Research on the Update Method of CP-ABE Access Control Strategy Based on Smart Contract

Yu Hao, Bo Cui(✉), Ru Li, Tingting Song, and Wenhan Hou

College of Computer Science, Inner Mongolia Key Laboratory of Wireless Networking and Mobile Computing, Inner Mongolia University, Hohhot, China
{31909020,31809125,cshwh}@mail.imu.edu.cn, {cscb, csliru}@imu.edu.cn

Abstract. The CP-ABE access control method based on blockchain realizes fine-grained access control of data and ensures the storage of ciphertext security. However, there are some problems in updating the access control policy. For example, the blockchain cannot be used as ample data storage. The ciphertext size increases with the number of attributes and the complexity of the access control policy, and the CP-ABE based on bilinear mapping is expensive to calculate. Therefore, this paper proposes dividing the ciphertext into data-related ciphertext and policy-related ciphertext, which are stored in blockchain and IPFS. It is worth noting that this paper uses the RSA-based CP-ABE encryption method, which can effectively reduce the computational cost of encryption and decryption and achieve a constant ciphertext size. In addition, we also systematically analyze and compare the advantages and costs generated by CP-ABE based on bilinear mapping and CP-ABE based on RSA. Through experimental analysis, security analysis, and formal analysis, compared with the existing access control policy update methods, the scheme proposed in this paper shows better performance when frequently updating the access control policy.

Keywords: Blockchain · CP-ABE · IPFS · Access Control · Policy update

1 Introduction

The ciphertext-policy attribute-based encryption (CP-ABE) [1] has attracted widespread attention as soon as it was proposed. The CP-ABE encryption method was initially used to store the ciphertext in a central server or cloud. However, there are some problems in traditional storage methods, such as single point of failure and trust crisis, which may cause the ciphertext to be lost and altered. Therefore, the combination of blockchain [2] and CP-ABE is used to achieve access control [3–8], which effectively solves the problems of traditional storage methods. But, blockchain is a tamper-proof distributed digital ledger. The old ciphertext cannot be deleted when an access control policy is

H. Gao et al. (Eds.): CollaborateCom 2022, LNICST 460, pp. 494–513, 2022.
https://doi.org/10.1007/978-3-031-24383-7_27

updated. The new ciphertext is continuously uploaded to the blockchain network, resulting in an increasing storage burden of blockchain. At the same time, as the ciphertext size increases, the update time also increases.

Researchers [9, 10] proposed reducing the time consumption and the size of the ciphertext when the access control policy is updated by decentralized storage of the ciphertext related to the data and the ciphertext related to the policy. But all data are stored in blockchain. Although the size of policy-related ciphertexts is reduced to reduce the consumption of update time, the storage burden of blockchain is not solved. Zhang et al. [11] suggested that data ciphertext should be stored in the cloud, and smart contract and CP-ABE should be combined to maintain an access control list. Although it can solve the burden of blockchain, the cloud is not trusted, and there will still be the threat of ciphertext tampering. Gao et al. [12] used the combination of InterPlanetary File System (IPFS) [13], which is tamper-proof, and blockchain to carry out the distributed storage of ciphertext. IPFS can effectively reduce the storage burden of the blockchain to realize the big data storage, make up for the defects of low storage efficiency and high cost of the blockchain, and avoid single points of failure and trust issues.

However, the CP-ABE encryption method is based on bilinear mapping in [12], and bilinear mapping is a complex calculation. Moreover, the time consumption of encryption and decryption and the ciphertext size is positively related to the number of attributes and complexity of the access control policy [14]. In other words, although scheme [12] relatively reduces the storage burden and update time consumption of blockchain, when the access control policy is frequently updated, the time consumed by encryption and decryption and the size of ciphertext generated will continue to increase with the increase of attributes and the complexity of the access control policy.

To further improve the performance of the access control policy updates, we propose to use the combination of IPFS and blockchain to reduce the burden of blockchain storage. Meanwhile, we use the RSA-based CP-ABE encryption method to avoid the time cost caused by bilinear mapping [15], effectively reducing the time consumption of encryption and decryption while achieving a constant ciphertext size. This method is more suitable for limited storage space, and where frequent access control policy updates are required. Besides, a Storage Smart Contract (SSC) and a Query Smart Contract (QSC) are designed to manage the new and old ciphertexts during the update process. Most importantly, we analyzed and compared CP-ABE based on bilinear mapping and CP-ABE based on RSA, and we conducted a safety analysis of the model and gave formal proof.

2 Related Work

CP-ABE can realize fine-grained access control, but traditional centralized ciphertext storage risks tampering and single point failure. Blockchain has a strong tamper-proof capability but lacks access control and privacy protection for the data. Literatures [3–8] proposed the integration of CP-ABE and blockchain technology to achieve secure data storage and fine-grained access control.

However, due to the limited storage capacity of blockchain, it is not suitable for ample data storage. To reduce the storage burden of blockchain, literature [16–21] proposed the

on-chain and off-chain collaborative storage modes. In this mode, the big data generated by encryption is stored in the cloud or IPFS, and the data attributes, indexes, access rights, and other contents are stored on the blockchain. But none of the above researches considers the issue of access control policy update. Reference [22–24] studied the update of access control policy. Mounnan et al. [22] uploaded the ciphertext to the cloud, and entrusted the cloud to update the policy by updating the key. Ma et al. [23] proposed to entrust a trusted third party to update the access control policy through a ciphertext conversion algorithm. The ciphertext and the transform ciphertext are stored on the blockchain. Guo et al. [24] proposed dividing the cloud into storage and computing clouds. The storage cloud is used to store the ciphertext, and the computing cloud is used to convert the ciphertext to update the access control policy.

The three access policy update methods above need to update or convert all ciphertexts when updating, which is computationally expensive. Therefore, literature [9–12] proposed to store data-related ciphertexts and policy-related ciphertexts separately. Only the policy-related ciphertext can be updated to reduce the calculation cost when the policy is updated. Tian et al. [10] divided ciphertext encrypted by CP-ABE into data ciphertext and policy ciphertext, stored in the transaction block and policy block, respectively, and only the policy block needs to be updated when the policy is updated. In [9, 11, 12], data encrypted in other encryption modes is stored off-chain, and its decryption key is encrypted by CP-ABE and stored on the blockchain. When the policy is updated, only the ciphertext formed by the decryption key needs to be updated. The difference is the storage location of the data ciphertext in these three schemes. Yu et al. [9] used the chameleon hash to construct a redactable key chain and a standard data chain to store the ciphertext of the decryption key and the data ciphertext, respectively. Zhang et al. [11] recommended using cloud storage to store the data ciphertext and a smart contract to generate the corresponding access control list. Gao et al. [12] suggested using the distributed IPFS system to store the ciphertext of the data, and the smart contract realizes the management of the ciphertext and users.

However, the CP-ABE encryption methods used in the current research are all based on bilinear mapping, which will lead to expensive calculation costs. At the same time, as the number of attributes in an access control policy and the policy complexity increase, the ciphertext size and calculation time consumption also increase. For this reason, it is necessary to find a more efficient CP-ABE encryption method and control the size of ciphertext to improve the performance of the access control policy update.

3 Overview of Our Scheme

This section will introduce the access control policy update scheme combining on-chain and off-chain, as shown in Fig. 1. The proposed method combines IPFS and blockchain, and applies the RSA-based CP-ABE algorithm and smart contracts. There are five prominent roles in this method: Data Owner (DO), Data Requester (DR), blockchain, IPFS, and Certificate Authority (CA). In this paper, the default trusted CA is fully trusted, and its function description is shown in Table 1. The whole process of method is divided into nine processes:

Table 1. Function description of main roles

Name	Functional description
DO	Encrypt data, upload ciphertext and policy ciphertext, deploy smart contracts, update access control policies, maintain data information, query historical records
DR	Query data, decrypt policy ciphertext and data ciphertext
IPFS	Store data ciphertext
Blockchain	Store policy ciphertext, compile, store and execute smart contracts
CA	Generate system parameters and distribute public and private keys of RSA-based CP-ABE algorithm

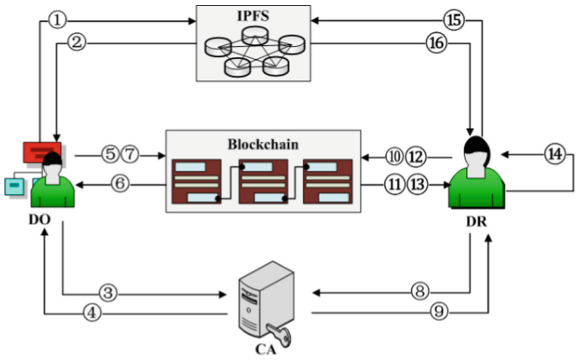

Fig. 1. Overall structure diagram.

- Upload data ciphertext: DO first encrypts the original data using symmetric encryption technology, then DO uploads the encrypted data ciphertext to IPFS. IPFS will calculate a hash value according to the content of the uploaded data ciphertext and return it to DO, where the hash value is used to find the uploaded data, corresponding to step 1 and step 2 in Fig. 1.
- Apply for public and private keys: CA first executes the initialization algorithm in the CP-ABE based on RSA and calculates the master public key (MPK) and the master secret key (MSK). When the DO request is received, the MPK used for encryption is sent to the DO, corresponding to steps 3 and 4 in Fig. 1.
- Upload policy ciphertext: DO designs an access control policy to limit DR access to the data. And then DO encrypts the hash value returned by IPFS, the decryption key of the data ciphertext, and the designed access control policy with MPK to form a policy ciphertext, and upload it to the blockchain to get the storage location of the transaction, corresponding to steps 5 and 6 in Fig. 1.
- Deploy smart contracts and upload data related information: DO first deploys SSC and QSC on the blockchain, then DO adds the relevant summary information of the data, the transaction location and the upload time of the policy ciphertext on the blockchain, as shown in step 7 in Fig. 1.

- Apply for a user private key: DR sends attributes to CA to apply for the user's private key. CA executes the key generation algorithm and sends the user private key corresponding to the user attribute to DR, as shown in step 8 and Step 9 in Fig. 1.
- Query policy ciphertext: DR retrieves whether the queried data exists by calling QSC. After receiving the query request, QSC calls SSC to query whether there is a record corresponding to the input information in the SSC. For example, to query if they have the same file name. If the record exists, the QSC returns the latest policy ciphertext storage location of the file to the DR according to the storage record of the SSC. Among them, only DO can choose to view the history of the policy ciphertext, corresponding to step 10 and step 11 in Fig. 1.
- Decrypt policy ciphertext: After receiving the storage location of the policy ciphertext returned by the QSC, the DR downloads the policy ciphertext from blockchain and decrypts the policy ciphertext with the user private key. Suppose the user's attributes satisfy the access control policy in the policy ciphertext. In that case, the policy ciphertext can be decrypted to obtain the IPFS's hash and the decryption key of the data ciphertext. as shown in steps 12 to 14 in Fig. 1.
- Decrypt data ciphertext: DR downloads the corresponding data ciphertext from IPFS through the IPFS's hash and decrypts the data ciphertext with the decryption key in the policy ciphertext to obtain the original data, corresponding to step 15 and step 16 in Fig. 1.
- Access control policy update: Only the DO can perform the update operation and update the relevant information of the policy ciphertext in the SSC. After DO decrypts the policy ciphertext, a new policy ciphertext is obtained by re-encryption. After it is re-uploaded to the blockchain, the new data information is published to the SSC. SSC first compares the data, if the data already exists, it will replace it and store the historical data in the historical information database. Otherwise, SSC will records the new information.

4 Implementation Details of Our Scheme

IN this section, the RSA-based CP-ABE algorithm used in this paper and other bilinear-mapping-based CP-ABE algorithms are firstly summarized and analyzed. Then, we will introduce the specific content of SSC and QSC.

4.1 Comparison of CP-ABE Based on RSA and CP-ABE Based on Bilinear Mapping

Different access structures are used to express the CP-ABE access policy based on the bilinear mapping. The most commonly used are tree, threshold, Linear Secret-Sharing Scheme (LSSS), and AND gates. The tree structure is the first proposed access structure. In the tree structure, the non-terminal nodes are composed of threshold gates, and the leaves are composed of attributes. LSSS is an improvement based on the tree structure, which is represented through a matrix. When forming the LSSS matrix, it makes the number of rows of the matrix equal the number of leaf nodes. In the access structure constructed by the threshold, the threshold t is set. (A, t) is used to encrypt the data so

that the data requester can decrypt the ciphertext only if t attributes are met in attribute A. In the AND gate structure, a string composed of 0 and 1 is used to express whether the attribute exists. We assume that the access control policy attribute set and user's attributes set, where if and if, and the same is true. Then, $P \subseteq A$ only if for all, so we can say that the attribute set A satisfies the access policy P only if $P \subseteq A$.

The access tree and LSSS matrix methods are more flexible and convenient in the expression of access control policy. However, there are many non-attribute nodes in the access tree structure, and for a complex access control policy, the access tree structure will become more prominent. Therefore, more computational overhead is undoubtedly increased in encryption and decryption. Although the LSSS matrix effectively reduces the number of non-attribute nodes and tree structure, it does not fundamentally solve the computational overhead caused by non-attribute nodes. The threshold and AND gate methods are weaker and more straightforward in structure. Because it only involves the values of related attributes, it solves the problem of computational consumption of accessing non-attribute nodes in the tree and LSSS matrix. So, the threshold method and the AND gate method have certain advantages in terms of key length, ciphertext size, and encryption and decryption time consumption. Some scenarios with many attributes have a better practical application effect. Therefore, the algorithm used in this paper adopts the AND gate method to construct the access control policy.

LSK represents length of user secret key and LCT represents length of cipher-text. We use |G| and |Gt| to represent the prime order pairing (the group G is multiplicative group ZN, where $N = pq$), and use |Gc| and |Gtc| to represent the composite order pairing. Besides, the n represents the number of universal attributes, the nA represents the average number of values assigned to each attribute in attribute set A and the L represents length of plain-text M [15] in Table 1. In Table 2, we use p and e denote paring computation and exponentiation. Meanwhile, S denotes the size of an access formula, |I| denotes the number of attributes in user's decryption private key, m denotes the upper bound of attribute number in universal attributes U and s denotes the number of attributes in the chosen attribute set U [26].

This paper makes an overall summary analysis. Table 2 and Table 3 summarize the relevant data of some bilinear map-based CP-ABE under different access structures. Although they are all based on bilinear mapping, encryption and decryption time consumption are different in different access structures. For example, during encryption, BSW needs to perform two exponentiations on each leaf node, HLR needs to perform m+t+1 exponentiation, and SYGH needs to perform $s + 3$ exponentiations and a pairing computation. Although some CP-ABE schemes do not need pairing computation when encrypting, the key point is calculating $e(g,g)^{\alpha}$ during decryption, where α is a random parameter selected in the pairing calculation during the setup or the encryption phase. In the decryption stage, HLR and EMNOS need three pairing computations, MSC needs two pairing computations, AC needs six pairing computations, and so on. In addition, in the BSW, CN, LOSTB, and Waters schemes, the number of pairing computations depends on the number of attributes. Each attribute must be paired during the decryption process. But bilinear mapping is also an expensive calculation method, almost three times the time consumption of other calculations [32].

Table 2. CP-ABE encryption method ciphertext and attribute private key size

Scheme	Access structure	LSK	LCT
BSW [1]	Tree	(2|A| + 1) |G|	(2|P| + 1) |G| + |Gt|
HLR [25]	Threshold	(n + |A|) |G|	2|G| + |Gt|
SYGH [26]	Threshold	(n + |A|) |G|	2|G| + |Gt|
GZCMZ [27]	Threshold	2n (n + |A|) |G|	2|G| + Gt
EMNOS [28]	(n-n) Threshold	2|G|	2|G| + |Gt|
LOSTW [29]	LSSS	(|A| + 1) |Gc|	(2|P| + 1) |Gc| + |Gtc|
Waters [30]	LSSS	(|A| + 1) |G|	(2|P| + 1) |G|+ |Gt|
LW [31]	LSSS	(3 + |A|) |Gc|	(2|P| + 2) Gt + Gtc
AC [32]	LSSS	(3|A| + 6) |G|	(3|P| + 3) |G| + |Gt|
MST [33]	LSSS	(|A| + 2) |G|	(|P| + 1) |G| + |Gt|
DJ [34]	AND gate	(nA|A| + 2) |Gc|	2|Gc| + |Gtc|
ZZCLL [45]	AND gate	(n + 1) |G|	2|G| + |Gt|
CN [36]	AND gates	2(|A| + 1) |G|	(|P| + 1) |G| + |Gt|
ZH [37]	AND gates	(|A| + 1) |G|	2|G| + |Gt|
GSWV [38]	AND gates	2|G|	(n-P + 2) |G| + |Gt| + L
YWGWDG [39]	AND gates	(|A| + 2) |G|	2|G| + |Gt|

Table 3. Encryption and decryption time of CP-ABE

Scheme	Encryption time	Decryption time	Scheme	Encryption time	Decryption time
BSW	(2S + 2)e	(|I|)p	EMNOS	3e	2p + 3e
HLR	(m + t + 1)e	$3p + O(t^2 + m)e$	Waters	(2S + 2)e	(|I|)p
SYGH	p + (s + 3)e	$2p + O(t^2 + s)e$	MST	(m + 1)e	2p
GZCMZ(CPA)	3e	2p + 2Se	DJ	3e	2p
GZCMZ(CCA)	6e	6p + (2S + 2)e	CN	(n + 2)e	(n + 1)p

The RSA-based CP-ABE encryption method eliminates the process of pairing calculation. Only exponentiation, multiplication, and hashing computation are involved in the encryption and decryption process. Therefore, the RSA-based CP-ABE used in our paper can effectively avoid the time consumption caused by complex paired calculation in the process of encryption and decryption. At the same time, the RSA-based CP-ABE achieves a constant ciphertext size, that this algorithm gets the attribute private key size is 2|G| and the ciphertext size generated is 3|G| + L. Although there are methods to

achieve constant ciphertext in the CP-ABE scheme based on bilinear mappings, such as HLR, SYGH, EMNOS, DJ, GZCMZ, and other schemes, as shown in Table 2. But these schemes consume more time for encryption and decryption. To sum up, RSA-based CP-ABE encryption has the advantages of minor time consumption and fixed ciphertext.

4.2 Sample Smart Contract Design

In order to implement the retrieval and management of the policy ciphertext after the access control policy is updated, two smart contracts with different functions are designed, which are SSC and QSC. SSC is responsible for recording data-related information, including data digest such as the file name, the position of transactions on the blockchain and upload time. When storing information in SSC, it is necessary to compare the uploaded data digest. SSC first calls the comparison function. Because the latest file update access control policy is more likely, this paper uses reverse order comparison can effectively save the time consumption of comparison. During the comparison, if there is a file with the same name as the newly uploaded file, the existing record will be stored in the historical information database. Then the new file position and the uploading time will be used to replace the original storage information of this file. If the same file name does not exist, a new record is added. It is important to note that DO can only call the SSC. QSC is responsible for the inquiry work. DR calls QSC to input the file name to be queried, and then QSC calls SSC to query whether the file name exists in SSC. If the file name exists, QSC returns the latest position of transactions on the blockchain corresponding to the file name. If not, it returns empty information. In addition, the DO has the right to call the SSC through the QSC to view the content in the historical information base.

Two data-related information structures, Massage and Record, are defined in SSC. Massage structure is used to store information related to the latest data, and Record structure is used to store historical data. Both structures contain three fields, namely Dataname, BCHash, and Time. Dataname is the summary information of the index data for the query. BCHash refers to the transaction position of the policy ciphertext in the blockchain. Time refers to the upload Time of the policy ciphertext to the blockchain. Meanwhile, two array variables, Mass and Historical, are defined for Massage and Record structures.

The Compare function is an internal function of the contract and can only be called by the SSC contract, as shown in Table 4. For the two input strings, first, compare whether the lengths of the strings are equal. If the lengths are not equal, return false directly. When the lengths are equal, compare the hash values of the strings. When the hash values of the two strings are the same, return true. Otherwise, return false.

The saveMassage function can only be executed by the contract creator, as shown in Table 5. That is, only the data owner DO can perform information management and update the policy ciphertext. To summarize the data uploaded by DO, the saveMassage function first calls the comparison function to perform a reverse order comparison to determine whether the information exists in the Message. If it does not exist, it means that the corresponding policy ciphertext is newly uploaded. The SSC contract records the new information (dataname, BChash, time) in the Massage for the freshly uploaded information. If the data exists, it means that the corresponding policy ciphertext has undergone an access control policy update operation. For the updated information, when

the same summary information is found, the original record is stored in the Record, and then replace the original record with the new transaction location and upload time to complete an access control policy update.

Table 4. Compare function in the SSC

Input	Summary information about two data points (string a, string b)
Output	True or False
1.	if bytes(a).length != bytes(b).length then
2.	return false
3.	else
4.	return keccak256(a) == keccak256(b)
5.	end if

Table 5. SaveMassage function in the SSC.

Input	Data related information (dataname, BChash, time)
Output	success or fail
1.	If msg.sender is not owner then
2.	throw exception ("Do not have permission")
3.	end if
4.	for compare in reverse order
5.	if compare (Massage. Dataname, dataname) = true
6.	Historical.push(Record (Dataname, BCHash, Time))
7.	Mass.BCHash = BChash
8.	Mass.Time = time
9.	end
10.	else
11.	Mass.push(Massage (dataname, BChash, time))
12.	end if
13.	end for

The QueryMassage function is called by the QSC contract to query through the input data digest information, as shown in Table 6. This function does not perform authentication, and the query location is the data in the Massage. The function returns matching data-related information by comparing the input data digest with the digest information stored in the Message (BCHash, Time).

The QueryRecord function is called by the QSC contract to query the information in the Record, but the function needs to judge the caller's identity during execution. Only the contract creator can call this function through the QSC to query the data history, as shown in Table 7.

In QSC smart contract, the main function is to call the query function in the SSC to query according to the information input by the user and return the query result to the user through the QSC, as shown in Table 8. Anyone can call Research functions. First, the querier enters the information to be queried. The QSC automatically calls

Table 6. QueryMassage function in the SSC

Input	Data summary information (dataname)
Output	Transaction storage location (BCHash)and Upload time (Time)

```
for compare in reverse order
  if compare (Mass. Dataname, dataname) = true then
    return (BCHash, Time)
    break
  end
  else
    throw exception (“No result is found”)
  end if
end for
```

Table 7. QueryRecord function in the SSC

Input	Data summary information (dataname)
Output	Transaction storage location (BCHash)and Upload time (Time)

```
If msg.sender is not owner then
  throw exception ("Do not have permission")
end if
for compare in Record
  if compare (Historical. Dataname, dataname) = true then
    return (BCHash, Time)
  end
    throw exception ("No result is found")
  end if
end for
```

the QueryMassage function in the SSC to query the information in the Massage and returns the query result to the querier. At the same time, the inquirer can choose to query the history records. Querying the history begins by determining whether the caller is the originator of the contract. If not, an exception is thrown. Otherwise, all history is returned.

Table 8. Research function in the QSC

Input	Data summary information (dataname)
Output	Transaction storage location (BCHash)and Upload time (Time)

```
function Research (dataname)
  Call the SSC QueryMassage function
  if Querying Historical Records
    Call the SSC QueryRecord function
  end if
end
```

5 Security and Performance Analysis of the Proposed Scheme

5.1 Security Analysis

The CP-ABE algorithm used in this paper is proposed in the literature [15]. The author of the literature has fully proved the security of this algorithm, so we will not repeat the proof of its safety. In this section, we analyze the security of the proposed model in terms of privacy, security, and access control and provide formal proof.

Privacy Analysis. This paper uses symmetric encryption and CP-ABE encryption to protect privacy. The data is symmetrically encrypted and stored in IPFS, while its decryption key is encrypted and stored in blockchain by CP-ABE. To obtain the decryption key of the data, DR can decrypt the policy ciphertext encrypted by CP-ABE only when his attributes meet the access control policy. The decryption key is used to decrypt the data ciphertext on IPFS to obtain the original data. therefore, the privacy of the data is well guaranteed.

Security Analysis. IN this paper, we adopt a combination of IPFS and blockchain to ensure the security of data storage, publishing, and sharing. The original data is encrypted and stored on IPFS, and the IPFS'S hash of the data and decryption key are encrypted by CP-ABE and then released to the blockchain. Meanwhile, DR needs to query the data by invoking a smart contract to obtain relevant information for data sharing. Both IPFS and blockchain are distributed technologies. IPFS splits data and stores it in different nodes and computes hash from the data content, while blockchain stores data in a block and connects with the previous block for hash calculation. The features of IPFS and blockchain make data tamper-proof, ensuring data integrity and security while avoiding single points of failure.

Access Control Analysis. IN this paper, CP-ABE is used to achieve fine-grained data access control. It associates ciphertext with access control policy and the key with attributes to achieve access control with the granularity that can be refined to the attribute level. DO can make different access control policies according to the attributes of the DR to achieve fine-grained access control. Meanwhile, because DO completely determines the access control policy, DO has complete access control over the data.

Formal Analysis. WE divide the model into two layers. the top layer includes four alternative transitions: *Upload Data Ciphertext*, *Upload Policy Ciphertext*, *Data Information Operation* and *Decrypt Ciphertext*, which corresponds to the four bottom models respectively. The four places *DO*, *DR*, *IPFS*, and *BC* represent four main roles. the top-level model is shown in Fig. 2. It is worth noting that during the modeling process in this paper, the default DO has obtained the MPK required for encryption from the CA, and the DR has obtained the user's private key from the CA.

The Upload data ciphertext layer's content includes DO encrypt the original data, uploading the data ciphertext to IPFS, and receiving the hash value, as shown in Fig. 3. In the process of uploading data ciphertext, the original data can only be uploaded to IPFS after symmetric encryption, and IPFS can only return the hash value to DO after the data ciphertext is uploaded successfully. The Upload data ciphertext subpage first

triggers the transition *symmetric encryption* to encrypt the data. The place *G1* is used to restrict only the original data to be encrypted to form the data ciphertext. After the data ciphertext is formed, the transitions *a1* and *b1* are fired. Then the transition *upload to IPFS* is fired, which satisfies the constraint that the data ciphertext can only be uploaded to IPFS after the data is encrypted. At this time, there is data ciphertext in IPFS. After the transition *c1* is fired, the transition *calculate the hash* and *Return Hash to DO* are fired successively to return the IPFS's hash value to DO, which satisfies the constraint that the hash value can be returned to DO only after the data ciphertext exists on IPFS.

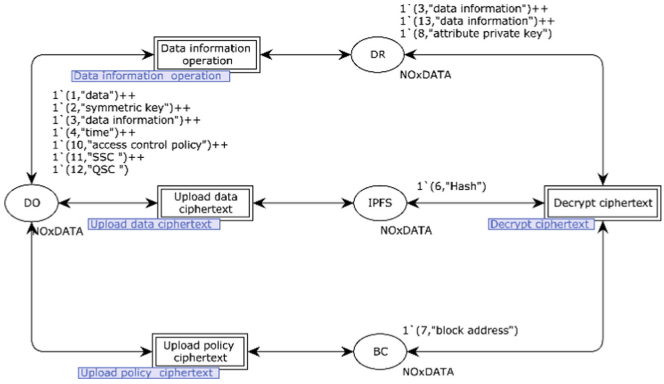

Fig. 2. The top layer of CPN model.

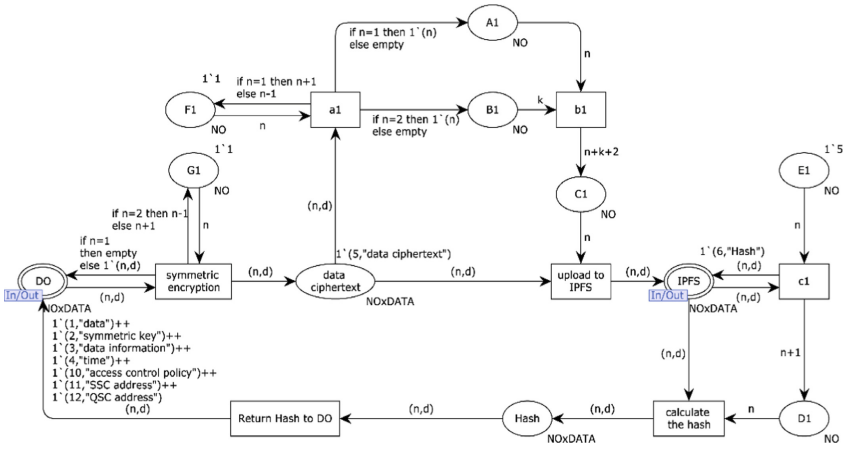

Fig. 3. The upload data ciphertext layer of CPN model.

The content of the Upload policy ciphertext layer includes DO uploading the policy ciphertext formed by RSA-based CP-ABE encryption to blockchain and obtaining the transaction position of the policy ciphertext in the blockchain, as shown in Fig. 4. Three constraints must be satisfied in this part. Firstly, RSA-based CP-ABE encryption can be performed only when the decryption key of the data ciphertext, the hash value returned by

IPFS, and the access control policy exist simultaneously. Secondly, it can be uploaded to the blockchain after encryption into the policy ciphertext. Finally, the transaction position can be returned only after the policy ciphertext is uploaded to the blockchain. When the DO receives the IPFS's hash in the Upload data ciphertext layer, the transition *blind1* is fired to bind the access control policy, decryption key, and IPFS's hash into a token. After the binding is completed, the transitions *CP-ABE encryption*, *upload to blockchain*, and *return block address to DO* are successively fired to complete the following operations, including RSA-based CP-ABE encryption, policy ciphertext upload to the blockchain, and return to the transaction location. Among them, places *A2*, *B2*, *G2*, and transitions *Blind1*, *C2* are used to restrict access control policy, decryption key, and IPFS's hash to be present simultaneously when RSA-based CP-ABE encryption is performed. Places *C2*, *D2* and transition *a2* are used to restrict only the policy ciphertext to be uploaded to the blockchain. Meanwhile, places *E2*, *F2* and transition *b2* are used to limit the return to the transaction position only after the policy ciphertext is uploaded to the blockchain. Through the restrictions of the above places and transitions, the Upload policy ciphertext layer satisfies the constraints.

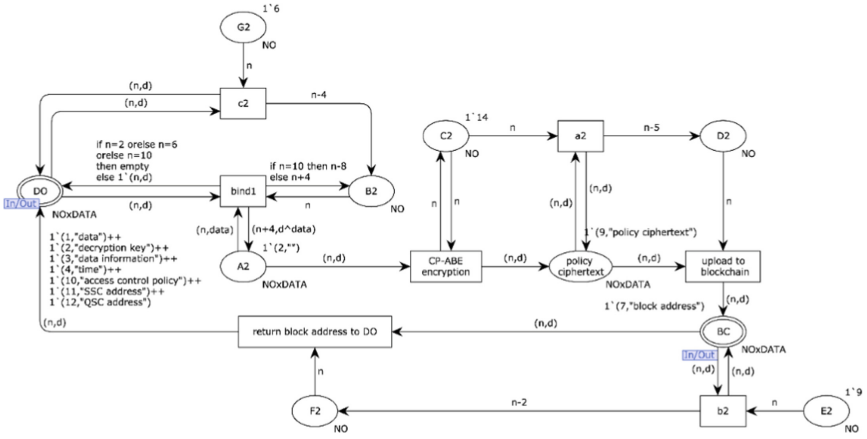

Fig. 4. The Upload policy ciphertext layer of CPN model.

The content of the data information operation layer includes DO deploying SSC and QSC, DO uploading data-related information to SSC, and DR querying information through QSC, as shown in Fig. 5. In this layer, four constraints need to be met: SSC and QSC can be deployed at any time, DR can call QSC to query after successful contract deployment, QSC returns query results to DR, and DR can always send query requests to QSC. Transition *Deploying the SSC* and *Deploying the QSC* can be fired at any time to deploy smart contracts. After that, the DR can trigger the transition *inquire* and *call SSC* to query the information, satisfying the first and second constraints. At this time, the transition *Release data information* is fired to upload the data-related information to the SSC. When the SSC is called by the QSC and receives the queried information, the transition *judge* can be fired for information matching. After the matching is successful, the transaction location and upload time are returned to the QSC through the trigger

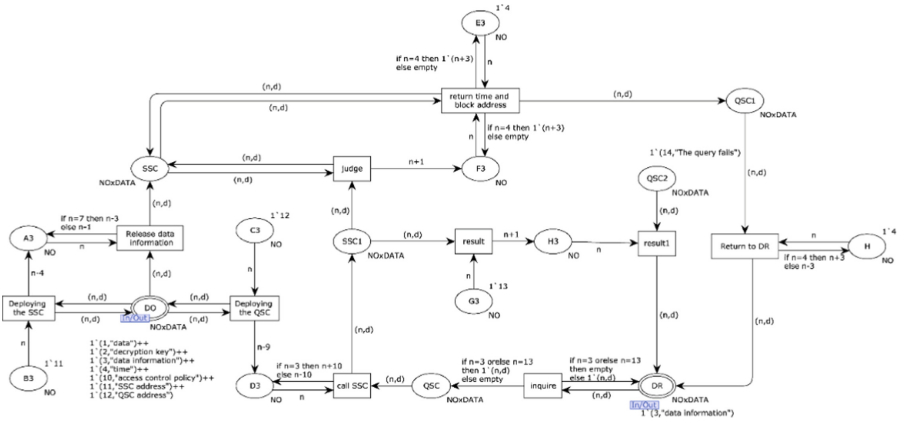

Fig. 5. The Data information operation layer of CPN model.

transition *return time and block address*, and then the transition *Return to DR* is fired to return the final query result to the DR. If the matching is unsuccessful, the transition *result* and *result1* will be fired successively and return the query failure information to the DR, thereby satisfying the third constraint condition. In addition, the transition *inquire* can always be fired to satisfy the fourth constraint. The places *SSC* and *SSC1* represent the same contract, and *QSC, QSC1* and *QSC2* represent the same contract, represented by different places because they are in different states.

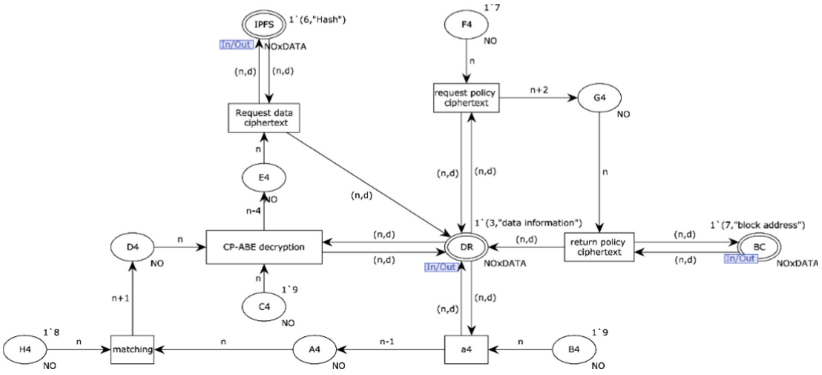

Fig. 6. The decrypt ciphertext layer of CPN model

The content of the Decrypt ciphertext layer includes DR downloading the data ciphertext from IPFS after DR decrypts the policy ciphertext, as shown in Fig. 6. In the process of decrypting the ciphertext, it is required to obtain the policy ciphertext first before the RSA-based CP-ABE decryption operation can be performed, and the data ciphertext can be downloaded only after the decryption is successful. The transition *request policy ciphertext* is fired first, and then the transition *return policy ciphertext* is fired to return the policy ciphertext from the blockchain. At this time, the transition *matching* is fired for

matching, whether the user attribute meets the requirements of the access control policy in the policy ciphertext. If the matching is successful, the transition *CP-ABE decryption* can be fired to perform RSA-based CP-ABE decryption. Finally, the data ciphertext is obtained by triggering the transition of the *Request data ciphertext*. In the decryption process, the constraints are satisfied by the restriction of place *A4* to *H4* and transition *a4*.

We use CPN tools to generate a complete state space report. The data in the state space report is shown in Table 9. It can be seen from the activity data information in the state space report that the model designed in this paper does not have dead marking or dead transitions, but there is an active transition *inquire*. Because in the model of this paper, the transition *inquire* can always be activated cyclically, that is, the DR can continuously send query requests to the QSC, which meets the design requirements. The built model passes the verification and satisfies various constraints through the simulation verification and state space analysis of CPN tools. Therefore, the scheme corresponding to the model is correct and safe.

Table 9. Data in the state space

Variable	Value
State Space (Nodes)	619
State Space (Arcs)	2490
State Space (Secs)	0
State Space (Status)	Full
Scc Graph (Nodes)	619
Scc Graph (Arcs)	1706
Scc Graph (Secs)	0
Dead markings	None
Dead transition instances	None
Live transition instances	Data_information_operation'inquire 1

5.2 Performance Analysis

This section will test the associated time consumption of different CP-ABE encryption methods when updating the access control policy. The experimental environment requirements are shown in Table 10.

We first compare the performance of CP-ABE based on bilinear mapping and CP-ABE based on RSA. The test mainly aims at policy ciphertext generation. In our scheme, the data encrypted by CP-ABE is the IPFS's hash and the decryption key of the data ciphertext. Since the hash size of the IPFS address is fixed and the decryption key of the data ciphertext is also fixed once it is determined, we assume that the size of the data encrypted by CP-ABE is 128 bytes. We manually designed access control policies during

Table 10. Requirements of the experimental environment

Demand	Message
Processor	Intel(R) Core (TM) i5-3230M CPU @ 2.60 GHz
Operating system	64-bit Windows 7
Memory	4.00 GB
Programming language	Java with IPBC library
Blockchain	Ethereum

the test to ensure that the access control policies under different structures were the same. Simultaneously we set the number of attributes in the access control policy to 10–50 and the number of attributes in the private attribute key to 5–25. In the experiment, this paper compares the first proposed tree structure BSW scheme, the LW scheme under the LSSS structure, and the GZCMZ scheme under the threshold structure. GZCMZ is further upgraded to selected ciphertext attack (CCA) security under the proposed CP-ABE scheme of selective plaintext attack (CPA) security. Reference [14] makes a detailed comparison of the CP-ABE scheme under the AND gate structure, so this paper does not conduct a comparison test.

As shown in Fig. 7, the RSA-based CP-ABE scheme is significantly better than the other CP-ABE schemes based on bilinear mapping in terms of encryption and decryption time. Although the GZCMZ scheme also shows a better time advantage in the encryption stage, the time has increased rapidly in the decryption process. Because the GZCMZ does not need to perform pairing calculations during encryption, the encryption time consumption is relatively small during the encryption process. However, it is still necessary to perform pairing calculations on attributes in the decryption process. As the number of attributes increases, the calculation cost is increasing.

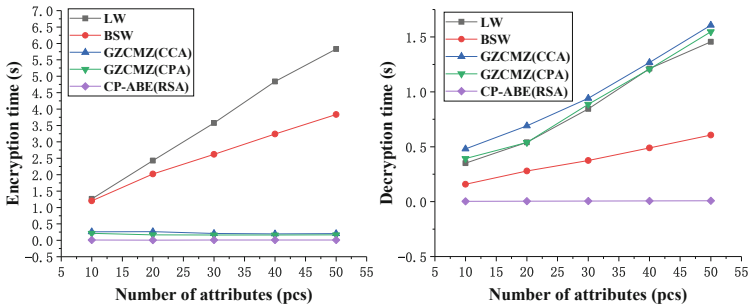

Fig. 7. Encryption time (a) and decryption time (b).

Moreover, the ciphertext size, policy ciphertext upload time, and total access control policy update time generated under different CP-ABE schemes are tested. The test results

are shown in Fig. 8 and Fig. 9 (a). As shown in Fig. 8 (a), the ciphertext size of BSW and LW increases almost linearly with the number of attributes. Both the CP-ABE based on RSA and GZCMZ achieve a constant ciphertext size. Although the ciphertext size of the key generated by the CP-ABE based on RSA is larger than GZCMZ in terms of the ciphertext size, both ciphertexts do not exceed 1KB. However, from Fig. 7 (b), GZCMZ consumes about 480-1600ms more than the RSA-based CP-ABE in the decryption process. When the access control policy is updated, the update time consumption includes the sum of the decryption time, the re-encryption time, and the policy ciphertext upload time. As shown in Fig. 8 (b) and Fig. 9 (a), the RSA-based CP-ABE scheme offers higher update efficiency when updating the access control policy.

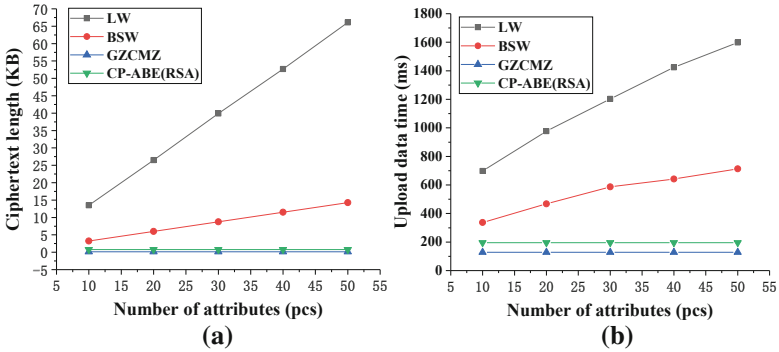

Fig. 8. Ciphertext length (a) and upload blockchain time (b).

We also conducted performance tests on SSC and QSC. In the process of querying and updating information, as shown in Fig. 9(b), the information stored first takes a higher time in querying and updating information. This is because the update and query operation designed in this paper use reverse order comparison when comparing data summaries. In this way, in the process of information update and query, the data stored later will be matched first, while the data stored first will be matched for more times, so it will consume more time.

According to the statistics in Fig. 9(b), in the process of information update, the time consumption is about 15–20 ms higher than the storage time consumption, because each information update is one step more than the information storage operation to put the historical records into the Record array. The information query time is about 350–400 ms higher than the storage time, because when querying, the SSC contract needs to be called through the QSC contract, and there will be a certain time cost in the process of calling the contract. For SSC, when the amount of stored data is less than 50, the storage and update times do not exceed 1200 ms. For QSC, the query time does not exceed 1550 ms. To sum up, the CP-ABE access control policy update method based on smart contract proposed in this paper shows better performance in time.

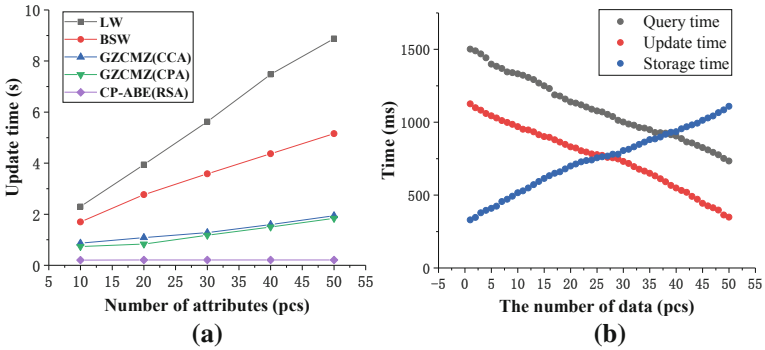

Fig. 9. Policy update time (a) and smart contract time(b).

6 Conclusion

In this paper, we propose an access control policy update scheme. In this paper, the RSA-based CP-ABE algorithm is used to improve the update efficiency of re-encryption. In addition, two smart contracts with different functions are designed to realize the storage management of old and new policy ciphertext and the retrieval of relevant information. Finally, the experiment results show that the proposed scheme has better strategy update efficiency in an environment with many attributes and frequent personnel changes. Our future work will focus on improving the functions of smart contracts of our scheme, which could implement the copyright control, keyword, and old policy ciphertext search functions.

Acknowledgment. This paper is supported by the National Natural Science Foundation of China (61962042) and Natural Science Foundation of Inner Mongolia (2018MS06028, 2022MS06020) and Science and Technology Program of Inner Mongolia Autonomous Region (2019GG376, 2020GG0188), and Open Topic of Inner Mongolia Discipline Inspection and Supervision Big Data Laboratory (IMDBD202008).

References

1. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: 2007 IEEE symposium on security and privacy (SP'07), pp. 321–334. Berkeley, California (2007)
2. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. Consulted (2008)
3. Wen, Q., Gao, Y., Chen, Z., Wu, D.: A blockchain-based data sharing scheme in the supply chain by IIoT. In: 2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS), pp. 695–700. Taipei, China (2019)
4. Huang, D., Chung, C.J., Dong, Q., Luo J., Kang, M.: Building private blockchains over public blockchains (PoP) an attribute-based access control approach. In: Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, pp. 355–363. Limassol, Cyprus (2019)
5. Yan, B., Yu, J., Wang, Y., Guo, Q., Chai B., Liu, S.: Blockchain-based service recommendation supporting data sharing. In: International Conference on Wireless Algorithms, Systems, and Applications, pp. 580–589. Qingdao, China (2020)

6. Huang, S., Chen, L.W., Fam, B.B.: Data security sharing method based on CP-ABE and blockchain. Comput. Syst. App. **28**(11), 79–86 (2019)
7. Qiu, Y.X., Zhang, H.X., Cao, Q., Zhang, J.C., Chen, X.S., Jin, H.J.: Blockchain data access control scheme based on CP-ABE algorithm. Chinese J. Netw. Inf. Secur. **6**(3), 88–98 (2020)
8. Qin, X., Huang, Y., Yang, Z., Li, X.: A blockchain-based access control scheme with multiple attribute authorities for secure cloud data sharing. J. Syst. Architect. **112**, 101854 (2021)
9. Yu, G., et al.: Enabling attribute revocation for fine-grained access control in blockchain-IoT systems. IEEE Trans. Eng. Manage. **67**(4), 1213–1230 (2020)
10. Tian, Y.L., Yang, K.D., Wang, Z., Feng, T.: Algorithm of blockchain data provenance based on ABE. J. Commun. **40**(11), 101–111 (2019)
11. Zhang, Y., He, D., Choo, K.K.R.: BaDS: blockchain-based architecture for data sharing with ABS and CP-ABE in IoT. In: Wireless Communications and Mobile Computing, pp. 1–9 (2018)
12. Gao, H., Ma, Z., Luo, S., Xu, Y., Wu, Z.: BSSPD: a blockchain-based security sharing scheme for personal data with fine-grained access control. In: Wireless Communications and Mobile Computing, pp. 1–20 (2021)
13. Benet, J.: IPFS - content addressed, versioned, P2P file system (DRAFT 3). arXiv preprint arXiv:1407.3561 (2014)
14. Odelu, V., Das, A.K., Khan, M.K., Choo, K.K.R., Jo, M.: Expressive CP-ABE scheme for mobile devices in IoT satisfying constant-size keys and ciphertexts. IEEE Access **5**, 3273–3283 (2017)
15. Khandla, D., Shahy, H., Bz, M.K., Pais, A.R., Raj, N.: Expressive CP-ABE scheme satisfying constant-size keys and ciphertexts. In: IACR Cryptol. ePrint Arch, p. 1257 (2019)
16. Jiang, S., Liu, J., Wang, L., Yoo, S.M.: Verifiable search meets blockchain: A privacy-preserving framework for outsourced encrypted data. In: ICC 2019–2019 IEEE International Conference on Communications (ICC), pp. 1–6. Tokio, Japan (2019)
17. Sun, S., Du, R., Chen, S.: A secure and computable blockchain-based data sharing scheme in IoT system. Information **12**(2), 47 (2021)
18. Li, X., Tan, M.: Electronic certificate sharing scheme with searchable attribute-based encryption on blockchain. J. Phys: Conf. Ser. **1757**(1), 012161 (2021)
19. Sun, J., Yao, X., Wang, S., Wu, Y.: Blockchain-based secure storage and access scheme for electronic medical records in IPFS. IEEE Access **8**, 59389–59401 (2020)
20. Pham, V.D., et al.: B-Box-a decentralized storage system using IPFS, attributed-based encryption, and blockchain. In: 2020 RIVF International Conference on Computing and Communication Technologies (RIVF), pp. 1–6. Ho Chi Minh City, Vietnam (2020)
21. Tan, H.B., et al.: Archival data protection and sharing method based on blockchain. J. Softw. **30**(9), 2620–2635 (2019)
22. Mounnan, O., Mouatasim, A.E., Manad, O., Outchakoucht, A., Es-samaali H., Boubchir, L.: A novel approach based on blockchain to enhance security with dynamic policy updating. In: 2020 7th International Conference on Internet of Things: Systems, Management and Security (IOTSMS), pp. 1-6. Paris, France (2020)
23. Ma, W., et al.: Attribute revocable data sharing scheme based on blockchain and CP-ABE. In: Proceedings of the 4th International Conference on Computer Science and Application Engineering, pp. 1–7. Sanya, China (2020)
24. Guo, R., Yang, G., Shi, H., Zhang, Y., Zheng, D.: O^3-R-CP-ABE: an efficient and revocable attribute-based encryption scheme in the cloud-assisted IoMT system. IEEE Internet of Things J. **8**(11), 8949–8963 (2021). https://doi.org/10.1109/JIOT.2021.3055541
25. Herranz, J., Laguillaumie, F., Ràfols, C.: Constant size ciphertexts in threshold attribute-based encryption. In: Nguyen, P.Q., Pointcheval, D. (eds.) Public Key Cryptography – PKC 2010. LNCS, vol. 6056, pp. 19–34. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13013-7_2

26. Susilo, W., Yang, G., Guo, F., Huang, Q.: Constant-size ciphertexts in threshold attribute-based encryption without dummy attributes. Inf. Sci. **429**, 349–360 (2018). https://doi.org/10.1016/j.ins.2017.11.037
27. Ge, A., Zhang, R., Chen, C., Ma, C., Zhang, Z.: Threshold ciphertext policy attribute-based encryption with constant size ciphertexts. In: Susilo, W., Mu, Y., Seberry, J. (eds.) Information Security and Privacy. LNCS, vol. 7372, pp. 336–349. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31448-3_25
28. Emura, K., Miyaji, A., Nomura, A., Omote, K., Soshi, M.: A ciphertext-policy attribute-based encryption scheme with constant ciphertext length. In: Bao, F., Li, H., Wang, G. (eds.) Information Security Practice and Experience. LNCS, vol. 5451, pp. 13–23. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00843-6_2
29. Lewko, A., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption. In: Gilbert, H. (ed.) Advances in Cryptology – EUROCRYPT 2010. LNCS, vol. 6110, pp. 62–91. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_4
30. Waters, B.: Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) Public Key Cryptography – PKC 2011. LNCS, vol. 6571, pp. 53–70. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19379-8_4
31. Lewko, A., Waters, B.: New proof methods for attribute-based encryption: achieving full security through selective techniques. In: Safavi-Naini, R., Canetti, R. (eds.) Advances in Cryptology – CRYPTO 2012. LNCS, vol. 7417, pp. 180–198. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_12
32. Agrawal, S., Chase, M.: FAME: fast attribute-based message encryption. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp.665–682. Dallas, TX, USA (2017)
33. Malluhi, Q.M., Shikfa, A., Trinh, V.C.: A ciphertext-policy attribute-based encryption scheme with optimized ciphertext size and fast decryption. In: Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, pp. 230–240. United Arab Emirates, Dubai (2017)
34. Doshi, N., Jinwala, D.C.: Fully secure ciphertext policy attribute-based encryption with constant length ciphertext and faster decryption. Sec. Commun. Netwv **7**(11), 1988–2002 (2014)
35. Zhang, Y., Zheng, D., Chen, X., Li, J., Li, H.: Computationally efficient ciphertext-policy attribute-based encryption with constant-size ciphertexts. In: Chow, S.S.M., Liu, J.K., Hui, L.C.K., Yiu, S.M. (eds.) Provable Security. LNCS, vol. 8782, pp. 259–273. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-12475-9_18
36. Cheung, L., Newport, C.: Provably secure ciphertext policy ABE. In: Proceedings of the 14th ACM Conference on Computer and Communications Security, CCS 2007, pp. 456–465. New York, NY, USA (2007)
37. Zhou, Z.B., Huang, D.J.: On efficient ciphertext-policy attribute based encryption and broadcast encryption: extended abstract. In: Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, pp. 753–755. New York, NY, USA (2010)
38. Guo, F., Mu, Y., Susilo, W., Wong, D.S., Varadharajan, V.: CP-ABE with constant size keys for lightweight devices. IEEE Trans. Inf. Forensics Secur. **9**(5), 763–771 (2014)
39. ang, W., Wang, R., Guan, Z., Wu, L., Du, X.J., Guizani, M.: A lightweight attribute based encryption scheme with constant size ciphertext for Internet of Things. In: ICC 2020 IEEE International Conference on Communications (ICC), pp. 1–6. Dublin, Ireland (2020)