Hindawi
Security and Communication Networks
Volume 2021, Article ID 6689448, 13 pages
https://doi.org/10.1155/2021/6689448

Research Article

Blockchain-Based Fine-Grained Data Sharing for Multiple Groups in Internet of Things

Teng Li, Jiawei Zhang, Yangxu Lin, Shengkai Zhang, and Jianfeng Ma

School of Cyber Engineering, Xidian University, Xi'an, China

Correspondence should be addressed to Jiawei Zhang; zjw8512@126.com

Received 20 November 2020; Revised 29 January 2021; Accepted 11 February 2021; Published 28 February 2021

Academic Editor: Kim-Kwang Raymond Choo

Cloud-based Internet of Things, which is considered as a promising paradigm these days, can provide various applications for our society. However, as massive sensitive and private data in IoT devices are collected and outsourced to cloud for data storage, processing, or sharing for cost saving, the data security has become a bottleneck for its further development. Moreover, in many large-scale IoT systems, multiple group data sharing is practical for users. Thus, how to ensure data security in multiple group data sharing remains an open problem, especially the fine-grained access control and data integrity verification with public auditing. Therefore, in this paper, we propose a blockchain-based fine-grained data sharing scheme for multiple groups in cloud-based IoT systems. In particular, we design a novel multiauthority large universe CP-ABE scheme to guarantee the fine-grained access control and data integrity across multiple groups by integrating group signature into our scheme. Moreover, to ease the need for a trusted third auditor in traditional data public auditing schemes, we introduce blockchain technique to enable a distributed data public auditing. In addition, with the group signature, our scheme also realizes anonymity and traitor tracing. The security analysis and performance evaluation show that our scheme is practical for large-scale IoT systems.

1. Introduction

The Internet of Things (IoT) brings the power of the Internet, data processing, and analytics to the real world of physical objects. The fast development of IoT has greatly facilitated a variety of applications all over the world, such as the Internet of Vehicles (IoV), Industrial Internet of Things (IIoT), and Health Internet of Things (HIoT) [1]. All explosive numbers of embedded internet-enabled sensors provide an incredibly rich set of data that device owners can use to gather data about the safety of their operations, track assets, and reduce manual processes. The device providers can also use the IoT to gather data about people's preferences and behavior, though that can have serious privacy and security implications. Therefore, how to store and share these data can be a challenging problem. With cloud computing assistance, the enormous storage and computation resources can accommodate those massive data in IoT applications. As shown in Figure 1, the ubiquitous IoT devices gather all kinds of data, for example, health records, location data, and personal privacy information, and upload these data to the cloud, which can significantly lower their local burden for data storage, processing, and sharing.

Although cloud-based IoT provides many conveniences for users, data security and privacy have become serious problems when vast amounts of sensitive and private IoT data are stored and shared in untrusted cloud servers. There have been many catastrophic accidents in data sharing services of cloud, such as iCloud and AWS Cloud. The cloud service providers may unconsciously leak sensitive information or even delete these outsourced user data [2]. Therefore, many researchers have dedicated their research to data security in sharing scenarios. The works in [3–6] solve the data access control for data sharing in various scenarios and the proposals in [7–10] focus on the data integrity verification and public auditing. However, as the scale of data and services in the cloud grows, the multiple group data sharing emerges as a promising application for data sharing between users in different groups [11–13], such as multiple health study organizations and multiple businesses.

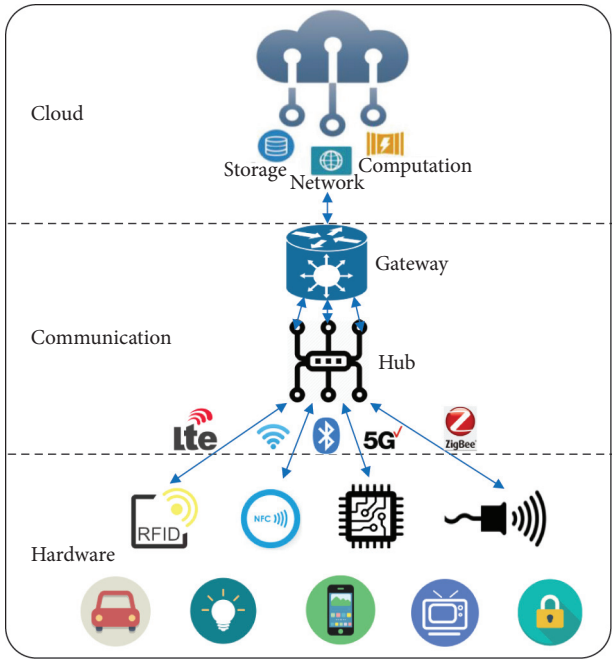

Figure 1: : An overview of cloud-based IoT.

However, all of the above schemes fail to support data security protection in multiple group data sharing scenarios, which is a big challenge. The approach in [14] proposes a multiple group data sharing scheme that provides the data auditing and integrity verification and trace mechanism with the help of blockchain. Nevertheless, the approach suffers from the lack of data confidentiality and fine-grained access control, especially the single point failure problem in large-scale IoT systems, which can be another challenge for data sharing in multiple groups.

To address these challenging issues in multiple group data sharing, we propose a comprehensive data sharing approach to guarantee data security from the aspects of data fine-grained access control, data integrity verification, and public auditing. Besides, with multiauthority attribute-based encryption and blockchain techniques, our proposal enables data integrity without a trusted third party. It makes the data access control more flexible in a large attribute universe, user traceability, and multiple authorities. The security analysis and experimental evaluation demonstrate the security and efficiency of our work. In conclusion, our contribution can be summarized as follows:

We firstly present blockchain-based fine-grained data sharing for multiple groups in cloud-based IoT system, which can achieve the data integrity verification, public auditing, and fine-grained access control simultaneously providing a comprehensive security for data security.

A multiauthority ciphertext-policy access control scheme is designed for fine-grained data sharing, which can support large attribute universe and avoid single point failure for authority in large-scale systems. Our scheme extends to multiple group data sharing with the group signature and blockchain techniques and enables user anonymity and traceability with data integrity.

We present a detailed security analysis to demonstrate that our scheme satisfies the security goals of our system. Extensive experiment with the implementation is also provided, and the evaluation results depict the efficiency of our scheme, which proves that our scheme is secure and practical.

The remainder of this paper is structured as follows. Section 2 introduces and reviews the related work on identity authentication. Then, in Section 3, a motivating example of our scheme is given together with attacker mode. Our approach's overview is presented in Section 4. Following this, in Section 5, we describe in detail the system design and, in Section 6, we show the performance evaluation for our proposal. Finally, a conclusion is presented for our work in Section 7.

2. Related Work

This section reviews some related works on attribute-based encryption (ABE) and blockchain techniques.

2.1. Attribute-Based Encryption. To guarantee data fine-grained access control, ABE was first introduced by literature [15] as a promising paradigm. Then, [16] divided this scheme into two classes: CP-ABE and KP-ABE (Key-Policy ABE), where the former requires the data owner to designate an access policy for ciphertext and the latter for user key. As the access policy is embedded into ciphertext enabling data owner to control data access on his own, CP-ABE attracts great attention and, for the design of flexible access control scheme according to data owner's requirements, we emphasize on the CP-ABE scheme in our work. Later, many studies dedicated to CP-ABE are proposed, such as large universe CP-ABE [17], traceable CP-ABE, and revocable CP-ABE [18].

Traditionally, in design of CP-ABE schemes, the system model involves just one trusted authority for attribute management and key distribution. However, this brings the failure of single point and has become the bottleneck in large-scale system. Thus, literature [19] introduced a decentralized architecture for multiple attribute authority that collaboratively manages users and attribute universe. Recently, the proposal in [20] has realized multiauthority CP-ABE with large attribute universe and white-box traceability, while the ability of user tracing is limited and just suitable for certain scenarios. The multiauthority CP-ABE scheme [21] highlights the efficiency in traceable decentralized CP-ABE scheme, while it also suffers from the same drawback with [20].

2.2. Blockchain. Blockchain can be regarded as a distributed database based on blockchain technology [22–25]. It has the characteristics of openness, tamper-resistance, decentralization, and autonomy. The data recorded on the blockchain cannot be tampered with or modified. A blockchain can be regarded as a decentralized, trusted third party. The decentralization of blockchain provides a feasible solution for

the construction of a security scheme without a trusted third party. Blockchain enables users to build a shared, distributed, and fault-tolerant database [26]. Ghoshal et al. [27] proposed the first auditing mechanism without a third-party requirement. Blockchain-based data auditing can provide tamper-proof records and enable data accountability in the cloud.

At present, blockchain network can be divided into three categories: public blockchain, private blockchain, and alliance blockchain. We present a specific data-sharing scheme for multiple groups based on the concept of consortium blockchain. The nodes in the alliance chain are well connected and the verification efficiency is high. While providing high-speed transaction processing, it can maintain the operation with the minimum cost, reduce the transaction cost, and have good scalability. The data can maintain privacy to a certain extent.

3. Preliminaries

The section presents several relevant notions and definitions employed in our paper.

3.1. Notations. We summarize several notations used in our scheme as well as their descriptions in Table 1.

3.2. Access Structure

Definition 1 (access structures [3]). Suppose that $\{L_1, \ldots, L_n\}$ is a parties set. One of the collections $L \subseteq 2^{\{L_1,\ldots,L_n\}}$ is considered to be monotone if $\forall M, N$: $M \in L$ and $M \subseteq M$, then $N \in L$. An access structure that is monotone is defined as one of the nonempty subsets L of $\{L_1, \ldots, L_n\}$, *i.e.*, $L \subseteq 2^{\{L_1,\ldots,L_n\}}/\varnothing$. The elements in L are defined as authorized sets and the other sets are defined as unauthorized sets. Without loss of generality, we can describe users with their attribute set.

3.3. Linear Secret Sharing Scheme (LSSS)

Definition 2 (LSSS [28]). A secret sharing scheme over the attribute set is called linear over $\mathbb{Z}_p$ if it satisfies the following: 1. the secret share for each attribute can form a vector over $\mathbb{Z}_p$, 2. there is a matrix M with h rows and n columns and a function for Π that maps each row M_j to an attribute. For any $j = 1, \ldots, h$, let the function φ define the attribute that labels the j^{th} row as $\varphi(j)$. Given the column vector $\overrightarrow{v} = (s, x_2, \ldots, x_n)^T$, in which T is the transpose of the vector $\overrightarrow{v}$, s is the secret that will be shared, and $x_2, \ldots, x_n \in \mathbb{Z}_p$ are uniformly chosen at random; then $M\overrightarrow{v}$ is the vector of h shares of the secret s based on Π. The share $(M\overrightarrow{v})_j$ belongs to the attribute $\varphi(j)$.

Let attribute set $S \in \mathbb{A} \wedge S \in \mathbb{S}$ be any anthorized attribute set, and let $J = \{j | j \in \{1, \ldots, h\} \wedge \varphi(j) \in S\}$. Then, there exist constants $\{n_j \in \mathbb{Z}_p\}_{j \in J}$ such that if $\{s_j\}_{j \in J}$ are vaild shares of a secret s according to Π, then $\Pi_{j \in J} n_j s_j = S$.

Table 1: Notation description.

Notations	Descriptions
$[n]$	$\{1, \ldots, n\}$
G^*, G_T^*	Two multiplicative cyclic groups
GPK	Global public key
$\text{SK}_{\text{GID},S}$	Secret attribute key for DU
APK	Authority public key
ASK	Authority secret key
C_s	Ciphertext component encrypted by symmetric algorithm
(A, ρ)	LSSS access policy
CT_p	The ciphertext component encrypted with ABE algorithms
CT	The whole ciphertext

3.4. Group Signature. First proposed by Chaum and van Heyst in 1991 [29], group signature aims to sign a message in the name of the group for any group member. In a group signature scheme, any group member can sign a message anonymously on behalf of the entire group. Similar to a general digital signature, a group signature is publicly verifiable and can be verified by a single group public key. Moreover, the actual identity of the signer in the system cannot be traced by the verifier. Only the group manager can identify the real signer. A group signature scheme should satisfy the following requirements:

Unforgeability. No one can generate a valid group signature except the members of the group.

Anonymity. Given a group signature, determining the identity of the signer is computationally infeasible for anyone except the manager of the group.

Traceability. The group manager can trace the real identity of a malicious user when a dispute occurs.

Unlinkability. Without opening the group signature, it is difficult to distinguish whether two different signatures are made by the same group member.

Nonframeability. No one, including the group manager, can generate a valid group signature in the name of other group members.

The advantageous property of a group signature is that it enables suitable anonymity in various scenarios.

3.5. Merkle Hash Tree. The Merkle hash tree [30] is a specific binary tree that can be used to authenticate data. As shown in Figure 2, assume that we want to use a MHT to authenticate a file F. We divide file F into L file blocks and construct a MHT with L leaves, which store the hash values of $f_1, \ldots, f_L$. Figure 2 depicts an example of a MHT. File F has been divided into 4 blocks. The verifier has the root hash value HR and requests f_2. He requires authentication of the received file. The prover provides the verifier the auxiliary authentication information $\Omega = H(f_1, H_B)$. The verifier receives Ω and then computes $H_A = H(H(f_1) \| H(f_2))$, $H_R' = H(H_A \| H_B)$. Finally, the verifier checks whether H_R' is the same as HR.

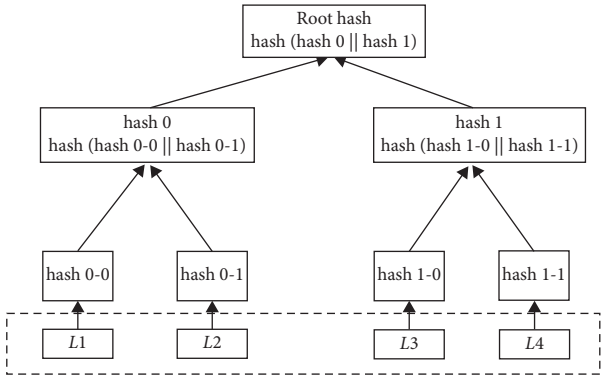

Figure 2: An example of MHT.

3.6. Blockchain

Definition 3 (blockchain [31]). The blockchain technology was first used for bitcoin, introduced by Nakamoto in 2008 [31]. In recent years, the attitude of bitcoin has been rising and falling in the world, but as one of the underlying technologies of bitcoin, blockchain technology has been paid more and more attention. In the process of bitcoin formation, blocks are storage units one by one, recording all the communication information of each block node in a certain period of time. Each block is linked by random hash (also known as hash algorithm). The latter block contains the hash value of the previous block. With the expansion of information exchange, one block is connected with another block, and the result is called blockchain [32]. In essence, it is a shared database. The data or information stored in it has the characteristics of "unforgeability," "whole process trace," "traceability," "openness and transparency," and "collective maintenance." Based on these characteristics, blockchain technology has laid a solid "trust" foundation, created a reliable "cooperation" mechanism, and has broad application prospects.

3.7. Cryptographic Background

Definition 4 (bilinear maps [3]). We consider two p-ordered G_0 and G_1 groups that are multiplicative cyclic, where p is a prime. ε, ε are two generators of group G_0. If the map $\widehat{e}$: $G_0 \times G_0 \longrightarrow G_1$ satisfies the following properties, then we call it a bilinear map:

(1) Bilinearity: $\widehat{e}(\varepsilon^a, \varepsilon^b) = \widehat{e}(\varepsilon, \varepsilon)^{ab}, \forall a, b \in Z_p, \varepsilon, \varepsilon \in G$

(2) Nondegeneracy: $\widehat{e}(\varepsilon, \varepsilon) \neq 1_{G_1}$, $\widehat{e}(\varepsilon, \varepsilon)$ is a generator of G_1

(3) Computability: $\widehat{e}(\varepsilon, \varepsilon)$ is efficiently computable for all $\varepsilon, \varepsilon \in G_0$

4. System Model and Security Requirements

We present the system and threat model as well as corresponding security requirements for our work in this section.

4.1. System Model. Figure 3 shows the model of our system for HTR-DAC. It consists of five entities: cloud service provider (CSP), attribute authorities (AAs), agencies (ACs), data user (DU), and data owner (DO), which are described as follows:

CSP: the CSP provides large amounts of resources, such as storage and computation. It can also publish various services.

AA: the AA is in charge of attribute assignment and secret key distribution and generation for users.

DO: the DO produces a large amount of data and outsources them to CSP for saving cost and data sharing. Before data uploading, he will encrypt the data and designate a specific access policy. He also generates corresponding group signature for data integrity verification.

DU: the DU enjoys the data sharing service. Only authorized DU can access and recover the plaintext by his secret key. Any user can publicly audit shared data through blockchain and interaction with CSP.

AC: the AC takes charge of user in a group. It is responsible for data uploading and blockchain operation. When a user uploads the data, AC will forward the ciphertext and record corresponding verification information into blockchain for public auditing.

4.2. Threat Model. In our proposal, the AAs, ACs, and DO are regarded as the fully trusted entities, while the CSP is considered to be untrusted, which may intentionally leak the sensitive information or even tamper and delete the shared data of users. Moreover, some unauthorized DU may illegally access the sensitive data by collusion attack, which will break the data security and privacy. Aiming to resist these attacks, we take the following security requirements into consideration:

Data confidentiality: DO should guarantee the confidentiality of his data and any DU can access the shared data if and only if his access rights satisfy the access policy of the data

Collusion resistance: any users in the system should not have the ability to combine their secret keys to make them satisfy the access policy and thus access the shared data illegally

Anonymity: as the real identity of user, especially in IoT systems, may contain some privacy, to prevent the user privacy from leakage, the ciphertext should be shared without leaking real identity

Auditability: each system user can publicly verify the integrity of the shared data instead of deploying a trusted auditor

Traceability: when a dispute occurs, the agencies can trace the real identity of a malicious user for traitor tracing

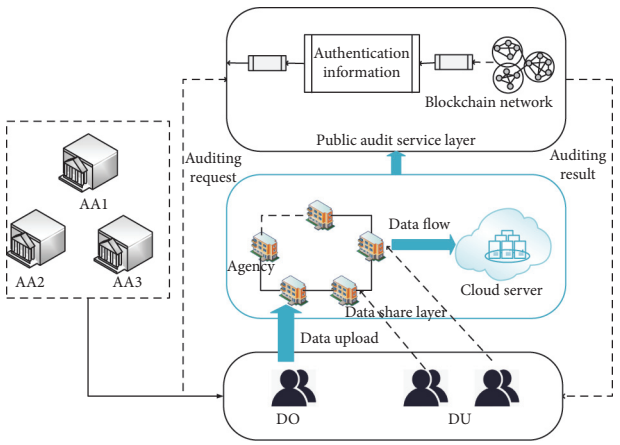

Figure 3: An overview of cloud-based IoT.

In addition to these security requirements, we also raise the following performance related design goals for our scheme:

Large universe: the system should afford large attribute universe; that is, any string can be employed as an attribute for system users and the system and attribute authorities need no global parameters for each attribute.

Multiauthority: the system should support multiple attribute authorities that collaboratively manage the attribute universe. That is, attribute authorities should take charge of user attribute assignment and attribute secret keys generation.

4.3. System Framework. According the above system model, the procedure of our system includes several phases as follows:

Initialization. In this phase, the system finishes initialization and generates global public key and publishes these parameters to the whole system. Then, the attribute authorities generate their own public keys and secret keys and also publish their public keys in system domain.

Authorization. In this phase, each user is registered into the system by a joining request. The attribute authorities collaboratively assign attribute secret key components to each user according to his attribute set. Moreover, the agencies generate signing secret key for each user. At the end, each user can get his secret key after successfully taking part in the system.

Encryption. In this phase, each DO encrypts his data produced by himself by a symmetric encryption algorithm with designating a specific access policy for fine-grained access control. Also, the user needs to create group signature for data integrity verification, public auditing, and traitor tracing. When an integrated ciphertext is generated, the user sends it to the agency in charge and, subsequently, the agency checks the ciphertext and uploads it to cloud and records the verification information into blockchain by related algorithms.

Data Auditing. In this phase, each user in system can publicly audit the data without a trusted auditor and many computations and communication overhead. As the verification information is recorded in blockchain without being tampered, the user can successfully execute data auditing algorithm and gets its results.

Decryption. In this phase, DU requests to access the data file according to the auditing result. If the DU is authorized, that is, he has enough access rights, then DU can decrypt the shared ciphertext and check its integrity.

User Tracing. In this phase, as we leverage the group signature and ABE technique, the user identity is hidden from the ciphertexts. Any time a dispute happens, the data owner will be traced by revealing its real identity with the group signature mechanism by corresponding agency. Moreover, the group signature also guarantees the avoidance of being framed.

Here, we describe the formal definition of our scheme. A blockchain-based fine-grained data-sharing scheme for multiple groups consists of the following algorithms:

$\text{Setup}_G(\kappa, k, l)$: given the security parameters κ, k, l, the algorithm initializes the whole system and generates global public parameter GPK.

$\text{Setup}_A(\text{GPK})$: on inputting the global public parameter GPK, the algorithm generates public key APK and secret key ASK for each attribute authority.

$\text{KeyGen}(\text{GPK}, \text{GID}, S, \{\text{ASK}_j\})$: after receiving the global public key GPK, the user's global identity GID, and attribute set S as well as the corresponding authority public key set $\{\text{ASK}_j\}$, the algorithm generates secret key for each system user.

$\text{Encrypt}(\text{GPK}, M, (A, \rho), \{\text{APK}_j\})$: given the global public key GPK, the message to be outsourced M together with its access policy (A, ρ), and the authority public key set $\{\text{ASK}_j\}$, the algorithm computes all components of ciphertext which are encrypted by symmetric encryption and CP-ABE.

$\text{SigGen}(\text{GPK}, C_s, C_p)$: the algorithm generates group signature for ciphertext output in Encrypt algorithm and returns the final ciphertext. The algorithm not only uploads the final ciphertext to cloud but also records the verification information into blockchain through relevant transaction.

$\text{Audit}(\text{GPK})$: the algorithm is executed by any system user to verify data stored in cloud publicly. Given the global public key GPK, the algorithm checks the integrity of shared ciphertext with verification information recorded in blockchain and the auxiliary information from cloud.

$\text{Decrypt}(\text{GPK}, \text{CT}, \text{SK}_{\text{GID},S})$: given the global public key GPK, user's secret key $\text{SK}_{\text{GID},S}$, and the ciphertext CT to be accessed, the algorithm is executed by user GID. It can verify the data with signature verification algorithm and decrypts CT if he has authorized secret key.

Trace(GPK, Sig_G): if a dispute happens, the data owner needs to be traced by exposing his real identity executed by corresponding agency AC_i. Given the global public key GPK and the signature of the ciphertext Sig_G, AC_i can recover the real identity of data owner denoted by his certificate.

4.4. Security Model. In this section, we present a static security model for multiauthority CP-ABE schemes by a security game between an adversary and a challenger.

Setup. The challenger runs the global setup algorithm of multiauthority CP-ABE and gives the global parameter GPK to the adversary.

Adversary's Queries. The adversary proceeds as follows:

It chooses a corrupt authority set $C \subset U$ and sends the public keys of these corrupt authorities to the challenger.
It chooses a good authority set $N \subset U$ and queries the public keys of these good authorities.
It makes secret key queries for a sequence $\{(S_j, \text{GID}_j)\}_{j=1}^{m}$, where GID_j is an identity and $S_j \subset U$ is an attributes set. In this sequence, we require that the identities $\{\text{GID}_j\}$ be different and none of these keys come from a corrupt authority; that is, $T(S_j) \cap C = \varnothing$.
It specifies two equal-length messages M_0, M_1 and an access structure (A, ρ) to the challenger for a challenge ciphertext. We require that, for each identity GID_j, this access structure (A, ρ) cannot be satisfied by $S_{C_\theta} \cup S_j$, where S_{C_θ} is the set of all the attributes that are controlled by the corrupt authorities.

Challenger's Replies. The challenger flips a random coin $b \in \{0, 1\}$ and gives the adversary with the following:

The public keys $\{\text{PK}_\theta\}_{\theta \in N}$ corresponding to the good authorities N
The secret keys $\{\text{SK}_{S_j}, \text{GID}_j\}_{j=1}^{m}$ corresponding to $\{(S_j, \text{GID}_j)\}_{j=1}^{m}$
The challenge ciphertext $\text{CT}^* \longleftarrow \text{Encrypt}(\text{GPK}, \text{PK}, M, (A, \rho))$

Guess. The adversary outputs a guess b' for b. The advantage of the adversary in this game is defined as $\Pr[b = b'] - (1/2)$.

Definition 5. A multiauthority CP-ABE scheme is statically secure (against static corruption of authorities) if all polynomial time adversaries have at most a negligible advantage in this security game.

5. Blockchain-Based Fine-Grained Data Sharing for Multiple Groups

In this section, we describe our proposal by giving the overview of the system and the concrete construction.

5.1. Overview. Our scheme aims to protect the shared data across multiple groups. It highlights the aspects of data integrity and fine-grained access simultaneously. To ease the overhead of a trusted third party as auditor, we introduce blockchain to fulfill public data auditing; and the feature of tamper-resistance in blockchain also makes it trusted and public. Moreover, we leverage group signature and ABE to achieve anonymity, fine-grained access control, traceability, and nonframeability.

We let U_a denote the attribute universe and U_A is the authority universe. In our scheme, the attribute universe in U_a is expressed as any string which satisfies the large universe attribute. As in [20], we also define a map $H: U_a \longrightarrow U_A$ from an attribute to the index of the authority in charge. That is, for $\forall \text{att}_i \in U_a$, it belongs to authority $AA_{H(\text{att}_i)}$.

5.2. Concrete Construction. Here, we describe the concrete construction of our proposal. In particular, our scheme involves such algorithms: Setup, CreateTopDA, CreateEntity, TKeyGen, Encrypt, TokenGen, Trap, $\text{Decrypt}_{\text{OUT}}$, Decrypt_U, and DecVerify, which are described in detail subsequently.

5.2.1. Initialization Phase. This phase consists of two algorithms for system setup and authority setup.

$\text{Setup}_G(\kappa, k, l)$: given the security parameter κ, k, l, the algorithm creates two big primes p, q of κ bits and generates a bilinear group $(G^*, G_T^*, p, g, \hat{e})$, where G^*, G_T^* are two multiplicative cyclic groups with order p, g is a generator of G^*, and $\hat{e}: G^* \times G^* \longrightarrow G_T^*$ is a bilinear map. It also chooses hash functions $H_0: \{0,1\}^* \longrightarrow G^*$, $H_1: U_a \longrightarrow G^*, H_2: \{0,1\}^* \longrightarrow Z_p$ and a pair of symmetric encryption algorithms $(\text{Enc}_s, \text{Dec}_s)$. Moreover, the algorithm computes $Q_1 = 2p + 1, Q_2 = 2q + 1$ and selects elements $h, a_1, a_2, g_0, g_1, g_2, \delta_1, \delta_2 \in_R \text{QR}(n)$, where $n = Q_1 Q_2$ and $\text{QR}(n)$ denotes the quadratic residue of Z_p^*. Finally, the algorithm publishes the global public key as $\text{GPK} = \{G^*, G_T^*, g, h, a_1, a_2, g_0, g_1, g_2, \delta_1, \delta_2, \hat{e}, H, H_0, H_1, H_2, (\text{Enc}_s, \text{Dec}_s)\}$.

$\text{Setup}_A(\text{GPK})$: each attribute authority $AA_i \in U_A$ randomly selects $\alpha_i, \beta_i \in Z_p$. Then, it picks $\theta_i \in Z_p$ and computes $Y_i = g^{\theta_i}$. Finally, it publishes its public key $\text{APK}_i = \{\hat{e}(g, g)^{\alpha_i}, g^{\beta_i}, Y_i\}$ and keeps their master key $\text{ASK}_i = \{\alpha_i, \beta_i, \theta_i\}$ privately.

5.2.2. Registration Phase. This phase consists of the key generation algorithm to generate secret key for each user according to his attribute set.

KeyGen(GPK, GID, S, $\{\text{ASK}_j\}$): given the global public key GPK, the user global identity GID, and his attribute set S as well as the master key $\{\text{ASK}_j\}$ of attribute authorities that each attribute belongs to (different attribute in S may belong to the same authority), for each $\text{att}_i \in S$, the corresponding attribute authority AA_j

$(j = H(\text{att}_i))$ chooses $\mu \in_R Z_p$ and generates the secret attribute key components $\{\text{SK}_{\text{GID},i}\}_{i\in S}$ for user GID according to attribute set S, where

$$\text{SK}_{\text{GID},i} = \left\{D_{\text{GID},i} = g^{\alpha_j} H_0(\text{GID})^{\beta_j} H_1(\text{att}_i)^{\mu}, D'_{\text{GID},i} = g^{\mu}\right\}. \tag{1}$$

Then, the user GID randomly picks $d_{\text{GID}} \in [2^{u_2}], e_{\text{GID}} \in [n]$ and computes $f_{\text{GID},1} = g_0^{d_{\text{GID}}} g_0^{e_{\text{GID}}}$. The user GID sends $f_{\text{GID},1}$ to the agency AC_i and receives $b_{\text{GID}}, c_{\text{GID}} \in_R [2^{u_2}]$ from AC_i after successfully verifying the user GID by checking if $f_{\text{GID}} \in \text{QR}(n)$. The user GID then computes $d'_{\text{GID}} = 2^{u_1} + (b_{\text{GID}} d_{\text{GID}} + c_{\text{GID}} \bmod 2^{u_2})$, $f_{\text{GID},2} = a_1^{d_{\text{GID}}}$ and sends $f_{\text{GID},2}$ to AC_i. After checking $f_{\text{GID},2} \in \text{QR}(n)$, AC_i randomly selects $L_{\text{GID}} \in B$ and computes $K_{\text{GID}} = (f_{\text{GID},2}^{a_2})^{1/L_{\text{GID}}}$ and returns $(K_{\text{GID}}, L_{\text{GID}})$ to the user GID. If successfully checked with $a_1^{d'_{\text{GID}} a_2} = K_{\text{GID}}^{L_{\text{GID}}}$, the algorithm generates the secret key for the user GID as $\text{SK}_{\text{GID},S} = \left\{K_{\text{GID}}, L_{\text{GID}}, \{\text{SK}_{\text{GID},i}\}_{i\in S}\right\}$, where $(K_{\text{GID}}, L_{\text{GID}})$ is the user certificate used for signature generation and user identifying.

5.2.3. Encryption Phase. The phase includes two steps: encrypting and signing. The former algorithm finishes the work of symmetric encryption and attribute-based encryption for fine-grained access, while the latter is responsible for group signature generation to ensure the data integrity and auditing.

Encrypt(GPK, M, (A, ρ), $\{\text{PK}_j\}$): when data owner decides to outsource and share the data M with a designated LSSS access policy (A, ρ), where A is a $l \times m$ share-generating matrix and ρ is a corresponding map from a row A_x of A to an attribute $\text{att}_x \in U_a$, that is, $\rho(A_x) = \text{att}_x$, where $x \in [l]$, we can also infer that, with the map H of GPK, we can get the index $j = H(\text{att}_x)$ of the corresponding attribute authority in charge of the attribute att_x; that is, H maps each row A_x to a specific authority AA_j.

First of all, the algorithm chooses a random element $k \in G_T^*$ and computes $C_s = \text{Enc}_s(k, M)$ to get the encrypted data, where Enc_s is the symmetric encryption algorithm in GPK.

Then, the algorithm selects two random vectors $\gamma = \{s, \gamma_2, \ldots, \gamma_m\}$ and $\gamma' = \{0, \gamma'_2, \ldots, \gamma'_m\}$, where $s, \gamma_i, \gamma'_i \in_R Z_p$. Thus, it can get $\lambda_x = A_x \cdot \gamma$ which is the share component of s for each attribute $\rho(i)$ corresponding to x-th row A_x, and $\lambda' = A_x \cdot \gamma'$ which is the share component of 0 for each attribute of access policy. Moreover, the algorithm computes the ciphertext according to the policy after choosing $r_x \in Z_p$, where $x \in [l]$, and gets $C_p = \left\{C, \left\{C_{x,1}, C_{x,2}, , C_{x,3}, , C_{x,4}\right\}_{x\in[l]}\right\}$, where

$$\begin{aligned} C &= k \cdot \hat{e}(g,g)^s, \quad \forall x \in [l]: \\ C_{x,1} &= \hat{e}(g,g)^{\lambda_x + \alpha_{H(\rho(x))} r_x}, \\ C_{x,2} &= g^{r_x}, \\ C_{x,3} &= g^{\beta_{H(\rho(x))} r_x} g^{\lambda'_x}, \\ C_{x,4} &= H_1(\rho(x))^{r_x}, \end{aligned} \tag{2}$$

SigGen(GPK, C_s, C_p): after receiving the encrypted data C_s with symmetric encryption, the algorithm divides it into M parts, that is, $\{C_s^1, C_s^2, \ldots, C_s^M\}$, and computes hash value of each part with hash function H_2 of GPK to get $H_f = \{H_2(C_s^1), \ldots, H_2(C_s^M)\}$ and its root hash value H_R according to MHT algorithm. Then, the algorithm selects random values $r \in 0, 1^2 l$ and computes $T_1 = K_{\text{GID}} Y_i^r, T_2 = g_0^r, T_3 = g_0^{L_{\text{GID}}} h^r$. Moreover, the algorithm selects other random numbers $r_1 \in \{0,1\}^{\kappa(\gamma_1 + k)}, r_2 \in \{0,1\}^{\kappa(\lambda_2 + k)}, r_3 \in \{0,1\}^{\kappa(\gamma_1 + 2l + k + 1)}$, $r_4 \in \{0,1\}^{\kappa(2l+k)}$ and computes $d_1 = T_1^{r_1} / (a_1^{r_2} Y_j^{r_3})$, $d_1 = T_2^{r_1} / g^{r_3}, d_3 = g^{r_4}, d_4 = g^{r_1} h^{r_4}$ as well as the hash value $H_v = H_2(g \| h \| Y \| a_1 \| a_2 \| T_1 \| T_2 \| T_3 \| d_1 \| d_2 \| d_3 \| d_4 \| H_R \| C_p)$. In addition, it outputs the signature $\text{Sig}_G = (c, s_1, s_2, s_3, s_4, T_1, T_2, T_3)$, where

$$\begin{aligned} s_1 &= r_1 - c(e_i - 2^{\gamma_1}), \\ s_2 &= r_2 - c\left(x_i - 2^{\lambda_2}\right), \\ s_3 &= r_3 - cre_i, \\ s_4 &= r_4 - cr. \end{aligned} \tag{3}$$

Finally, the algorithm outputs the ciphertext $\text{CT} = \left\{\text{ID}_f, C_s, C_p, \text{Sig}_G, H_f, H_R, M\right\}$, where ID_f is the identity of the file and M is the number of H_f, and outsources it to AC_i which is the managing agency in charge of the user GID.

Subsequently, the agency AC_i needs to verify the shared ciphertext CT by Algorithm 1 and (C_s, H_f, H_R) with MHT verification algorithm. If successful, AC_i uploads CT to the cloud and stores the verification information to blockchain as described in Algorithm 2.

5.2.4. Data Auditing

Audit(GPK): the algorithm is executed by any system user to audit the shared data. Due to the transparency of data in blockchain, any user can finish data auditing instead of a trusted public auditor. As described in algorithm Encrypt, the verification

information of each file is stored in blockchain, and the users in system can get the information including H_R, Sig_G, M. Thus, with the auxiliary information from cloud, each system user audits the data as follows:

(1) The algorithm chooses a number $t \in_R N^*$ and $M\%t = 0$ and divides the leaves of Merkle tree for the file ID_f into t subgroups. Then, it computes $rs = \mathrm{PR}(H_R^t, \delta)$, where PR is a cryptographic pseudorandom generator and $\delta \in [t-1]$.

(2) After receiving the auxiliary information from cloud, the algorithm computes $H_{R'}$ by constructing a path to the root of Merkle tree. It checks if $H_R = H_R'$ and $\mathrm{PR}(H_{R'}^t, \delta) = rs$. If the verification is successful, the data are validated.

5.2.5. Decryption Phase.

Decrypt$(\mathrm{GPK}, \mathrm{CT}, \mathrm{SK}_{\mathrm{GID},S})$: given the secret key $\mathrm{SK}_{\mathrm{GID},S}$ of data user and the ciphertext CT, the algorithm can find a set of constances $\{\omega_x \in Z_p\}$ which makes the equation $\sum_{x=1}^{l} w_x A_x = (1, 0, \ldots, 0)$ hold if the data user is authorized. Then, for each row $x \in [l]$ of access policy, the algorithm computes

$$\begin{aligned} R_x &= \frac{C_{x,1} \cdot \hat{e}\left(H_0(\mathrm{GID}), C_{x,3}\right) \cdot \hat{e}\left(D_{\mathrm{GID},\rho(x)}', C_{x,4}\right)}{\hat{e}\left(D_{\mathrm{GID},\rho(x)}, C_{x,2}\right)}, \\ &= \hat{e}(g,g)^{\lambda_x} \hat{e}\left(H_0(\mathrm{GID}), g\right)^{\lambda_x'}. \end{aligned} \tag{4}$$

Next, according to the constances $\{\omega_x\}$, the algorithm computes the following equation:

$$\begin{aligned} R &= \prod_{x=1}^{l} R^{\omega_x} = \hat{e}(g,g)^s, \\ k &= \frac{C}{R}, \\ Ml &= \mathrm{Dec}_s(k, C_s). \end{aligned} \tag{5}$$

Finally, the algorithm outputs the recovered plaintext M of data.

5.2.6. User Tracing

Trace$(\mathrm{GPK}, \mathrm{Sig}_G)$: if there exists a dispute each time, the DO should be traced by recovering his real identity by the agency AC_i in charge of the DO. As a member of blockchain, AC_i requests the blockchain for traitor tracing. The blockchain responds to the AC_i with the disputable data after running consensus protocol by which at least two-thirds of nodes in blockchain confirm the validity of the request and generate the new block as the proof of traitor tracing with synchronization.

Then, the algorithm computes $K_{\mathrm{GID}} = T_1 / T_2^{\theta_i}$ and gets the corresponding identity of user being traced.

6. Security Analysis

In this section, we give a brief security analysis for our proposal.

$$\begin{aligned} R_x &= \frac{C_{x,1} \cdot \hat{e}\left(H_0(\mathrm{GID}), C_{x,3}\right) \cdot \hat{e}\left(D_{\mathrm{GID},\rho(x)}', C_{x,4}\right)}{\hat{e}\left(D_{\mathrm{GID},\rho(x)}, C_{x,2}\right)} \\ &= \frac{\hat{e}(g,g)^{\lambda_x + \alpha_{H(\rho(x))} r_x} \hat{e}\left(H_0(\mathrm{GID}), g^{\beta_{H(\rho(x))} r_x} g^{\lambda_x'}\right)}{\hat{e}\left(g^{\alpha_j} H_0(\mathrm{GID})^{\beta_j} H_1(\mathrm{att}_i)^{\mu}, g^{r_x}\right)} \\ &\quad \cdot \frac{\hat{e}\left(g^{\mu}, H_1(\rho(x))^{r_x}\right)}{\hat{e}\left(g^{\alpha_j} H_0(\mathrm{GID})^{\beta_j} H_1(\mathrm{att}_i)^{\mu}, g^{r_x}\right)} \\ &= \hat{e}(g,g)^{\lambda_x} \hat{e}\left(H_0(\mathrm{GID}), g\right)^{\lambda_x'}, \\ R &= \prod_{x=1}^{l} R^{\omega_x} = \prod_{x=1}^{l} \left(\hat{e}(g,g)^{\lambda_x} \hat{e}\left(H_0(\mathrm{GID}), g\right)^{\lambda_x'}\right)^{\omega_x} \\ &= \hat{e}(g,g)^{\sum_{x=1}^{l} \omega_x \lambda_x} \hat{e}\left(H_0(\mathrm{GID}), g\right)^{\sum_{x=1}^{l} \omega_x \lambda_x'} \\ &= \hat{e}(g,g)^{s} \hat{e}\left(H_0(\mathrm{GID}), g\right)^{0} \\ &= \hat{e}(g,g)^{s}. \end{aligned} \tag{6}$$

Theorem 1. *The scheme satisfies the correctness of decryption.*

Proof. Our scheme can guarantee that the data user gets correct plaintext if and only if his attribute set S satisfies the LSSS access policy (A, ρ).

Theorem 2. *The scheme satisfies the requirement of fine-grained access and collusion resistance.*

Proof. In our scheme, on the one hand, the secret value s is separated according to secret sharing matrix A, and only the attribute set that is identical to an authorized set in A can satisfy matrix A and reconstruct s. That is, only users with enough access rights which are identical to authorized attribute set can recover the secret and plaintext. On the other hand, the authorities generate the attribute secret key for system users including the hash value of each user's global identity, which is unique for them. If two or more users want to launch collusion attack against our scheme, they have to combine their secret key components corresponding to different attributes. However, as the key component $\{D_{\mathrm{GID},i} = g^{\alpha_j} H_0(\mathrm{GID})^{\beta_j} H_1(\mathrm{att}_i)^{\mu}\}$ embeds $H_0(\mathrm{GID})$ which relates to user identity and μ which is randomized element for each user, it is impossible to conduct successful collusion attack against our scheme. □

Theorem 3. *The scheme satisfies the requirement of public auditability.*

Input: GPK: system public key
CT: message
σ_G: the group signature of message M
Output: True/False: the verification result.
(1) Compute the following values:
$d_1' = (a_1^c T_1^{s_1 - c2^{(\gamma_1)}} / a_1^{s_2 - c2^{\gamma_1}} Y^{s_3})$
$d_2' = (T_2^{s_1 - c2^{\gamma_1}} / g^{s_3})$
$d_3' = T_2^c g^{s_4}$
$d_4' = T_3^c g^{s_1 - c2^{\gamma_1}} h^{s_4}$
(2) GM randomly selects $e_i \in B$ computes $A_i = (C_2^{a_2})^{1/e_i}$, then sends (A_i, e_i) to UAV_i as its member certificate.
(3) User GID checks if $a_1^{x_i' a_2} = A_i^{e_i}$. If successful, then $\{A_i, e_i\}$ is its signing key.
(4) **if** $c' = c$ **then**
(5) **return** True
(6) **else**
(7) **return** False
(8) **end if**

Algorithm 1: Signature verification.

Input: GPK: system public key
VI: verification information $(H_R, M, \mathrm{Sig}_G, \mathrm{ID}_f, \omega)$
Output: True/False: the verification result.
(1) The nodes of the blockchain compute Merkle root value H_R' with ω, where ω is the information for the nodes to verify Merkle root H_R and get verification result Verby execute Algorithm 1 with Sig_G.
(2) **if** $H_R' = H_R$ and Ver = True **then**
(3) create a new block to store $(H_R, M, \mathrm{Sig}_G, \mathrm{ID}_f, t)$ where t is the timestamp.
(4) **else**
(5) abort
(6) **end if**

Algorithm 2: Storage transaction.

Proof. The security proof of these security requirements is similar to [14]. □

Theorem 4. *The scheme satisfies the requirements of anonymity and traceability.*

Proof. The security proof of these security requirements is similar to [14]. □

7. Performance Evaluation

In this section, we give a thorough performance analysis of our scheme by implementation and comparison with another existing related work [20] from theoretical computation and storage complexity and actual time and storage cost.

Table 2 summarizes the computation and storage complexity comparison between our scheme and the scheme in [20]. We analyze the computation complexity of Encrypt, Decrypt, and KeyGen of the two schemes. From the comparison, we can infer that as our scheme introduces group signature for data integrity verification, Encrypt and KenGen introduce some extra overhead in computation complexity, while the scheme in [20] also incurs some overhead by introducing tracing mechanism. On the other hand, we analyze the storage complexity of the two schemes from *SK Size* and *PP Size*, which denote the secret key size and public parameter size, respectively. We note that, for the same reason, the public key size and secret key of system users in our scheme increase for signature generation, while the storage complexity of the other scheme shows the same thing.

Figure 4 plots the comparison of encryption time cost for our scheme and the scheme in [20]. As we can see, the time cost in encryption algorithm in two schemes is affected by the complexity of policy. It is in proportion to the row number of access policy. We set the number of files from 1 to 5, and, no matter in which condition, the encryption time cost in our scheme is less than that in [20].

Figure 5 plots the comparison of encryption time cost for our scheme and the scheme in [20]. It is obvious that the time cost of decryption algorithm in both schemes is

Table 2: Complexity comparison.

Schemes	Encrypt	Decrypt	KeyGen	*SK Size*	*PP Size*
Scheme [20]	$6lE_1 + (2l+1)E_2 + lM_1 + (l+1)M_2$	$3E_1 + lE_2 + 3M_1 + (l+3)M_2 + 3P$	$5\|S\|E_1 + 2\|S\|M_1$	$3\|S\|\|G^*\|$	$3\|G^*\| + \|G_T^*\|$
Our scheme	$(4l+8)E_1 + (l+1)E_2 + (l+4)M_1 + M_2$	$lE_2 + (l+3)M_2 + 3P$	$(4\|S\|+1)E_1 + 2\|S\|M_1$	$(2\|S\|+1)\|G^*\|$	$2\|G^*\| + \|G_T^*\|$

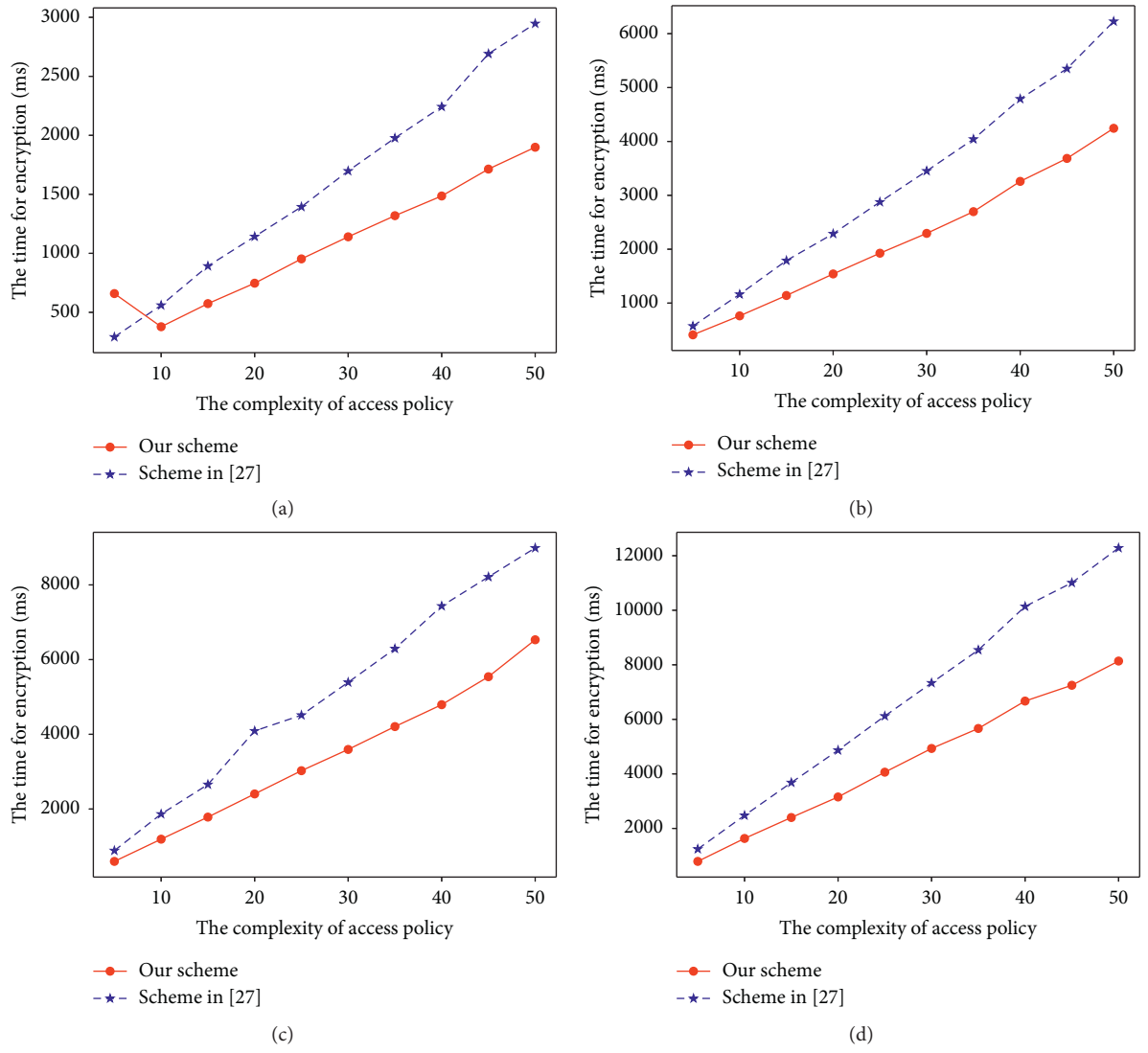

Figure 4: Encryption cost comparison for different number of files. (a) $|\overline{X}| = 0$. (b) $|\overline{X}| = 2$. (c) $|\overline{X}| = 4$. (d) $|\overline{X}| = 6$.

affected by policy complexity as in encryption. We test the time cost in different file number condition, and the result shows that the time cost in our scheme is far less than that in the other scheme for the same attribute size and same number of files.

Figure 6 plots the comparison of authority public parameter and user secret key storage cost for our scheme and the scheme in [20]. We note that both schemes have constant public parameter size having no relationship with the size of attribute. This is consistent with the property of large universe. The size of public parameter for each authority in our scheme is far less. Moreover, for the size of secret key, we note that our scheme has a smaller size in key length than the other scheme.

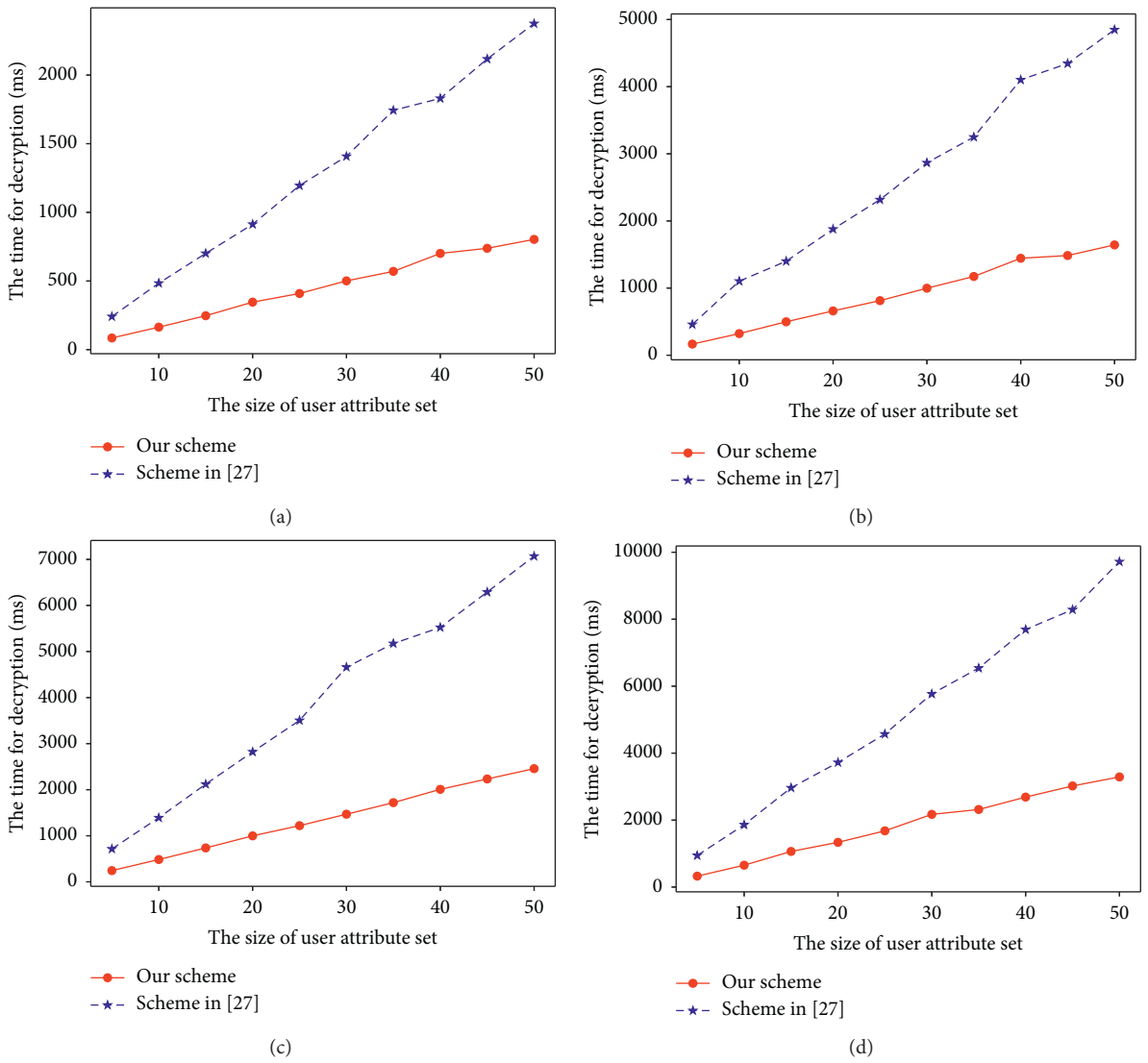

Figure 5: Decryption cost comparison for different number of files. (a) $|F| = 1$. (b) $|F| = 2$. (c) $|F| = 3$. (d) $|F| = 4$.

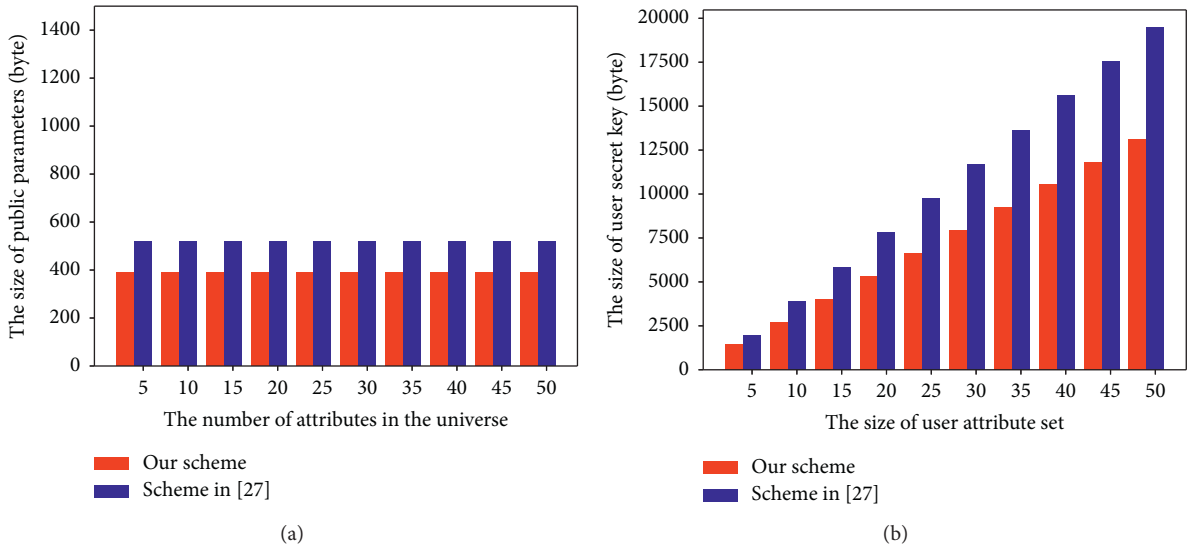

Figure 6: Comparison of the storage cost. (a) Public parameter size. (b) User key size.

8. Conclusion

In this paper, we propose a blockchain-based fine-grained data-sharing scheme for multiple groups in cloud-based IoT systems. In our proposal, we design a novel multiauthority large universe CP-ABE scheme to guarantee the fine-grained access control and data integrity across multiple group by integrating group signature into our scheme. Moreover, to ease the need for a third trusted auditor in traditional data public auditing schemes, we introduce blockchain technique to enable a distributed data public auditing. In addition, with the group signature, our scheme also realizes anonymity and traitor tracing. The security analysis and performance evaluation show that our scheme is practical for large-scale IoT systems.

Data Availability

Data are available upon request to the corresponding author.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was funded by the National Natural Science Foundation of China (nos. 61902291 and 62072352), China Postdoctoral Science Foundation Funded Project (2019M653567), National Natural Science Foundation of Shaanxi Province (2019JM-425), and the Fundamental Research Funds for the Central Universities (JB191507).

References

[1] C. Savaglio, M. Ganzha, M. Paprzycki, C. Bădică, M. Ivanović, and G. Fortino, "Agent-based internet of things: state-of-the-art and research challenges," *Future Generation Computer Systems*, vol. 102, pp. 1038–1053, 2020.

[2] H. Tabrizchi and M. K. Rafsanjani, "A survey on security challenges in cloud computing: issues, threats, and solutions," *The Journal of Supercomputing*, vol. 2, no. 1, pp. 1–40, 2020.

[3] J. Zhang, J. Ma, Z. Ma, N. Lu, and D. Wei, "Efficient hierarchical data access control for resource-limited users in cloud-based e-health," in *Proceedings of the 2019 International Conference on Networking and Network Applications (NaNA)*, Daegu, Korea, October 2019.

[4] Q. Huang, Y. Yang, and J. Fu, "Secure data group sharing and dissemination with attribute and time conditions in public cloud," *IEEE Transactions on Services Computing*, vol. 42, 2018.

[5] H. Xiong, H. Zhang, and J. Sun, "Attribute-based privacy-preserving data sharing for dynamic groups in cloud computing," *IEEE Systems Journal*, vol. 13, no. 3, pp. 2739–2750, 2018.

[6] P. Zhang, Z. Chen, J. K. Liu, K. Liang, and H. Liu, "An efficient access control scheme with outsourcing capability and attribute update for fog computing," *Future Generation Computer Systems*, vol. 78, pp. 753–762, 2018.

[7] W. Shen, J. Qin, J. Yu, R. Hao, and J. Hu, "Enabling identity-based integrity auditing and data sharing with sensitive information hiding for secure cloud storage," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 2, pp. 331–346, 2018.

[8] N. Garg, S. Bawa, and N. Kumar, "An efficient data integrity auditing protocol for cloud computing," *Future Generation Computer Systems*, vol. 109, 2020.

[9] S. Hiremath and R. S. Kunte, "Homomorphic authentication scheme for proof of retrievability with public verifiability," in *2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS).*, pp. 1017–1022, Madurai, India, June 2020.

[10] X. Lu, Z. Pan, and H. Xian, "An integrity verification scheme of cloud storage for internet-of-things mobile terminal devices," *Computers & Security*, vol. 92, Article ID 101686, 2020.

[11] C.-C. Lee, P.-F. Ho, and M.-S. Hwang, "A secure e-auction scheme based on group signatures," *Information Systems Frontiers*, vol. 11, no. 3, pp. 335–343, 2009.

[12] C.-C. Lee, Y.-M. Lai, and P.-J. Cheng, "An efficient multiple session key establishment scheme for vanet group integration," *IEEE Intelligent Systems*, vol. 31, no. 6, pp. 35–43, 2016.

[13] C.-T. Li, D.-H. Shih, C.-C. Wang, C.-L. Chen, and C.-C. Lee, "A blockchain based data aggregation and group authentication scheme for electronic medical system," *IEEE Access*, vol. 8, pp. 173 904–173 917, 2020.

[14] H. Huang, X. Chen, and J. Wang, "Blockchain-based multiple groups data sharing with anonymity and traceability," *Science China Information Sciences*, vol. 63, no. 3, Article ID 130101, 2020.

[15] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Annual International Conference On the Theory and Applications Of Cryptographic Techniques*, pp. 457–473, Aarhus, Denmark, May 2005.

[16] 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, Zurich, Switzerland, April 2006.

[17] 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. 33, 2020.

[18] P. K. Premkamal, S. K. Pasupuleti, and P. Alphonse, "Dynamic traceable cp-abe with revocation for outsourced big data in cloud storage," *International Journal of Communication Systems*, vol. 34, no. 4, Article ID e4351, 2020.

[19] A. Lewko and B. Waters, "Decentralizing attribute-based encryption," *Advances in Cryptology*, vol. 6597, pp. 568–588, 2011.

[20] K. Zhang, H. Li, J. Ma, and X. Liu, "Efficient large-universe multi-authority ciphertext-policy attribute-based encryption with white-box traceability," *Science China Information Sciences*, vol. 61, no. 3, Article ID 032102, 2018.

[21] K. Sethi, A. Pradhan, and P. Bera, "Practical traceable multi-authority cp-abe with outsourcing decryption and access policy updation," *Journal of Information Security and Applications*, vol. 51, Article ID 102435, 2020.

[22] H. Huang, X. Chen, Q. Wu, X. Huang, and J. Shen, "Bitcoin-based fair payments for outsourcing computations of fog devices," *Future Generation Computer Systems*, vol. 78, pp. 850–858, 2018.

[23] H. Huang, K. C. Li, and X. Chen, "Blockchain based fair three party contract signing protocol for fog computing," *Concurrency and Computation: Practice and Experience*, vol. 31, no. 22, Article ID e4469, 2019.

[24] C. Yang, X. Chen, and Y. Xiang, "Blockchain-based publicly verifiable data deletion scheme for cloud storage," *Journal of*

Network and Computer Applications, vol. 103, pp. 185–193, 2018.

[25] J. Liang, W. Han, Z. Guo et al., "Desc: enabling secure data exchange based on smart contracts," *Science China Information Sciences*, vol. 61, no. 4, Article ID 049102, 2018.

[26] E. Gaetani, L. Aniello, R. Baldoni, F. Lombardi, A. Margheri, and V. Sassone, "Blockchain-based database to ensure data integrity in cloud computing environments," 2017.

[27] S. Ghoshal and G. Paul, "Exploiting block-chain data structure for auditorless auditing on cloud data," in *Proceedings of the International Conference On Information Systems Security*, pp. 359–371, Jaipur, India, December 2016.

[28] X. Fu, X. Nie, T. Wu, and F. Li, "Large universe attribute based access control with efficient decryption in cloud storage system," *Journal of Systems and Software*, vol. 135, pp. 157–164, 2018.

[29] D. Chaum and E. Van Heyst, "Group signatures," in *Workshop on the Theory and Application of Cryptographic Techniques*, Springer, Berlin, Germany, 1991.

[30] N. Nesa and I. Banerjee, "Combining merkle hash tree and chaotic cryptography for secure data fusion in iot," in *Transactions on Computational Science*, Springer, Berlin, Germany, 2020.

[31] S. Nakamoto, "Bitcoin: a peer-to-peer electronic cash system," Manubot," Technical Report, 2019.

[32] S. Hakak, W. Z. Khan, G. A. Gilkar, M. Imran, and N. Guizani, "Securing smart cities through blockchain technology: architecture, requirements, and challenges," *IEEE Network*, vol. 34, no. 1, pp. 8–14, 2020.