Mathematical Biosciences and Engineering

http://www.aimspress.com/journal/MBE

MBE, 16(5): 4229–4249.
DOI: 10.3934/mbe.2019211
Received: 21 March 2019
Accepted: 05 May 2019
Published: 15 May 2019

Research article

A revocable storage CP-ABE scheme with constant ciphertext length in cloud storage

Yang Zhao, Xin Xie*, Xing Zhang and Yi Ding

University of Electronic Science and Technology of China, Chengdu, 610054 , China

* **Correspondence:** Email: xiex0814@gmail.com.

Abstract: The ciphertext policy attribute-based encryption (CP-ABE) is widely used in cloud storage. It not only provides a secure data sharing scheme but also has the characteristics of fine-grained access control. However, most CP-ABE schemes have problems such as the ciphertext length increases with the complexity of the access policy, the encryption scheme is complex, the computational efficiency is low, and the fine-grained revocation cannot be performed. In view of the above problems, this paper proposes an efficient CP-ABE scheme with fine-grained revocable storage and constant ciphertext length. The scheme combines proxy re-encryption with CP-ABE technology, adopts the flexible access strategy AND-gates on multi-valued attributes with wildcards (AND^*_m), and realizes revocable storage and fixed-length ciphertext. At the same time, in order to reduce the amount of user decryption calculation, the complex operation in the decryption process is outsourced to the third-party server and the decryption result is verified to ensure the correctness of the information. Finally, the security of the scheme is proved under the decisional bilinear Diffie-Hellman (DBDH) assumption. In addition, the performance analysis shows that the scheme is efficient and feasible in cloud storage.

Keywords: CP-ABE; revocable storage; constant ciphertext length; outsourcing decryption; AND-gates

1. Introduction

With the development of the internet of things [1], artificial intelligence [2] and cloud computing [3], more and more people use cloud storage servers to store large amounts of data. But these servers are untrustworthy, and information security has higher requirements in the new environment. Therefore, scholars have proposed an encryption and decryption system based on attribute: attribute-based encryption (ABE). It uses the access structure associated with the user attribute to decrypt the ciphertext if and only if the user's own attributes satisfy the access policy. Subsequently, more flexible key policy attribute-based encryption (KP-ABE) and CP-ABE schemes were proposed. Considering

the actual application scenarios, our scheme is based on CP-ABE.

In many systems that use CP-ABE, it is required to be able to flexibly add or delete user rights. According to the basic characteristics of ABE, such an application scenario can be implemented by adding and revoking attributes of users. Since attribute revocation is more complicated than attribute addition and is more difficult to implement, the research focus is more on revocation. According to the method of revocation, the granularity of revocation will be different, and the efficiency of revocation will also be different. The scheme [4] only implements user revocation, and its application scenarios are bound to be limited. Some schemes have not updated the ciphertext after the revocation, such as [5], that is, the revocable storage isn't implemented. At this time, the system has a forward security problem. In the scheme of implementing revocable storage, there is a problem that the computational complexity becomes larger as the complexity of the scheme becomes larger. For example, in scheme [6], the operation complexity of decryption will increase linearly with the attribute or revocation events, and the operation cost in the later stage of the scheme is huge. After using outsourcing decryption technology to solve complex operations and make up for the lack of time, there is still a spatial problem. The ciphertext length in most schemes increases linearly with the complexity of the access policy, resulting in a huge communication cost in data sharing. In particular, the partial revocation method also causes the ciphertext length to increase linearly with the number of revocation events. What is more serious is that the partial access structure is not flexible enough, which causes the scheme to lose the basic characteristics of ABE-flexibility.

It can be seen that constructing an outsourcing decryption CP-ABE scheme with fine-grained revocable storage, flexible access structure and constant ciphertext length has fatal research significance.

1.1. Related work

Through the efforts of scholars, there have been a lot of achievements in the field of cryptography [7–15]. After the development of identity-based cryptography for a period, attribute-based cryptography was proposed based on the fuzzy identity-based encryption technology [16]. It uses the attribute sets to represent the users' identities. The encryptor only needs to specify some attributes of the decryptor and does not need to specify its specific identity, so the attribute-based encryption is suitable for scenarios where the decryption party is not fixed. In 2006, Goyal et al. [17] proposed key policy attribute-based encryption (KP-ABE). Its main idea is to add an access structure constructed from user attribute sets to the user's private key, if and only if the access structure in the private key is satisfied by the attribute set in the ciphertext, the ciphertext can be decrypted. In 2007, the ciphertext policy attribute-based encryption scheme corresponding to the KP-ABE scheme was proposed by Bethencourt et al. [18]. The user's attribute set is stored in the user's private key, and all users' attribute sets in the system are stored in the ciphertext together with the access structure constructed by the access rules. Decryption can only be performed when the user attribute satisfies the access structure in the ciphertext.

In recent years, there have been a lot of researches on ABE, such as ABE with AND-gates structure, revocation ABE, ABE with fixed length ciphertext and outsourcing decryption ABE, etc.

In the study of the AND-gates, Cheung and Newport [19] proposed a CP-ABE scheme that supports AND-gates on positive and negative attributes with wildcards ($AND^{*}_{+,-}$), and proved its security under the standard model. Emura et al. [20] proposed the first ciphertext length constant CP-ABE scheme. However, the scheme only supports AND gates on positive and negative attributes ($AND_{+,-}$), includ-

ing the scheme of AND-gates on multi-valued attributes (AND_m) structure proposed later, which has serious drawbacks, that is, a user can decrypt the ciphertext if and only if the attributes in his private key are exactly the same as those contained in the ciphertext access structure, this has lost the basic principle of attribute-based encryption: one-to-many. Later, Nishide et al. [21] proposed AND^*_m structure scheme, which greatly improved the flexibility of access. On this basis, Zhang et al. [6] realized the fixed ciphertext length and reduced the storage cost. Constant-size ciphertext is also one of the key directions of ABE research. In recent years, there have been many achievements such as [22–26].

In addition to the study of access structures, the attribute revocation mechanism is also critical in ABE. In recent years, scholars at home and abroad have proposed a series of attribute revocation schemes, which are divided into indirect revocation and direct revocation. Indirect revocation requires the authority to periodically update the key, while direct revocation puts the revocation information into the ciphertext at the time of encryption. The indirect revocation of attributes scheme was first proposed by Pirretti et al. [27], which sets an expiration date for each attribute, and the authority periodically updates the key, but it cannot suspend the validity period of the key at any time. In response to its related issues, Bethencourt et al. [18] proposed the idea of setting the attribute suspension date, but the solution authority still has a large workload. Later, Attrapadung et al. [28] divided the revocation mechanism into two types: direct revocation and indirect revocation. Then they proposed a hybrid revocation scheme. The encryption party chooses either of the two mechanisms for encryption, the user can use keys to decrypt ciphertexts of any types. Most of the direct revocation schemes use broadcast encryption technology such as [29–31]. In order to reduce the workload of the authorized organization, Yu et al. [5] use the version number to mark the key and ciphertext, and introduce the proxy server to combine the proxy re-encryption technology with CP-ABE, which greatly reduced the workload of the authorized organization and achieved attribute revocation under a smaller overhead. To solve the forward security problem, Sahai et al. [32] proposed the concept of revocable storage, that is, ciphertext update must be performed when the revocation occurs. The direct revocation scheme proposed by Zhang et al. [6] introduces the revocation auxiliary judgment function applicable to the multi-attribute environment to determine whether the ciphertext needs to be updated when the attribute revocation event occurs. Zhao et al. [4] use the Chinese remainder theorem to construct a direct revocation scheme with constant length of ciphertext and key. After the revocation, the purpose of revocation can be achieved by only updating the ciphertext.

Another drawback in ABE is that decryption operations are very expensive, so they are not suitable for resource-constrained devices. In order to improve the efficiency of the scheme, Green et al. [33] proposed a solution for outsourcing the computationally expensive decryption operation to the cloud server. The main solution is the user uses the proxy re-encryption method to obtain the conversion key, sends it and ciphertext to the third-party server. The server performs the decryption operation to obtain a simpler form ciphertext, and the user can recover the plaintext with only a small computational overhead. After that, scholars tried to use outsourcing algorithm at different stages of ABE, such as generating private key phase [34, 35], encryption phase [36] and decryption phase [37, 38]. However, a malicious third-party server may tamper the original ciphertext, and the user will not be aware of it. The security of most outsourcing solutions can only ensure that malicious cloud servers can't understand any information about encrypted messages, but it does not guarantee the correctness of their conversion operations. In order to solve this problem, Lai et al. [39] proposed a verifiable outsourcing decryption CP-ABE scheme to ensure the correctness of the conversion operation. Li et al. [34] also proposed an

Table 1. Property comparison of different revocation schemes.

Scheme	Revocable Storage	Revocation Granularity	Verifiable Outsourcing Decryption	Constant Ciphertext Length	Access Policy	Ciphertext* Type
[5]	×	Attribute Revocation	×	×	$AND^*_{+,-}$	1
[6]	✓	Attribute Revocation	×	✓	AND^*_m	4
[32]	✓	User Revocation	×	×	Tree	1

* This item represents the number of ciphertext types that is generated by the scheme, which means that there are corresponding number of decryption algorithms and revocation algorithms.

outsourcing decryption ABE scheme that effectively solves the verifiability.

This paper mainly studies efficient revocation schemes. In the existing revocation schemes, there are some defects in revocation granularity, storage consumption, etc., as shown in Table 1. [5] does not implement revocable storage and the access structure is only $AND^*_{+,-}$, so its permissions are less flexible and users can illegally access the ciphertext which is encrypted before revocation. [6] contains a complex revocation algorithm and a decryption algorithm. For different types of ciphertexts, users divide them into 4 cases for revocation and decryption, which brings huge computational overhead to users. In scheme [32], revocation only reaches the user level, which cannot handle the requirement of flexible permission change. In summary, in the context of cloud storage, it is necessary to study more efficient and multi-functional revocation schemes.

1.2. Our contribution

In this paper, based on the study of revocable storage, ciphertext size and decryption efficiency, we propose a constant ciphertext length CP-ABE scheme with revocable storage and verifiable outsourcing decryption (RVOC-CP-ABE).

For the revocable storage problem, we improve the version numbering method which is based on proxy re-encryption to adapt to the characteristics of fixed-length ciphertext, and realize fine-grained attribute revocation and ciphertext update. If the user is the revocation target, cannot access the corresponding files, especially the encrypted files before revocation. In order to improve the decryption efficiency, we propose an outsourcing decryption mechanism, which can reduce the user's local computation and provide verification function to ensure decryption accuracy. Then, we combined the above two mechanisms to construct RVOC-CP-ABE scheme on the basis of AND^*_m access structure which is more flexible than traditional And-gate. Our scheme ensures that the attribute permissions are correctly revoked while reducing the decryption complexity and storage complexity to a constant level. We further describe the cloud server architecture and data flow of the scheme, providing a complete and efficient solution for cloud storage security.

Finally, under the DBDH assumption, we prove RVOC-CP-ABE scheme can resist chosen plaintext attacks (CPA), and the performance comparison with similar schemes shows that the scheme is efficient

and feasible.

1.3. Organization

The structure of this paper is as follows: Chapter 2 introduces the relevant theoretical knowledge used in this scheme; Chapter 3 will introduce the access structure used in this scheme, then introduce the design and security model of the scheme, and finally detail the implementation of the scheme. The fourth chapter will carry out security proof. The fifth chapter analyzes the performance of the scheme and compares it with similar schemes. Finally, the sixth chapter draws the conclusions of this program.

2. Preliminaries

2.1. Bilinear map

Assuming two multiplicative cyclic groups G and G_t, They have the same prime number p. Then let g be a generator element of cyclic group G. e is an effective bilinear mapping from G to G_t: $G \times G \rightarrow G_t$. e satisfies the following three properties:

1. **Bilinearity**: $\forall a, b \in Z_p$, $e(g^a, g^b) = e(g, g)^{ab}$.
2. **Non-Degeneracy**: $\exists u, v \in G$, $e(u, v) \neq 1$.
3. **Computability**: $\forall u, v \in G$, $e(u, v)$ can be calculated effectively.

2.2. Complexity assumption

Let $a, b, c, \theta \in_R Z_p$. There isn't a polynomial-time algorithm probabilistic has non-negligible advantage to distinguish the DBDH-tuple $\left[g, g^a, g^b, g^c, g^{abc}\right]$ from the random five-tuple $\left[g, g^a, g^b, g^c, g^{\theta}\right]$.

3. Revocable and verifiable outsourcing CP-ABE with constant ciphertext length

In this chapter, we will detail RVOC-CP-ABE. The scheme uses AND_m^* access structure to implement a fixed ciphertext length CP-ABE, which implements efficient revocable storage using version numbering method and proxy re-encryption, while outsourcing complex decryption calculations to a third-party server and verifying the decryption results. This chapter mainly has the following parts:

1. Introduce AND_m^* access structure.
2. Show the basic design of this program.
3. Introduce the security model of this scheme.
4. Describe the specific implementation of this program.

3.1. AND-Gates on multi-valued attributes with wildcards

As mentioned earlier, the $\text{AND}_{+,-}$ and AND_m access strategies are too restrictive and should not be used. Compared with AND_m^* and $\text{AND}_{+,-}^*$, the same access strategy using AND_m^* expression is much simpler than using $\text{AND}_{+,-}^*$ expression.

Table 2. Access structure.

Attribute	w_1	w_2	w_3
Attribute Name	School	Duty	Gender
Attribute Value	Univ.A	Teacher	Female
	Univ.B	Student	Male
	Univ.C		
Access Policy	Univ.C	*	Male

Now assume there is an attribute set as shown in the table 2, using $\text{AND}^*_{+,-}$ to indicate the access policy:

$$AP = w_1^- \wedge w_2^- \wedge w_3^+ \wedge w_4^* \wedge w_5^* \wedge w_6^- \wedge w_7^+$$

The w^+ represents the attribute in the access policy, the w^- represents the access policy not have the attribute, and the wildcard w^* represents no effect on the attribute. Using AND^*_m to indicate the access policy:

$$AP = w_{1,3} \wedge * \wedge w_{3,2}$$

Each of these is the value of the corresponding attribute, and the wildcard means that the attribute has no effect. Through the above comparison, it is obvious that the access strategy AND^*_m is simple and clear, so this paper adopts this structure.

Given a list of attributes L and an access policy W, $L| = W$ means that L matches W, and $L| \neq W$ means that the two do not match. Given a list of attributes $L = [L_1, L_2, ..., L_n]$ and access policy $W = [W_1, W_2, ..., W_n] = \wedge_{i \in I_W} W_i$. For $1 \leqslant i \leqslant n$, if $L_i = W_i$ or $W_i = *$, then $L| = W$, otherwise $L \neq W$. Where $I_W = \{i | 1 \leqslant i \leqslant n, W_i \neq *\}$ is a subscript index set, which clarifies the attributes of non-wildcards that appear in W. In the calculation, only the values corresponding to these subscripts are taken for calculation (I_L also take these indices to calculate). And does not appear in W the attribute is represented as a wildcard $*$, and whether the user has the corresponding attribute does not affect its decryption ability.

3.2. Scheme design idea

This scheme implements revocable storage and outsourcing decryption, so compared to the general CP-ABE, it will use two more servers: the revocation server and the outsourcing decryption server. The revocation server also completes the function of proxy re-encryption and updates the ciphertext after the revocation occurs. In this scheme, the users whose key needs to be updated does not need to send the decryption key information in the key to the revocation server, and only needs to exchange the irrelevant part, then they can decrypt the new ciphertext correctly, which is safer than the scheme [5]. Our scheme uses the version number to mark the ciphertext and key to complete the revocation. Therefore, the version number of each part should be consistent with the system version number. If it is inconsistent, the key or ciphertext should obtain the revocation related information from the revocation server to update the version to the latest. RVOC-CP-ABE requires the following six participants: data encryptor (DE), data decryptor (DD), authority center (AC), data storage server (DSS), revocation server (RS), and outsourcing decryption server (ODS).

- **Data Encryptor (DE)**: The data encryptor is the user who owns the original plaintext. Considering the advantages of mixed encryption, it encrypts the plaintext with a symmetric key. Then it requests AC to obtain the system public key, and encrypts the symmetric key using the public key and the access policy, and then stores the ciphertexts to DSS.
- **Data Decryptor (DD)**: The data decryptor is the user who accesses the system to obtain the plaintext. The user obtains the private key corresponding to the attribute from AC, and then converts the private key, sends the conversion key and the ciphertext to ODS, and the returned converted ciphertext is further decrypted to obtain the symmetric key, finally, the symmetric decryption is performed to obtain the plaintext. If the user finds that the private key version is inconsistent with the system version, the user should obtain the latest version of the revocation key from the RS, which is independent of the decryption key information in the private key.
- **Authority Center (AC)**: The Authority center is a fully trusted server that initializes the scheme system to generate the system's public key and master secret key. Provide user registration function and generate a decryption private key for DD by the master secret key and user attributes.
- **Data Storage Server (DSS)**: The data storage server is used to store the ciphertext encrypted by DE. Because it is a cloud server, it can easily expand its storage capacity and it is also an untrusted third-party server. All information should be encrypted when stored in it. When the version number of the ciphertext which is accessed is inconsistent with the system version number, DSS is responsible for passing the old version of the ciphertext to RS for re-encryption to complete the ciphertext update after the revocation is completed.
- **Revocation Server (RS)**: The revocation server mainly stores the relevant information of the revocation: the re-encryption parameter table and the attribute revocation table. When the system revokes some users' attributes, RS is responsible for generating a new version of the revocation parameters and promoting the system version. After the storage server sends the low version ciphertext to RS, the server uses the re-encryption parameter table to re-encrypt the old version ciphertext, thereby implementing the ciphertext update and completing the revocable storage. When the user sends the low version revocation key to RS, the server uses the re-encryption parameter table and the attribute revocation table to calculate a new revocation key and return. In this scheme, if there are multiple revocation versions, ciphertexts and revocation keys of them can be updated in one time, which greatly reduces the resource overhead of re-encryption.
- **Outsourcing Decryption Server (ODS)**: Due to the complex ABE decryption calculation, this scheme puts the most complicated part of the decryption process into the outsourcing decryption server. Because the user passes the blinded conversion key to the three-party server during decryption, it overcomes the untrustworthy problem of outsourcing decryption. After the complicated bilinear calculation, ODS only obtains the intermediate ciphertext.

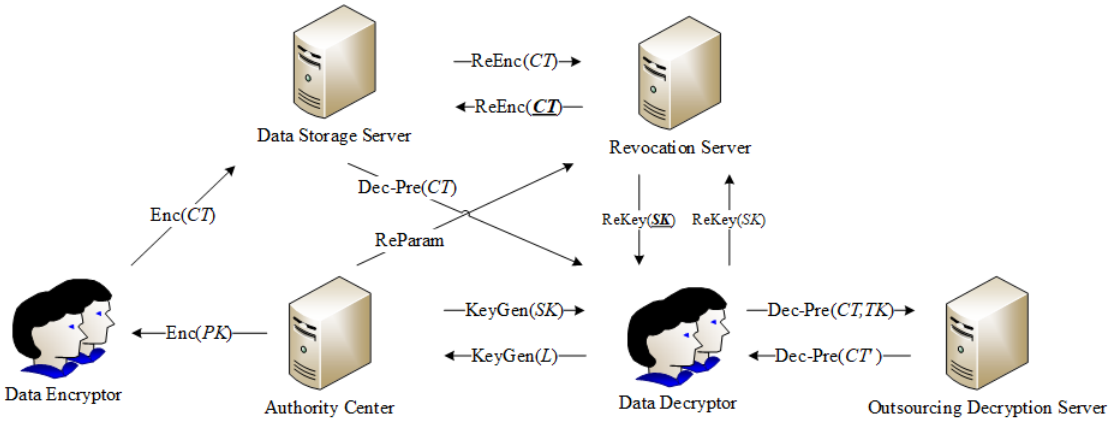

Figure 1. Framework of RVOC-CP-ABE.

The program also adds decryption verification to ensure that revocation and outsourcing are performed correctly. The framework of RVOC-CP-ABE scheme is shown as Figure 1. The Authority Center initializes the system to generate the public key (PK) and master secret key (MSK). The data encryptor first performs symmetric encryption, then obtains the PK from AC, generates the ciphertext in combination with AND^*_m access policy and the symmetric key, and stores the ciphertexts in the data storage server. The data decryptor obtains the private key (SK) corresponding to the own attributes from the AC, blinds it into a conversion key, and sends the conversion key and ciphertext to the outsourcing decryption server. The server performs complex operations of decryption to obtain the converted ciphertext and return it to DD. The decryptor first performs preliminary verification on the converted ciphertext, then performs a simple operation to obtain the symmetric key, and verifies the symmetric key. If the verification passes, symmetric decryption is performed to obtain plaintext. If the SK or ciphertext is inconsistent with the system version during the decryption process, the corresponding update is performed by RS.

According to the above process, the RVOC-CP-ABE algorithm is mainly divided into eight parts, namely: initialization algorithm $Setup$, key generation algorithm $KeyGen$, encryption algorithm Enc, pre-decryption algorithm $Dec-Pre$, decryption algorithm Dec, re-encryption parameter algorithm $ReParam$, re-encryption ciphertext algorithm $ReEnc$, re-encryption key algorithm $ReKey$. The latter three mainly occurred after the revocation event.

1. **Setup**$(1^\lambda) \rightarrow (ver, PK, MSK)$: AC runs the system initialization algorithm, taking the security parameter λ as input, and outputs the system's PK and MSK. The authority center will publicize the PK and save the MSK safely.
2. **KeyGen**$(PK, MSK, L) \rightarrow (SK)$: The key generation algorithm is also run by AC. DD submits its attribute list L to AC. The server uses PK, MSK and L to generate the corresponding user private key SK.
3. **Enc**$(PK, M, W) \rightarrow (CT, E(M))$: The encryption algorithm is run by DE. The algorithm first uses the symmetric key ck to encrypt the plaintext M to generate the symmetric ciphertext $E(M)$, then inputs the system's PK, the symmetric key ck, and the access policy W, outputs the corresponding ciphertext CT, then saves the CT and $E(M)$ to DSS.
4. **Dec-Pre**$(CT, PK, SK) \rightarrow (CT')$: The pre-decryption algorithm is mainly to perform outsourcing

decryption operations. DD obtains the CT, and then uses the blind parameter z to blindly encrypt the own key SK to obtain the conversion key TK, then sends (CT, TK) to ODS for outsourcing decryption, and returns the intermediate ciphertext CT'.

5. **Dec**$(CT', z, E(M)) \rightarrow (M)$: After DD obtains the intermediate ciphertext, it uses CT' and the blind parameter z to obtain the symmetric key ck, then verifies the decryption result. If the decryption succeeds, the symmetric decryption is continued to obtain the plaintext M.
6. **ReParam**(μ, IDs): After the attribute revocation occurs in the system, RS obtains the attribute set μ that needs to be updated and its corresponding revocation user set IDs to perform system version upgrade, and generates a new version of the re-encryption parameter and the attribute revocation information.
7. **ReEnc**$(CT) \rightarrow (\overline{CT})$: DSS can check the ciphertext version when a DD accesses the ciphertext. If the version is not up-to-date, the ciphertext CT is sent to RS for ciphertext re-encryption to obtain the latest version of the ciphertext $\overline{CT}$.
8. **ReKey**$(SK) \rightarrow (\overline{SK})$: After obtaining the ciphertext CT, DD finds the version of its private key SK is inconsistent with CT's version, then DD sends its old version revocation key (a part of SK) to RS. RS checks whether it has the revocation attribute according to the user ID, and only updates the parameters corresponding to the unrevoked attributes of the user. After the user gets the new version revocation key, update SK to $\overline{SK}$.

3.3. Security model

This scheme mainly needs the following aspects of security: ciphertext security, outsourcing decryption computing security, and security of revocation algorithm. The CP-ABE encryption system guarantees the security of ciphertext storage, and unauthorized users cannot successfully decrypt and obtain ciphertext. When the scheme performs the outsourcing decryption algorithm, the user needs to blind the key before sending it to the third-party server, so the security can be converted into the security of the scheme algorithm. On the other hand, the revocable storage of this scheme is implemented by the revocation server, it mainly generates new re-encryption parameters for different versions. Although the server directly updates the ciphertext, it only updates revocation key of the user's private key, does not involve the key part of decryption in the private key, so its security can also be transferred to the security of the scheme.

For the above security issues, this paper considers the indistinguishability under chosen plaintext attack (IND-CPA), which is simulated by a security game containing adversary $\mathcal{A}$ and challenger C. The process is as follows:

1. **Initialization**: Adversary $\mathcal{A}$ will select the access structure $\mathbb{W}^*$ and a version number ver^* used in the game and send them to challenger C.
2. **Setup**: C runs the $Setup$ algorithm in this scheme to initialize the system, create the system's public key PK and master secret key MSK, and run $ReParam$ algorithm to upgrade the system version from 1 to $ver*$. Finally, PK is sent to $\mathcal{A}$.
3. **Phase 1**: $\mathcal{A}$ can randomly select the attribute set $S = \{S_1, S_2, ..., S_n\}$ from the all attributes set, S is limited to not satisfy $\mathbb{W}^*$. $\mathcal{A}$ sends S to C query for private key. C calculates the private key SK corresponding to ver=1 by running the calculation, and then returns SK to the adversary. $\mathcal{A}$ runs $ReKey$ algorithm and raises the private key version to the $ver*$ version.

4. **Challenge**: $\mathcal{A}$ selects two challenged plaintexts M_0, M_1 and sends them to C. C randomly selects a number b from {0,1} then uses M_b as plaintext, inputs M_b, PK and $\mathbb{W}^*$ to the encryption algorithm to calculate the corresponding ciphertext CT_b. After completing the above operation, C runs $ReEnc$ algorithm updates CT_b to $\overline{CT_b}$ which version is $ver*$, then returns the ciphertext $\overline{CT_b}$ to $\mathcal{A}$.
5. **Phase 2**: $\mathcal{A}$ repeats the same query as **Phase 1**.
6. **Guess**: $\mathcal{A}$ guesses b' that b' belongs to {0, 1}. If $b' = b$ then $\mathcal{A}$ wins this safe game. The probability of $\mathcal{A}$ winning a safe game in this game is defined as follows:

$$Adv_{\mathcal{A}} = |Pr[b' = b] - \frac{1}{2}|$$

If polynomial time adversary $\mathcal{A}$ has no probability to break the security game with a non-negligible advantage, then the adversary has no advantage, and it can be theoretically shown that RVOC-CP-ABE is safe.

3.4. Scheme realization

This section will give a concrete implementation of the revocation algorithm, the outsourcing decryption algorithm, and the basic CP-ABE scheme.

Let G and G_t be two multiplicative cyclic groups with the order is a large prime p, g is a generator of G, and map $e : G \times G \to G_t$ is a bilinear map. Suppose that the attribute domain of the system has a total of n attributes, and its attribute set is recorded as $U = \{u_1, u_2, ..., u_n\}$, each attribute has multiple values, and the i-th attribute u_i has n_i values. The corresponding set of values is denoted as $V_i = \{v_{i,1}, v_{i,2}, ..., v_{i,n_i}\}$. $H : G_t \to Z_p^*$ is a collision-resistant hash function.

1. **Setup**$(1^\lambda) \to (ver, PK, MSK)$: The authority center runs the $Setup$ algorithm to initialize the system. Select $u, v, d \in G, x_{i,j}, y_{i,j} \in_R Z_p^* (i \in [1, n], j \in [1, n_i])$. Then initialize the system version initial value ver=1, generate the public key and the master secret key of system as follows:

$$X_{i,j} = g^{-x_{i,j}}, Y_{i,j} = e(g, g)^{y_{i,j}} (1 \leqslant i \leqslant n, 1 \leqslant j \leqslant n_i) \tag{3.1}$$

$$PK = (g, u, v, d, \{X_{i,j}, Y_{i,j}\}_{1 \leqslant i \leqslant n, 1 \leqslant j \leqslant n_i}) \tag{3.2}$$

$$MSK = (\{x_{i,j}, y_{i,j}\}_{1 \leqslant i \leqslant n, 1 \leqslant j \leqslant n_i}) \tag{3.3}$$

2. **KeyGen**$(PK, MSK, L) \to (SK)$: The user with the attribute list $L = [L_1, L_2, ..., L_n]$ applies for the attribute private key from the authority center. Let $L_i = v_{i,j}$, the authority center first selects a random number $r \in_R Z_p^*$, the revocation key is $\overline{RK_i^{(1)}} = 1, 1 \leqslant i \leqslant n$, $\overline{RK_i^{(k)}}$ represents i-th attribute L_i corresponding to the k version of the revocation key, then calculates as follows:

$$K_i = g^{y_{i,j} + r x_{i,j}} (1 \leqslant i \leqslant n), K = g^r, ver = 1 \tag{3.4}$$

So the user private key SK is:

$$SK = (ver, L, K, \{K_i\}_{1 \leqslant i \leqslant n}, \overline{RK_i^{(ver)}}_{1 \leqslant i \leqslant n}) \tag{3.5}$$

3. **Enc**$(PK, M, W) \rightarrow (CT, E(M))$: First, the symmetric key ck is selected to symmetrically encrypt the plaintext M to generate the ciphertext $E(M)$, then input public key PK, symmetric key ck and access policy W. Let $ver = 1, W = \wedge_{I_W} W_i, W_i = v_{i,j}$, then the data encryptor selects a random number $s \in_R Z_p^*$, calculates as follows:

$$X_W = \prod_{i \in I_W} X_{i,j}, Y_W = \prod_{i \in I_W} Y_{i,j} \tag{3.6}$$

$$C_0 = ckY_W^s, C_1 = g^s, C_2 = X_W^s \tag{3.7}$$

Due to the need of verification function, the encryptor continues to select $s' \in_R Z_p^*, \widetilde{ck}$, then calculates:

$$C_0' = \widetilde{ck} Y_W^{s'}, C_1' = g^{s'}, C_2' = X_W^{s'}, \hat{C} = u^{H(ck)} v^{H(\widetilde{ck})} d \tag{3.8}$$

Therefore, the output ciphertext is $CT = (ver, W, \hat{C}, C_0, C_1, C_2, C_0', C_1', C_2')$.

4. **Dec-Pre**$(CT, PK, SK) \rightarrow (CT')$: First, the user obtains the ciphertext CT from the data storage server. The data storage center detects whether the ver in the ciphertext is consistent with the system version number. If the inconsistency occurs, the server executes the $ReEnc$ algorithm to obtain the new version of the ciphertext. After obtaining the ciphertext, the user checks whether the ver of SK is consistent with the ver in the obtained CT. If not, the $ReKey$ algorithm is executed to obtain a new revocation key. If they are consistent, the user generates a conversion key for outsourcing computing, chooses an outsourcing parameter $z \in_R Z_p^*$ and save it, then calculates:

$$K' = K^{\frac{1}{z}}, K_i' = K_i^{\frac{1}{z}} (1 \leqslant i \leqslant n) \tag{3.9}$$

$$TK = (ver, L, K', \{K_i'\}_{1 \leqslant i \leqslant n}, \overline{RK_i^{(ver)}}_{1 \leqslant i \leqslant n}) \tag{3.10}$$

Then the user send (CT, TK) to outsourcing decryption server for outsourcing decryption operation. If the ver in the TK is inconsistent with the ver in the CT, the server directly suspends the operation and returns the decryption failure. Check user's attribute set L and the access policy W in ciphertext, If $L| \neq W$, return decryption failed. Otherwise, the version number is the same and $L = W$, calculate $\overline{RK} = \prod_{i \in I_L} \overline{RK_i^{(ver)}}$. Assuming this is the decryption process of ver=1, so $\overline{RK}$=1, then doing the following calculation:

$$T' = e(C_1, (\prod_{i \in I_L} K_i')^{\overline{RK}}) \cdot e(C_2, K') = e(g, g)^{\frac{s \sum_{i \in I_L} y_{i,j}}{z}} \tag{3.11}$$

$$T'' = e(C_1', (\prod_{i \in I_L} K_i')^{\overline{RK}}) \cdot e(C_2', K') = e(g, g)^{\frac{s' \sum_{i \in I_L} y_{i,j}}{z}} \tag{3.12}$$

The conversion ciphertext $CT' = (\hat{T} = \hat{C}, T_0 = C_0, T_0' = C_0', T', T'', \overline{RK})$ is returned to the data decryptor.

5. **Dec**$(CT', z, E(M)) \rightarrow (M)$: The user first checks whether part of the information in the converted ciphertext is consistent with the ciphertext: $\hat{T} = \hat{C}, T_0 = C_0, T_0' = C_0'$, and if there is an inequality, the decryption is aborted. After verification, first assuming this is the decryption process of ver=1, then $\overline{RK}$=1, the decryption process is as follows:

$$ck = (\frac{C_0}{(T')^z})^{\frac{1}{\overline{RK}}} = \frac{ck \cdot e(g, g)^{s \sum_{i \in I_W} y_{i,j}}}{e(g, g)^{s \sum_{i \in I_L} y_{i,j}}}, \widetilde{ck} = (\frac{C_0'}{(T'')^z})^{\frac{1}{\overline{RK}}} \tag{3.13}$$

Then verify: $\hat{T} = u^{H(ck)}v^{H(\widetilde{ck})}d$. If the equation is true, then use ck to symmetric decrypt $E(M)$ to get the message M, otherwise the decryption fails.

6. **ReParam**(μ, IDs): Run this algorithm when the system has an attribute revocation. Input the attribute set μ that needs to be updated and the revocation user set IDs corresponding to each attribute. We use $rk_{i,j}^{(k)}$ represent re-encryption parameter of attribute $v_{i,j}$ in the k-th version. In the new version $ver + 1$, the re-encryption parameter $rk_{i,j}^{(ver+1)}$ is selected for each attribute. If the revocation attribute is $v_{a,b}$, then $rk_{a,b}^{(ver+1)} \in_R Z_p^*, rk_{(i,j)\neq(a,b)}^{(ver+1)} = 1$, that is, the non-revoked attribute re-encryption parameter is 1. The revocation server maintains two tables: the re-encryption parameter table and the attribute revocation table. Each time a revocation occurs, the system increments ver by 1. The re-encryption parameter table records the new version corresponding to the re-encryption parameters, as shown in the Table 3.

Table 3. Re-encryption parameter table.

ver	$v_{1,1}$	$v_{1,2}$	$v_{2,1}$	$\cdots$	$v_{i,j}$	$\cdots$	v_{n,n_i}
k	$rk_{1,1}^{(k)}$	$rk_{1,2}^{(k)}$	$rk_{2,1}^{(k)}$	$\cdots$	$rk_{i,j}^{(k)}$	$\cdots$	$rk_{n,n_i}^{(k)}$
$\cdots$				$\cdots$			
2	$rk_{1,1}^{(2)}$	$rk_{1,2}^{(2)}$	$rk_{2,1}^{(2)}$	$\cdots$	$rk_{i,j}^{(2)}$	$\cdots$	$rk_{n,n_i}^{(2)}$
1	1	1	1	1	1	1	1

At the same time, save the user ID of each revocation attribute according to IDs, as shown in Table 4. According to the table, only the unrevoked attributes are re-encrypted when the relevant user performs key re-encryption.

Table 4. Attribute revocation table.

attribute	user ID			
$v_{i,j}$	ID_1	ID_2	$\cdots$	ID_5
$v_{i,j+1}$	ID_2	ID_3	$\cdots$	ID_4

7. **ReEnc**$(CT) \rightarrow (\overline{CT})$: The system after a revocation, does not need to update the ciphertext immediately. When a user accesses a ciphertext with an unupgraded version, the data storage server sends the ciphertext to the revocation server and updates the ciphertext multiple versions at a time. Assuming the old ciphertext differ with the latest version of the k version, first calculate based on the attributes contained in the access structure in the ciphertext:

$$\overline{RK_W} = \prod_{i\in I_W} rk_{i,j}^{(ver+1)} \times \prod_{i\in I_W} rk_{i,j}^{(ver+2)} \times \cdots \times \prod_{i\in I_W} rk_{i,j}^{(ver+k)} \tag{3.14}$$

Then calculate:

$$\bar{C}_0 = C_0^{\overline{RK_W}}, \bar{C}_2 = C_2^{\overline{RK_W}}, \bar{C}'_0 = C_0'^{\overline{RK_W}}, \bar{C}'_2 = C_2'^{\overline{RK_W}} \tag{3.15}$$

Return new ciphertext $\overline{CT} = (ver + k, W, \hat{C}, \bar{C}_0, C_1, \bar{C}_2, \bar{C}'_0, C_1, \bar{C}'_2)$.

8. **ReKey**$(SK) \rightarrow (\overline{SK})$: When the user accesses the ciphertext and finds that the version number ver in SK is not equal to that in the ciphertext. Assuming there is a difference of k versions between the two, the user sends $(ver, L, \{\overline{RK_i^{(ver)}}\}_{1\leqslant i\leqslant n})$ to the revocation server to obtain a new revocation key$\{\overline{RK_i^{(ver+k)}}\}$. Here we can see that the revocation server gets only the insignificant private key information, not the core decryption parameters. The revocation server first determines whether there is an attribute of the user that is revoked according to the user's ID. If there is, the re-encryption parameter of the attribute is not included in the calculation, and I_R is set to the attribute set that the user is revoked, and the following calculation is performed:

$$\overline{RK_i^{(ver+k)}} = \overline{RK_i^{(ver)}} \times \prod_{h=ver+1}^{ver+k} rk_{i,j}^{(h)}, i \in I_L, i \notin I_R \tag{3.16}$$

The revocation $\overline{RK_i^{(ver+k)}}$ of the user's revoked attribute remains the original value, finally return $(ver+k, \{\overline{RK_i^{(ver+k)}}\}_{1\leqslant i\leqslant n})$. The user updates SK to $\overline{SK} = (ver+k, L, K, \{K_i\}_{1\leqslant i\leqslant n}, \overline{RK_i^{(ver+k)}}_{1\leqslant i\leqslant n})$.

To facilitate understanding, a specific revocation example is used to illustrate the process of the revocation algorithm and its correctness.

Now assume that the first time the user is revoked attribute $v_{1,2}$, the second time the user is revoked attribute $v_{1,2}, v_{2,3}$, the system version is from 1 to 3, and the ciphertext containing these two attributes is re-encrypted as follows:

$$\overline{RK_W} = \prod_{i\in I_W} rk_{i,j}^{(2)} \times \prod_{i\in I_W} rk_{i,j}^{(3)}$$

Other attributes are not revoked, $rk_{i,j\neq(1,2)}^{(2)} = 1$, $rk_{i,j\neq(1,2),(2,3)}^{(3)} = 1$, so $\overline{RK_W} = rk_{(1,2)}^{(2)} rk_{(1,2)}^{(3)} rk_{(2,3)}^{(3)}$, output new ciphertext $\overline{CT}$ like the result of $ReEnc$ algorithm.

A user with the $v_{1,2}, v_{2,3}$ attributes and version number 1 accesses the above ciphertext, uses the Rey algorithm to calculate:

$$\overline{RK_1^{(3)}} = \overline{RK_1^{(1)}} \times rk_{1,2}^{(2)} \times rk_{1,2}^{(3)}, \overline{RK_2^{(3)}} = \overline{RK_2^{(1)}} \times rk_{2,3}^{(2)} \times rk_{2,3}^{(3)}$$

Because of $\overline{RK_1^{(1)}} = 1, \overline{RK_2^{(1)}} = 1, rk_{2,3}^{(2)} = 1, \overline{RK_{i\neq1,2}^{(3)}} = 1$, get the value of the revocation key $\overline{RK_1^{(3)}} = rk_{1,2}^{(2)} \times rk_{1,2}^{(3)}, \overline{RK_2^{(3)}} = rk_{2,3}^{(3)}$.

Then the outsourcing decryption operation as follows:

$$\overline{RK} = \prod_{i\in I_L} \overline{RK_i^{(ver)}} = rk_{(1,2)}^{(2)} rk_{(1,2)}^{(3)} rk_{(2,3)}^{(3)}$$

$$\begin{aligned} T' &= e(C_1, (\prod_{i\in I_L} K_i')^{\overline{RK}}) \cdot e(\bar{C}_2, K') \\ &= e(g^s, g^{\frac{\overline{RK}}{z}\sum_{i\in I_L}(y_{i,j}+rx_{i,j})}) \cdot e(g^{-s\overline{RK_W}\sum_{i\in I_W} x_{i,j}}, g^{\frac{r}{z}}) \\ &= e(g,g)^{\frac{s\overline{RK}\sum_{i\in I_L} y_{i,j}}{z}} \end{aligned}$$

$$T'' = e(C_1', (\prod_{i \in I_L} K_i')^{\overline{RK}}) \cdot e(\bar{C}_2', K') = e(g,g)^{\frac{s'\overline{RK}\sum_{i \in I_L} y_{i,j}}{z}}$$

Finally, the user local Calculation as follows:

$$\begin{aligned} ck &= (\frac{\bar{C}_0}{(T')^z})^{\frac{1}{\overline{RK}}} \\ &= \left(\frac{ck^{\overline{RK_W}} \cdot e(g,g)^{s\overline{RK_W}\sum_{i \in I_W} y_{i,j}}}{e(g,g)^{s\overline{RK}\sum_{i \in I_L} y_{i,j}}} \right)^{\frac{1}{\overline{RK}}} \\ &= \left(ck^{\overline{RK_W}}\right)^{\frac{1}{\overline{RK}}}, \widetilde{ck} = (\frac{\bar{C}_0'}{(T'')^z})^{\frac{1}{\overline{RK}}} \end{aligned}$$

Then verify $\hat{T} = u^{H(ck)} v^{H(\widetilde{ck})} d$, if the equation is true, then use ck to symmetric decrypt $\underline{E(M)}$ to get the message M, otherwise the decryption fails. If the user has the attribute revoked, then $\overline{RK} \neq \overline{RK_W}$, so it can't get the correct plaintext.

4. Security analysis

In this section, it will be demonstrated that the scheme RVOC-CP-ABE is CPA-safe under the DBDH assumption.

Theorem 1. *If an adversary $\mathcal{A}$ wins the IND-CPA game with a non-negligible advantage ε, then there is a challenger C to solve the decision bilinear Diffie-Hellman problem with the advantage $\frac{\varepsilon}{2}$.*

Proof. In the CPA game, the challenger C selects two multiplicative cyclic groups G and G_t of the order p, g is the generator of G, mapping e, hash function H, and $a, b, c \in_R Z_p^*$. C throws one coin σ, if $\sigma = 0$, then set $z = abc$, if $\sigma = 1$, then set $z \in_R Z_p^*$. C has a five-tuple $(g, A, B, C, Z) = (g, g^a, g^b, g^c, e(g,g)^z)$, then simulates the IND-CPA security game as follows.

- **Initialization**. The adversary $\mathcal{A}$ generates a challenge access policy $\mathbb{W}^* = \wedge_{i \in I_{\mathbb{W}^*}} W_i$, $I_{\mathbb{W}^*} = \{i_1, i_2, ..., i_m\} (m < n)$ represents the subscript set of attributes which are not wildcards in the access policy $\mathbb{W}^*$. Then $\mathcal{A}$ generates a version number $ver^* (ver^* \geqslant 1)$, and sends $\mathbb{W}^*$ and ver^* to C.
- **Setup**. C runs the $Setup$ algorithm of the scheme to calculate PK and MSK. For each $i \in U, j \in U_i (U = \{U_1, U_2, \cdots, U_n\}$ is a set of all attributes in system), select a random number $\partial_{i,j} \in_R Z_p^*$, calculate $X_{i,j} = g^{-\partial_{i,j}}, Y_{i,j} = e(B, X_{i,j}^{-1}) = e(g,g)^{b\partial_{i,j}} (1 \leqslant i \leqslant n, 1 \leqslant j \leqslant n_i)$, and return PK to $\mathcal{A}$. Then, C runs the $ReParam$ algorithm to simulate the upgrade of $ver^* - 1$ versions. For the k-th version, set $rk_{i,j}^{(k)} \in_R Z_p^*$ if the attribute is revoked, if not $rk_{i,j}^{(k)} = 1$.
- **Phase 1**. $\mathcal{A}$ selects a set of attributes $L = \{L_1, L_2, ..L_n\}, L \subseteq U, L_i = v_{i,j}$,but L does not satisfy $\mathbb{W}^*$. $\mathcal{A}$ sends L to C for private key query. C selects $\delta \in_R Z_p^*$, performs calculation $K_i = g^{(b+\delta)\partial_{i,j}}, K = g^{\delta}, ver = 1$, the revocation key $\overline{RK_i^{(1)}} = 1, 1 \leqslant i \leqslant n$. C returns SK to $\mathcal{A}$, $\mathcal{A}$ can choose to update the key, that is, run the $ReKey$ algorithm, send parameters $(ver = 1, L, \{\overline{RK_i^{(1)}}\}_{1 \leqslant i \leqslant n})$ to C. C calculates $\overline{RK_i^{(ver^*)}} = \overline{RK_i^{(1)}} \times \prod_{h=2}^{ver^*} rk_{i,j}^{(h)}$ according to $L, ver*$ and user ID. After completion, $\mathcal{A}$ has the latest version of the key.

- **Challenge**. $\mathcal{A}$ submits two equal-length messages M_0 and M_1 to C, the challenger chooses $b \in_R \{0, 1\}$ and calculates $C_0 = M_b Y^c_{\mathbb{W}^*}, C_1 = g^c, C_2 = X^c_{\mathbb{W}^*}$. Let $\sum_{i \in I_{\mathbb{W}^*}} \partial_i = a + \eta$, there is

$$C_0 = M_b e(g, g)^{c \cdot \sum_{i \in I_{\mathbb{W}^*}} b\partial_i} = M_b e(g, g)^{cb(a+\eta)} = M_b Z e(g, g)^{cb\eta}$$

  Then C runs the *ReEnc* algorithm on CT, updates it to the latest version ver^*, and returns $\overline{CT}$ to $\mathcal{A}$.
- **Phase 2**. $\mathcal{A}$ repeats the same query as **Phase 1**.
- **Guess**. $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$, if $b' = b$, C will output $\sigma' = 0$, this means (g, A, B, C, Z) is a valid DBDH group. Otherwise, C corresponds to output $\sigma' = 1$, where z is a random number, indicating a random five-tuple.
  If $Z = e(g, g)^{abc}$, $\mathcal{A}$ gains a valid ciphertext, its advantage is:

$$Pr[b' = b | Z = e(g, g)^{abc}] = \frac{1}{2} + \varepsilon$$

  If $Z = R$, the ciphertext is completely random to $\mathcal{A}$, its advantage is:

$$Pr[b' = b | Z = R] = \frac{1}{2}$$

  Finally, the simulator's advantage in this game is described as follows:

$$Adv = \frac{1}{2} Pr[b' = b | Z = e(g, g)^{abc}] + \frac{1}{2} Pr[b' = b | Z = R] - \frac{1}{2} = \frac{\varepsilon}{2}$$

□

5. Performance analysis

This chapter mainly analyzes the performance of the RVOC-CP-ABE scheme through theoretical analysis and experimental simulation.

5.1. Theoretical analysis

Scholars have done a lot of research on the fixed ciphertext length CP-ABE and the CP-ABE that supports revocable storage. We compare the new scheme with other related schemes.

Table 5. Schemes comparison I.

Scheme	$\|CT\|$	$\|SK\|$	Decryption
[20]	$2\|G\| + \|G_t\|$	$2\|G\|$	$2T_e$
[39]	$(4n + 3)\|G\| + 2\|G_t\|$	$(n + 2)\|G\|$	$2T_{G_t}$
[32]	$(2n + 1)\|G\| + \|G_t\|$	$(n + 2)\|G\|$	$2nT_e + nT_{G_t}$
[5]	$(n + 1)\|G\| + \|G_t\|$	$(2n + 1)\|G\| + \|Z_p\|$	$(n + 1)T_e$
[6]	$4\|G\| + \|G_t\|$	$(n + 1)\|G\| + \|Z_p\|$	$\geqslant (n + 1)T_e$
[4]	$4\|G\| + 2\|Z_p\|$	$2\|G\| + 2\|Z_p\|$	$4T_{G_t}$
Ours	$5\|G\| + 2\|G_t\|$	$(n + 1)(\|G\| + \|Z_p\|)$	$4T_{G_t}$

Table 6. Schemes comparison II.

Scheme	Revocable Storage	Revocation Granularity	Revocation Method	Verifiable Outsourcing Decryption	Constant Ciphertext Length	Access Policy
[20]	×	×	×	×	✓	AND_m
[39]	×	×	×	✓	×	LSSS
[32]	✓	User Revocation	Indirect	×	×	Tree
[5]	×	Attribute Revocation	Indirect	×	×	$\mathrm{AND}^*_{+,-}$
[6]	✓	Attribute Revocation	Direct	×	✓	AND^*_m
[4]	✓	User Revocation	Direct	✓	✓	AND
Ours	✓	Attribute Revocation	Indirect	✓	✓	AND^*_m

As shown in Table 5 and Table 6, the main comparison indicators are ciphertext length $|CT|$, users' attribute private key size $|SK|$, decryption cost and implementation function. $|G|$ and $|G_t|$ in the table represent the bit length of one element in the groups G and G_t, n represents the number of attributes contained in the attribute field. For the convenience of comparison, only the bilinear operation and the exponential operation in the decryption algorithm are included in the comparison range, T_{G_t} represents the exponential operation time in the group G_t, and T_e represents the bilinear operation time.

It can be seen from Table 5 that because our scheme achieves the constant ciphertext length, the ciphertext length is only $5|G| + 2|G_t|$. Compared with the scheme [5] that implements the version numbering revocation algorithm and the scheme for implementing verifiable outsourcing decryption [39], the ciphertext length does not increase as the number of attributes in the access policy increases. Scheme [20] has a private key length of only $2|G|$ because of its special access policy. To implement a flexible access policy, the private key will grow linearly with the number of user attributes n increasing. From the perspective of the decryption cost, our scheme will hand over the complex bilinear part which will dynamic change with the attribute set to the outsourcing decryption server, local only requires simple operations $4T_{G_t}$, so the data decryptor's operation and the decryption complexity will be dramatically decreased compared to the schemes [32] [5] and [6] which is affected by the number of revocation.

The AND-gates on positive and negative attributes with wildcards access policy of the scheme [5] and the AND-gates on multi-valued attributes access policy of the scheme [20] are not flexible enough. This scheme realizes the more flexible AND-gates on multi-valued attributes with wildcards access policy. Compared with the scheme [6], this scheme not only realizes the outsourcing decryption but also uses relatively simple revocation algorithm. The revocation algorithm of scheme [6] divides the ciphertext into four types of storage, its decryption algorithm also has four corresponding types, and also stores more revocation parameters, which increases the complexity of the scheme. [32] is the first

scheme to propose the concept of revocable storage. It combines the nature of the ”ciphertext proxy” to directly update the ciphertext when the revocation occurs. Although the revocation algorithm of this scheme refers to scheme [5], [5] does not implement revocable storage, and there are certain security problems. Our scheme also updates the core parameters of ciphertext in the revocation process while realizing the constant ciphertext length to ensure the accuracy of the revocation. Scheme [4] also implements revocable storage. Its ciphertext and key are fixed length, but its revocation granularity only reaches user revocation, our scheme reaches attribute revocation which is the finest granularity. Although our scheme implements indirect revocation, multiple revocations can be completed with only one update, which greatly reduces the overhead of updating ciphertext and private key. So our solution is more suitable for the actual scene.

In general, other schemes have more or fewer problems in implementing the related functions of attribute-based encryption. Our scheme reduces storage cost and operation complexity while solving their shortcomings.

5.2. Experimental simulation

In order to verify the theoretical analysis of the previous section, we use the PBC library developed in C language to simulate the scheme and other schemes. The experimental computer configuration is as follows: Intel i7-7700 processor, 8GB memory and 3.60GHz.

To show the comparison results more clearly, in the figure, we call [5] as Yu’s Scheme, [4] as Zhao’s Scheme, [6] as Zhang’s Scheme, [39] as Lai’s Scheme.

First, we simulated the ciphertext size, as shown in Figure 2. From the storage situation of the scheme [39] [5], the ciphertext size increases linearly with the number of attributes. When the number of attributes is 20, the ciphertext size will reach 8KB. However, the encryption method of our scheme will fix the ciphertext length, which will only occupy 0.4KB. Whether encrypting 1 attribute or multiple attributes, although the encryption complexity will be different (since encryption will only be done once, we don’t consider the encryption cost), the ciphertext obtained last is constant length.

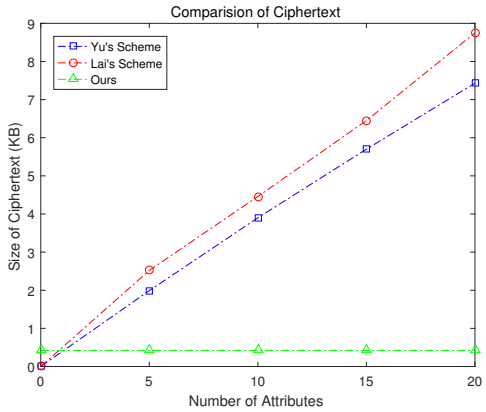

Figure 2. Ciphertext size.

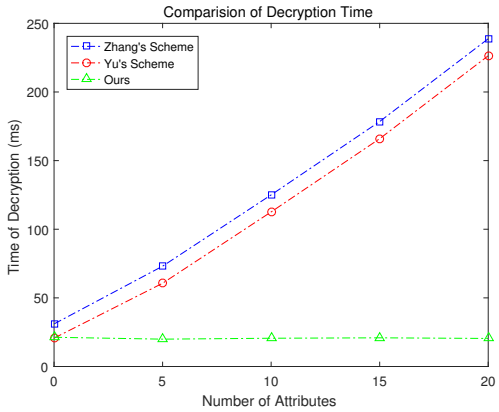

Figure 3. Decryption time.

Then we simulated the decryption, monitoring the decryption time of [5] [6] and our scheme, as shown in Figure 3. The scheme [6] is divided into four types of decryption, and user’s decryption process is related to the revocation user set. We only include the simplest decryption method in the comparison and the actual decryption time of [6] should be more. Under the above conditions, the de-

cryption time of [6] is similar to that of [5]. When the decryption involves 20 attributes, the decryption time is about 230ms. Since this solution implements complex computing outsourcing, the resource-consuming calculation steps are all calculated by the outsourcing server. We only need to perform a simple calculation on the intermediate ciphertext. In theory, only $4T_{G_t}$ is needed (if not verify part for $2T_{G_t}$), and the index operation on the group G_t has less running time than the bilinear pairing operation, so the decryption calculation time of this scheme is only about 20ms which is similar to the scheme [20].

Next, we compared the ciphertext update time of [4], [6] and ours, as shown in Figure 4. We first fixed the total number of users in the system at 30, revoked 5 attributes each time, and observed the impact of the number of revoked users each time on the ciphertext update time. Obviously, the update time of all schemes is inversely proportional to the number of revoked users, because the more users involved in revoking attributes, the fewer relevant parameters of legitimate users that need to be updated. Since different revocation schemes adopt different methods, the space and time for the corresponding revocation are also different. As can be seen from the figure, the time consumption of our scheme is at a medium level. When there are 20 revoked users, the calculation time for updating the ciphertext is about 400ms. However, our solution does not need to update every time, it only updates the ciphertext when a user accesses a ciphertext with a new version, which also makes up for the drawback of time-consuming revocation.

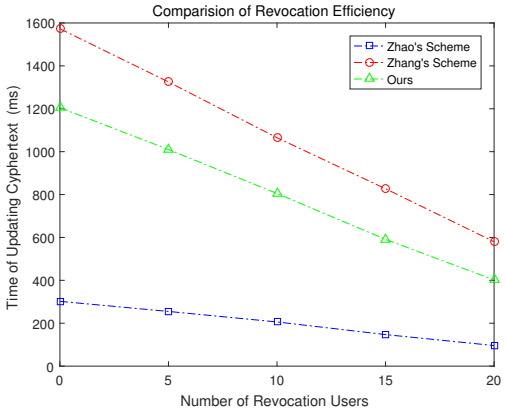

Figure 4. Revocation time.

It can be seen from the specific experimental data that this scheme has certain advantages in decryption calculation and data storage, and the consumption of updated ciphertext is at a medium level.

6. Conclusion

In this paper, we propose a CP-ABE scheme for efficient revocable storage and verifiable outsourcing decryption. With our proposed scheme, the fine-grained attribute revocation is implemented by the version numbering method and the proxy re-encryption technology, while revocable storage ensures forward security of the system. Due to the features of lower storage overhead and fixed ciphertext length, our scheme is more suitable for cloud storage scenarios with multi-attribute and multi-user. Moreover, to reduce the cost of users' local decryption calculations, our scheme securely outsources

complex operations to a powerful third-party server, and users only need to perform a small amount of calculation and verification to obtain the correct information. Performance analysis demonstrates that this scheme has improved in all aspects compared to similar schemes.

In future research, we will consider altering the access structure of this scheme, because the single AND-gate structure is still not flexible enough to satisfy more demanding access control strategies. With the development of technology, malicious attacks are becoming more and more unpredictable, so improving the security of the scheme is also an important research direction.

Acknowledgments

This work was supported in part by Sichuan science and technology project under Grant 2018GZ0236 and Grant 2017FZ0004.

Conflict of interest

The authors declare no conflict of interest.

References

1. K. H. Yeh, A secure transaction scheme with certificateless cryptographic primitives for iot-based mobile payments, *IEEE Syst. J.*, **12** (2018), 2027–2038.
2. Z. Qin, Y. Wang, H. Cheng, et al., Demographic information prediction: a portrait of smartphone application users, *IEEE T. Emerg. Top. Com.*, **6** (2018), 432–444.
3. H. Xiong, H. Zhang and J. Sun, Attribute-based privacy-preserving data sharing for dynamic groups in cloud computing, *IEEE Syst. J.*, 1–22.
4. Y. Zhao, M. Ren, S. Jiang, et al., An efficient and revocable storage cp-abe scheme in the cloud computing, *Computing*, (2018), 1–25.
5. S. Yu, C. Wang, K. Ren, et al., Attribute based data sharing with attribute revocation, in Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, *ACM*, (2010), 261–270.
6. Y. Zhang, D. Zheng, J. Li, et al., Attribute directly-revocable attribute-based encryption with constant ciphertext length, *J. Cryptologic Res.*, **1** (2014), 465–480.
7. Q. Jiang, Y. Qian, J. Ma, et al., User centric three-factor authentication protocol for cloud-assisted wearable devices, *Int. J. Commun. Syst.*, e3900.
8. H. Xiong, Q. Mei and Y. Zhao, Efficient and provably secure certificateless parallel key-insulated signature without pairing for iiot environments, *IEEE Syst. J.*.
9. C. M. Chen, B. Xiang, K. H. Wang, et al., A robust mutual authentication with a key agreement scheme for session initiation protocol, *Appl. Sci.*, **8** (2018), 1789.
10. J. Sun, Y. Bao, X. Nie, et al., Attribute-hiding predicate encryption with equality test in cloud computing, *IEEE Access*, **6** (2018), 31621–31629.

11. H. Xiong, Y. Zhao, L. Peng, et al., Partially policy-hidden attribute-based broadcast encryption with secure delegation in edge computing, *Future Gener. Comp. Sy.*.
12. T. Y. Wu, C. M. Chen, K. H. Wang, et al., A provably secure certificateless public key encryption with keyword search, *J. Chin. Inst. Eng.*, **42** (2019), 20–28.
13. H. Xiong and J. Sun, Comments on verifiable and exculpable outsourced attribute-based encryption for access control in cloud computing, *IEEE T. Depend. Secure*, **14** (2017), 461–462.
14. T. Y. Wu, C. M. Chen, K. H. Wang, et al., Security analysis and enhancement of a certificateless searchable public key encryption scheme for iiot environments, *IEEE Access*, **7** (2019), 49232–49239.
15. H. Xiong, Q. Wang and J. Sun, Comments on circuit ciphertext-policy attribute-based hybrid encryption with verifiable delegation, *Inform. Process. Lett.*, **127** (2017), 67–70.
16. A. Sahai and B. R. Waters, Fuzzy identity-based encryption., in *Eurocrypt*, Springer, **3494** (2005), 457–473.
17. V. Goyal, O. Pandey, A. Sahai, 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, *ACM*, (2006), 89–98.
18. J. Bethencourt, A. Sahai and B. Waters, Ciphertext-policy attribute-based encryption, in Security and Privacy, 2007. SP'07. IEEE Symposium on, *IEEE*, (2007), 321–334.
19. L. Cheung and C. Newport, Provably secure ciphertext policy abe, in Proceedings of the 14th ACM conference on Computer and communications security, *ACM*, (2007), 456–465.
20. K. Emura, A. Miyaji, A. Nomura, et al., A ciphertext-policy attribute-based encryption scheme with constant ciphertext length., in *ISPEC*, Springer, **9** (2009), 13–23.
21. T. Nishide, K. Yoneyama and K. Ohta, Attribute-based encryption with partially hidden encryptor-specified access structures, in *International Conference on Applied Cryptography and Network Security*, Springer, (2008), 111–129.
22. C. Chen, J. Chen, H. W. Lim, et al., Fully secure attribute-based systems with short ciphertexts/signatures and threshold access structures, in *Cryptographers Track at the RSA Conference*, Springer, (2013), 50–67.
23. N. Doshi and D. C. Jinwala, Fully secure ciphertext policy attribute-based encryption with constant length ciphertext and faster decryption, *Secur. Commun. Netw.*, **7** (2014), 1988–2002.
24. J. Herranz, F. Laguillaumie and C. Ràfols, Constant size ciphertexts in threshold attribute-based encryption, in *International Workshop on Public Key Cryptography*, Springer, (2010), 19–34.
25. Y. Zhang, D. Zheng, X. Chen, et al., Computationally efficient ciphertext-policy attribute-based encryption with constant-size ciphertexts, in *International Conference on Provable Security*, Springer, (2014), 259–273.
26. Z. Zhou and D. Huang, On efficient ciphertext-policy attribute based encryption and broadcast encryption, in Proceedings of the 17th ACM conference on Computer and communications security, *ACM*, (2010), 753–755.
27. M. Pirretti, P. Traynor, P. McDaniel, et al., Secure attribute-based systems, *J. Comput. Secur.*, **18** (2010), 799–837.

28. N. Attrapadung and H. Imai, Attribute-based encryption supporting direct/indirect revocation modes, in *IMA International Conference on Cryptography and Coding*, Springer, (2009), 278–300.
29. M. Naor and B. Pinkas, Efficient trace and revoke schemes, in *International Conference on Financial Cryptography*, Springer, (2000), 1–20.
30. D. Boneh, C. Gentry and B. Waters, Collusion resistant broadcast encryption with short ciphertexts and private keys, in *Crypto*, Springer, **3621** (2005), 258–275.
31. A. Lewko, A. Sahai and B. Waters, Revocation systems with very small private keys, in 2010 IEEE Symposium on Security and Privacy (SP), *IEEE*, (2010), 273–285.
32. A. Sahai, H. Seyalioglu and B. Waters, Dynamic credentials and ciphertext delegation for attribute-based encryption, in *Advances in Cryptology–CRYPTO 2012*, Springer, (2012), 199–217.
33. M. Green, S. Hohenberger, B. Waters, et al., Outsourcing the decryption of abe ciphertexts., in *USENIX Security Symposium*, **2011** (2011).
34. J. Li, X. Huang, J. Li, et al., Securely outsourcing attribute-based encryption with checkability, *IEEE T. Parall. Distr.*, **25** (2014), 2201–2210.
35. R. Zhang, H. Ma and Y. Lu, Fine-grained access control system based on fully outsourced attribute-based encryption, *J. Syst. Software*, **125** (2017), 344–353.
36. J. Li, C. Jia, J. Li, et al., Outsourcing encryption of attribute-based encryption with mapreduce, in *International Conference on Information and Communications Security*, Springer, (2012), 191–201.
37. K. Li and H. Ma, Outsourcing decryption of multi-authority abe ciphertexts, *IJ Network Security*, **16** (2014), 286–294.
38. B. Qin, R. H. Deng, S. Liu, et al., Attribute-based encryption with efficient verifiable outsourced decryption, *IEEE T. Inf. Foren. Sec.*, **10** (2015), 1384–1393.
39. J. Lai, R. H. Deng, C. Guan, et al., Attribute-based encryption with verifiable outsourced decryption, *IEEE T. Inf. Foren. Sec.*, **8** (2013), 1343–1354.

AIMS Press