexts $ct_{\mathbb{A}} = ((M, \rho), C_0, C_1, \{C_{\tau,1}, C_{\tau,2}, C_{\tau,3}\}_{\tau \in [l]})$ and the encapsulated key $K_{DU} = e(g,g)^{\alpha' s\mu}$.
- **KeySanityCheck**$(pp, tk_{\mathcal{S},ID}) \to 1/0$: With the input of public parameter pp and transformation key $tk_{\mathcal{S},ID}$, this algorithm run by chaincode (invoked by DU) first verifies whether $tk_{\mathcal{S},ID}$ can pass the key sanity check, which should satisfy $K_0, K_1, K_2, K_{\tau,1}, K_{\tau,2} \in \mathbb{G}$ and $K_3 \in \mathbb{G}_T$ Then, it checks whether the following formula is satisfied
$$e(K_0, g^\beta g^L) = e(w, K_1)^L \cdot e(w, K_2) \cdot K_3$$
If it passes, this algorithm will output 1 which means the key is actually used by the key owner. Otherwise, 0 is output which means the key is used by someone else.
- **Decrypt**$_{BC}(pp, ct_{\mathbb{A}}, tk_{\mathcal{S},ID}) \to pct/\perp$: On input $ct_{\mathbb{A}}$ and $tk_{\mathcal{S},ID}$, the algorithm executed by chaincode (invoked by DU) first checks whether the key is used by its owner by algorithm **KeySanityCheck**. If not, it outputs $\perp$. Otherwise, it checks whether $\mathcal{S}$ satisfies the access structure $\mathbb{A}$. If not, it also outputs $\perp$. Otherwise, let $I = \{i : \rho(i) \in \mathcal{S}\}$, there must be constants $\{\omega_i \in \mathbb{Z}_p\}_{i \in l}$ such that $\sum_{i \in I} \omega_i M_i = (1, 0, ..., 0)$. It then calculates
$$pct = \frac{e(C_0^L C_1, K_0)}{\prod_{i \in l} (e(C_{\tau,1}, K_1^L K_2) e(C_{\tau,2}, K_{\tau,1}) e(C_{\tau,3}, K_{\tau,2}))^{\omega_i}}.$$
And it finally outputs the transformed ciphertext pct.
- **Decrypt**$_{DU}(pct, sk_u) \to key$: With the input of the transformed ciphertext pct and the master secret key of user $sk_u = d_u$, the algorithm executed by data user just easily calculates
$$key = (ct')^{1/d_u} = (e(g, P_u)^{\alpha' s\mu})^{1/d_u} = e(g,g)^{\alpha' s\mu}.$$
It outputs key which is actually the needed encapsulated key K_{DU}.

Figure 2: Construction of our LU-MAABE-OD scheme.

receiving, $\mathcal{B}$ forwards $(\mathcal{S}, ID)$ to $\mathcal{C}$ and obtains a secret key $sk'_{\mathcal{S},ID} = (K'_0 = g^\alpha w^r, K'_1 = g^r, \{K'_{\tau,2} = g^{r_\tau}, K'_{\tau,3} = (u^{A_\tau} h)^{r_\tau} v^{-r}\}_{\tau \in \mathcal{S}})$. Next, $\mathcal{B}$ randomly chooses $d'_u \in \mathbb{Z}_p$ and computes $P_u = g^{d'_u}$. To formalize, the user public key is set as $pk_u = P_u$ while user master key is $sk_u = d'_u$. Then, $\mathcal{B}$ computes $K_0 = (K'_0)^{d'_u/\beta + \mathcal{H}(ID)}$, $K_1 = (K'_1)^{d'_u/\beta + \mathcal{H}(ID)}$, $K_2 = (K_1)^\beta$, $K_3 = (e(g,g)^\alpha)^{d'_u}$, $\{K_{\tau,1} = (K_{\tau,2}')^{d'_u}, K_{\tau,2} = (K_{\tau,3}')^{d'_u}\}_{\tau \in \mathcal{S}}$. Let $r^* = r \cdot d'_u/\beta + \mathcal{H}(ID)$, $r^*_\tau = r_\tau \cdot d'_u$, and then r^*, r^*_τ are random numbers due to the randomness of r, r_τ,

- Correctness of **KeySanityCheck**:

$$e(K_0, g^{\beta} g^{L}) = e(P_u^{\frac{\alpha'\mu}{\beta+\mathcal{H}(ID)}} w^{r'\mu}, g^{\beta+\mathcal{H}(ID)}) = e(P_u, g)^{\alpha'\mu} e(w, g)^{r'\mu(\beta+\mathcal{H}(ID))} \quad (1)$$

$$e(w, K_1)^{L} \cdot e(w, K_2) \cdot K_3 = e(w, g^{r'\mu\mathcal{H}(ID)}) e(w, g^{r'\mu\beta}) e(P_u, g)^{\alpha'\mu} = e(w, g)^{r'\mu(\beta+\mathcal{H}(ID))} e(P_u, g)^{\alpha'\mu} \quad (2)$$

It is obviously that $e(K_0, g^{\beta} g^{L}) = e(w, K_1)^{L} \cdot e(w, K_2) \cdot K_3$.

- Correctness of **Decrypt**$_{BC}$:

$$e(C_0^{L} C_1, K_0) = e(g^{s\cdot\mathcal{H}(ID)} g^{s\beta}, (P_u^{\frac{\alpha'}{\beta+\mathcal{H}(ID)}} w^{r'})^{\mu}) = e(g, P_u)^{\alpha' s\mu} e(g, w)^{r' s\mu(\beta+\mathcal{H}(ID))} \quad (3)$$

$$e(C_{\tau,1}, K_1^{L} K_2) = e(w^{\lambda_\tau} v^{t_\tau}, g^{r'\mu\mathcal{H}(ID)} g^{r'\mu\beta}) = e(w, g)^{\lambda_\tau r'\mu(\beta+\mathcal{H}(ID))} e(v, g)^{t_\tau r'\mu(\beta+\mathcal{H}(ID))} \quad (4)$$

$$e(C_{\tau,2}, K_{\tau,1}) = e((u^{\rho(\tau)} h)^{-t_\tau}, g^{r'_\tau\mu}) = e(u^{\rho(\tau)} h, g)^{-t_\tau r'_\tau\mu} \quad (5)$$

$$e(C_{\tau,3}, K_{\tau,2}) = e(g^{t_\tau}, (u^{A_\tau} h)^{r'_\tau\mu} v^{-r'\mu(\beta+\mathcal{H}(ID))}) = e(g, u^{A_\tau} h)^{t_\tau r'_\tau\mu} e(g, v)^{-t_\tau r'\mu(\beta+\mathcal{H}(ID))} \quad (6)$$

Finally, it calculates $ct' = \frac{e(C_0^{L} C_1, K_0)}{\prod_{i\in I} (e(C_{\tau,1}, K_1^{L} K_2) e(C_{\tau,2}, K_{\tau,1}) e(C_{\tau,3}, K_{\tau,2}))^{\omega_i}} = e(g, P_u)^{\alpha' s\mu}$.

Figure 3: Correctness of our LU-MAABE-OD scheme.

respectively. Therefore, $sk_{\mathcal{S},ID} = (K_0 = P_u^{\alpha/\beta+\mathcal{H}(ID)} w^{r^*}, K_1 = g^{r^*}, K_2 = g^{r^*\beta}, K_3 = e(g, P_u)^{\alpha}, \{K_{\tau,1} = g^{r^*_\tau}, K_{\tau,2} = (u^{A_\tau} h)^{r^*_\tau} v^{-r^*(\beta+\mathcal{H}(ID))}\}_{\tau\in\mathcal{S}^*})$ inherits the randomness. It sets the transformation key $tk_{\mathcal{S},ID} = sk_{\mathcal{S},ID}$ and the entry $(j, (\mathcal{S}, ID), sk_{\mathcal{S},ID}, tk_{\mathcal{S},ID}, pk_u, sk_u)$ is stored in T.

(ii) **Corrupt.SK(i)**: $\mathcal{B}$ checks whether the *i-th* entry $(i, (\mathcal{S}, ID), sk_{\mathcal{S},ID}, pk_u, sk_u)$ is in table T. It then returns $sk_{\mathcal{S},ID}, pk_u, sk_u)$ to $\mathcal{A}$, or $\perp$ if there is no such entry.

(iii) **Corrupt.TK(i)**: $\mathcal{B}$ checks whether the *i-th* entry $(i, (\mathcal{S}, ID), tk_{\mathcal{S},ID})$ is in table T. It then returns $tk_{\mathcal{S},ID}$ to $\mathcal{A}$, or $\perp$ if there is no such entry.

(4) **Challenge**: $\mathcal{A}$ declares two messages m_0, m_1 of the same length which are forwarded to $\mathcal{B}$. Then, they are sent to $\mathcal{C}$ by $\mathcal{B}$ and achieve a challenge ciphertext $ct^* = ((M^*, \rho^*), C^*, C_0^*, \{C_{\tau,1}^*, C_{\tau,2}^*, C_{\tau,3}^*\}_{\tau\in[l]})$. Then, $\mathcal{B}$ selects a random bit $b_{\mathcal{B}} \in \{0, 1\}$ to calculate $key_{b_{\mathcal{B}}} = C^*/m_{b_{\mathcal{B}}}$ and returns to $\mathcal{A}$ the new challenge ciphertext $ct = ((M, \rho), key_{b_{\mathcal{B}}}, C_0 = C_0^*, C_1 = (C_0^*)^{\beta}, \{C_{\tau,1} = C_{\tau,1}^*, C_{\tau,2} = C_{\tau,2}^*, C_{\tau,3} = C_{\tau,3}^*\}_{\tau\in[l]})$.

(5) **QueryPhase-2**: the same as **QueryPhase-1.**

(6) **Guess**: $\mathcal{A}$ outputs a guess bit $b_{\mathcal{A}} \in \{0, 1\}$. If $b_{\mathcal{A}} = 1$, it denotes that $key_{b_{\mathcal{B}}}$ is a random key and $\mathcal{B}$ outputs $1 - b_{\mathcal{B}}$. Otherwise, it means that $key_{b_{\mathcal{B}}}$ is guessed as the key encapsulated by ct and $\mathcal{B}$ outputs $b_{\mathcal{B}}$.

Obviously, if $\mathcal{A}$ can selectively break our scheme Σ with a non-negligible advantage ϵ, $\mathcal{B}$ can also selectively break scheme Σ' with the same advantage. □

Theorem 2. *In our LU-MAABE-OD scheme, the advantage of any PPT adversary winning in the key sanity check game* $Game_{\Sigma,\mathcal{A}}^{KSC}$ *is negligible.*

Proof. It is similar to the proof in [11]. We build a PPT adversary $\mathcal{A}$ in our LU-MAABE-OD scheme, which can be denoted by Σ. It produces the public parameter pp, a ciphertext ct, and two different transformation keys $tk_{\mathcal{S},ID}^{(i)} = (\mathcal{S}, L, K_1^{(i)}, K_2^{(i)}, K_3^{(i)}, K_4^{(i)}, \{K_{\tau,1}^{(i)}, K_{\tau,2}^{(i)}\}_{\tau\in\mathcal{S}})$, $i = 1, 2$. $\mathcal{A}$ wins the game $Game_{\Sigma,\mathcal{A}}^{KSC}$ only if the following three requirements are all satisfied.

(i) $\text{Check}(pp, tk_{\mathcal{S},ID}^{(i)}) = 1, i = 1, 2$.

(ii) $\text{Dec}(pp, ct, tk_{\mathcal{S},ID}^{(i)}) \neq \perp, i = 1, 2$.

(iii) $\text{Dec}(pp, ct, tk_{\mathcal{S},ID}^{(1)}) \neq \text{Dec}(pp, ct, tk_{\mathcal{S},ID}^{(2)})$.

Based on the first condition, we have for $i = 1, 2$, it satisfies.

(i) $K_0, K_1, K_2, K_{\tau,1}, K_{\tau,2} \in \mathbb{G}, K_3 \in \mathbb{G}_T$.

(ii) $e(K_0, g^{\beta} g^{L}) = e(w, K_1)^{L} \cdot e(w, K_2) \cdot K_3$.

Based on the second condition, we have for $i = 1, 2$, there is

$$B^{(i)} = \frac{e\left(C_0^{L} C_1, K_0^{(i)}\right)}{\prod_{j\in I} \left(e\left(C_{j,1}, K_1^{(i)L} K_2^{(i)}\right) e\left(C_{j,2}, K_{\rho(j),1}^{(i)}\right) e\left(C_{j,3}, K_{\rho(j),2}^{(i)}\right)\right)^{\omega_j}}. \quad (3)$$

Therefore, we will finally achieve $B^{(1)} = B^{(2)} = e(g, g)^{\alpha' s\mu}$, from which we can calculate that.

(i) $\text{Dec}(pp, ct, tk_{\mathcal{S},ID}^{(1)}) = C/B^{(1)}$.

(ii) $\text{Dec}(pp, ct, tk_{\mathcal{S},ID}^{(2)}) = C/B^{(2)}$.

It is obvious that the respective decryption result is the same which contradicts with the last condition. Thus, the advantage that $\mathcal{A}$ can win the game $Game_{\Sigma,\mathcal{A}}^{KSC}$ is negligible. □

Theorem 3. *In our LU-MAABE-OD scheme, semi-honest KGC or AA cannot recover the underlying encrypted data and secret key of DU since they are not allowed to collude with each other.*

Proof. Suppose $\mathcal{A}$ is a PPT adversary who is curious about obtaining the encapsulated key $K_{DU} = e(g, g)^{\alpha' s\mu}$ and DU's private key $sk_u = d_u$.

(i) For DU's private key, $\mathcal{A}$ can easily obtain the public parameter pp and DU's private key message $sk = \{e(g, P_u)^{\alpha'}, e(g, P_u)^{\alpha'\mu}\}$. To obtain DU's secret

key d_u, either a semi-honest KGC or a semi-honest AA as $\mathscr{A}$ is required to solve discrete logarithm problems of computing $\log_{e(g,g)^{\alpha'}}\{e(g,p_u)^{\alpha'}\}$ or $\log_g(p_u)$, which are considered to be impossible for any PPT adversary. Therefore, any semi-honest KGC or AA cannot illegally obtain DU's secret key.

(ii) For the encapsulated key, $\mathscr{A}$ can easily obtain the public parameter pp and ciphertext message $ct' = g^s$ related to calculating K_{DU}. If $\mathscr{A}$ is a semi-honest KGC (who cannot collude with AA), with the master secret key α', $\mathscr{A}$ can easily calculate $e(g^{\alpha'}, g^s) = e(g,g)^{\alpha' s}$. To obtain$K_{DU}$,at least one of DU's secret key d_u(which has been proved difficult for PPT adversary$\mathscr{A}$ to achieve), AA's secret key μ and random secret information s should be obtained; otherwise, KGC can never decrypt K_{DU} due to the discrete logarithm problem of computing $\log_g(g^s)$ or $\log_{e(g,g)^{\alpha'}} e(g,g)^{\alpha'\mu}$. If $\mathscr{A}$ is a semi-honest AA (who cannot collude with KGC), similarly, it is required to obtain α' or s or d_u; otherwise, $\mathscr{A}$ is required to handle the discrete logarithm problem of computing $\log_g(g^s)$ or $\log_{e(g,g)^{\mu}} e(g,g)^{\alpha'\mu}$. In a word, any semi-honest KGC or AA cannot illegally obtain encapsulated key. □

Lemma 1. *In our LU-MAABE-OD system, any PPT adversary cannot abuse public or leaked transformation key which is not owned by him/her.*

Proof. Suppose there exists a PPT adversary $\mathscr{A}$ (with his/her identity ID') who wants to abuse public (since the transformation key will be publicly recorded after it is used through BC) or leaked transformation key which is not owned by him/her. It is noted that due to our sophisticated design, chaincode invoked by $\mathscr{A}$ will make a check before decryption. If $\mathscr{H}(ID)$ in the transformation key does not satisfy $\mathscr{H}(ID) = \mathscr{H}(ID')$, chaincode will reject to provide decryption services and produce $\perp$. Otherwise, it will make a key sanity check to validate whether the transformation key is well-formed. If it passes, $\mathscr{A}$ can make a decryption through chaincode, or it will be rejected. According to Theorem 2, $\mathscr{A}$ can only win the game $Game^{KSC}_{\Sigma,\mathscr{A}}$ with a negligible advantage, and thus $\mathscr{A}$ cannot abuse public or leaked transformation key which is not owned by him/her. □

6. System Analysis

In this section, we will evaluate the practical performance and the challenges brought by blockchain system in our scheme.

6.1. Performance Evaluations. We evaluated the performance of our scheme and made a comparison with three other existing schemes [10–12] in terms of overheads of the following stages: Setup, KeyGen, Encrypt, and Decrypt. It should be noted that we evaluated all the costs in the former three stages while only the user decryption time was evaluated in the last stage.

We implemented our LU-MAABE-OD scheme in software based on Charm [40] (Version 0.50). To make a comparison, we also implemented the scheme in [10–12] and MNT224 elliptic curve is applied. All the experiments were executed on a computer equipped with 2.4 GHz Intel Core i5 with 8 GB of RAM running MacOS Mojave 10.14.6 and Python3. By increasing number of policy attributes from 1 to 100, we compared the total running time in stage of Setup, KeyGen, Encrypt, and Decrypt. For accuracy, we repeated 50 times for each scheme and the average was taken as the results in our figures and the time is all given in seconds.

As shown in Figure 4, we examine the time cost of Setup, from which we can find the overhead in [12] is linear to the attribute universe while that of our scheme and [10, 11] is constant. As the introduction of two authorities, costs in our scheme and in [11] are 0.02 s higher than those in [10] on average. Figure 5 shows the cost of KeyGen, from which we can follow that our scheme costs much more than [10, 12] for the reason of key blindness by two authorities, which will unavoidably increase considerable overheads. However, compared with another multi-authority scheme in [11], our scheme improves the efficiency of nearly 2.8 s on average when the user attribute size is 100. From Figure 6 which tests the cost of Encrypt, it is shown that scheme in [10, 11] and our scheme achieve nearly the same encryption efficiency while scheme in [12] is around 2.6 s faster on average when the policy attribute size is 100 owing to the low-paid small-universe construction of access tree. Figure 7 displays the Decrypt cost in user side, which only costs a constant time of 0.004 s on average in our scheme because most of cost is taken by blockchain. Despite the use of outsourced decryption, scheme in [10] can also achieve constant decryption cost, but it is about 0.013 s lower than that of our scheme as the verification of decryption result from cloud is needed while it is unnecessary in our scheme. It is noted that although blockchain-assisted decryption is adopted in scheme [21], heavy computation is still required for users which is linear to the attribute size and in scheme [11], the whole decryption task is taken by users so that huge overheads will be put on the user side. As a result, our scheme achieves higher security compared with [10, 12] while efficiency on KeyGen and Encrypt has been sacrificed; however, the high performance of Decrypt is realized so that our scheme is made especially applicable for resource-constrained data users. Besides, with the same high security, our scheme performs better than scheme in [11] in terms of overall overheads.

6.2. Blockchain Analysis. As mentioned above, our system implementation is based on blockchain, and the IPFS data storage and partial decryption by smart contracts/chaincodes are the keys to solve on-chain storage burden and user computing overheads in traditional schemes. Therefore, we need to consider some problems that may be brought by the introduction of blockchain platform. In terms of public

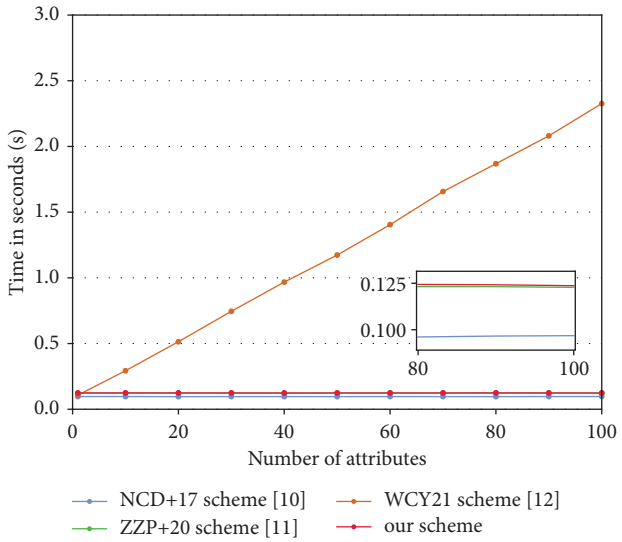

Figure 4: The Setup cost in NCD + 17, ZZP + 20, WCY21, and our scheme.

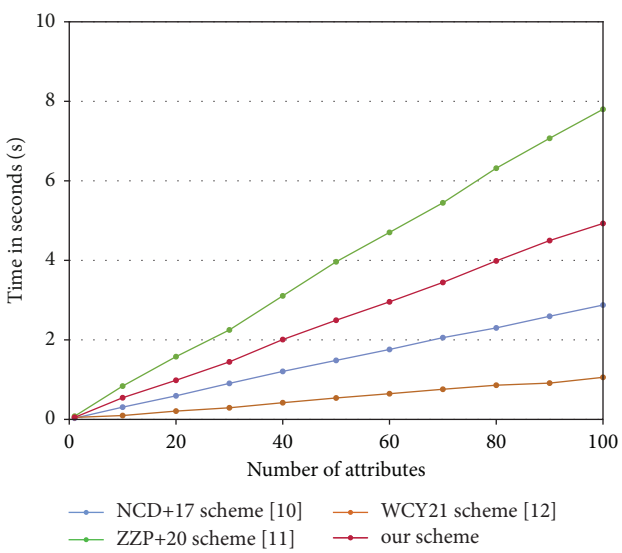

Figure 5: The KeyGen cost in NCD + 17, ZZP + 20, WCY21, and our scheme.

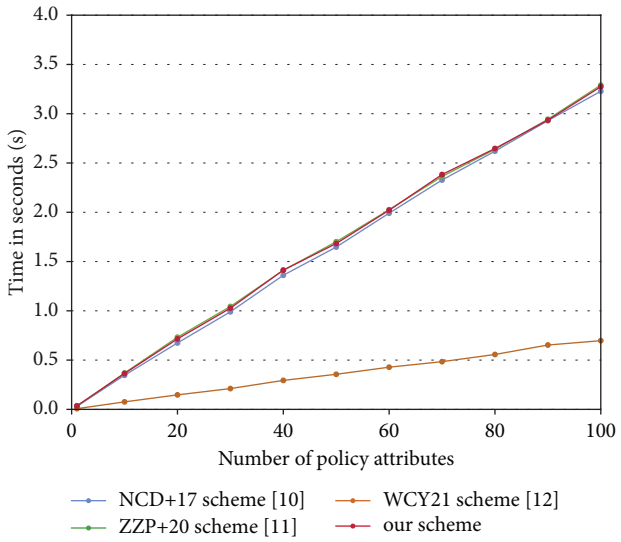

Figure 6: The Encrypt cost in NCD + 17, ZZP + 20, WCY21, and our scheme.

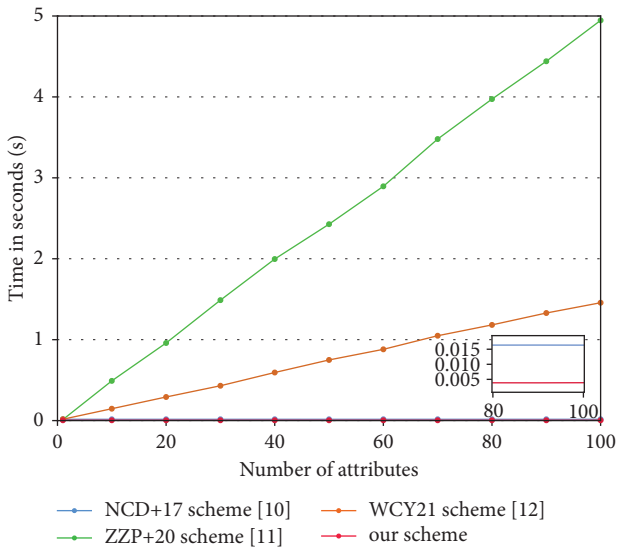

Figure 7: The Decrypt cost in NCD + 17, ZZP + 20, WCY21, and our scheme.

blockchain like Ethereum, resource-limited DU can ask blockchain to make outsourced decryption and the execution cost of smart contracts needs to be paid by DU for the help of decryption. However, due to the technique problem that smart contract cannot well support pairing operations in ABE currently, it is not recommended to be deployed in Ethereum until the related language library (i.e., Solidity) is introduced. Besides, regular problems like network congestion remains to be further discussed. In terms of permissioned blockchain like Hyperledger, it can be deployed in scenarios like healthcare and IoT data sharing, in which the system is maintained by a number of authorized nodes (i.e., data center managers). Since the key received by chaincode is just the transformation key which can be public and the final secret key to decrypt is always kept by DU themselves, the underlying data will also never be achieved illegally by those nodes. Besides, even if these nodes can collude with one of AA and KGC, according to the analysis of Theorem 3, as long as AA and KGC do not collude with each other, the underlying data and the user's secret key will not be disclosed, and thus the collusion between the node and AA/KGC will not affect the security. Therefore, under the assumption of this article mentioned in Section 2, the system is still secure.

7. Conclusion

In this paper, an efficient large-universe multi-authority CP-ABE scheme with blockchain-assisted outsourced

decryption (LU-MAABE-OD) for data sharing system is proposed. In particular, we extended the basic outsourced scheme based on [10] and introduced two authorities (namely, KGC and AA) to avoid key escrow problem which was ignored in [10] while key leakage resilience was reserved. Then, the technique of blockchain and IPFS was applied to be responsible for outsourced decryption and data storage, solving the issue of malicious decryption in traditional cloud computation scenarios and storage burden on chain, respectively. As of independent interest, our scheme was also key abuse resistant due to the key owner and key sanity check through chaincode on blockchain. Furthermore, we evaluated the performance and discussed the security of our LU-MAABE-OD scheme. As a result, our scheme achieved higher security at the expense of some efficiency. However, LU-MAABE-OD was user-friendly in terms of the most lightweight user decryption cost, which was especially needed for resource-limited data users. For the future work, we think it is promising to explore more significant functions like attribute revocation, user revocation, and policy hiding and devote to promoting overall efficiency in such blockchain-based access control and data sharing system.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This study was supported in part by the National Natural Science Foundation of China (62002120), Shanghai Rising-Star Program (no. 22QA1403800), Innovation Program of Shanghai Municipal Education Commission (2021-01-07-00-08-E00101), NSFC-ISF Joint Scientific Research Program (61961146004), and the "Digital Silk Road" Shanghai International Joint Lab of Trustworthy Intelligent Software (Grant No. 22510750100).

References

[1] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 457–473, Springer, Berlin, Heidelberg, 2005.

[2] 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 Virginia USA, October 2006.

[3] A. Lewko and B. Waters, "Unbounded hibe and attribute-based encryption," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 547–567, Springer, Berlin, Heidelberg, 2011.

[4] Z. Liu, Z. Cao, and D. S. Wong, "White-box traceable ciphertext-policy attribute-based encryption supporting any monotone access structures," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 1, pp. 76–88, 2013.

[5] J. Ning, X. Dong, Z. Cao, L. Wei, and X. Lin, "White-box traceable ciphertext-policy attribute-based encryption supporting flexible attributes," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 6, pp. 1274–1288, 2015.

[6] G. Yu, Y. Wang, Z. Cao, J. Lin, and X. Wang, "Traceable and undeniable ciphertext-policy attribute-based encryption for cloud storage service," *International Journal of Distributed Sensor Networks*, vol. 15, no. 4, 2019.

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

[8] A. Lewko and B. Waters, "Decentralizing attribute-based encryption," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 568–588, Springer, Berlin, Heidelberg, 2011.

[9] J. Li, X. Chen, S. S. Chow, Q. Huang, D. S. Wong, and Z. Liu, "Multi-authority fine-grained access control with accountability and its application in cloud," *Journal of Network and Computer Applications*, vol. 112, pp. 89–96, 2018.

[10] J. Ning, Z. Cao, X. Dong, K. Liang, H. Ma, and L. Wei, "Auditable σ-time outsourced attribute-based encryption for access control in cloud computing," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 1, pp. 94–105, 2017.

[11] Z. Zhang, P. Zeng, B. Pan, and K.-K. R. Choo, "Large-universe attribute-based encryption with public traceability for cloud storage," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 10314–10323, 2020.

[12] Y. Wang, C. Cao, and L. You, "A novel personal privacy data protection scheme based on blockchain and attribute-based encryption," *Journal of Cryptologic Research*, vol. 8, no. 1, pp. 14–27, 2021.

[13] M. Green, S. Hohenberger, and B. Waters, "Outsourcing the decryption of abe ciphertexts," in *Proceedings of the USENIX Security Symposium*, San Francisco, CA, August 2011.

[14] J. Lai, R. H. Deng, C. Guan, and J. Weng, "Attribute-based encryption with verifiable outsourced decryption," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 8, pp. 1343–1354, 2013.

[15] J. Li, X. Huang, J. Li, X. Chen, and Y. Xiang, "Securely outsourcing attribute-based encryption with checkability," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 8, pp. 2201–2210, 2014.

[16] B. Qin, R. H. Deng, S. Liu, and S. Ma, "Attribute-based encryption with efficient verifiable outsourced decryption," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 7, pp. 1384–1393, 2015.

[17] S. Lin, R. Zhang, H. Ma, and M. Wang, "Revisiting attribute-based encryption with verifiable outsourced decryption," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 10, pp. 2119–2130, 2015.

[18] X. Mao, J. Lai, Q. Mei, K. Chen, and J. Weng, "Generic and efficient constructions of attribute-based encryption with verifiable outsourced decryption," *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 5, pp. 533–546, 2016.

[19] H. Ma, R. Zhang, Z. Wan, Y. Lu, and S. Lin, "Verifiable and exculpable outsourced attribute-based encryption for access control in cloud computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 14, no. 6, pp. 679–692, 2017.

[20] H. Guo, W. Li, M. Nejad, and C.-C. Shen, "Access control for electronic health records with hybrid blockchain-edge architecture," in *Proceedings of the 2019 IEEE International Conference on Blockchain (Blockchain)*, pp. 44–51, IEEE, Atlanta, GA, USA, July 2019.

[21] S. Wang, D. Zhang, and Y. Zhang, "Blockchain-based personal health records sharing scheme with data integrity verifiable," *IEEE Access*, vol. 7, pp. 102887–102901, 2019.

[22] X. Yang, T. Li, X. Pei, L. Wen, and C. Wang, "Medical data sharing scheme based on attribute cryptosystem and blockchain technology," *IEEE Access*, vol. 8, pp. 45468–45476, 2020.

[23] J. Tao and L. Ling, "Practical medical files sharing scheme based on blockchain and decentralized attribute-based encryption," *IEEE Access*, vol. 9, pp. 118771–118781, 2021.

[24] Q. He, Y. Xu, Z. Liu, J. He, Y. Sun, and R. Zhang, "A privacy-preserving internet of things device management scheme based on blockchain," *International Journal of Distributed Sensor Networks*, vol. 14, no. 11, 2018.

[25] Y. Zhang, D. He, and K.-K. R. Choo, "Bads: blockchain-based architecture for data sharing with abs and cp-abe in iot," *Wireless Communications and Mobile Computing*, vol. 2018, Article ID 2783658, 9 pages, 2018.

[26] J. Zhang, Y. Xin, Y. Gao, X. Lei, and Y. Yang, "Secure abe scheme for access management in blockchain-based iot," *IEEE Access*, vol. 9, pp. 54840–54849, 2021.

[27] J. Benet, "Ipfs-content Addressed, Versioned, P2p File System," arXiv preprint, 2014.

[28] Y. Hassanzadeh-Nazarabadi, A. Küpçü, and Ö. Özkasap, "Lightchain: A Dht-Based Blockchain for Resource Constrained Environments," arXiv preprint, 2019.

[29] S. Wang, Y. Zhang, and Y. Zhang, "A blockchain-based framework for data sharing with fine-grained access control in decentralized storage systems," *IEEE Access*, vol. 6, pp. 38437–38450, 2018.

[30] V.-D. Pham, C.-T. Tran, T. Nguyen et al., "B-box-a decentralized storage system using ipfs, attributed-based encryption, and blockchain," in *Proceedings of the 2020 RIVF International Conference on Computing and Communication Technologies (RIVF)*, pp. 1–6, IEEE, Ho Chi Minh City, Vietnam, October 2020.

[31] B. Waters, "Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization," in *International Workshop on Public Key Cryptography*, pp. 53–70, Springer, Berlin, Heidelberg, 2011.

[32] A. Lewko and B. Waters, "New proof methods for attribute-based encryption: achieving full security through selective techniques," in *Annual Cryptology Conference*, pp. 180–198, Springer, Berlin, Heidelberg, 2012.

[33] Y. Rouselakis and B. Waters, "Practical constructions and new proof methods for large universe attribute-based encryption," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, pp. 463–474, Berlin, Germany, November 2013.

[34] A. Beimel, *Secure Schemes for Secret Sharing and Key Distribution*, Phd Thesis, israel institute of Technology Technion, Haifa, Israel, 1996.

[35] S. Nakamoto, "Bitcoin: a peer-to-peer electronic cash system," *Decentralized Business Review*, 2008, https://bitcoin.org/bitcoin.pdf., Article ID 21260.

[36] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 15132 pages, 2014.

[37] E. Androulaki, A. Barger, V. Bortnikov et al., "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proceedings of the Thirteenth EuroSys Conference*, pp. 1–15, Porto, Portuga, April 2018.

[38] N. Szabo, "Formalizing and Securing Relationships on Public Networks," *First monday*, vol. 2, no. 9, 1997.

[39] V. Shoup, "A proposal for an iso standard for public key encryption (version 2.1)," *IACR e-Print Archive*, vol. 112, 2001.

[40] J. A. Akinyele, C. Garman, I. Miers et al., "Charm: a framework for rapidly prototyping cryptosystems," *Journal of Cryptographic Engineering*, vol. 3, no. 2, pp. 111–128, 2013.