BCAS: A blockchain-based ciphertext-policy attribute-based encryption scheme for cloud data security sharing

Yuting Zuo, Zhaozhe Kang, Jian Xu and Zhide Chen

Abstract
It is the most important and challenging problem to share the data safely in cloud computing. Some so-called trusted third parties may also infringe users' data privacy. It is an urgent problem for data owners to share data safely with the designated users rather than the third party or other users. Traditional encryption schemes utilize different keys to produce multiple encrypted copies of the same data for users. It is no longer applicable for cloud data sharing security. Attribute-based encryption can solve above problems, but it needs to rely on trusted third parties to protect the users' privacy. In this article, in order to address the above problems, we propose a blockchain-based ciphertext-policy attribute-based encryption scheme for cloud data secure sharing without relying on any trusted third parties. Blockchain-based ciphertext-policy attribute-based encryption scheme can protect the rights and security of data owner. Compared with existing cloud security schemes, the proposed scheme has more advantages in terms of the six aspects: (1) data owners have the authority to decide who can decrypt the data; (2) the operations of users are retained permanently, and all records are tamper-proof; (3) our proposed scheme has the characteristic of "one-to-many" encryption, and data is encrypted only once; (4) our scheme does not rely on any trusted third party; (5) in terms of the discrete logarithm problem and decisional q parallel-bilinear Diffie–Hellman exponent problem, we prove that our proposed scheme is secure; and (6) experiment shows that our proposed scheme is more efficient than the comparative scheme.

Keywords
Blockchain, ciphertext-policy attribute-based encryption, data sharing security, cloud computing

Date received: 1 January 2021; accepted: 8 February 2021

Handling Editor: Yanjiao Chen

Introduction
Cloud computing provides cost-effective and powerful data storage and management service on the Internet. The data sharing of this model is more flexible and convenient than that of the traditional storage ones. However, there are many challenges that have not yet been satisfactorily addressed in cloud storage security, such as user authentication, identity management, and data security sharing and storage. At present, the data of most users are uploaded to the open cloud, which is easy to be retrieved or even attacked by others. Currently, most cloud data storage relies on the so-called “trusted third party,” but ordinary users cannot trust verification on the third party. In order to ensure our legal identity, the third party needs us to provide personal identity information. We

Fujian Normal University, Fuzhou, China

Corresponding author:
Yuting Zuo, Fujian Normal University, Cangshan Campus, No. 8 Shangsan Road, Cangshan District, Fuzhou 350007, Fujian, China.
Email: 1415435249@qq.com

Creative Commons CC BY: This article is distributed under the terms of the Creative Commons Attribution 4.0 License (https://creativecommons.org/licenses/by/4.0/) which permits any use, reproduction and distribution of the work without further permission provided the original work is attributed as specified on the SAGE and Open Access pages (https://us.sagepub.com/en-us/nam/open-access-at-sage).
cannot guarantee that the third-party platform will not damage our identity and data privacy. For trusted third parties, users have no privacy. Users cannot monitor the behavior of the trusted third party. In addition, users cannot know whether the so-called trusted third party will illegally obtain our information and data. It is also impossible for users to verify whether the trusted third party is really trusted. When using the third-party platform, we only passively choose to accept the agreements of the third party. Therefore, it is an urgent problem to be solved how to realize the trusted and secure cloud data sharing without the third party.

We use cryptography to solve most security problems in cloud computing. Key management is a very important part of cloud computing. There are many challenges in terms of generating, managing, and distributing keys. In most cases, the key and its encryption and decryption mechanism in cloud storage services are provided by the cloud service providers. It is difficult for users to learn the real security of the system, and there are many hidden dangers in the security of users' data. Many scholars have proposed a variety of encryption schemes for cloud storage. The attribute-based encryption (ABE) scheme was first proposed by Sahai and Waters. ABE is suitable for cloud storage systems, which has the characteristic of "one-to-many" encryption. What's more, ABE is better suitable for cloud data sharing. Ciphertext-policy attribute-based encryption (CP-ABE) is one of the categories of ABE. In the CP-ABE scheme, ciphertext is associated with an access policy, while the user’s decryption key is identified by a set of descriptive attributes. In the CP-ABE scheme, an attribute authority manages attribute sets and is responsible for issuing keys to users based on the attributes. The data owner (DO) defines the access policy and encrypts the data. The user can decrypt the ciphertext if and only if the attributes associated with its secret key satisfy the access policy in the ciphertext. CP-ABE is a very promising encryption technique for secure data sharing in the context of cloud computing environment. DOs are allowed to fully control the access policy associated with their data which to be shared. A CP-ABE scheme consists of four algorithms: Setup, Encrypt, KeyGen, and Decrypt.

Setup (λ, U). The Setup algorithm takes as input the security parameter λ and the unique description of the attributes universe U. It outputs the public parameter PK and the master key MK.

Encrypt (PK, M, A). The Encrypt algorithm takes as input the public parameters PK, a message M, and an access structure A over the universe of attributes. The algorithm will encrypt M and output a ciphertext CT such that only a user who possesses a set of attributes satisfying the access structure will be able to decrypt the message. We assume that the ciphertext implicitly contains A.

KeyGen (MK, S). The KeyGen algorithm takes as input the master key MK and a set of attributes S. It outputs a private key SK.

Decrypt (PK, CT, SK). The Decrypt algorithm takes as input the public parameters PK, a ciphertext CT which contains an access policy A and a private key SK. If the attributes set S satisfies the access structure A, then the algorithm will decrypt the ciphertext and return a message M.

Blockchain originates from bitcoin which is a digital crypto-currency system proposed by S Nakamoto. It is essentially an open and distributed decentralized database. The database based on cryptography and P2P network connects each block through a hash function. The database stores specific transactions in the block by utilizing the Merkle tree structure which links all transactions together. Once a transaction has been tampered, it will lead to change the root of Merkle tree hash, resulting in changing the hash of the block. If we find that the hash of a block cannot correspond to the hash of the whole network, we can request the other relevant blocks or transaction information to forward trace the relevant transactions and determine which transaction has been tampered, so as to achieve the unforgeability, non-tamperability, and traceability. If a malicious user wants to tamper with a transaction, but he or she cannot obtain 51% of the computing power in the whole network. He or she also cannot get the recognition of other nodes in the whole network according to the consensus mechanism. In addition, he or she will not get the recognition of the blockchain. All transactions will be published in the whole network, and operate automatically employing the workload proof mechanism. All transactions will not be controlled by a third party. Consequently, the blockchain technology has the characteristics of open, autonomous, and independent of a trusted third party. Blockchain uses cryptography technology to ensure anonymity, which makes the blockchain-based system more secure than traditional data storage systems.

The fusion of blockchain technology and cloud computing technology has a great prospect. A lot of literatures applied blockchain technology to solve the problems of cloud security. At present, the blockchain technology is used to protect cloud data security.

Related work

Cloud computing provides the facility to store and share data and avail software, platform, and infrastructure as services. However, the security problem of cloud computing, especially the security problem of cloud data and cloud storage, has become an urgent
problem to be solved, and many solutions have been put forward successively.

Most schemes use cryptography to secure cloud data, this requires the DO to encrypts the data before uploading it, so an important part of this is key management. Now, the question arises whether the keys are to be stored with DO or any other third party. In some schemes, the data which are stored in the cloud are to be shared among multiple members of the particular members using group key.

In the research work, Kumar et al. introduce a public key infrastructure certificate scheme to secure the cloud application program to encrypt the key, which also binds the key with files based on File-Id attributes for personal storage. It is easy to maintain and manage the keys to update, but it is required to update the user credentials regularly. Suppose if the user credentials are weak, then the security of file cannot be assured. Here, the author encrypts the stored cloud data using an elliptic curve.

Cryptographic Key Management System (CKMS) provides encryption, decryption, identity authentication, integrity, and digital signatures with the help of certificate authority (CA). The owner of the data, which is to be uploaded in the cloud environment, has to register with CKMS by creating an account with login credentials. The same login credentials can be used even to upload the list of users and their permissions with whom the uploaded file can be shared. CKMS uses asymmetric encryption to generate both public and private key pair; the private key is used to encrypt the data and the public key is divided into two parts by CKMS: one part of it is transmitted to the owner, and the rest is kept with CKMS. To a certain extent, CKMS system supports “one-to-many” encryption, but it does not support fine-grained access control. However, key management relies on CA and secure channel. Once the public key is obtained in the insecure channel transmission, or CA is on CA and secure channel. Once the public key is

The tuCP-ABE Scheme belongs to the CP-ABE scheme. Aiming at the existing security problems of cloud computing, the tuCP-ABE Scheme provides a secure encryption strategy. The key is generated and managed by two authorities central authority (CA) and outsourcing authority (OA). If users want to obtain data, they need to cooperate with two authorities CA and OA to generate the decryption key. As the user can access the data after the identity authentication, the tuCP-ABE Scheme has high security. A single CA or OA can neither get the decryption key, nor decrypt the data. To some extent, it is difficult to collude. The tuCP-ABE Scheme is a solution suitable for fine-grained access control in cloud storage systems, and it has high security. However, this scheme does not involve the user who provides the data and the user who is responsible for the data encryption. Key negotiation only involves two authorities and the user who requests the data. The user who provides the data cannot control the data provided by himself. If the user who provides the data cannot control who can access the data, the security is meaningless. In a real environment, to decrypt a small amount of data, two authorities are needed to generate the key, which is a waste of
resources. In other words, the scheme relies on a trusted third party and the efficiency of key generation is low. Data integrity is an important part of data security and privacy, including key integrity, ciphertext integrity, and message integrity.

Blockchain-based ciphertext-policy attribute-based encryption scheme (BCAS) stores the hash of the key, the hash of the ciphertext, and the hash of the decrypted data in the blockchain. On one hand, the malicious user cannot obtain the ciphertext, the key, and the initial data through the hash. On the other hand, it can be used for data integrity verification. When the DO uploads the encrypted data to the cloud storage, he or she needs to upload the hash of the ciphertext and the hash of the initial data to the blockchain. The system will verify the ciphertext and the hash of ciphertext, and the data can be uploaded successfully only after the data verification is passed. After DR requests the data through the data label, it verifies whether the encrypted data are damaged. Then, the key needs to be generated by DO and DR. Then, each party uploads its own hash of the key, and both parties confirm the key in the blockchain. After the key generation, DR decrypts the encrypted data and then submits the hash of the decrypted data to compare with the initial data hash to verify the integrity of the decrypted data.

tuCP-ABE scheme verifies the integrity of the key, Wei et al.27, and Block-Secure scheme verifies the integrity of the encrypted data, in which Block-Secure scheme divides the data into blocks and provides a better replacement strategy for the file copy. CKMS system does not talk about integrity verification.

The results of comparison are shown in Table 1. FGAC means that it supports fine-grained access control, and O2M means that it supports “one-to-many” encryption, one user to encrypt and upload data, and multiple users to decrypt with their own independent key. TTP-free means that it does not rely on the trusted third party, SKIV is short for Secret Key Integrity Verification, CTIV is short for Ciphertext Integrity Verification, and MIV is short for Message Integrity Verification.

### Our contributions
This article is distinct from other research works in related fields. We combine the advantages of blockchain and ABE and propose a BCAS. The main contributions of this article are as follows:

We have defined the concept of cloud data security sharing and its important indicators clearly. On this basis, we propose a BCAS scheme for cloud data security sharing.

In our scheme, DOs have the authority to decide who can decrypt the data. DOs encrypt the data and upload the encrypted data to the cloud environment through the blockchain system (BCS). Data requesters (DRs) must negotiate the key through the BCS that can be used to decrypt data with the DO.

In our scheme, all operations of users need to be conducted in the BCS. The operations are retained in the blockchain permanently, and all records are tamper-proof.

Our proposed scheme has the characteristic of “one-to-many” encryption. Data are only encrypted once. Users who meet the attribute requirements can negotiate the key with the DO.

In our scheme, the global public parameters are initialized by the setup mechanism in the BCS, and the users negotiate the key in the BCS without relying on the trusted third party.

We prove that our proposed scheme is secure based on Discrete Logarithm (DL) assumption and decisional $q$ parallel-bilinear Diffie–Hellman exponent (BDHE) assumption.

We compare and analyze our scheme with the other four schemes in terms of “one-to-many” encryption, integrity verification, and so on. According to the comparison, we find that our scheme is superior to other schemes.

We compared our scheme with the tuCP-ABE scheme through four experiments. When the number of the messages is the same, with the increase in attributes, the time and storage of the whole process in our scheme are obviously less than those in tuCP-ABE scheme. For different processes with the same attribute, the time and storage of party key generation step and decryption key generation step are obviously better than those of tuCP-ABE scheme. What’s more, DOs in our scheme have the right to decide who can decrypt the data, which would achieve the real security.

### Table 1. Comparison of schemes.

| Scheme       | FGAC | O2M | TTP-free | SKIV | CTIV | MIV |
|--------------|------|-----|----------|------|------|-----|
| CKMS         |      |     |          |      |      |     |
| Wei et al.   |      |     |          |      |      |     |
| Block-Secure |      |     |          |      |      |     |
| tuCP-ABE     | ✓    | ✓   | ✓        | ✓    | ✓    | ✓   |
| BCAS         | ✓    | ✓   | ✓        | ✓    | ✓    | ✓   |

BCAS: blockchain-based ciphertext-policy attribute-based encryption scheme.
Organization
The rest of this article is organized as follows. In section “Background,” we introduce the definition of cloud data security sharing and related assumptions of our scheme including DL assumption and decisional q parallel-BDHE assumption. Then, we propose the BCAS and introduce transaction structure and process in section “Architecture design.” We introduce a specific encryption and decryption scheme for BCAS in section “Our construction of BCAS scheme.” In section “Security and performance analysis,” we proof the security of our scheme, and present the performance analysis by experiment. Finally, we conclude this article in section “Conclusion.”

Background
Definition 1: cloud data security sharing. Cloud data security sharing focuses on data security storage and sharing in the cloud storage platform (CSP). DO can upload the encrypted data. Other users who want to decrypt the data need to obtain a unique decryption key with the consent of the DO. In other words, the data only need to be encrypted once, and the users who meet the requirements have their own unique key to decrypt. DO has its own absolute control over the uploaded data and is also responsible for its own uploaded data. All the actions of users will remain unchanged and cannot be tampered with. Users cannot deny their actions. In general, to realize cloud security sharing, we need to achieve the following:

Support Fine-grained Access Control: support fine-grained access control policy, DO encrypts the data, and other users can negotiate with the DO to generate a unique decryption key after meeting the attribute requirements.

TTP-free: the operation of key management and data storage does not depend on the third party. The third party cannot interfere with the generation of key and data encryption of the DO.

Integrity: the key, ciphertext, and plaintext should have integrity. Once one integrity is destroyed, the user cannot get the correct decrypted data.

DO Authority: DO have control over their uploaded data. The cloud platform is an open platform, and everyone may access the encrypted uploaded data. However, if users want to decrypt the data, users must negotiate the key that can be used to decrypt data with the DO. Otherwise, users cannot decrypt the data.

Behavior Record: whether it is DOs or DRs, all operations need to be preserved, and the operations cannot be tampered with or forged. Users cannot deny their actions. Prevent users from performing malicious operations.

Definition 2: bilinear maps. g is a prime-order bilinear group generator which takes a security parameter λ as input and outputs prime-order bilinear groups (G, GT, e, p), where p is a prime, G, GT are the two cyclic groups of order p, and e : G × G → GT is a bilinear maps. The bilinear map e has the following properties:

Bilinearity: ∀g, h ∈ G and a, b ∈ Zp, we have e(ga, hb) = e(g, h)ab.

Non-degeneracy: ∃g ∈ G such that e(g, g) has order p in GT.

Computability: e can be efficiently computed.

Definition 3: DL problem. Inputting g a, where g ∈ G, a ∈ ℤp, the DL problem is to compute a.

Assumption 1: DL assumption. The advantage of an algorithm A in solving the DL problem is defined to be

\[ \text{Adv}_{\text{DL}}(A) = \Pr[A(g, g^a) = a : g, g^a \leftarrow \mathcal{R}G] \]

We say that G satisfies the DL assumption if Adv_{DL}(A) is a negligible function of security parameter λ for any polynomial algorithm A.

Definition 4: decisional q parallel-BDHE problem. Inputting

\[ \overline{s} = g, g^a, g^{a^2}, \ldots, g^{a^q} \]

\[ g^{ab}, g^{a(b/h)}, g^{a((r+1)/h)}, \ldots, g^{a((r+q)/h)} \]

∀1 ≤ j ≤ q

\[ g^{(ab_j/h)}, \ldots, g^{a(b_j/h)} \]

the decisional q parallel-BDHE problem is to distinguish e(g, g)^(a(r+1)/h) from a random element in GT, where \( g \in G, a, s, b_1, \ldots, b_q \in ℤ_p \).

Assumption 2: decisional q parallel-BDHE assumption. The advantage of an algorithm A in solving the decisional q parallel-BDHE problem is defined to be

\[ \text{Adv}_{\text{BDHE}}(A) = \Pr[A(\overline{s}, e(g, g)^{ah}) = 1] \]

We say that G satisfies the decisional q parallel-BDHE assumption if Adv_{BDHE}(A) is a negligible function of security parameter λ for any polynomial algorithm A.
Architecture design

We proposed a new cloud secure storage and sharing scheme for a series of security problems in cloud computing, namely, BCAS. Compared with the existing cloud security schemes, the proposed scheme is more advantages:

1. Authority of DO: DOs have the authority to decide who can decrypt the data.
2. Tamper-Proof: operations of users will be retained permanently in the blockchain.
3. Fine-grained Access Control: we have developed the CP-ABE scheme to achieve the secure data sharing.
4. “One-to-Many” Encryption: the data are encrypted by DO only once.
5. TTP-free: the scheme does not use any TTP throughout the execution of the scheme.

Architecture

In this section, we present an architecture for cloud storage and sharing called BCAS. As it is shown in Figure 1, the architecture consists of four entities like CSP, BCS, DO, and DR.

DO and DR are different identities merely for different options on cloud. A user is a DO when uploading data and a DR when requesting to download data. When the user is used for different identity, the corresponding ID is different:

1. DO is an identity of the user which encrypts and uploads the data into the CSP and has full control over it.
2. DR is an identity of the user which would like to accesses and decrypts data stored in CSP.
3. BCS is a data and user management system based on blockchain. Each operation of the user will be stored in the blockchain as a transaction. Ensure the integrity and credibility of transactions using the tamper-proof of blockchain. BCS plays an important role in our architecture, which provides confidentiality (encryption and decryption), authentication of an identity, data authentication, integrity, User authority control, key management, and storage of user operation records. BCS will reduce the malicious operation of users, malicious operation will remain in BCS permanently, and users will be punished accordingly. Blockchain is a distributed database which structure as shown in Figure 2, \( T_i \) \( (i = 1, 2, 3, \ldots) \) is a specific user operation, including viewing, accessing, registering, uploading, downloading and other operations as well as accessing users and timestamps.
4. CSP is an open storage platform, which is only used to store encrypted data. Other operations are managed through BCS.

The whole process that DR requests the encrypted data uploaded to DO through BCS and obtains the decryption key to decrypt the data successfully is called transaction in our architecture. A complete transaction includes the following contents:

First, users who want to upload or download data in the cloud environment must provide identity certificate information to register identity with BCS, which generates independent data upload identity and data download identity for users. Then, when DO needs to submit its own data upload identity to upload

![Figure 1. BCAS architecture.](image-url)
data. After passing the BCS verification, DO needs to submit the data encrypted by ABE technology and relevant information. After passing the BCS verification information, BCS will save the relevant information and user’s operations in the blockchain and upload the encrypted data to the cloud. Finally, if DR wants to access the data in the cloud, it needs to provide the data download identity. After BCS verifies the information is legal and the DO agrees the request of DR, DO and DR negotiates the key, which is used for DR to decrypt the data. After BCS verification, DR can applies for the data to the BCS, and then DR uses the key to decrypt the data.

A specific transaction is all or part of a complete transaction. BCS will store the contents of the transaction and its timestamp on the blockchain. If DR determines that the data uploaded by the DO are illegal, the blockchain can provide relevant evidence to prove that, if a user accesses the data in the cloud storage space maliciously, it will also be recorded in the blockchain. For users who often maliciously operate, the blockchain will store the user’s data If Id is added to blacklist, the user’s operation permission in cloud storage will be restricted or even punished.

**Transaction structure**

A complete transaction shall include the identity of both parties, data-related information, key, operation, and corresponding timestamp. The structure of the transaction is shown in Figure 3.

It should include the following:

**IDInfo.** IDInfo is the identity information of the transaction user, including the identity of the data uploader and the identity of the DR. When users perform different operations, they need to provide the corresponding identity. The storage format of identity in the transaction is as follows:

```
"IDInfo": {
  "data owner": "IDInfo of data owner",
  "data requester": "IDInfo of data requester"
}
```

**Data.** Data include the hash, address, and label of the encrypted data. The encrypted data are uploaded to the cloud storage. The hash value and label of the data ciphertext are stored in the blockchain. The label is
used by the user to find the data. The address is the storage address of the encrypted data in the cloud storage. The data are downloaded from the cloud storage to the system. The hash is used to verify whether the corresponding data of the label are complete. The format of data is as follows:

```
"data": {  
"data Hash": "Hash of raw data",
"ciphertext hash": "Hash of encrypted data",
"address": "the address of encrypted data in cloud storage",
"label": []}
```

**Key.** The key stores the public key of both parties and the hash value of the final key. The final key hash is used to ensure the integrity of the key. If the hash values obtained by both parties are different, the final key needs to be discussed again; otherwise, the data cannot be decrypted:

```
"key": {  
"pk_DO": "the public key of DO",
"pk_DR": "the public key of DR",
"DO_SK_DRS_hash": "the hash value of SK_DRS provided by DO",
"DR_SK_DRS_hash": "the hash value of SK_DRS provided by DR"
}
```

**Operations and timestamps.** Operations and timestamps include all operations of users in the blockchain, including user identity registration, key upload, data request, key negotiation, and so on. All operations in a transaction are listed in the order of timestamps:

```
"Operations & timestamps": {  
"ts1": "op1",
"ts2": "op2",
"ts3": "op3",
"ts4": "op4",
......
}
```

**Transaction process**

**Global setup.** GlobalSetup (λ) → GPP: the global setup algorithm is executed by BCS. BCS generates the global public parameter GPP and publishes it on the blockchain. If users want to upload or download data in the cloud, they will receive the GPP. Users use GPP to generate the corresponding key.

**Identity register.** IDRegister (ID) → (IDInfoDO, IDInfoDR): the ID of users register algorithm is run by BCS. All users who want to upload or download data in the cloud need to register identity information in the blockchain. Users include DO and DR. BCS will generate unique identity information IDInfoDO and IDInfoDR for different identities of users. The blockchain will store the user’s register operation and the hash value of IDInfoDO and IDInfoDR. Users need to provide their own upload or download data, which can be followed up only after the BCS audit.

**User key generation.** UKeyGen (GPP) → (SK, PK): the user key generation algorithm is run by user. Each user who wants to upload or download data in the cloud needs to generate his own public–private key pair (SK, PK) for authentication and encryption/decryption key generation, where SK means the secret private key and PK means the public key. The public–private key pair of DO is (SKDO, PKDO), and the public–private key pair of DR is (SKDR, PKDR). To ensure security, SK used by the same user in different identities should be different, and the corresponding PK should also be different. The SK of the user is kept by himself, and the PK is disclosed in the whole network.

**Encryption and upload.** Encrypt (GPP, SKDO, (W, ρ), M) → C(W, ρ): the encryption algorithm is run by DO. According to the given plaintext M, and access structure (W, ρ), DO encrypts the data with secret private key SKDO and GPP, uploads the ciphertext, and provides the corresponding label for other users to access. BCS will store the hash of the ciphertext, the corresponding label, and the address in the cloud storage.

In the upload process, the blockchain will record the DO’s upload data application and operation. DO provides their own IDInfoDO for the blockchain. After the authentication is passed, the encrypted data will be uploaded into CSP. Once the user is reported to upload the malicious data, the corresponding data in the CSP will be cleared, but the upload record will be permanently saved. It will affect the user’s own permissions and ensure that the vast majority of users upload data legally. In order to ensure that the data are wrong or tampered with in the process of transmission process, DO also needs to upload the hash value of the data. When the block link receives the encrypted data uploaded by the user, it needs to verify the integrity of data first. If it does not meet the requirements, it needs to be uploaded again.

**Decryption key generation.** DecryptKeyGen (GPP, (SKDO, PKDO), (SKDR, PKDR), S, θ) → SKDR,S: the decryption key generation algorithm is run by DO, DR, and BCS. DR needs to submits IDInfoDR to BCS to verify
identity. When request data from DO, then DO determines the identity of DR. DR uses key $SK_{DR}$ to generate signature $L$, and submits identity and attributes together with signature $L$ to DO. DO verifies the signature $L$, causing DO to confirm DR’s identity. After confirming the identity, DO generates part of the key, respectively, DO sends the key by secret channel to DR, and DR summarizes all the keys as the final decryption key. In terms of the correctness of the final decryption key, DO and DR calculate the hash value and verify it in the BCS.

Decryption. Decrypt $(W,S,SK_{DR},S,C_{(W,ρ)}) → M$: the decryption algorithm is executed by DR. DR obtains the necessary additional decryption parameters for decryption, downloads the encrypted data from the cloud, and decrypts the data with the decryption key and decryption parameters.

Our construction of BCAS scheme

According to the structure and specific process we proposed, we designed a detailed BCAS Scheme.

Global parameter setup

BCS chooses the security parameter $\lambda$, and generates two cyclic groups $G, G_T$ of prime order $p$, $g, g_1$ are the two random generators of $G$. $e : G \times G \rightarrow G_T$ is a bilinear map. For each attribute $atti \in U$, BCS chooses randomly $t_i \in Z_p$, and lets $T_i = g^{t_i}$. The algorithm picks two secure hash functions $H_i : Z_p \times G \times \{0,1\}^d \rightarrow G, H_2 : Z_p \times G \times G \times \{0,1\}^d \times G \rightarrow Z_p$, where $d$ represents the bit length of the identity information.

The global public parameter $GPP$ is $(p, G, G_T, e, g, g_1, \{T_i\}_{atti \in U}, H_1, H_2)$. $GPP$ is published in the BCAS system, and all nodes in the network will receive parameters.

Identity register

Users register their personal identity in BCS, and submit their identity information to BCS. Users get two identities $IDInfo_{DO}$ and $IDInfo_{DR}$ generated by BCS.

User key generation

User randomly selects elements $α_1, α_2, α_3, x_{DR} \in Z_p$. The user is DO if the user uploads the data to the cloud storage. DO chooses $α_1, α_2, α_3 \in Z_p$ as his private key $SK_{DO}$, and publishes $(P_1 = e(g, g)^{α_1}, P_2 = e(g, g)^{α_2}, P_3 = g^{α_3})$ as his public key $PK_{DO}$. The user is DR if the user downloads the data from the cloud storage. DR chooses $x_{DR} \in Z_p$ as his private key and publishes $PK_{DR} = g^{x_{DR}}$ as his public key. The public key of the user is published throughout the network.

The process of user identity registration and key generation is shown in Figure 4.

Encryption and upload

Given the message $M$ (raw data that users need to encrypt and share) and the access structure $(W, ρ)$, where $W$ is a $l \times k$ matrix, the ciphertext $C_{(W, ρ)} = (C, C', \{C_i\}_{i \in [l]})$ is generated according to Algorithm 1.

Figure 5 shows the data encryption and upload operation. After the message $M$ is encrypted and the personal information is verified, the ciphertext $C_{(W, ρ)}$, the data label, the message hash (MH), and the ciphertext hash (CH) are submitted by DO to BCS. Then, BCS verifies the ciphertext and the CH. If verification is correct, the ciphertext and the label are uploaded to the cloud. All operations of DO, CH, and MH are retained in the blockchain.

Decryption key generation

DR needs to do the following to access the data uploaded by DO:
Figure 5. Data encryption and upload.

Algorithm 1. Ciphertext generation algorithm.

**Input:** the message \( M \), the access structure \((W, \rho)\).

**Output:** the ciphertext \( C_{(W, \rho)} = \left( C, C', \{ C_i, D_i \}_{i \in [l]} \right) \).

1. DO chooses a random vector \( \vec{v} = (r_1, r_2, \ldots, r_l) \in \mathbb{Z}_p^l \) and random elements \( r_1, \ldots, r_l \in \mathbb{Z}_p \).
2. DO computes \( C' = g', C = M e(g, g)^{r_1} e(g, g)^{r_2} \)′.
3. For \( i = 1 \) to \( l \), DO computes \( C_i = g_{W, \rho}^{D_i} D_i = g_i \).
4. The ciphertext \( C_{(W, \rho)} = \left( C, C', \{ C_i, D_i \}_{i \in [l]} \right) \).

After the calculation is done, DO and DR will upload the hash value of the key, respectively, to verify the correctness of the key.

The process of DO and DR negotiating to generate the DR decryption key is shown in Figure 6.

**Decryption**

DR requests and obtains encrypted data \( C_{(W, \rho)} = \left( C, C', \{ C_i, D_i \}_{i \in [l]} \right) \) from BCS. For the encrypted data, DR is obtained \( \{ W_i : i \in I \} \) such that \( \sum_{i \in I} W_i = (1, 0, \ldots, 0) \), where the index set is \( I = \{ i : att_{(i)} \in S' \} \). If the attribute set \( S' \) satisfies the access structure \((W, \rho)\), DR can recover the message as follows

\[
M = \sum_{i \in I} \left( e(C_i, K_{h^2} + \theta) e(D_i, K_{h^2} + \theta) \right)^{w_i} / e(K', C'h_{h^2}) e(K', C^h)
\]

**Security and performance analysis**

**The proof of security**

The confidentiality (IND-CPA security) of the proposed scheme can be proved directly based on the decision \( q \) parallel-BDHE assumption. We denote the CP-ABE scheme by Waters\(^{10}\) as WCP-ABE. For simplicity, we will reduce the security of the proposed scheme to that of WCP-ABE scheme.

**Lemma 1.** We say the WCP-ABE scheme is IND-CPA secure if the (decision) \( q \) parallel-BDHE assumption holds.

**Proof.** The details of proof are referred to Waters.\(^{9}\)

**Lemma 2.** If DO is a trusted data provider, the confidentiality of the proposed BCAS scheme can be reduced to that of the WCP-ABE scheme.
The Decryption Key Generation algorithm is as follows:

1. DR selects a one-time random $r_{DR} \in \mathbb{Z}_p$ and computes $L = h_{1}^{r_{DR}}$, where $h_{1} = H_{1}(r_{DR}, PK_{DR}, IDInfo_{DR})$ and sends $(r_{DR}, L, h_{1}, PK_{DR}, IDInfo_{DR}, S)$ secretly to DO.

2. DO verifies the correctness of $L$ by $e(L, g) = e(h_{1}, PK_{DR})$, where $h_{1} = H_{1}(r_{DR}, PK_{DR}, IDInfo_{DR})$. If it holds, DO searches $(IDInfo_{DR}, *, *, *)$ in list $L_{DO}$. Else, he stores $(IDInfo_{DR}, L, PK_{DR}, S, h_{2})$ in $L_{DO}$, where $h_{2} = H_{2}(P_{DO}, L)$. DO chooses $\theta, s \in \mathbb{Z}_p$, and computes

$$K = g^{\theta}, K' = g^{\theta \cdot h_{1} \cdot g^{-1}}, K'' = g^{\theta \cdot h_{1} \cdot s \cdot g^{-1}}, \{K_{i} = T_{i}^{\alpha} \}_{\alpha \in \mathbb{G}}$$

DO sends $(\theta, K, K', K'', \{K_{i} \}_{\alpha \in \mathbb{G}})$ to DR secretly. DR computes $h_{2} = H_{2}(P_{DO}, L)$. Then, DR obtains the decryption key as follows

$$SK_{DR,S} = (h_{2}, \theta, K, K', K'', \{K_{i} \}_{\alpha \in \mathbb{G}})$$

If there is an adversary $A$ that can break BCAS scheme with advantage $Adv_{A}$, we can construct an adversary $B$ against WCP-ABE with advantage $Adv_{B}$ such that $Adv_{B} = Adv_{A}$. The construction is as follows, where $C$ simulates the queries both from $A$ and $B$.

**Setup.** $C$ gives $B$ the WCP-ABE public parameters $(p, G, G_{T}, e, g, e(g, g)^{p}, g_{1}, g^{u}, \{T_{i} \}_{\alpha \in \mathbb{G}})$. $C$ randomly selects $\alpha_{1}, \alpha_{2}, \alpha_{3} \in \mathbb{Z}_{p}$ and gives $(g, g^{u}, e(g, g)^{p}, g_{1}, \{T_{i} \}_{\alpha \in \mathbb{G}})$ to $A$ as public parameters of the BCAS scheme, which implicitly sets $\alpha_{1} = \alpha_{2} = \alpha_{3} = \alpha$.

**Phase 1.** $A$ submits a attribute set $S$ with identity $U$ for DecryptKeyGen oracle. $C$ generates $\hat{K} = g^{\theta}, \hat{K}' = g^{\theta \cdot h_{1} \cdot g^{-1}}, \hat{K}'' = g^{\theta \cdot h_{1} \cdot s \cdot g^{-1}}, \{\hat{K}_{i} = T_{i}^{\alpha} \}_{\alpha \in \mathbb{G}}$ by running the KeyGen algorithm (or a KeyGen oracle queried by $B$) of WCP-ABE. Then, $C$ computes

$$K = (\hat{K})^{1/\theta} = g^{u/\theta}, K' = K'' = (\hat{K}')^{1/\theta} = g^{u/\theta}, \{K_{i} = (\hat{K}_{i})^{1/\theta} = T_{i}^{\alpha/\theta} \}_{\alpha \in \mathbb{G}}$$

and returns $SK_{DR,S} = (h_{2}, \theta, K, K', K'', \{K_{i} \}_{\alpha \in \mathbb{G}})$ to $A$, where implicitly sets $h_{2} = \theta$.  

**Figure 6.** Decryption key generation.
Challenge. A submits two messages $M_0, M_1$ of equal length along with target access structure $W$. $C$ flips a random coin $b \in \{0, 1\}$ and generates the BCAS challenge ciphertext $C_{W^b} = (C, C', \{C_i, D_i\}_{i \in [l]})$ of $M_b$. $C$ returns $C_{W^b}$ to $A$ and sends $((\sqrt{e(g, g)}^{aq}e(g, g)^{aq}), C', \{C_i, D_i\}_{i \in [l]})$ to $B$.

Phase 2. Same with Phase 1.

Guess. $A$ outputs a guess bit $b'$, then $B$ outputs the same $b'$ for WCP-ABE.

From the above simulation, the distributions of the public parameter, decryption keys, and challenge ciphertext are indistinguishable from the real scheme, we have $Adv_B = Adv_A$.

Lemma 3. A curious-but-honest illegal user cannot decrypt any ciphertext.

Proof. Here, we give a heuristic analysis. A curious-but-honest illegal user needs to get $r$ to decrypt ciphertext $M(e(g, g)^{a_1}e(g, g)^{a_q})^y$.

$$M = e(g, g)^{a_1}e(g, g)^{a_q}$$

However, it is related to DL to compute $r$. Under the DL assumption, the illegal user cannot decrypt any ciphertext.

Theorem 1. If the (decision) q parallel-BDHE assumption holds, then the proposed blockchain-based CP-ABE scheme is IND-CPA secure.

Proof. It follows from Lemmas 1–3.

Comparison and analysis

We make a deeper comparison between our scheme with tuCP-ABE scheme.

Efficiency. We compare the efficiency of our scheme with that of tuCP-ABE scheme. The main difference between the two schemes lies in the key generation process. In order to guarantee the security, tuCP-ABE scheme choose two third parties CA and OA to assist users in generating encryption and decryption keys. CA and OA will not collude with each other to ensure the security of the keys. The interaction process is shown in Figure 7.

Our BCAS scheme uses blockchain instead of two authorities, which can simplify the interaction process and improve the efficiency of key generation. To facilitate the comparison, we build Table 2 to compare the process of key generation between the two parties.

In the actual application scenario, the encryption data are uploaded to the cloud first. If DR finds the file and wants to access the file, DR can negotiate the key with DO to decrypt the data. Therefore, in our scheme, we first encrypt and upload the data, and then negotiate the key. However, in tuCP-ABE scheme, the key is negotiated before the encryption of data. In order to facilitate the comparison, we exchange the order of our encryption and key generation, which has no impact on our specific comparison results:

Step 1: GlobalSetup. Generate global public parameters. The two schemes use similar GPP. In the first step, the efficiency of the two schemes is similar.

Step 2: IDReg. User identity registration, the tuCP-ABE scheme does not mention how to generate the ID. Compared with the following key generation processes, the ID generation process is relatively simple. We think the efficiency of the two schemes is similar.

Step 3: ParKeyGen. The key generation phase of the participants. In BCAS scheme, users generate their own public key/private key pair. This process is equivalent to that in tuCP-ABE, CA and users generate their own public key/private key pair, respectively. In this step, the efficiency of the two schemes is similar.

Step 4: DecKeyGen. Generate decrypt key. In BCAS scheme, DO and DR generate the decryption key. The operation of DO is equivalent to the operation of CA in tuCP-ABE scheme, and the operation of DR is equivalent to the operation of U in tuCP-ABE scheme. Compared with tuCP-ABE scheme, BCAS scheme saves the negotiation between CA and OA, and directly calculates the final key. In addition, BCAS scheme has less calculate $D, K''$ than tuCP-ABE scheme, and the calculation process of $K'$, $K''$ is also simplified. In the process of calculating the key, the parameters and the calculation are reduced. So, in this step, the efficiency of BCAS is higher than tuCP-ABE scheme.

Step 5: Encrypt. In this step, both BCAS scheme and tuCP-ABE scheme are encrypted, but the ciphertext generation process and the ciphertext are different. BCAS scheme has a smaller amount of calculation than tuCP-ABE scheme, BCAS scheme does not need to calculate $C'' = g^{a_q}$, the calculation process of $C$ is simplified. So, in the encryption step, the efficiency of BCAS scheme is higher than that of tuCP-ABE scheme.

Step 6: Decrypt. In this step, both BCAS scheme and tuCP-ABE scheme are decrypted. The decryption process of BCAS scheme is relatively simpler, mainly
because BCAS scheme reduces bilinear pairing operation. It has higher efficiency.

In addition to the efficiency difference in the above steps, there are four secret transmissions between users and third parties in tuCP-ABE scheme. In our scheme, it only takes two times. With the decrease in secret transmissions, the security dependence of the transmission channel decreases. In the process of key generation, tuCP-ABE scheme needs CA to calculate the parameters involved in generating intermediate key and ciphertext. As it is faster in the process of encryption and decryption, our scheme is more efficient in actual use.

**Figure 7.** Process for tuCP-ABE scheme.

**Table 2.** Comparison between BCAS and tuCP-ABE.

| Step | BCAS               | tuCP-ABE               |
|------|--------------------|------------------------|
| 1    | GlobalSetup        | GlobalSetup            |
| 2    | IDReg              | IDReg                  |
| 3: ParKeyGen | UKeyGen           | CA/OA Setup & USignKeyGen |
| 4: DecKeyGen | DecryptKeyGen     | UDecryptKeyGen          |
| 5    | Encrypt            | Encrypt                |
| 6    | Decrypt            | Decrypt                |

BCAS: blockchain-based ciphertext-policy attribute-based encryption scheme.
Storage. To ensure the security, tuCP-ABE scheme uses two trusted third parties CA and OA to help users in generating the key and generates multiple intermediate keys in the process of generating the final decryption key. Compared with BCAS scheme, the final key generated by tuCP-ABE scheme has more parameters, longer key length, longer ciphertext, and larger storage space. BCAS scheme simplifies the process and parameters, so the final key and ciphertext take up less memory.

In our BCAS scheme, the decryption key obtained by DR is
\[ SK_{DR,S} = (h_2, \theta, K, K', K'' , \{ K_i \}_{a \in \mathcal{S}}) \]
while the user's key in tuCP-ABE scheme is
\[ SK_{U,S} = (r_2, \theta, h_2, K, D, K', K'' , \{ K_i \}_{a \in \mathcal{S}}) \]
our key is more compact and occupies less memory space.

In our BCAS scheme, the ciphertext of DO encryption is: \( C_{(W,p)} = (C, C', \{ C_i, D_i \}_{i \in \mathcal{I}}) \), where the data corresponding to the relevant characters in the ciphertext are as follows
\[
C = M(e(g,g)^{a_1} e(g,g)^{a_2})^r, \\
C' = g'^r, C_i = g_1^{w_i} T_{\rho_i}^{-r_i} D_i = g_i^{-r_i}
\]

The encrypted ciphertext in tuCP-ABE scheme is:
\[ CT_{(W,p)} = (C, C', C'', \{ C_i, D_i \}_{i \in \mathcal{I}}) \]
where the data corresponding to the relevant characters in the ciphertext are as follows
\[
C = M(e(g,g)^{a_1} e(g,g)^{a_2} e(g,g)^{a_3})^r, \\
C' = g'^r, C'' = g_i^{r_i}, C_i = g_1^{w_i} T_{\rho_i}^{-r_i} D_i = g_i^{-r_i}
\]

When the same plaintext \( M \) is being encrypted, the memory of \( C \) in tuCP-ABE scheme is larger than that in BCAS scheme, and it takes additional memory space for the ciphertext to store \( C'' \).

DO authority. In tuCP-ABE scheme, DO is not mentioned. In the process of key generation, DR negotiates key with CA and OA. To a certain extent, the DO has no controlling authority over the data, and the control authority over the data is in CA and OA. Although the security of the tuCP-ABE scheme is high, this data security can only ensure that malicious users cannot obtain the key for data decryption, but the DO may not have the authority to manage the access authority of other users. Even if CA and OA audit the DR with the authorization of the DO, but the DO cannot control the decryption key, it may lose the control of the data to some extent. In this way, data security will lose its most fundamental significance: only when the DO has the right to authorize other users to access the data, without the authorization of the DO, it is safe to access the plaintext.

In our scheme, first, the user’s identity is authenticated through the BCS. In the key negotiation process, DO and DR negotiate the key. DO has legal control over own data. DO can not only know who will access the data but also control who can decrypt the data. Only the DR who is authenticated by the DO. Only in this way can we negotiate with the DO to obtain the decryption key and decrypt the data. That is, our scheme realizes the DO’s control authority to the data provided by himself on the premise of ensuring security.

Experimental results. To evaluate the difference of performance between the proposed BCAS scheme and tuCP-ABE scheme, we implement the BCAS scheme and the tuCP-ABE scheme in C with the PBC library Microsoft Visual C++ 6.0. We run the experiments on a Windows 10 system which is equipped with a quad-core Intel CPU and 8 GB RAM.

Figure 8(a) and (b) shows the comparison of performance between two schemes of the same \( M \), attributes, and access structure in different steps. Figure 8(a) shows that BCAS scheme takes less time than tuCP-ABE scheme, and Figure 8(b) shows that BCAS scheme consumes less storage space than tuCP-ABE scheme, especially in Step 3 (ParKeyGen) and Step 4 (DecKeyGen). BCAS scheme has better performance than tuCP-ABE scheme by reducing the interaction and computation of the third party.

Figure 9(a) and (b) shows the comparison of performance between two schemes of the same \( M \) in different attributes. Figure 9(a) shows that with the increase in attributes, the runtime of whole process in tuCP-ABE scheme is longer than that in BCAS scheme, and Figure 9(b) shows that with the increase in attributes, the storage space of whole process in tuCP-ABE scheme is larger than that in BCAS scheme.

Conclusion
In this article, we propose the BCAS cloud data sharing security scheme without any trusted third party. Using blockchain to authenticate users’ identities and control their access authority, we achieve the cloud security sharing without a trusted third party. Users need to register their identity to access, upload, and download the data in CSP. In our architecture, blockchain plays an important role. It guarantees the traceability of user’s operations and the verification of data integrity through its own characteristics. In this scheme, the data in the CSP will not be accessed by illegal visitors unless the user is authenticated. In addition, our scheme combines the CP-ABE scheme to protect the data security...
in the CSP and enables the DOs to control their own data. Finally, we also prove the security of the scheme and analyze the security of the model. The experimental results show that our scheme has better performance than the contrast scheme. The BCAS can protect the user’s private key and support fine-grained access control policy. Compared with the previous methods, BCAS scheme does not rely on the cloud service providers or the trusted third parties. It can protect the authority of the DO and achieve the secure data sharing. All the operations are not be tampered and the security is better guaranteed.

Declaration of conflicting interests
The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding
The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported by the Key-Area Research and Development Program of Guangdong Province (grant no. 2019B010137002) and the Natural Science Foundation of Fujian, China (grant no. 2020J01171).

ORCID iD
Yuting Zuo https://orcid.org/0000-0002-2783-8351

References
1. Ali M, Khan SU and Vasilakos AV. Security in cloud computing: opportunities and challenges. Inform Sciences 2015; 305: 357–383.
2. Senyo PK, Addae E and Boateng R. Cloud computing research: a review of research themes, frameworks, methods and future research directions. *Int J Inform Manage* 2018; 38(1): 128–139.

3. Deshpande P, Sharma SC, Peddoju SK, et al. Security and service assurance issues in Cloud environment. *Int J Syst Assur Eng Manage* 2018; 9(1): 194–207.

4. Malathi PS and Yanathithi D. A study of data storage security issues in cloud computing. *Bonfr Int J Softw Eng Soft Comput* 2019; 9(2): 5–7.

5. Rath M. Security challenges and resolution in cloud computing and cloud of things. In: Kecskemeti G (ed.) Applying integration techniques and methods in distributed systems and technologies. Hershey, PA: IGI Global, 2019, pp.79–102.

6. Pradeep KV and Vijayakumar V. Survey on the key management for securing the cloud. *Procedia Comput Sci* 2015; 50: 115–121.

7. Hegde N and Manvi SS. Thesis proposal summary: key management authentication and non repudiation for information transaction in vehicular cloud environments. In: *Proceedings of the IEEE international conference on cloud computing in emerging markets (CCEM)*, Bangalore, India, 19–21 October 2016, pp.157–160. New York: IEEE.

8. Ethelbert O, Moghaddam FF, Wieder P, et al. A JSON token-based authentication and access management schema for Cloud SaaS applications. In: *Proceedings of the IEEE 5th international conference on future Internet of Things and cloud (FiCloud)*, Prague, 21–23 August 2017, pp.47–53. New York: IEEE.

9. Rajarajeswari S and Somasundaram K. Data confidentiality and privacy in cloud computing. *Indian J Sci Technol* 2016; 9(4): 1–8.

10. Sahai A and Waters B. Fuzzy identity-based encryption. In: *Proceedings of the IEEE conference on theory and applications of cryptographic techniques*, Aarhus, 22–26 May 2005, vol. 3494, pp.457–473. Berlin: Springer.

11. Bethencourt J, Sahai A and Waters B. Ciphertext-policy attribute-based encryption. In: *Proceedings of the IEEE symposium on security and privacy (SP’07)*, Berkeley, CA, 20–23 May 2007, pp.321–334. New York: IEEE.

12. Nakamoto S. Bitcoin: a peer-to-peer electronic cash system, 2008, https://www.bitcoin.org

13. Yli-Huumo J, Ko D, Choi S, et al. Where is current research on blockchain technology?—A systematic review. *PLoS ONE* 2016; 11(10): e0163477.

14. Kan J, Chen S and Huang X. Improve blockchain performance using graph data structure and parallel mining. In: *Proceedings of the 1st IEEE international conference on hot information-centric networking (HotICN)*, Shenzhen, China, 15–17 August 2018, pp.173–178. New York: IEEE.

15. Gai K, Guo J, Zhu L, et al. Blockchain meets cloud computing: a survey. *IEEE Commun Surv Tut* 2020; 22(3): 2009–2030.

16. Xu H, Cao J, Zhang J, et al. A survey: cloud data security based on blockchain technology. In: *Proceedings of the IEEE 4th international conference on data science in cyberspace (DSC)*, Hangzhou, China, 23–25 June 2019, pp.618–624. New York: IEEE.

17. Zhang Y, Xiang Y and Zhang LY. Cloud computing security. In: Zhang Y, Xiang Y and Zhang LY (eds) *Secure compressive sensing in multimedia data, cloud computing and IoT*. Singapore: Springer, 2019, pp.63–82.

18. Lv J, Wang Y and Liu J. A security problem in cloud auditing protocols. In: *Proceedings of the international conference on machine learning, big data and business intelligence (MLBDBI)*, Taiyuan, China, 8–10 November 2019, pp.43–46. New York: IEEE.

19. Vijayakumar V, Priyan MK, Ushadevi G, et al. E-health cloud security using timing enabled proxy re-encryption. *Mobile Netw Appl* 2019; 24(3): 1034–1045.

20. Taylor ME and Aboagye-Darko D. Security approaches and crypto algorithms in mobile cloud storage environment to ensure data security. In: *Proceedings of the international conference on artificial intelligence and security*, New York, 26–28 July 2019, pp.516–524. Berlin: Springer.

21. Bacis E, Di Vincenzo SDC, Foresti S, et al. Securing resources in decentralized cloud storage. *IEEE T Inf Foren Sec* 2019; 15: 286–298.

22. Zhang Q, Wang X, Yuan J, et al. A hierarchical group key agreement protocol using orientable attributes for cloud computing. *Inform Sciences* 2019; 480: 55–69.

23. Harn L and Hsu CF. A practical hybrid group key establishment for secure group communications. *Comput J* 2017; 60(11): 1582–1589.

24. Bicakci K, Yavuz DD and Gurkan S. TwinCloud: secure cloud sharing without explicit key management. In: *Proceedings of the IEEE conference on communications and network security (CNS)*, Philadelphia, PA, 17–19 October 2016, pp.581–585. New York: IEEE.

25. Kumar A, Lee BG, Lee HJ, et al. Secure storage and access of data in cloud computing. In: *Proceedings of the international conference on ICT convergence (ICTC)*, Jeju, Korea, 15–17 October 2012, pp.336–339. New York: IEEE.

26. Pradeep KV, Vijayakumar V and Subramaniyaswamy V. An efficient framework for sharing a file in a secure manner using asymmetric key distribution management in cloud environment. *J Comput Network Comm* 2019; 2019: 9852472.

27. Wei PC, Wang D, Zhao Y, et al. Blockchain data-based cloud data integrity protection mechanism. *Future Gener Comp Sy* 2020; 102: 902–911.

28. Li J, Wu J and Chen L. Block-secure: blockchain based scheme for secure P2P cloud storage. *Inform Sciences* 2018; 465: 219–231.

29. Goyal V, Pandey O, Sahai A, et al. Attribute-based encryption for fine-grained access control of encrypted data. In: *Proceedings of the 13th ACM conference on computer and communications security*, Alexandria, VA, 30 October–3 November 2006, pp.89–98. New York: ACM.

30. Yu G, Wang Y, Cao Z, et al. Traceable and undeniable ciphertext-policy attribute-based encryption for cloud storage service. *Int J Distrib Sens N*. Epub ahead of print 4 April 2019. DOI: 10.1177/1550147719841276.