A revocable attribute-based access control system using blockchain

Xin Liu1,2*, Yue-guang Zheng3,4 and Xu-zhou Li5

1 School of Information Engineering, Shandong Youth University of Political Science, Jinan, 250103, China
2 Key Laboratory of Information Security and Intelligent Control in Universities of Shandong (Shandong Youth University of Political Science), Jinan, 250103, China
3 Lianlao Distributed Storage Intelligent Industry Technology Research Institute (Shandong) Co. Ltd, Jinan, 250014, China
4 Shenzhen Zhongkai Xinlian Blockchain Co., Ltd, Shenzhen, 518052, China
5 Shandong Youth University of Political Science, Jinan, 250103, China
*Corresponding author’s e-mail: lx@sdyu.edu.cn

Abstract. Attribute-based access control is an effective cryptographic mechanism that allows a data owner to perform attribute-level access control over users’ access capabilities. In the application background of blockchain, it is urgent to use attribute-based access control to solve a series of security problems. Although there exist several similar schemes, they do not consider attribute revocation. By adopting the attribute revocation technique based on binary trees, we extended Waters’ ciphertext policy attribute-based encryption (CP-ABE) scheme into a revocable CP-ABE (RCP-ABE) scheme. By combining our RCP-ABE with blockchain, we constructed a revocable attribute-based access control system. The new system supports expressive access control policies and allows the attribute authority to revoke users’ attributes or part of their attributes. The security analysis shows that the new system satisfies several ideal properties, i.e., forward security, backward security, confidentiality, and integrity.

1. Introduction
Cloud storage service is a typical application of cloud computing technology, which allows governments, enterprises, and individuals to outsource their business data to cloud service providers, thus significantly reducing the cost of data management and maintenance. Attribute-based access control [1, 2] is a promising cryptography mechanism, which aims to use attribute-based encryption (ABE) to solve a series of security issues in the field of cloud storage services. In other words, data owners can use attribute-based access control systems to tackle the following main problems: (1) How to prevent cloud service providers and unauthorized users’ access to sensitive data. (2) How to use attribute-based policies to constrain the access rights of legitimate users.

Blockchain is a distributed chained storage structure constructed by cryptographic hash functions. It realizes the immutability and traceability of on-chain data through collision-resistant hash functions. At the same time, it introduces consensus mechanisms to ensure the consistency of on-chain data [3]. With the popularity of blockchain applications, more and more data are stored in the blockchain. Therefore, when data owners use blockchain to achieve data sharing, the issue of fine-grained access...
control on data in the blockchain also needs to be solved. Recently, several blockchain-oriented data access control systems have been proposed. In [3], Wang et al. put forward a secure electronic health record system by combining ABE with blockchain and provided the application of their system in the field of insurance claims. In [4], Wang et al. proposed a distributed data access control system that used a consensus mechanism to select some nodes from the blockchain nodes to act as attribute authorities. In [5], Lu et al. proposed a data access control framework based on ABE and blockchain. Moreover, they provided performance analysis in the consortium chain environment based on Hyperledger Fabric. In [6], Qiu et al. proposed an access control scheme based on blockchain. Their scheme was obtained by extending the Fabric-CA module in the super ledger. However, a main disadvantage of the above schemes is that they do not consider the issue of attribute revocation, i.e., it should allow the attribute authority to revoke all or part of a malicious user’s attributes while ensuring that the access rights of other users who share the same attributes with the malicious user are not affected.

1.1. Our contribution
By extending Waters’ ciphertext policy ABE (CP-ABE), we put forward a revokable CP-ABE (RCP-ABE). Then, we constructed an attribute-based access control system based on the new RCP-ABE scheme and blockchain. Compared with existing similar systems, our system has the following advantages: (1) It supports the flexible LSSS (Linear Secret Sharing Scheme) type access policy. (2) It allows the attribute authority to revoke all or part of users’ attributes and meet the ideal forward security and backward security. (3) It inherits the unique decentralization, immutability, and traceability of blockchain.

1.2. Organization
In Section 2, we introduced the key techniques used in this article. In section 3, we propose a new RCP-ABE scheme that supports attribute revocation. In section 4, we constructed a new attribute-based access control system based on our RCP-ABE and blockchain. In section 5, we provide the security analysis and performance analysis. Finally, we summarize in section 6.

2. Preliminaries

2.1. Attribute-based encryption
 Attribute-based encryption [7] is a kind of 1-to-many public key encryption. In such a scheme, the attribute authority generates attribute decryption keys for users based on their attributes. A data owner can define an attribute-based access policy and use it to encrypt sensitive data that they want to share. A user can decrypt successfully if he owns a set of attributes that meet the access policy defined by the data owner.

2.2. Attribute revocation technique based on binary trees
In [8], Hur et al. proposed an attribute revocation technique based on binary trees. To implement attribute revocation, a specific revocation authority \( RA \) needs to be introduced. \( RA \) forms attribute groups based on attributes (for example, all users with attribute \( \tau \) form group \( G_\tau \)). Meanwhile, \( RA \) selects the group key \( GK_\tau \) for \( G_\tau \). \( GK_\tau \) will be used to blind the elements (corresponding to the attribute \( \tau \) ) in users’ attribute decryption key and ciphertext so that users can successfully execute decryption only if they own \( GK_\tau \). For each user \( U \) in \( G_\tau \), \( RA \) distributes \( GK_\tau \) in the following steps: (1) Build a binary tree called key encrypting key (KEK) tree so that each user corresponds to a different leaf node. (2) Select a KEK for each node. (3) Select the minimum cover set of \( G_\tau \) (i.e., the nearest common ancestors of all user nodes in \( G_\tau \)), and set the KEK of these nodes as the KEK of \( G_\tau \), denoted as \( KEK(G_\tau) \). (4) Generate a unique \( PathKey_\tau \) for each user \( U \) in \( G_\tau \), which consists of the KEKs of each node located on the path from the root node to the user node. (5) Provide each user in
with $K_EK(G_t)$ and corresponding $PathKey_t$, through a secret channel. Meanwhile, publish the ciphertext $\{Enc_k(GK_t)\}_{k \in KEK(G_t)}$, where $Enc()$ represents a symmetric encryption algorithm. (6) When a user is removed from $G_t$, $RA$ re-selects the group key $GK'_t$ and the minimum cover set of $G_t$. At the same time, $RA$ updates the KEKs corresponding to these nodes and provides the updated $KEK'(G_t)$ and the corresponding path keys to all non-revoked users in $G_t$ through a secret channel. In addition, $RA$ updates the ciphertext $\{Enc_k(GK_t)\}_{k \in KEK(G_t)}$ to $\{Enc_k(GK'_t)\}_{k \in KEK(G_t)}$. Obviously, for a revoked user, he cannot decrypt $\{Enc_k(GK'_t)\}_{k \in KEK(G_t)}$ with the previous $KEK(G_t)$ and path key.

2.3. Blockchain and inter planetary file system
Blockchain is essentially a chained storage structure, which makes use of a redundancy mechanism to make multiple blocks store the same data [4]. At the same time, blockchain adopts the mechanism of “the minority is subordinate to the majority” to ensure the consistency of data, i.e., to resist the 51% attack [9]. Since this kind of attack is almost impossible to succeed in a distributed blockchain structure, users’ data in blockchain can be considered secure. Inter planetary file system (IPFS) is a peer-to-peer distributed file system, which can make up for the disadvantage that blockchain is not suitable for storing large volumes of business data. When data owners store their files in the IPFS system, they can get a storage address returned by the system. Thereafter, they encrypt the address by using CP-ABE and the resulting ciphertexts can be published in blockchain.

3. A new revocable CP-ABE scheme
We assume that the scheme in this section is constructed on the group pair $(\mathbb{G}_1, \mathbb{G}_2)$, and there exists a bilinear mapping $\hat{e}$, s.t., $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are prime cyclic groups of order $p$. In addition to $RA$ and users, there are two types of participants, where $AA$ represents an attribute authority and $DO$ represents a data owner.

3.1. Setup
$AA$ performs the following steps:
(1) Define the attributes universe $\mathcal{U} = \{1, 2, ..., |\mathcal{U}|\}$.
(2) Select $g, h_1, ..., h_{|\mathcal{U}|} \in \mathbb{G}_1$, select $\alpha, \alpha' \in \mathbb{Z}_p$, and calculate $\hat{e}(g, g)^\alpha, g^\alpha$.
(3) Publish $PK = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, p, \mathcal{U}, g, h_1, ..., h_{|\mathcal{U}|}, e(g, g)^\alpha, g^\alpha)$ and keep $MK = g^\alpha$ secretly.

3.2. Key generation
Phase 1: Decryption key generation
In this phase, $AA$ generates the decryption key $SK_S$ for user $U_i$ who owns an attribute set $S$. The specific steps are as follows:
(1) $U_i$ provides $S$ to $AA$.
(2) $AA$ selects $t \in \mathbb{Z}_p$ and calculates $K = g^a g^{\alpha t}, L = g^i$. For each attribute $\tau \in S$, $AA$ computes $K_\tau = h_\tau^i$. Finally, $AA$ sends $SK_S$ to $U_i$.

Phase 2: Key encrypting key generation
$AA$ provides $RA$ with the composition of each attribute group $G_t(\tau \in \mathcal{U})$. $RA$ creates a KEK tree and generates the $KEK(G_t)$ and a $PathKey_t$ for each user $U_i$ in $G_t$ in the manner described in section 2.2. The $KEK(G_t)$ and $PathKey_t$ are then sent to $U_i$ over a secret channel.
3.3. Data outsourcing
Assume that DO wants to share some file $M \in \mathbb{G}_r$. DO defines the following access policy: users’ attribute set must satisfy the access structure $(M, \rho)$ embedded in the ciphertext, where $M$ represents an $l \times n$ matrix and $\rho$ represents the mapping used to map the $i$th row of $M$ to the attribute $\rho(i)$. DO performs the following steps:

1. Select $s, y_1, \ldots, y_n \in \mathbb{Z}_p^*$ and define a random vector $\vec{v} = (s, y_2, \ldots, y_n)^T \in \mathbb{Z}_p^n$. For $i = 1, \ldots, l$, calculate $\lambda_i = M \vec{v}$. Compute $C = \alpha \cdot e(g, g)^{\lambda_i}, D = g^{\lambda_i}$.

2. Provide the ciphertext $CT = (\langle M, \rho \rangle, C, C', \{(C_i, D_i)\}_{i=1}^l)$ to RA.

3.4. Data encryption
Before distributing $CT$, RA performs the following steps:

1. For each attribute group $\mathcal{G}_r(\in \mathcal{U})$, select a group key $GK_r \in \mathbb{Z}_p^*$. 

2. Parse $CT$ into $\langle (\langle M, \rho \rangle, C, C', \{(C_i, D_i)\}_{i=1}^l) \rangle$. For each $\mathcal{G}_r$ (satisfying $\rho(i) = r$), update the ciphertext element $D_i = g^{\lambda_i}$ into $D'_i = (g^{\lambda_i})^{GK_r}$, thus obtaining the updated ciphertext $CT' = \langle (\langle M, \rho \rangle, C, C', \{(C_i, D_i')\}_{i=1}^l) \rangle).

3. Select the minimum cover set of $\mathcal{G}_r$ in the KEK tree to obtain the set $KEK(\mathcal{G}_r)$ composed of these nodes’ KEKs.

4. Generate a header message $Hdr = \{\forall \mathcal{G} \in \mathcal{U}: Desc(\mathcal{G}) \cap \{Enc_K(GK_r)\}_{K \in KEK(\mathcal{G})}\}$, i.e., for each attribute $\mathcal{G}$ in $S$, use each key $K$ in $KEK(\mathcal{G})$ to perform symmetric encryption on $GK_r$, so that any user in $\mathcal{G}$ who has not been revoked must have a key $K$ in $KEK(\mathcal{G})$, and use it to recover $GK_r$ from the ciphertext set $\{Enc_K(GK_r)\}_{K \in KEK(\mathcal{G})}$.

3.5. Data decryption
Assume that a user $t_U$ has the decryption key $SSK$ and they want to access the shared file. For this end, $t_U$ performs the following steps:

1. Receive the $(Hdr, CT')$ provided by RA and parse $CT'$ into $CT' = \langle (\langle M, \rho \rangle, C, C', \{(C_i, D_i')\}_{i=1}^l) \rangle$. Then, check if $S$ satisfies $(\langle M, \rho \rangle)$, and if not, terminate.

2. For each attribute $r$ in $Hdr$, find the ciphertext $\{Enc_K(GK_r)\}_{K \in KEK(\mathcal{G}_r)}$ corresponding to $\mathcal{G}_r$ by using $desc(r)$. Calculate $KEK^* = PathKey_{r} \cap KEK(\mathcal{G}_r)$ and use it to decrypt $\{Enc_K(GK_r)\}_{K \in KEK(\mathcal{G}_r)}$, thus recovering $GK_r$.

3. Update $SSK_s = (K, L, \{K_r\}_{r \in \mathcal{S}})$ into $SSK_s = (K, L, \{K_r = (K_r)^{Enc_K} \}_{r \in \mathcal{S}})$.

4. Define $I = \{i: \rho(i) \in S\}$, and calculate the set of integers $\{\omega_i\}_{i \in I}$, s.t., $(\omega_1, \omega_2, \ldots, \omega_n) \cdot M_I = (1, 0, \ldots, 0)_{n \times l}$, where $M_I$ represents the submatrix corresponding to $I$.

5. Compute $e(C', K) \prod_{i \in I} (e(C_i, L) e(D_i', K_r^{Enc}))^{\omega_i} = e(g, g)^{\omega_i}$ and output $M = C / e(g, g)^{\omega_i}$.

3.6. Key update
Assume that RA receives a notification from $AA$ that a user in $\mathcal{G}_r$ is revoked (i.e., they no longer own the attribute $\mathcal{G}_r$). At this point, RA does the following steps:
(1) Reselect a new group key $GK'_r$ for $G_r$. Reselect the minimum cover set of $G_r$ and the corresponding $KEK'(G_r)$ in the KEK tree. Then, redistribute the path key $PathKey_{U_i}$ for each unrevoked user $U_i$ in $G_r$.

(2) Select $s' \in \mathbb{Z}_p$ and update $(Hdr, CT')$, where

$$
Hdr = \{ \{ Enc_e(GK'_s) \}_k \in KEK(G_r) : \forall \tau \in S \setminus \{ \tau' \} : \text{desc}(\tau), \{ Enc_e(GK_s) \}_k \in KEK(G_r) \}, CT' = ((M, \rho),

$$

$$
C = M \cdot e(g, g)^{a(t + s')}, C_i = C_i \cdot H(\rho(i))^{s'}, D_i = (\langle D_i \rangle^{GK_{r(i)}} g^{s'})^{GK_{r(i)}} \cdot (g^{s'})^{GK_{r(i)}} \cdot \prod_{\tau \in S \setminus \{ \tau' \}} \langle C_i \rangle = C_i \cdot H(\rho(i))^{s'}, D'_i = D'_i \langle g^{s'} \rangle^{GK_{r(i)}} \cdot \prod_{\tau \in S \setminus \{ \tau' \}} \langle C_i \rangle = C_i \cdot H(\rho(i))^{s'}, D'_i = D'_i \langle g^{s'} \rangle^{GK_{r(i)}} \cdot \prod_{\tau \in S \setminus \{ \tau' \}} \langle C_i \rangle = C_i (K, L, K'_r = (K_r)^{GK_r}, K' = (K'_r)^{GK_r, r \in S \setminus \{ \tau' \}}).
$$

4. A new attribute-based access control system

In this section, we propose a new attribute-based access control system. Before providing the detailed description of our system, we define several symbols, which represent the algorithms or protocols of the RCP-ABE scheme in Section 3. Concretely, $Setup()$ represents the setup algorithm, $AKeyGen()$ represents the key generation protocol, $Encrypt()$ represents the encryption algorithm, $ReEncrypt()$ represents the re-encryption algorithm, $Decrypt()$ represents the decryption algorithm, $AUpdate()$ represents the attribute update algorithm. In addition, $Enc()$ and $Dec()$ represent a symmetric encryption algorithm and a symmetric decryption algorithm respectively.

4.1. Setup

$AA$ performs the $Setup()$ algorithm of the RCP-ABE scheme in Section 3.

4.2. Key generation

$U_i$ provides their attribute set $S$ to $AA$. Then, $AA$ and $RA$ respectively execute the $AKeyGen()$ algorithm of the RCP-ABE scheme in Section 3. Finally, $U_i$ obtains $SK_S$, $PathKey_{U_i}$, and $\{ KEK(G_r) \}_{r \in S}$. In addition, $AA$ publishes its ID, $U_i$’s ID, and the time of the generation of $U_i$’s decryption key to the blockchain.

4.3. Data outsourcing

Assuming that $DO$ wants to share a file $M \in G_r$, he performs the following steps:

(1) Select an encapsulated key $key$ and execute $C_i \leftarrow Enc(key, M)$. Store $C_i$ in the IPFS system, and get the storage address $Addr$ returned by the IPFS system.

(2) Define the access policy $(M, \rho)$, and execute $C_2 \leftarrow Encrypt((M, \rho), key || Addr)$. Then, calculate $Hfile \leftarrow H(M)$ and publish the information of $M$’s ID, $DO$’s ID, $(M, \rho)$, $Hfile$, and $C_2$ to the blockchain, where $H(\cdot)$ represents a collision-resistant hash function.

4.4. Data access

Assuming $U_i$ has $S, SK_S, PathKey_{U_i}$, and $\{ KEK(G_r) \}_{r \in S}$, they performs the following steps:

(1) Search the ciphertext $C_2$ from the blockchain, and check whether $S$ satisfies $(M, \rho)$. If not, terminate.

(2) Perform $key || Addr \leftarrow Decrypt(PK, SK_S, C_2)$, then download $C_i$ from the IPFS system according to $Addr$.

(3) Restore the file $M$, i.e., $M \leftarrow Dec(key, C_i)$. If $Hfile = H(M)$, publish their ID, $M$’s ID, their access time, and other information to the blockchain.
4.5. Attribute revocation
When receiving AA’s notification that a user has been revoked from the attribute group $G_i$, RA executes the $AUpdate()$ algorithm of the RCP-ABE scheme in section 3.

5. Security analysis and performance comparison
First, we analyze the security of our system. The details are as follows:

**Correctness.** Assume that $U_i$ owns the attribute set $S$ that satisfies $(\mathbf{M}, \rho)$. Also, for each $i \in I$, $U_i$ is affiliated with $G_i$ (satisfying $\rho(i) = \tau$) and owns $KEK(G_i), PathKey_{G_i}$. Given the cipher-text $C_2 = ((\mathbf{M}, \rho), C, C', \{(C_i, D_i)\}_{i \in I})$ and $Hdr$, we prove that $U_i$ can successfully decrypt $C_2$. Concretely, $U_i$ can firstly get $\{GK_{\rho(i)}\}_{i \in I}$ from $Hdr$ by decryption and then compute
\[
\hat{e}(C', K) = \prod_{i \in I} \left( \hat{e}(C_i, L) \hat{e}(D'_i, K'_{\rho(i)}) \right)^{\alpha_i}
= \prod_{i \in I} \left( \hat{e}(g^e, h^e_{\rho(i)}) \hat{e}(g^e, h'_{\rho(i)}) \hat{e}(g^e, h'_i) \hat{e}(g^e, h'_i) \right)^{\alpha_i}
= \prod_{i \in I} \left( \hat{e}(g^e, h^e_{\rho(i)}) \hat{e}(g^e, h'_i) \hat{e}(g^e, h'_i) \hat{e}(g^e, h'_i) \right)^{\alpha_i}
= \hat{e}(g, g)^{\alpha_i} \sum_{i \in I} \hat{e}(g, g)^{\alpha_i} = \hat{e}(g, g)^{\alpha_i}.
\]

Next, $U_i$ recovers the encapsulated key by calculating $key || Addr = C / e(g, g)^{\alpha_i}$. Finally, $U_i$ uses $key$ to decrypt the ciphertext $C_1$ and recover $M$.

**Forward security.** For a newly registered user, we assume that the ciphertext $C_2$ is generated after they have registered. It’s easy to see that as long as they own enough attributes, they can still decrypt $C_2$.

**Backward security.** As described in section 2.2, when a user $U^*$ is revoked from $G_i$ at time $t$, RA will generate the updated group key $GK'_{\tau}$ and perform updates on $Hdr$ and $C_2$. Because RA does not distribute $GK'_{\tau}$ to $U^*$, $U^*$ cannot decrypt $C_2$ and any newly published ciphertext using their decryption key (assuming that the access policy embedded in these ciphertext requires that $U^*$ must own the attribute $\tau$).

**Confidentiality.** Before publishing sensitive data to the blockchain, $DO$ encrypts it with our RCP-ABE scheme, which is obtained by adding an attribute revocation mechanism to the Waters CP-ABE scheme. According to the chosen-plaintext attack (CPA)-security of the Waters scheme, our RCP-ABE scheme also satisfies the CPA-security to ensure that sensitive data cannot be obtained by unauthorized users who do not meet the access policy $(\mathbf{M}, \rho)$.

**Integrity.** Our system satisfies integrity because $DO$ publishes the RCP-ABE scheme ciphertext $C_2$ of the sensitive file $M$ and the hash value $H\text{file}$ to the blockchain. In the data access phase, the user verifies whether $H\text{file} = \mathcal{H}(M')$ is satisfied for the decrypted file $M'$ and the hash value $H\text{file}$ downloaded from the blockchain. According to the collision-resistance of $\mathcal{H}(\cdot)$, this equation will not be satisfied if $M$ is tampered with.

Below, we provide a performance comparison between our system and multiple attribute access control systems (see Table 1). Both our system and the system in [3] support the LSSS type access policy. Both systems in [5] and [6] support access policy based on attribute tree. The two strategies are
equivalent. The system in [4] supports a threshold type access policy, so its application scenarios are limited. Currently, blockchain can be divided into three types, namely, public chain, consortium chain, and private chain. In addition to the fact that the system in [6] supports a consortium chain, the rest systems support a public chain. It can be seen from the comparison that our system is the only one that supports attribute revocation.

| System | The type of access policy | Attribute revocation | The type of access blockchain |
|--------|--------------------------|---------------------|-----------------------------|
| [3]    | LSSS                     | No                  | Public chain                |
| [4]    | Threshold                 | No                  | Public chain                |
| [5]    | Attribute tree            | No                  | Public chain                |
| [6]    | Attribute tree            | No                  | Consortium chain            |
| Ours   | LSSS                     | Yes                 | Public chain                |

6. Conclusions
We propose a revocable attribute-based access control system for the blockchain application environment. The advantage of our system is supporting flexible access policies and user revocation at the attribute level. At the same time, our system satisfies the following security properties, i.e., forward security, backward security, confidentiality, and integrity. In future research, we will define the security model of revocable attribute-based access control systems, describe the security games for each security property, design revocable attribute-based access control systems suitable for the consortium chain environment, and so on.

Acknowledgments
This work was supported in part by the Project of Shandong Province Higher Educational Science and Technology Program under Grant J17KA081, in part by the Undergraduate Education Reform Project of Higher Education in Shandong Province under Grant M2018X245, in part by the Scientific Research Project of Shandong Youth University of Political Science under Grant XJPY2021.

References
[1] Liu, X., Xu, Q., Zhang, B., et al.(2018) k-times attribute-based authentication scheme using direct anonymous attestation. Journal on Communications, 39: 113-133.
[2] Zhang, L., Xiong, H., Huang, Q., et al. (2019) Cryptographic solutions for cloud storage: challenges and research opportunities. IEEE Transactions on Services Computing, Doi: 10.1109/TSC.2019.2937764. (Early Access)
[3] Wang, H., Song, Y. (2018) Secure cloud-based ehr system using attribute-based cryptosystem and blockchain. Journal of medical systems, 42:1-9.
[4] Wang, J., Xie, Y., Wang, G., et al. (2021) A method of privacy preserving and access control in blockchain based on attribute-based Encryption. Netinfo Security, 20: 47-51.
[5] Lu, X., Fu, S. (2021) A trusted data access control scheme combining attribute based encryption and blockchain. Netinfo Security, 21: 7-14.
[6] Qiu, Y., Zhang, H., Cao, Q., et al.(2020) Blockchain data access control scheme based on CP-ABE algorithm. Chinese Journal of Network and Information Security, 2020, 6: 88-98.
[7] Waters, B.(2011) Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization. In: Proceedings of PKC 2011, Berlin, pp. 53-70.
[8] Hur, J., Noh, D. K. (2010) Attribute-based access control with efficient revocation in data outsourcing Systems. IEEE Transactions on Parallel and Distributed Systems, 22: 1214-1221.
[9] Ye, C., Li, G., Cai, H., et al. (2018) Analysis of security in blockchain: case study in 51%-attack detecting. In: Proceedings of DSA 2018, NewYork, pp.15-24.