Efficient revocable CP-ABE for big data access control in cloud computing

Praveen Kumar Premkamal*

Department of Computer Applications,
National Institute of Technology,
Tiruchirappalli, India
Email: tvp.praveen@gmail.com
*Corresponding author

Syam Kumar Pasupuleti

Institute for Development and Research in Banking Technology,
Castle Hills, Road No.1, Masab Tank,
Hyderabad, India
Email: psyamkumar@idrbt.ac.in

P.J.A. Alphonse

Department of Computer Applications,
National Institute of Technology,
Tiruchirappalli, India
Email: alphonse@nitt.edu

Abstract: Due to huge volume of big data, cloud is a better choice to store big data. Since the cloud is not trustworthy, privacy and access control is a big concern. Ciphertext policy attribute-based encryption (CP-ABE) is a promising technique to enable both privacy and access control in the cloud. However, directly applying CP-ABE scheme for big data in the cloud is a challenging task because of revocation. Existing CP-ABE with revocation schemes are lacking in efficiency. In this paper, we propose an efficient revocable CP-ABE (R-CP-ABE) scheme for big data access control in cloud using proxy-based updates in which the proxy server performs the ciphertext and secret key updates instead of data owner and data user respectively during revocation. This outsourced updates during revocation reduces the communication and computation overhead of data owner and data users. In security analysis, we prove that our R-CP-ABE scheme is secure against chosen plain-text and user collusion attacks. In addition, we also show that our scheme achieves forward and backward secrecy. The performance analysis demonstrates that our method is efficient when comparing with existing schemes.

Keywords: cloud computing; privacy; access control; CP-ABE; big data; user revocation; attribute revocation.

Reference to this paper should be made as follows: Premkamal, P.K., Pasupuleti, S.K. and Alphonse, P.J.A. (2019) 'Efficient revocable CP-ABE for big data access control in cloud computing', *Int. J. Security and Networks*, Vol. 14, No. 3, pp.119–132.

Biographical notes: Praveen Kumar Premkamal received his MTech in Computer and Information Technology from the Manonmaniam Sundaranar University. He is currently doing his PhD at the National Institute of Technology, Tiruchirappalli. His research interests include cloud computing, big data, and cryptography.

Syam Kumar Pasupuleti received his MTech in Computer Science and Technology from the Andhra University and PhD in Computer Science from the Pondicherry University. He is an Assistant Professor in Institute for Development and Research in Banking Technology (IDRBT), Hyderabad. His research interests are in the area of cloud computing, security and privacy, cryptography and IoT. He is a member of the IEEE.

P.J.A. Alphonse received his MTech in Computer Science from the Indian Institute of Technology, Delhi and PhD in Mathematics and Computer Science from the National Institute of Technology, Tiruchirappalli. He is currently working as a Professor in National Institute of Technology, Tiruchirappalli. His research interests include graph theory and its algorithms, wireless and ad hoc networks, cryptography and network security. He is a life member of the ISTE and ISC.

1 Introduction

Big data is an enormous volume of unstructured, semi-structured and structured data gathered from different sources at rapid speed (Khan et al., 2014). The big data analysis helps the organisation to stay agile and work faster. It is cost effective, and also delivers insightful discovery and optimisation. But, the traditional storage devices are not suitable for big data because it has limited storage space and not scalable. The better alternative to store and process big data is cloud computing.

The cloud storage consent to store and access data anywhere and anytime through the internet and it minimises the storage, maintenance and management cost of organisation by avoiding large capital investment, hardware upgrading expenditure at their premises (Voorsluys et al., 2011). It also supports scalability, so even when the data is produced in a range of capabilities rapidly, cloud can accommodate it.

Although the cloud storage provides huge benefits, the organisations need to address the foremost important security and privacy challenges (Takabi et al., 2010; Gupta et al., 2016) before moving the data in to the cloud such as:

1 access or reveal the private information by untrusted cloud

2 data access by unauthorised users.

Hence, encrypting the data to keep the sensitive information safe and providing the access control is essential. CP-ABE is providing both encryption and access control in cloud environment. In CP-ABE, the data owner encrypts the data under defined access policy and generates ciphertext. The access policy is a Boolean formula which consists of set of predefined attributes. After generating the ciphertext, uploads the ciphertext into the cloud. The decryption is only possible for the user of those attributes that satisfies the access policy in the ciphertext. For example, the data owner defines the access policy to access the data is ((CLOUD AND FACULTY) OR HOD). The data users having the attributes either HOD or (CLOUD and FACULTY) can access the data, that is the head of the department or the faculty who is handling cloud subject can access the data.

However, the security challenge of revocation in CP-ABE scheme limits its practical applications. Revocation handling is a challenging issue in CP-ABE scheme because a single attribute that shared by different users. So the revoking a single attribute affects the other legitimate users those who are having the same attribute. The revocation may be either user revocation or attribute revocation. The user revocation deals with the user exits from the system. The attribute revocation deals with the dynamic requirement of changing user's attributes, for example the user may be promoted from one designation to another designation or transferring from one department to another department.

In user revocation, the user who exits from the system must be prohibited to access the previously accessed data as well as subsequently outsourced data. At the same time, when some attributes are revoked from the user, the user must provide appropriate access permissions according to the new attribute set. In addition, we also restrict the attribute revoked user to access the previously accessed data if their attributes do not satisfy the attribute set. Further, the revocation scheme must support forward secrecy and backward secrecy. The forward secrecy means that the newly joined user must be able to obtain the plaintext from the previously uploaded data in the cloud if their attributes satisfies the access policy. The backward secrecy means that the revoked user must not obtain the plaintext from the subsequently uploaded data in the cloud which requires the revoked attributes. Thus, the CP-ABE should support user and attribute revocations in order to maintain the access control effectively (Kumar et al., 2018).

Various efforts have been taken to address CP-ABE with the revocation issue in the literature (Xu and Martin 2012; Xiao et al., 2015; Xu et al., 2016; Balani and Ruj, 2014; Liu et al., 2014; Sultan and Barbhuiya, 2016; Wang et al., 2011; Yang et al., 2013; Yang and Jia, 2014; Zhang et al., 2016; Zhou et al., 2017; Hur and Noh, 2011; Xie et al., 2013; Wang et al., 2017; Li et al., 2018; Liu et al., 2018). Nevertheless, these schemes do not directly apply for big data in the cloud because of the following issues such as:

1 The ciphertext update depends on the untrusted cloud server.

2 Not all schemes support both forward and backward secrecy.

3 Higher computation overhead during encryption and decryption process.

Moreover, there are three main methods that are available in the literature to address the revocation issue such as temporal-based (Balani and Ruj, 2014; Liu et al., 2014; Sultan and Barbhuiya, 2016), key encryption key (KEK) tree or binary tree-based (Hur and Noh, 2011; Xie et al., 2013; Wang et al., 2017; Li et al., 2018; Liu et al., 2018) and update keying method (Wang et al., 2011; Yang et al., 2013; Yang and Jia, 2014; Zhang et al., 2016; Zhou et al., 2017).

In the temporal-based revocation schemes, the access structure is associated with the time period. After the time period, the user secret key is expired and the new key will be given to all non-revoked users. The temporal-based revocation schemes do not provide the immediate revocation, and it also only supports user revocation. The KEK tree-based revocation schemes provide effective attribute revocation using group key distribution. Here, the group key also needs to be encrypted and decrypted. This required additional computational overhead during encryption and decryption. The update keying method provides an efficient attribute revocation. In this method, only required portion of ciphertext and secret key are updated rather re-encrypted entirely. However, when revocation occurs, the non-revoked users necessitate to update their secret key, which increases their

communication and computation overhead. Thus, it requires to design an efficient CP-ABE scheme which supports user and attributes revocations for big data access control in cloud.

In this paper, we propose an efficient R-CP-ABE scheme for big data access control in cloud computing which supports user and attribute revocations. The salient features of our R-CP-ABE scheme are given below.

1 Our R-CP-ABE scheme reduces the data owners and data users communication and computation overhead by outsourcing the ciphertext update and secret key update to the proxy server during revocation.

2 Our R-CP-ABE scheme also reduces the storage overhead of data users through storing the part of user secret key in the proxy server.

3 Our R-CP-ABE scheme is secure against chosen plain-text and user collusion attacks. In addition, it aids to accomplish both forward and backward secrecy.

4 Our R-CP-ABE scheme achieves better efficiency than the existing system because of short ciphertext and minimum pairing operations are required.

The rest of the paper is organised as follows: the related works in the literature are discussed in Section 2. Section 3 gives the mathematical background, basic definitions and security model required to design our scheme and Section 4 describes the proposed scheme system model. The detailed algorithmic construction is given in Section 5. The security and performance of the proposed scheme is proved in Section 6 and Section 7, respectively. The paper is concluded in Section 8.

2 Related works

Sahai and Waters (2005) initially put forward the attribute-based encryption (ABE) scheme named as fuzzy identity-based encryption which was an extension of identity-based encryption. Key-policy ABE (KP-ABE) and CP-ABE are the two diverse approaches of ABE. Goyal et al. (2006) initially proposed the KP-ABE. In KP-ABE, the message is encrypted with the set of attributes and the user secret key associated with defined access policy. The CP-ABE scheme was initially proposed by Bethencourt et al. (2007) with access tree as access policy. CP-ABE is the better option than KP-ABE for outsourcing environment because the data owner has the control over the data. Waters (2011) designed a new CP-ABE scheme with linear secret sharing scheme (LSSS) as access structure. Recently, Li et al. (2017) enhanced the CP-ABE scheme by proposing a non-monotonic access structure ordered binary decision diagram. All of the above basic CP-ABE schemes are not suitable for practical applications because of revocation issues.

Xu and Martin (2012) proposed user revocation scheme without key update. In this scheme they divided secret share into two, one is used for generating secret key and another one is used by the cloud. Whenever the user wants to access the data, the user sends a request to the cloud. After getting the request, the cloud checks whether the user is revoked user or not. If not, then the cloud will issue another part of secret share. The major issue of this scheme is the part of the secret distribution depending on untrusted cloud. Balani and Ruj (2014) proposed the temporal access control-based user revocation scheme which means the access structure is associated with access time. Here, the part of decryption process was outsourced, but this scheme suffered with high computation overhead. Liu et al. (2014) and Sultan and Barbhuiya (2016) proposed CP-ABE scheme with user revocation using proxy re-encryption method in which the access structure is associated with access period. Here, the user's access right expires automatically after predetermined periods, but it only supports user revocation. But, the major issue of Liu et al. (2014) and Sultan and Barbhuiya (2016) schemes are that the revoked users are able to get the access to the previously accessed data until the end of the period. Wang et al. (2011) designed hierarchical attribute-based encryption scheme that supports user revocation. They used proxy re-encryption and lazy re-encryption techniques to address the user revocation. Recently, Wang et al. (2017) proposed efficient directly revocable CP-ABE scheme, but this scheme does not support attribute revocation.

Xiao et al. (2015) constructed the attribute revocation scheme without proxy re-encryption for big data access control, but this scheme requires re-generating the revoked user secret key for the revoked user again. Yang et al. (2013) proposed multi-authority CP-ABE scheme with attribute revocation using the update keying method. In this scheme, decryption task is outsourced to the cloud and the update key is generated for all the non-revoked users separately which increases the communication and computation complexity. Hong et al. (2015) proved that Yang et al. (2013) scheme failed to support the user collusion attack. Yang and Jia (2014) constructed an improved multi-authority CP-ABE scheme with attribute revocation as same like Yang et al. (2013) scheme except decryption outsourcing. This scheme also suffered with user collusion attack and it failed to achieve the forward and backward secrecy. Zhou et al. (2017) proposed improved version of Yang and Jia (2014) scheme, but this scheme suffered with efficiency. Xu et al. (2016) achieved the effectiveness in multi-authority CP-ABE scheme using weighted access policy with revocation using proxy re-encryption method. Zhang et al. (2016) proposed the CP-ABE scheme with both user revocation and attribute revocation using the update keying method, but their scheme suffered with more decryption time. However, in all the update keying schemes, the user have to update their key during revocation which increases the data user's computation and communication overhead.

Hur and Noh (2011) proposed the KEK tree or binary tree-based user revocation and attribute revocation scheme for data outsourcing system, but this scheme suffered in collusion attack. The generic group model is used to prove

the security of this scheme, so it is not possible to use in practical applications. Xie et al. (2013) improved the efficiency of Hur and Noh (2011) scheme during the key update process. Li et al. (2018) proposed a scheme that is based on Hur and Noh (2011) scheme which solved the collusion attack issue. Liu et al. (2018) proposed practical ABE, which address the attribute revocation. However, all the above KEK tree-based schemes suffered with higher computation overhead during encryption and decryption process.

3 Preliminaries

In this section, we present the mathematical background, definitions and security model which are required to design the R-CP-ABE scheme.

3.1 Bilinear pairing

Definition 3.1: A bilinear mapping function over additive cyclic group G_s and multiplicative cyclic group G_T of prime order p is defined as $\ell : G_S \times G_S \to G_T$, which satisfies the below properties (Bethencourt et al., 2007; Waters, 2011).

1 Bilinearity: for $g \in G_S$, $a, b \in Z_p$, $\ell(g^a, g^b) = \ell(g, g)^{ab}$ where g be the generator of group G_S.

2 Non-degeneracy: $\ell(g, g) \neq 1$.

3 Computability: there exists an efficient algorithm to compute $\ell(g, g)$.

3.2 Decisional bilinear Diffie-Hellman assumption

Definition 3.2: Let G_S and G_T are group of prime order p. Let g be the generator of group G_S and $a, b, c \in Z_p$. We define the decisional bilinear Diffie-Hellman (DBDH) problem as follows.

For the given input $g, g^a, g^b, g^c \in G_S$, the adversary must distinguish a valid tuple $\ell(g, g)^{abc} \in G_T$ from the random element $R \in G_T$. An algorithm $\mathfrak{B}$ that outputs $\sigma \in \{0, 1\}$ has advantage ϵ in solving the DBDH problem in G_S (Waters, 2011) if:

$$\left| P_r\left[\mathfrak{B}\left(g, g^a, g^b, g^c, \ell(g, g)^{abc} = 0\right)\right] - P_r\left[\mathfrak{B}\left(g, g^a, g^b, g^c, R = 0\right)\right]\right| \geq \epsilon.$$

Definition 3.3: The DBDH assumption holds if no probabilistic polynomial time algorithm has a non-negligible advantage in solving DBDH problem (Waters, 2011).

3.3 Shamir secret sharing

Shamir presented the effective secret sharing method in which secret s is divided into shares and is given to M participants. Out of M participants, N participants can combine together and reconstruct the secret value s if $N \leq M$, but if there are less than N participants, then it cannot reconstruct the secret s which is accomplished by polynomial interpolation. There are two stages such as share construction phase and share reconstruction phase. The following steps describe the share construction.

1 Select the secret value s.

2 Choose the random polynomial $f(x)$ of degree $N - 1$ where the coefficients are random positive numbers $a_1, a_2, \ldots, a_{N-1}$. Let $f(x) = s + a_1x_1 + \ldots + a_{N-1}x^{N-1}$, where $f(0) = s$.

3 Compute share value s_i and give the share value to M participants. Share value $s_i = f(x_i)$ where $i = 1$ to M.

Using Lagrange interpolation formula, reconstruct the secret s value from N share values. The formula is given below.

$$f(x) = \sum_{j=1}^{N} f_j(x)$$

$$f_j(x) = s_j \prod_{k=1, k\neq j}^{N} \frac{x - x_k}{x_j - x_k}$$

After computing $f(x)$, substitute $x = 0$ to get the secret value $f(0) = s$. The participants role is taken by the attributes in CP-ABE schemes. The Shamir secret sharing technique is used to provide access control in CP-ABE schemes with tree access structure.

3.4 Access tree

In our scheme, we employ the access tree as access policy. The access tree consists of attributes in leaf nodes and threshold gates in non-leaf nodes. All the child nodes are assigned with numbers beginning with 1 and ends with n. Let C_x be the available children for the node x. Let r_x be the threshold value of a node x and it satisfies $0 < r_x < C_x$. If r_x is 1 then the threshold gate is OR gate, and if $r_x = C_x$ then the threshold gate is AND gate. Let us define the functions which facilitate the functioning of access tree. The parent(x) function returns the parent of the node x in the access tree. The att(x) function returns the attribute corresponding to the node x in the access tree. The purpose of an index(x) is to return the integer value associated with the node x.

Definition 3.4: Satisfying an access tree: Let T be an access tree with root r. T_x denotes the subtree of T rooted at the node x. Hence, T is the same as T_r. If a set of attributes Γ satisfies the access tree T_x, then it denotes $T_x(\Gamma) = 1$. We compute $T_x(\Gamma)$ recursively as follows; if x is a non-leaf node, evaluate $T'_x(\Gamma)$ for all the children x' of node x. $T_x(\Gamma)$ returns 1 if and only if at least r_x children return as 1. If x is a leaf node, then $T_x(\Gamma)$ returns 1 if and only if att(x) $\in \Gamma$ (Bethencourt et al., 2007).

3.5 Security model

Here, we have defined a security game between adversary ($\hat{A}$) and challenger (ς) to prove the security of our R-CP-ABE scheme against chosen plain-text (CPA). The steps involved in this game are given below.

- *Initialising game:* the $\hat{A}$ selects the defy access structure (τ^*) and provides the same to the ς. *Setup phase:* the ς executes the algorithm SystemSetup and generates the public key (PK) and master secret key (MSK). After generating, the ς sends the PK to the $\hat{A}$.
- *Query phase 1:* the $\hat{A}$ requests the secret key for any attributes set Γ^* to the ς. The ς generates the secret key for the given set of attributes by executing KeyGeneration algorithm and sends to the $\hat{A}$.
- Challenge: $\hat{A}$ selects the two messages M_0 and M_1 and gives it to the ς. The size of the messages M_0 and M_1 must be same. The ς receives message and generates the random bit value $\sigma \in \{0, 1\}$. Based on the σ value, the challenger chooses the message and encrypts it. The ς returns the C_σ = Encryption (M_σ, τ^*, PK) to the $\hat{A}$.
- *Query phase 2:* the $\hat{A}$ could request further secret keys from ς as similar to query phase 1 and the ς also does the same as in query phase 1.
- *Guess:* the $\hat{A}$ must submit the guess $\sigma' \in \{0, 1\}$ for σ. The $\hat{A}$ wins the game when $\sigma' = \sigma$. The adversary's advantage of winning the game is defined as $\left(\Pr[\sigma = \sigma'] - \frac{1}{2}\right)$.

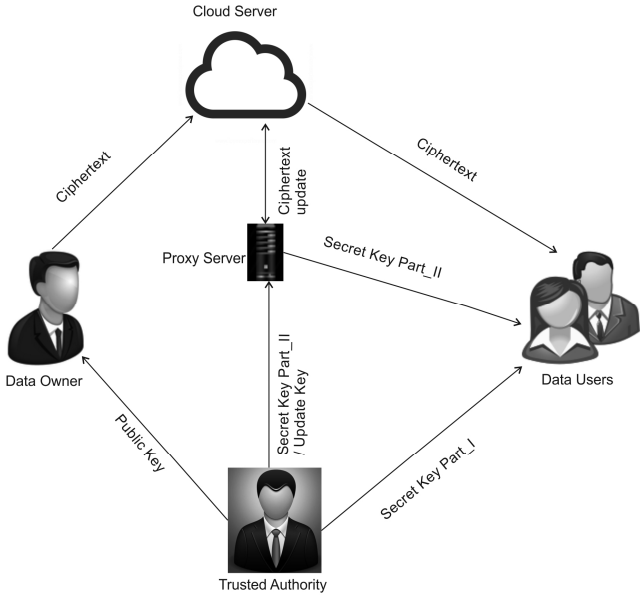
Definition 3.5: The proposed R-CP-ABE scheme is said to be secure against CPA if no probabilistic polynomial time adversaries have non-negligible advantage in the above game (Waters, 2011).

4 System model

The architecture of the proposed model has five different actors which are data owner, data users, trusted authority, cloud server and proxy server as shown in Figure 1.

- *Trusted authority* (*TA*)*:* authority creates the PK, MSK, user identification (UID) number, user secret key according to their attributes, and it also generates update key whenever revocation occurs.
- *Data owners* (*DO*)*:* the fully trusted data owner performs data encryption using PK and defined access policy to preserve the privacy of the data and uploads the data into the cloud.
- *Data users* (*DU*)*:* the users obtain the plain-text from the ciphertext if the user attributes satisfies the access policy in the ciphertext. The data users are untrusted in our scheme.
- *Cloud server* (*CS*)*:* the untrusted cloud server consists of huge storage capacity in which the data owners store their data and gives the data access to the non-revoked data users only. The cloud server also maintains the proxy server.
- *Proxy server* (*PS*)*:* the proxy server is the dedicated entity for the organisation in the cloud. Hence, it is semi-trusted entity. The semi-trusted means honest but curious. The proxy server will perform the assigned task correctly, but it is curious to obtain the plain-text. The proxy server performs the ciphertext and secret key updates on behalf of the data owner and data user respectively, during revocation while storing one portion of the user's secret key.

Figure 1 Architecture of our R-CP-ABE scheme



4.1 Objectives

The major intention of the proposed work is to develop efficient R-CP-ABE scheme for big data access control in the cloud with the following objectives.

- *Privacy:* the personal and sensitive data must be kept safe without disclosing it to the others. It is also necessary to secure data against chosen plain-text attack and user collusion attack.
- *Access control:* give the permission to read the data only for the authorised users of the system.
- *Revocation:* both user revocation and attribute revocation should support in order to maintain the access control effectively. It also supports forward and backward secrecy.

- *Efficiency:* improve the efficiency of the scheme by reducing storage, communication and computation overhead.

4.2 Security threats

In our R-CP-ABE scheme, we consider the following threats that breach the security of our scheme by attackers.

1 Chosen plain-text attack: the adversary gets the ciphertext for arbitrary plain-text and adversary reveals all or part of the information from the ciphertext.

2 User collusion attack: two or more users combined their secret key and try to obtain the data.

4.3 Framework

The framework of proposed scheme consists of nine algorithms. Here, we explain the high level design of nine algorithms.

- SystemSetup(d) → (PK, MSK): this algorithm is executed by trusted authority. It inputs the security parameter d and produces the PK and MSK as output.
- UserSetup(UL) → UID: whenever the user joins the system, TA executes this algorithm. This algorithm generates the unique UI number and add the UID into the user list (UL). It takes UL as input and outputs UID and updated UL.
- Encryption(M, PK, τ) → C: the data owner performs this algorithm. It takes the input message (M), PK, and an access policy (τ) and it returns the ciphertext (C).
- KeyGeneration(MSK, UL, UID, Γ) → S(S_Key1, S_Key2) or ϕ: TA runs and creates the secret key (S) for the given input MSK, user id (UID), UL and set of user's attributes (Γ). The key is generated separately for every registered user. It returns ϕ value for non-registered users.
- Decryption(C, S_Key1, S_Key2) → M or ϕ: the data user executes this algorithm with ciphertext (C), users secret key S_Key1 and S_Key2 as an input and returns the message (M) or null value ϕ.
- UUpdateKeyGen(UID, UL, RUL, PK, MSK) → upd_keys (flag, uk_ct, uk_sk): whenever the users exit the system, TA executes the user update key generation (UUpdateKeyGen) algorithm. It intakes the user id (UID), UL, revoked user list (RUL), PK, and MSK as input and outputs the update key (upd_keys) which consists of *flag* ='USR', ciphertext update key *uk_ct*, and secret update key *uk_sk* for all non-revoked users in the system.
- AUpdateKeyGen(UID, UL, RUL, PK, MSK, old_atts, new_atts) → upd_keys (flag, uk_ct, uk_sk, uk_sk_au, old_atts): whenever the users revoked the attributes, TA executes attribute update key generation (AUpdateKeyGen) algorithm. It inputs the user id (UID), UL, RUL, PK, MSK, set of revoked attributes *old_atts*, set of new attributes *new_atts* and outputs the update key (upd_keys) which consists of *flag* = 'ATT', ciphertext update key *uk_ct*, users secret key update key *uk_sk* for all attribute non-revoked users, update key for attribute revoked user *uk_sk_au* and revoked attribute list *old_atts*.
- CTUpdate(C, upd_keys) → C′: ciphertext update algorithm is executed by proxy server. It accepts the ciphertext (C), the update key (upd_keys) as input and returns the updated ciphertext (C′).
- SKUpdate(S_Key2, upd_keys) → S_Key2′: secret key update (SKUpdate) algorithm is run by proxy server. It inputs the users secret key S_Key2, the update key (upd_keys) and returns the updated user secret key S_Key2′.

5 R-CP-ABE scheme construction

The R-CP-ABE scheme contains five different phases namely setup, user key generation, encrypting data, decrypting the data, and revocation. The algorithmic constructions of our R-CP-ABE scheme are as follows.

5.1 Setup

In this phase, TA generates the PK, MSK and UID number using SystemSetup and UserSetup algorithms respectively.

5.1.1 SystemSetup

Trusted authority gets the security parameter d and creates the PK and MSK. Let U be the universal attribute set. Let UL be the UL in the system and RUL is the RUL. Let G_S, G_T are the bilinear group of prime order p and g be the generator of group G_S. Let $\ell : G_S \times G_S \to G_T$, which denotes the bilinear map.

1 Initialise UL = ϕ, RUL = ϕ.

2 Select random value α, β from Z_p.

3 Choose random value t_j for each attributes in U from Z_p and compute $T_j = g^{\beta . t_j}$, j = 1 to n; n denotes the number of attributes in U.

4 Compute $Y = \ell(g, g)^{\alpha}$.

5 Return PK = (g, Y, T_j: j = 1, 2, …, n), MSK = (α, β, (t_1, t_2, t_3, …, t_n)).

After generating the PK and MSK, TA sends the PK to the data owner and keeps the MSK. TA stores the UL into the cloud.

5.1.2 UserSetup

Trusted authority creates unique UID number for each new user and add the new UID number into the UL in the cloud.

First TA retrieves the UL from cloud and gives the UL as input to this algorithm.

1 Generates unique identification number (UID) from Z_p and adds the UID in UL

2 Return UID to user and stores the UL back in the cloud.

5.2 User key generation

In this phase, the TA creates user secret key for all users using KeyGeneration algorithm.

KeyGeneration

The trusted authority creates the secret key for every user with the inputs of MSK, UL, UID, and the set of user's attributes Γ.

1 If UID ∈ UL, then performs the following otherwise return S = ϕ.

2 Choose random number $\gamma \in Z_p$.

3 Compute $S\_Key1 = g^{\alpha+\gamma}$.

4 For each attribute a_j in Γ, compute $SK_j = g^{\frac{\gamma}{\beta.t_j}}$.

5 S_Key2 = (UID, SK_j: ∀ $a_j \in \Gamma$).

6 Return S = (S_Key1, S_Key2).

After secret key generation, TA sends one part (S_Key1) to user and stores another part (S_Key2) in the proxy server. Even though the part of secret key (S_Key2) is stored in the proxy server, the PS is not able to decrypt the data because a part of the key (S_Key1) is with the user. Without S_Key1, the PS cannot get the value $\ell(g, g)^{\alpha s}$ exactly.

5.3 Encrypting the data

The data owner encrypts the data and uploads the ciphertext into the cloud.

Encryption

The data owner encrypts the message (M) using PK under the defined access structure τ as mentioned below.

1 Choose random number $s \in Z_p$.

2 Compute $CT_1 = g^s$.

3 Compute $CT_2 = M.Y^s = M.\ell(g, g)^{\alpha s}$.

4 Lets r0 be the root node of access tree and r_x is the threshold value of the node x. Perform the following steps for each node of the access tree starting from the root node.

a if x = r0 then

- Polynomial q_{r0} is chosen randomly with the degree $r_{r0} - 1$ and $q_{r0}(0) = s$.
- The secret s is divided into number of shares according to the number of child nodes available for root node r0. Find out the shares s_y for each child node using secret sharing scheme $s_y = q_{r0}(\text{index(y)})$, y denotes the child node attributes of r0.

b else

- Polynomial q_x with the degree $r_x - 1$ is chosen randomly for every child node.
- Check x is leaf node or not, if leaf node compute $q_x(0)$ as $q_x(0) = s_x = q_{parents}(x)(\text{index(x)})$.
- For all non-leaf nodes x, again s_x is further divided into shares s_y using secret sharing scheme and assigning it to each child node. $S_y = q_x(\text{index(y)})$.

5 Let LN denotes all the available leaf nodes in the access tree and for each leaf node x in LN, Compute CT_j $CT_j = T_j^{q_x(0)} = g^{\beta.t_j q_x(0)}$.

6 Return ciphertext C, C = (τ, CT_1, CT_2, ∀ $x \in$ LN: CT_j).

5.4 Decrypting the data

Whenever the user wants to access the plain-text, first the user sends the request to the cloud for ciphertext. The cloud verifies whether the user is non-revoked user or not by checking the user id in the UL. If the user is non-revoked user, then the cloud sends the ciphertext. The revoked user will not get the ciphertext from the cloud. After getting the ciphertext from the cloud, the user also gets the part of secret key S_Key2 from the PS, and executes decryption algorithm.

Decryption

The user executes this algorithm with the input of C, S_Key1 and S_Key2. Decryption algorithm returns null value ϕ for the following case 1 and case 2 or it returns message (M).

Case 1 The users revoked from the system are not able to decrypt the data since the TA deletes their part of secret key S_Key2 which is available in the proxy server during the revocation event. If PS returns no secret key then this algorithm returns ϕ. Moreover, the revoked user will not get the ciphertext from the cloud.

Case 2 The user, whose attributes does not satisfies the access policy in C is also not able to decrypt, so it returns ϕ.

Case 3 If users attributes satisfies the access policy then they can decrypt the data and get M. Let us first find out the secret share value of all nodes in the access tree. Let's use SV_x to store the output.

1 For each leaf nodes $x \in LN$, LN denotes the available leaf nodes in the access tree and let a_j be the attribute related with leaf node x. If $a_j \notin \Gamma$ then returns ϕ, otherwise compute SV_x.

$$\begin{aligned} SV_x &= \ell\left(SK_j, CT_j\right) \\ &= \ell\left(g^{\frac{\gamma}{\beta.t_j}}, g^{\beta t_j q_x(0)}\right) \\ &= \ell(g,g)^{\frac{\gamma}{\beta.t_j}\beta t_j q_x(0)} \\ &= \ell(g,g)^{\gamma q_x(0)} \end{aligned}$$

2 Perform this step from top to bottom in the access tree. For each non-leaf nodes in the access tree, compute the share values using Lagrange interpolation formula. Let ρ denote all child nodes of non-leaf node x and Γ_x be the attributes of ρ. If all child nodes share value of node x is ϕ, then $SV_x = \phi$, otherwise compute SV_x. Let $i = \text{index}(\rho)$ and $\Gamma'_x = \{\text{index}(\rho) : \rho \in \Gamma_x\}$.

$$\begin{aligned} SV_x &= \prod_{\rho \in \Gamma_x} \left(SV_p\right)^{\triangle_{i,\Gamma'_x}(0)} \\ &= \prod_{\rho \in \Gamma_x} \left(\ell(g,g)^{\gamma q_x(0)}\right)^{\triangle_{i,\Gamma'_x}(0)} \\ &= \prod_{\rho \in \Gamma_x} \left(\ell(g,g)^{\gamma q_{parent(\rho)}(index(\rho))}\right)^{\triangle_{i,\Gamma'_x}(0)} \\ &= \prod_{\rho \in \Gamma_x} \ell(g,g)^{\gamma q_\rho(i)\triangle_{i,\Gamma'_x}(0)} \\ &= \ell(g,g)^{\gamma q_x(0)} \end{aligned}$$

3 If $SV_{r0} = \phi$, then this algorithm returns ϕ which denotes that the user attributes are not satisfying the access policy τ. Otherwise it decrypts the message as follows:

$$\begin{aligned} M &= \frac{CT_2}{\ell\left(S\_Key1, CT_1\right) / SV_{r0}} \\ &= \frac{CT_2}{\ell\left(g^{(\alpha+\gamma)}, g^s\right) / SV_{r0}} \\ &= \frac{CT_2}{\ell\left(g,g\right)^{(\alpha+\gamma)s} / \ell\left(g,g\right)^{\gamma q_{r0}(0)}} \\ &= \frac{CT_2}{\ell\left(g,g\right)^{\alpha s}\, \ell\left(g,g\right)^{\gamma s} / \ell\left(g,g\right)^{\gamma q_{r0}(0)}} \\ &= \frac{CT_2}{\ell\left(g,g\right)^{\alpha s}\, \ell\left(g,g\right)^{\gamma s} / \ell\left(g,g\right)^{\gamma s}} \\ &= \frac{M.\ell\left(g,g\right)^{\alpha s}}{\ell\left(g,g\right)^{\alpha s}} \end{aligned}$$

4 Return M.

5.5 Revocation

This phase addresses the user and attributes revocations. The CTUpdate and SKUpdate algorithms are common for both user and attribute revocations. The parameter *flag* is used to differentiate user or attribute revocation in the CTUpdate and SKUpdate algorithms. If *flag* value is 'USR' then it denotes user revocation and if the *flag* value is 'ATT' then it refers attribute revocation.

5.5.1 User revocation

Whenever the users exit the system, TA deletes the revoked users secret key S_Key2 which is available in the proxy server and generates the update keys using UUpdateKeyGen algorithm. After generating update key, it sends update key to the proxy server. The proxy server executes the ciphertext update algorithm and users secret key update algorithm. In order to prevent the data access of revoked users with their old secret key, it is required to update the ciphertext and secret key of non-revoked users.

UUpdateKeyGen

TA performs the following steps using the inputs UID, UL, RUL, PK, and MSK:

1 Delete the secret key S_Key2 in the proxy server for all revoked users.

2 Add the revoked user ids UID to RUL and delete the UIDs from UL.

3 Set the flag = 'USR' and choose random number $\beta' \in Z_p$.

4 Compute ciphertext update key uk_ct = β'/β and secret key update key uk_sk = $g^{\beta/\beta'}$.

5 Update T_j value in PK for all attributes; $T_j = (T_j)^{uk\_ct}$.

6 Update the β value in MSK as β'.

7 Return upd_keys(flag, uk_ct, uk_sk).

After the update key generation, the TA sends the upd keys (flag, uk_ct, uk_sk) to the proxy server.

CTUpdate

Upon receiving the update key from TA, the proxy server gets the ciphertext C from cloud and updates the component CT_j of C using uk_ct in upd_keys as follows.

1 If flag = 'USR' then

a compute $CT'_j = \left(T_j^{q_x(0)}\right)^{uk\_ct} = \left(g^{\beta.t_j.q_x(0)}\right)^{uk\_ct}$

2 If flag = 'ATT' then

a compute

$CT'_j = \left(T_j^{q_x(0)}\right)^{uk\_ct} = \left(g^{\beta.t_j.q_x(0)}\right)^{uk\_ct_j}$;

$\forall x \in old\_atts,\ j = att(x)$

b $CT'_j = CT_j$; $\forall x \notin old\_atts,\ j = att(x)$

3 Return $C' = (CT_1 = CT_1, CT_2 = CT_2, CT'_j)$.

SKUpdate

The proxy server updates the user's secret key S_Key2 which is available with PS itself after receiving update key from TA. The non-revoked users key need to be updated during user revocation, but both revoked and non-revoked users key need to be updated during attribute revocation.

1 If flag = 'USR' then // non-revoked users
   a For all S_Key2 available in the proxy server, update $SK_j = \left(g^{\frac{\gamma}{\beta.t_j}}\right) * uk\_sk.$

2 If flag = 'ATT' then For $\forall\, x \in$ old_atts, j = att(x)
   a Update for attribute revoked user as $SK'_j = \left(g^{\frac{\gamma}{\beta.t_j}}\right) * uk\_sk\_au_j.$
   b Update for attribute non-revoked users as $SK'_j = \left(g^{\frac{\gamma}{\beta.t_j}}\right) * uk\_sk_j.$

3 Returns $S\_Key2' = (\text{UID}, (SK_j = SK'_j))$.

5.5.2 *Attribute revocation*

Whenever the user revoked the attributes, TA initiates the update process and update the user attributes. The TA generates update key using AUpdateKeyGen algorithm and sends the update key to the proxy server. The proxy executes the ciphertext update algorithm and users secret key update algorithms. The new random attribute value is generated for all revoked attributes and it updates the new attribute value in ciphertext and non-revoked users secret key in order to avoid the revoked users data access with their old secret key.

AUpdateKeyGen

TA does the following step using the inputs UID, UL, RUL, PK, MSK, old_atts and new_atts:

1 If UID ∈ UL and UID ∉ RUL then
   a Generation of update key for attribute revoked user's uk_sk_au.
      - Perform the following: $\forall\, x \in$ old_atts, j = att(x) and $\forall\, y \in$ new_atts, k = att(y), Fetch the attribute random number (t) generated in setup phase corresponding to j and k and compute $uk\_sk\_au_j = g^{t_j/t_k}$
   b Generation of update key for ciphertext uk_ct and secret key for all attribute non-revoked users uk_sk.
      - Choose a new random value t'_j for all revoked attributes in old_atts.
      - Compute $\text{uk\_ct}_j = t'_j / t_j$ and $\text{uk\_sk}_j = g^{t_j/t'_j}$; $\forall\, x \in$ old_atts, j = att(x).
      - Update the t_j as t'_j in MSK; $\forall$ x ∈ old_atts, j = att(x).
      - Update T_j in PK. $T_j = (g^{\beta.t_j})^{uk\_ct_j}$; $\forall\, x \in$ old_atts, j = att(x).

2 Return upd_keys(flag = 'ATT', uk_ct = uk_ct_j, uk_sk = uk_sk_j, uk_sk_au = (UID, uk_sk_au_j), old_atts).

After the update key generation, the TA sends the upd_keys (flag = 'ATT', uk_ct, uk_sk, uk_sk_au, old_atts) to proxy server. The proxy server executes the same CTUpdate and SKUpdate algorithms in the user revocation.

6 Security analysis

In this part, we propose the security evidence for the efficient R-CP-ABE scheme against the CPA, user collusion attack and also prove that our scheme supports both forward and backward secrecy.

Theorem 1: Polynomial-time adversary $\hat{A}$ cannot selectively crack R-CP-ABE scheme with a challenge access structure in the security game mentioned in Section 3.5 if DBDH hold its assumption.

Proof: In the proof, we have reduced security of the chosen-plaintext attack into the DBDH assumptions. It implies that if DBDH problem is solved, then the adversary can breach this scheme successfully. Construct the simulator (χ) which performs the challenger task. If $\hat{A}$ wins this selectively CPA secure game with a non-negligible advantage ε, then the χ can solve the DBDH assumptions in $\frac{\epsilon}{2}$.

The challenger (ς) creates a basic setup to play game. Let G_S and G_T be the prime order group p and let g be the generator of G_S. The bilinear mapping function defined as $\ell : G_S \times G_S \to G_T$ and pick a, b, $c \in Z_p$ as random values. Let $\Re$ be the random value of G_T. The ς toss the coin $\sigma \in \{0, 1\}$ and defines $\wp = \ell(g, g)^{abc}$ when $\sigma = 0$, otherwise define $\wp = \Re$. Now the ς gives the basic setup to simulator and the responsibility to play the game. The challenge given to the simulator is $(g, A, B, C, \wp) = (g, g^a, g^b, g^c, \wp)$. Now the challenger for adversary is simulator and simulator starts playing the game with the adversary.

Initialising game: The $\hat{A}$ selects the defy access structure τ^* and gives the same to χ.

Setup phase: In this phase, χ generates the PK and sends it to $\hat{A}$. For generating PK, the simulator randomly chooses $\zeta \in Z_p$ and let $\alpha = ab + \zeta$.

Compute $\ell(g, g)^{\alpha} = \ell(g, g)^{(ab+\zeta)} = \ell(g, g)^{ab}\, \ell(g, g)^{\zeta}$.

Choose a random number for all attributes in adversary access tree $k_j \in Z_p$, and let $t_j = b/k_j$ if $a_j \notin \tau^*$, $t_j = k_j$ if $a_j \in \tau^*$.

Compute $T_j = g^{\beta t_j} = g^{\beta k_j}$ if $a_j \in \tau^*$; $T_j = g^{\beta t_j} = g^{\frac{\beta b}{k_j}} = B^{\frac{\beta b}{k_j}}$ if $a_j \notin \tau^*$.

The simulator also executes the UserSetup algorithm to generate the UID for the adversary. Now the simulator sends the PK and UID to the adversary.

Query phase 1: Adversary requests the secret key query to simulator for the set of attributes $\Gamma_j = \{a_j | a_j \in U^*\}$ provided $a_j \notin \tau^*$. The challenger picks the random number on every request $\gamma' \in Z_p$ and $\text{S\_Key1}^* = g^{\zeta+\gamma' b} = g^{\alpha-ab+\gamma' b} = g^{\alpha+(\gamma' b-ab)}$, thus $\gamma = \gamma' b - ab$. For each $a_j \in \Gamma_j$, the simulator generates secret key component as per the original algorithm $SK_j^* = g^{\frac{\gamma}{\beta t_j}}$. For each $a_j \notin \tau^*$, the simulator generates $SK_j^* = g^{\frac{(\gamma' b-ab)k_j}{\beta b}} = g^{\frac{\gamma' k_j - ak_j}{\beta}} = A^{\frac{k_j}{\beta}}.g^{\frac{-\gamma' k_j}{\beta}}$. After generating the secret key, the simulator sends the secret key to the adversary.

Further, the adversary may request update key query when the attribute x need to be updated to a new attribute y. For each update request, the simulator generates the update key and updates the secret key using the AUpdateKeyGen and SKUpdate algorithms as per the following steps. The update key will generate only if the adversary UID is in the UL. That means the adversary is not a revoked user. If the adversary UID is not in UL, then the update key is not generated and the secret key is also not updated.

Update key generation for secret key: $\forall\ x \in$ old_atts, j = att(x) and $\forall\ y \in$ new_atts, k = att(y), fetch the attribute random number (t) generated in setup phase corresponding to j and k and compute $uk\_sk\_au_j^* = g^{t_j / t_k}$.

Secret key update: $SK_j' = \left(g^{\frac{\gamma}{\beta.t_j}}\right) * uk\_sk\_au_j^*$. After the secret key update, the updated secret key send to the adversary.

Challenge: adversary selects and gives two identical size messages M_0 and M_1 to the χ. The χ tosses a binary coin (σ), and based on the value σ the message is encrypted. After encrypting the message, χ sends the ciphertext $M\sigma$ to $\hat{A}$. The ciphertext is computed as below.

$$CT_1^* = g^s = g^c$$
$$CT_2^* = M.\ell(g,g)^{\alpha s} = M_\sigma.\ell(g,g)^{\alpha c} = M_\sigma.\ell(g,g)^{(ab+\zeta)c}$$
$$= M_\sigma.\ell(g,g)^{abc}.\ell(g,g)^{\zeta c} = M_\sigma.\wp.\ell(g,g)^{\zeta c}$$

Perform the following steps to compute the value of CT_j^*: set the secret value c to the root node in the access tree. First the polynomial q_x with the degree $k_x - 1$ is chosen randomly for every child node, then the share value for each leaf and non-leaf node is found out to assign the share value to c_j. For all leaf nodes in the access tree, $CT_j^* = g^{c_j \beta k_j}$. Send the challenge ciphertext $C^* = (\tau^*, CT_1^*, CT_2^*, \forall a_j \in \tau^* : CT_j^*)$ to adversary. If $\sigma = 0$, $\wp = \ell(g,g)^{abc}$, which denotes that ciphertext C^* is valid for the message M_σ. If $\sigma = 1$, $\wp = \Re$ which is mapped to some random value in the target group, then decryption fails.

Query phase 2: Adversary may repeatedly ask secret key as same as in query phase 1 with same condition. The simulator also does the same process of query phase 1 whenever there is a request from $\hat{A}$.

Guess: Adversary submits the guess $\sigma' \in \{0, 1\}$ to the simulator for σ. If $\sigma' = \sigma$ then simulator guesses that $\sigma = 0$, so $\wp = \ell(g,g)^{abc}$ which is a well-founded cipher-text, it means that simulator provides ideal simulation. Hence, the advantage of adversary is $Pr[\sigma' = \sigma \mid \wp = \ell(g,g)^{abc}] = \frac{1}{2} + \epsilon$. Otherwise the simulator guesses that $\sigma = 1$, so $\wp = \Re$, which is mapped to random. So the disadvantage of the adversary which is not to get information is $Pr[\sigma' \neq \sigma \mid \wp = \Re] = \frac{1}{2}$. Thus on the whole advantage of the simulator to solve the DBDH assumption is:

$$\frac{1}{2}\left(Pr[\sigma' = \sigma \mid \sigma = 0] + \frac{1}{2} Pr[\sigma' \neq \sigma \mid \sigma = 1] - \frac{1}{2}\right) = \frac{\epsilon}{2}.$$

Theorem 2: Our R-CP-ABE scheme is secure against user collusion attack.

Proof: User secret key is generated for all authorised users of the system according to their attributes using KeyGeneration algorithm. During the key generation, there is a separate unique random component γ which is added to the secret key $g^{\alpha+\gamma}$ and $g^{\frac{\gamma}{\beta t_j}}$ for every user. If the users are trying to combine the secret key component to decrypt, they will not able to decrypt the message exactly because the random component γ value is different for all users. Thus, our scheme is secure against user collusion attack.

Theorem 3: Our R-CP-ABE scheme promises the forward and backward secrecy against the newly joined user and revoked user respectively.

Proof: When the attribute revocation occurs, the random value corresponding to the revoked attributes are re-generated and newly generated random value is updated in the ciphertext component $CT_j = (g^{(\beta.t_j.q_x(0))})^{uk\_ct}$. Whenever the new ciphertext C' is published, then automatically the revoked user's secret key becomes invalid to decrypt the newly published ciphertext because the secret key of revoked user is not updated and also consists of old attribute value for the revoked attribute. So the attribute revoked user can not decrypt the new ciphertext that is required by the revoked attributes, thus the backward secrecy is achieved. The same process is also used for the user revocation, but instead of updating the attribute value,

it updates the β value. Moreover, the revoked user will not get the updated ciphertext from the cloud because the cloud only provides the ciphertext to the user whose UID is in the UL. During the revocation handling the revoked user UID is removed from UL by TA. So the revoked user will not be able to get the ciphertext and obtain the plain text. Therefore backward secrecy is achieved in the user revocation.

During revocation the ciphertext gets updated while correspondingly the PK and MSK are also updated. If the new user joins the system their secret key is generated only from updated MSK, so the newly joined user can access the previously published data if their attributes satisfies the access policy. Thus the forward secrecy is achieved.

7 Performance analysis

In this part, we analyse the performance of our R-CP-ABE scheme in terms of storage, communication and computation overhead. The different notations used in the performance analysis are listed in Table 1.

7.1 Storage overhead

Here, we compute the storage requirement of each entity in our R-CP-ABE scheme and compared it with other existing systems. In our R-CP-ABE scheme, the storage overhead of cloud server depends on ciphertext length, DU is based on part of secret key, PS is used to store another part of user secret key, DO is based on size of PK and TA is based on number of attributes in universal attributes, and size of MSK.

Table 1 Notations used in performance analysis

Notation	*meaning*
η_{UA}	Total number of universal attributes
η_u	Total number of users
η_r	Total number of revoked users
η_{ra}	Total number of revoked attributes
η_Γ	Total number of user attributes
η_t	Total number attributes in the ciphertext
η_{nru}	Total number of non-revoked user having the revoked attributes
η_{nln}	Total number of non-leaf nodes in the access tree
L_S	The length of group G_S
L_T	The length of group G_T
L_z	The length of group Z_p
t_e	Time required for one exponentiation operation
t_p	Time required for one pairing operation

Table 2 shows the comparisons of storage cost of our R-CP-ABE scheme with other existing CP-ABE schemes. It is clearly observed from Table 2 that our scheme requires minimum storage overhead in DO, DU, TA, and CS than existing schemes due to the size of PK, secret key, MSK and length of the ciphertext respectively. Especially our R-CP-ABE scheme achieved very less storage overhead for DU because the part of secret key is stored in the PS without compromising the security.

7.2 Communication overhead

Here, we evaluate the communication overhead of our scheme and compared it with the existing schemes. The ciphertext size determines the major communication overhead in CP-ABE schemes because ciphertext needs to be stored and retrieved to/from the cloud. From Table 2 in the storage overhead, it is clearly noted that the ciphertext size (cloud column) in our scheme is less than other schemes.

Moreover, during the revocation the communication overhead mainly depends on sending the ciphertext update key and secret key update key to the entity that performs the update task. The communication overhead of user revocation and attribute revocation is almost same in our R-CP-ABE scheme. However, the attribute revocation communication overhead grows based on number of revoked attributes. In our R-CP-ABE scheme the communication overhead of revocation depends on sending both update keys from authority to PS, because PS performs both the updates, but in other existing schemes there is a communication between authority and CS as well as between authority and DU, because the CS performs the ciphertext update and data user performs the secret key update. Moreover, our R-CP-ABE scheme has the communication overhead between cloud server and proxy server during revocation which is not available with other existing schemes.

The communication overhead comparison of our R-CP-ABE scheme with other existing schemes during attribute revocation is shown in Table 3. From Table 3, it is clearly observed that the communication overhead during secret key update is constant in our R-CP-ABE scheme (authority and proxy), but it is linear with respect to number of users in other schemes (authority and user). Thus, our scheme removes the communication overhead from data user during revocation. But, our R-CP-ABE scheme requires additional communication overhead between cloud server and proxy server to eliminate the data user communication overhead. Since the proxy server is a part of cloud, the additional communication does not affect much in our scheme.

7.3 Computation overhead

Here, we compare the theoretical computational efficiency of our R-CP-ABE scheme with the existing schemes and it is tabulated in Table 4. From Table 4, it is clearly observed that our R-CP-ABE requires less computation overhead during secret key generation, encryption and decryption because of short secret key, short ciphertext and less number of pairing operations is required respectively.

Table 2 Comparison of storage overhead

Scheme	*Owner*	*User*	*Authority*	*Cloud*	*Proxy*
Yang and Jia (2014)	$(2\eta_{UA} + 4)L_S$	$(\eta_\Gamma + 2)L_S$	$(2\eta_u + 3)L_z$	$(4\eta_t + 2)L_S + L_T$	N/A
Zhang et al. (2016)	$(\eta_{UA} + 2)L_S + L_T$	$(2\eta_\Gamma + 2)L_S$	$(\eta_{UA} + 1)L_z + L_S$	$(3\eta_t + 2\eta_r + 1)L_S + L_T$	N/A
Zhou et al. (2017)	$(2\eta_{UA})L_S$	$(2\eta_\Gamma)L_S$	$(2\eta_{UA})L_z$	$(3\eta_t)L_S + L_T$	N/A
Our scheme	$(\eta_{UA} + 1)L_S + L_T$	L_S	$(\eta_{UA} + 2)L_z$	$(\eta_t + 1)L_S + L_T$	$\eta_\Gamma L_S$

Table 3 Comparison of communication overhead during attribute revocation

Scheme	*Authority and user*	*Authority and proxy*	*Authority and cloud*	*Cloud and proxy*
Yang and Jia (2014)	$\eta_u(\eta_{ra}L_S)$	–	$2\eta_{ra}L_z$	
Zhang et al. (2016)	$\eta_u(\eta_{ra}L_S)$	–	$\eta_{ra}L_z$	–
Zhou et al. (2017)	$\eta_u(\eta_{ra}L_S)$	–	$\eta_{ra}L_z$	–
Our scheme	–	$2\eta_{ra}L_s + \eta_{ra}L_z$	–	$(\eta_t + 1)L_S + L_T$

Table 4 Comparison of computation overhead

Scheme	*Secret key generation time*	*Encryption time*	*Decryption time*
Yang and Jia (2014)	$(2\eta_\Gamma + 4)t_e$	$(4\eta_t + 2)t_e + t_p$	$(2\eta_\Gamma + 4)t_p$
Zhang et al. (2016)	$(3\eta_\Gamma + 2)t_e$	$(3\eta_t + 2\eta_r + 1)t_e + t_p$	$(3\eta_\Gamma + 2\eta_r + 1)t_p + (\eta_{nln} + \eta_r)t_e$
Zhou et al. (2017)	$(2\eta_\Gamma)t_e$	$(3\eta_t)t_e + t_p$	$(4\eta_\Gamma + 2)t_p + 2\eta_{nln}t_e$
Our scheme	$(\eta_\Gamma + 1)t_e$	$(\eta_t + 1)t_e + t_p$	$(\eta_\Gamma + 1)t_p + \eta_{nln}t_e$

Table 5 Comparison of revocation overhead

Scheme	*Update key generation time*	*Ciphertext updater*	*Ciphertext update time*	*Secret key updater*	*Secret key update time*	*Forward secrecy*	*Backward secrecy*
Yang and Jia (2014)	$\eta_u(\eta_{ra}t_e)$	Cloud	$(2\eta_{ra})t_e$	User	$\eta_u(\eta_{ra})$	×	×
Zhang et al. (2016)	$2\eta_{ra}t_e$	Cloud	$\eta_{ra}t_e$	User	$\eta_{nru}(\eta_{ra}) + \eta_r(\eta_{ra}t_e)$	×	×
Zhou et al. (2017)	$\eta_u(\eta_{ra}t_e)$	Cloud	$\eta_{ra}t_e$	User	$\eta_u(\eta_{ra})$	×	×
Our scheme	$2\eta_{ra}t_e$	Proxy	$\eta_{ra}t_e$	Proxy	$\eta_u(\eta_{ra})$	✓	✓

Figure 2 Encryption computation time comparison

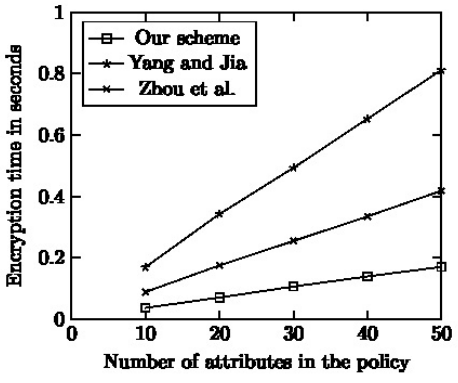


Table 5 shows the comparison of revocation overhead between our R-CP-ABE scheme and various existing revocation schemes. As shown in Table 5, Zhang et al. (2016) scheme achieves minimum computation overhead during update key generation and ciphertext update time, but this scheme suffered higher computation overhead during the secret key update. At the same time, Zhou et al. (2017) achieves minimum computation overhead during the ciphertext update time and secret key update time, but this scheme also suffered higher computation overhead during update key generation. Only our R-CP-ABE scheme achieves minimum computation overhead during update key generation, ciphertext update time, and user's secret key update time than other existing schemes.

Moreover, all the existing schemes (Yang and Jia, 2014; Zhang et al., 2016; Zhou et al., 2017) failed to achieve the forward secrecy and backward secrecy because the ciphertext update is performed by untrusted cloud. In addition, the user secret key update is carried out by the user itself, which increases the computation overhead for the data user. But, in our R-CP-ABE scheme, the proxy server performs both ciphertext update and secret key update which reduces the computation overhead for data owner, data user, and it also achieves the forward secrecy and backward secrecy.

7.4 Implementation results

We implemented our scheme, Yang and Jia (2014) scheme and Zhou et al. (2017) scheme in python 3.2 with pairing-based charm-crypto-0.42 cryptography library on Windows 8 operating system with the hardware configuration of Intel core i7 processor @ 2.50 GHz, 16 GB RAM and compared the implementation results. The experimental data is obtained as an average of ten runs and the experimental results are plotted as a graph. Figure 2 describes the relationship between computation time of encryption process and number of attributes involved in the access policy. From Figure 2, it is observed that our R-CP-ABE scheme achieved less encryption time because of short ciphertext when compared to other existing schemes. Figure 3 shows the relationship between decryption computation time and number of user attributes. It is clearly noted that, our R-CP-ABE scheme requires a less decryption time because of minimum pairing operations involved in the decryption. Figure 4 gives the relationship between times required to update the ciphertext and number of revoked attributes. Our R-CP-ABE scheme and Zhou et al. (2017) scheme achieved the less ciphertext update time. However, Zhou et al. (2017) required higher encryption time, decryption time, and update key generation time than our R-CP-ABE scheme. The simulated result matches the theoretical performance analysis. Thus our R-CP-ABE scheme is efficient.

Figure 3 Decryption computation time comparison

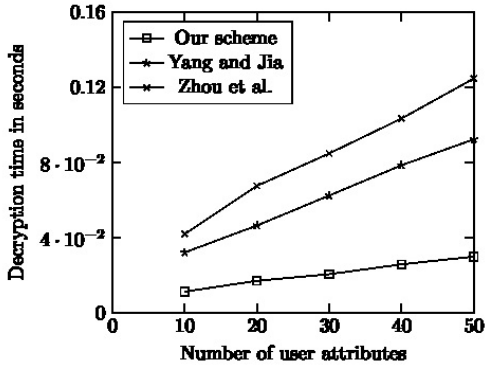
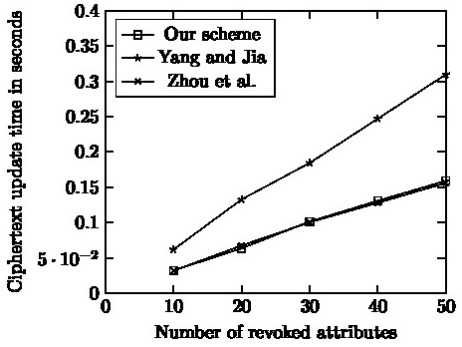


Figure 4 Comparison of ciphertext update time



8 Conclusions

In this paper, we proposed the efficient R-CP-ABE scheme for big data access control in cloud computing using proxy-based updates. Our R-CP-ABE scheme achieves privacy and access control along with user revocation and attribute revocation for big data in cloud. Our R-CP-ABE scheme reduced the complexity of data owner by delegating the ciphertext update to proxy server and also reduced the complexity of data user by delegating the secret key update to proxy server during revocation. Our scheme achieved the less storage overhead for data user by storing the part of secret key in the proxy server. The security of our R-CP-ABE scheme is proved against chosen plain-text and user collusion attacks along with forward and backward secrecy. Performance analysis shows that our scheme incurs less overhead in all aspects of the system. Thus R-CP-ABE scheme is more suitable for big data privacy and access control in cloud computing.

References

Balani, N. and Ruj, S. (2014) 'Temporal access control with user revocation for cloud data', *IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications* (*TrustCom*), IEEE, pp.336–343 [online] https://doi.org/10.1109/TrustCom.2014.45.

Bethencourt, J., Sahai, A. and Waters, B. (2007) 'Ciphertext-policy attribute-based encryption', *IEEE Symposium on Security and Privacy*, *SP'07*, IEEE, pp.321–334 [online] https://doi.org/10.1109/SP.2007.11.

Goyal, V., Pandey, O., Sahai, A. and Waters, B. (2006) 'Attribute-based encryption for fine-grained access control of encrypted data', *Proceedings of the 13th ACM Conference on Computer and Communications Security*, ACM, pp.89–98 [online] https://doi.org/10.1145/1180405.1180418.

Gupta, B., Agrawal, D.P. and Yamaguchi, S. (2016) *Handbook of Research on Modern Cryptographic Solutions for Computer and Cyber Security*, pp.1–589, IGI Global, Hershey, PA [online] https ://doi.org/10.4018/978-1-5225-0105-3.

Hong, J., Xue, K. and Li, W. (2015) 'Comments on dac-macs: effective data access control for multiauthority cloud storage systems/security analysis of attribute revocation in multiauthority data access control for cloud storage systems', *IEEE Transactions on Information Forensics and Security*, Vol. 10, No. 6, pp.1315–1317 [online] https://doi.org/10.1109/TIFS.2015.2407327.

Hur, J. and Noh, D.K. (2011) 'Attribute-based access control with efficient revocation in data outsourcing systems', *IEEE Transactions on Parallel & Distributed Systems*, Vol. 22, No. 7, pp.1214–1221 [online] https://doi.org/10.1109/TPDS.2010.203.

Khan, N., Yaqoob, I., Hashem, I.A.T., Inayat, Z., Ali, M., Kamaleldin, W., Alam, M., Shiraz, M. and Gani, A. (2014) 'Big data: survey, technologies, opportunities, and challenges', *The Scientific World Journal* [online] https://doi.org/10.1155/2014/712826.

Kumar, P.P., Kumar, P.S. and Alphonse, P. (2018) 'Attribute based encryption in cloud computing: a survey, gap analysis, and future directions', *Journal of Network and Computer Applications*, Vol. 108, pp.37–52 [online] https://doi.org/10.1016/j.jnca.2018.02.009.

Li, J., Yao, W., Han, J., Zhang, Y. and Shen, J. (2018) 'User collusion avoidance cp-abe with efficient attribute revocation for cloud storage', *IEEE Systems Journal*, Vol. 12, No. 2, pp.1767–1777 [online] https://doi.org/10.1109/JSYST.2017.2667679/.

Li, L., Gu, T., Chang, L., Xu, Z., Liu, Y. and Qian, J. (2017) 'A ciphertext-policy attribute-based encryption based on an ordered binary decision diagram', *IEEE Access*, Vol. 5, pp.1137–1145 [online] https://doi.org/10.1109/ACCESS.2017.2651904.

Liu, Q., Wang, G. and Wu, J. (2014) 'Time-based proxy re-encryption scheme for secure data sharing in a cloud environment', *Information Sciences*, Vol. 258, pp.355–370 [online] https://doi.org/10.1016/j.ins.2012.09.034.

Liu, Z., Jiang, Z.L., Wang, X. and Yiu, S. (2018) 'Practical attribute-based encryption', *Journal of Network and Computer Applications*, Vol. 108, No. C, pp.112–123 [online] https://doi.org/10.1016/j.jnca.2018.01.016.

Sahai, A. and Waters, B. (2005) 'Fuzzy identity-based encryption', *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Springer, pp.457–473 [online] https://doi.org/10.1007/1142663927.

Sultan, N.H. and Barbhuiya, F.A. (2016) 'A secure re-encryption scheme for data sharing in unreliable cloud environment', *IEEE World Congress on Services* (*SERVICES*), IEEE, pp.75–80 [online] https://doi.org/10.1109/SERVICES.2016.16.

Takabi, H., Joshi, J.B. and Ahn, G-J. (2010) 'Security and privacy challenges in cloud computing environments', *IEEE Security & Privacy*, Vol. 8, No. 6, pp.24–31 [online] https://doi.org/10.1109/MSP.2010.186.

Voorsluys, W., Broberg, J. and Buyya, R. (2011) 'Introduction to cloud computing', *Cloud Computing: Principles and Paradigms*, pp.1–41 [online] https://doi.org/10.1002/9780470940105.ch1.

Wang, G., Liu, Q., Wu, J. and Guo, M. (2011) 'Hierarchical attribute-based encryption and scalable user revocation for sharing data in cloud servers', *Computers & Security*, Vol. 30, No. 5, pp.320–331 [online] https://doi.org/10.1016/j.cose.2011.05.006.

Wang, H., Zheng, Z., Wu, L. and Li, P. (2017) 'New directly revocable attribute-based encryption scheme and its application in cloud storage environment', *Cluster Computing*, Vol. 20, No. 3, pp.2385–2392 [online] https://doi.org/10.1007/s10586-016-0701-7.

Waters, B. (2011) 'Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization', *International Workshop on Public Key Cryptography*, Springer, pp.53–70 [online] https://doi.org/10.1007/978-3-642-19379-84.

Xiao, M., Wang, M., Liu, X. and Sun, J. (2015) 'Efficient distributed access control for big data in clouds', *IEEE Conference on Computer Communications Workshops* (*INFOCOM WKSHPS*), IEEE, pp.202–207 [online] https://doi.org/10.1109/INFCOMW.2015.7179385.

Xie, X., Ma, H., Li, J. and Chen, X. (2013) 'New ciphertext-policy attribute-based access control with efficient revocation', *Information and Communication Technology-EurAsia Conference*, Springer, pp.373–382 [online] https://doi.org/10.1007/978-3-642-36818-941.

Xu, X., Zhou, J., Wang, X. and Zhang, Y. (2016) 'Multi-authority proxy re-encryption based on cpabe for cloud storage systems', *Journal of Systems Engineering and Electronics*, Vol. 27, No. 1, pp.211–223.

Xu, Z. and Martin, K.M. (2012) 'Dynamic user revocation and key refreshing for attribute-based encryption in cloud storage', *IEEE 11th International Conference on Trust*, *Security and Privacy in Computing and Communications* (*TrustCom*), IEEE, pp.844–849.

Yang, K. and Jia, X. (2014) 'Expressive, efficient, and revocable data access control for multi-authority cloud storage', *IEEE Transactions on Parallel and Distributed Systems*, Vol. 25, No. 7, pp.1735–1744 [online] https://doi.org/10.1109/TPDS.2013.253.

Yang, K., Jia, X., Ren, K., Zhang, B. and Xie, R. (2013) 'Dac-macs: effective data access control for multiauthority cloud storage systems', *IEEE Transactions on Information Forensics and Security*, Vol. 8, No. 11, pp.1790–1801 [online] https://doi.org/10.1109/TIFS.2013.2279531.

Zhang, P., Chen, Z., Liang, K., Wang, S. and Wang, T. (2016) 'A cloud-based access control scheme with user revocation and attribute update', *Australasian Conference on Information Security and Privacy*, Springer, pp.525–540 [online] https://doi.org/10.1007/978-3-319-40253-632.

Zhou, J., Duan, H., Liang, K., Yan, Q., Chen, F., Yu, F.R., Wu, J. and Chen, J. (2017) 'Securing outsourced data in the multi-authority cloud with fine-grained access control and efficient attribute revocation', *The Computer Journal*, Vol. 60, No. 8, pp.1210–1222 [online] https://doi.org/10.1093/comjnl/bxx017.