Journal of Ambient Intelligence and Humanized Computing (2019) 10:2693–2707
https://doi.org/10.1007/s12652-018-0967-0

ORIGINAL RESEARCH

A new verifiable outsourced ciphertext-policy attribute based encryption for big data privacy and access control in cloud

Praveen Kumar Premkamal[1] · **Syam Kumar Pasupuleti**[2] · **P. J. A. Alphonse**[1]

Received: 6 April 2018 / Accepted: 6 August 2018 / Published online: 14 August 2018

Abstract
The foremost security concerns for big data in the cloud are privacy and access control. Ciphertext-policy attribute based encryption (CP-ABE) is an effective cryptographic solution for above concerns, but the existing CP-ABE schemes are not suitable for big data in the cloud as they require huge computation time for encryption and decryption process. In this paper, we propose a new verifiable outsourced CP-ABE for big data privacy and access control in the cloud. Our scheme reduces the computational overhead of encryption and decryption by outsourcing the heavy computations to the proxy server. Our scheme also verifies the correctness of the data along with the outsourcing computations. Further, our scheme limits the data access for a set of users instead of providing an infinite number of times data access, which is essentially required for commercial applications. In security analysis, we prove that our scheme is secure against chosen plain-text attack, collusion and proxy attacks. Performance analysis proves that our scheme is efficient.

Keywords Cloud computing · Big data · Privacy · Access control · Outsourced CP-ABE · Correctness

1 Introduction

Due to the increased use of internet technology and digitization, the data has become a significant factor for the organizational growth. This lead to the advent of a new paradigm called the big data. The main characteristics of big data are volume, which refers to the huge size of data, the variety which refers to the data that may be structured, unstructured, or semi-structured and veracity which refers to the data that is generated in a rapid manner, so data could be collected and processed rapidly (Khan et al. 2014). The traditional database system like RDBMS do not support to storage and processing of big data due to huge volume and variety. So, the organizations require more investments in buying huge computing infrastructures, scalable storages, networking, distributed databases, platforms, frameworks and softwares, etc., but the cloud computing provides all the required facilities as pay-per-use model. It also reduces the infrastructure maintenance, software update, and management costs at local premises (Voorsluys et al. 2011). Hence, the cloud is an effective platform to store and process the big data.

Although cloud provides lots of benefits, the organizations are not moving big data to the cloud because of the security and privacy challenges (Takabi et al. 2010; Gupta et al. 2016) in the cloud. The foremost important security issues are disclosing of sensitive data to unauthorized users by the cloud and unauthorized users data access. Hence, an encryption is required to protect the sensitive information and also giving access rights only to authorized users for accessing the data.

CP-ABE scheme is a promising solution for data privacy along with fine-grained access control in cloud environment. In CP-ABE, the data owner encrypts the data with defined access structure or policy and stores it into the cloud. The data users can access the data only if their attributes satisfy the access structure in the cipher-text. Many CP-ABE schemes are in the literature (Bethencourt et al. 2007; Cheung and Newport 2007; Waters 2011; Deng et al. 2014; Zhang et al. 2016, 2017b; Jiang et al.

✉ Praveen Kumar Premkamal
tvp.praveen@gmail.com

Syam Kumar Pasupuleti
psyamkumar@idrbt.ac.in

P. J. A. Alphonse
alphonse@nitt.edu

[1] National Institute of Technology, Tiruchirappalli, India

[2] Institute for Development and Research in Banking Technology, Hyderabad, India

2017; Li et al. 2017b, 2018a, b), but directly applying these schemes for big data privacy and access control in cloud is a computationally intensive task because they suffered with huge encryption and decryption computation costs due to cipher-text size and requirement of exorbitant pairing operations respectively.

In order to reduce the computational overhead, the heavy encryption and decryption computations are outsourced to the cloud server (Ma et al. 2013; Xiang and Tang 2015; Ye et al. 2018) or to the proxy server (Kumar et al. 2018). Many outsourced CP-ABE schemes were proposed. Green et al. (2011) proposed CP-ABE scheme with outsourced decryption and Zhang et al. (2017a) proposed CP-ABE scheme with outsourced encryption and decryption computations, but these scheme have not checked the correctness of outsourced computations. Verifiable outsourced decryption in CP-ABE schemes were proposed (Qin et al. 2015; Lin et al. 2015; Mao et al. 2016; Li et al. 2017a; Wang et al. 2017b), but these schemes suffered with encryption computational overhead. Later, CP-ABE with verifiable outsourced encryption and decryption schemes were proposed (Ma et al. 2015; Wang et al. 2017a; Li et al. 2018c), but again these schemes are also not suitable for big data due to huge communication and computation overhead.

Moreover, all of the above schemes provide access control in an all or none fashion, that is the authorized users can access the data at any number of times, or no access is given to unauthorized users. Any commercial application such as accessing course tutorial samples, software trial download, application user interface experience, audio and video access, etc. may require restricted data access permission, such as two times access or three times access, etc. Hence, it is required to design a verifiable outsourced CP-ABE scheme that requires less communication and computation overhead along with a flexible access control (limited data access for the set of users) for big data.

In this paper, we propose a new verifiable outsourced CP-ABE (NVO-CP-ABE) for big data privacy and access control in the cloud with the following salient features.

1. In our scheme, all the inflated computations of encryption and decryption are outsourced to a proxy server to reduce the computation overhead.
2. Our scheme also reduces the data user's storage overhead by storing user's secret key in the proxy server without compromising the security.
3. Our scheme verifies the correctness of the encrypted message and also verifies the correctness of outsourced encryption and decryption computations.
4. Our scheme provides an option to restrict the set of authorized user's data access with fixed number of times instead of unlimited access which makes it convenient to use this scheme in the commercial applications.
5. Our scheme is secure against chosen plain-text attack (CPA), user collusion attack, proxy malfunction attack, and proxy-user collusion attack.
6. Our scheme achieves better efficiency when compared to other existing outsourced CP-ABE schemes in the literature.

The rest of the paper is structured as follows. Section 2 gives the related work in the literature. In Sect. 3, we introduce the mathematical preliminaries, definitions and security model associated with NVO-CP-ABE scheme. The NVO-CP-ABE model is discussed in Sect. 4. Section 5 describes the NVO-CP-ABE construction in detail. Sections 6 and 7 explains the security and performance analysis respectively, and the work ends with a conclusion in Sect. 8.

2 Related works

Attribute based encryption was introduced by Sahai and Waters (2005). There are two variations in attribute based encryption such as key-policy attribute based encryption (KP-ABE) and CP-ABE. Goyal et al. (2006) first introduced KP-ABE with an access tree as access structure. In KP-ABE, the data is encrypted with set of descriptive attributes and the user can decrypt the data only if the access structure of the user secret key satisfies the set of attributes in the cipher-text. Bethencourt et al. (2007) first introduced CP-ABE scheme with tree access structure. When compared to KP-ABE, the CP-ABE is more appropriate to provide access control because the data owner has the control over the data. CP-ABE schemes with different access structures were proposed. CP-ABE scheme with AND access structure was proposed by Cheung and Newport (2007), Linear Secret Sharing Scheme (LSSS) matrix access structure was proposed by Waters (2011), Ordered Binary Decision Diagram (OBDD) access structure was proposed by Li et al. (2017b). Generally, CP-ABE schemes suffered with huge encryption and decryption computational overhead.

Outsourcing heavy encryption and decryption computation tasks to the cloud server or to the proxy server is one of the efficient solution to reduce the computational overhead. De and Ruj (2015) proposed the decentralized access control in the cloud with fast encryption and outsourced decryption, but there was no mechanism to verify the correctness of the outsourced decryption computations in this scheme. Zhang et al. (2017a) proposed fully outsourced attribute based encryption scheme in which major computation task such as key generation, encryption and decryption are outsourced to the cloud server. Though the partial computations of different algorithms are based on untrusted cloud sever, there is no mechanism to verify whether the cloud server computed correct results or not. Moreover, they generated the cipher-text

from two outsourcing intermediate cipher-text, even though this improves the hardness of security, but it lacks its efficiency in computation and communication.

Qin et al. (2015) proposed symmetric key encryption based verifiable outsourced decryption CP-ABE scheme in which the hash function is used to verify the message correctness and decryption computation correctness. Lin et al. (2015) proposed the verifiable outsourced decryption for both key-policy and cipher-text policy context. Mao et al. (2016) proposed the generic verifiable outsourced decryption CP-ABE construction in which they proved the security in both chosen plain-text attack and chosen cipher-text attack. Li et al. (2017a) proposed a fully verifiable outsourced decryption for CP-ABE scheme, but this scheme suffered with heavy computation overhead because of some extra random message added to generate cipher-text to map with the unauthorized users. Wang et al. (2017b) proposed multi-authority based verifiable outsourced decryption for cloud access control. In all of the above schemes, the authorized data user can access the data any number of times, but some commercial applications require limited data access. Ning et al. (2018) proposed auditable outsourced attribute based encryption with sigma-time access for authorized users in cloud computing. This scheme only verifies the outsourced computation, but not the correctness of message.

All the above outsourced decryption schemes focus only on reducing data users higher decryption overhead with verification, but does not focus on data owners higher encryption computational overhead. Ma et al. (2015) proposed CP-ABE for access control in the cloud which supports verifiable and exculpable outsourcing encryption and decryption computation. Xiong and Sun (2017) proved Ma et al. (2015) scheme failing to provide the verifiable property for outsourced encryption, and concluded that outsourcing encryption with verification is a challenging open issue. Wang et al. (2017a) proposed the CP-ABE with outsourced and verifiable computation for key generation, encryption and decryption. However, this scheme suffered with large cipher-text size and could not verify the correctness of outsourced decryption. Li et al. (2018c) proposed the fuzzy encryption with efficient verifiable outsourced attribute based encryption in which the heavy computation task of encryption and decryption is outsourced to cloud. In this scheme, the authors verified the correctness of the message and outsourced computation using a hash function. All the above outsourced CP-ABE schemes are not specially designed for big data in the cloud.

3 Preliminaries

Here, we give all the mathematical preliminaries, definitions and security model associated with NVO-CP-ABE scheme. More precisely, the bilinear pairing, Decisional Bilinear Diffie–Hellman assumption, Shamir secret sharing, access structure, CP-ABE framework, and security model are presented.

3.1 Bilinear pairing

Definition 1 A bilinear mapping function over additive cyclic group G_s and multiplicative cyclic group G_t of prime order p is defined as $G_s \times G_s \rightarrow G_t$, which satisfies the below properties.

1. *Bilinearity* For $\mathfrak{g} \in G_s, a, b \in Z_p$, $\varrho(\mathfrak{g}^a, \mathfrak{g}^b) = \varrho(\mathfrak{g}, \mathfrak{g})^{ab}$ where $\mathfrak{g}$ be the generator of group G_s.
2. *Non-degeneracy* $\varrho(\mathfrak{g}, \mathfrak{g}) \neq 1$
3. *Computability* There exists an efficient algorithm to compute $\varrho(\mathfrak{g}, \mathfrak{g})$.

3.2 Decisional bilinear Diffie–Hellman (DBDH) assumption

Definition 2 Let G_s and G_t are group of prime order p. Let $\mathfrak{g}$ be the generator of group G_s and a, b, c $\in Z_p$. We define the DBDH problem as follows: For the given input $\mathfrak{g}$, $\mathfrak{g}^a$, $\mathfrak{g}^b$, $\mathfrak{g}^c \in G_s$, the adversary must distinguish a valid tuple $\varrho(\mathfrak{g}, \mathfrak{g})^{abc} \in G_t$ 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 if:

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

Definition 3 The DBDH assumption holds if no probabilistic polynomial time algorithm has a non-negligible advantage in solving DBDH problem.

3.3 Shamir secret sharing

Shamir secret sharing (Shamir 1979) is a cryptographic technique designed by Adi Shamir, which helps us to protect the secret value. The basic idea of this technique is to divide the secret into different pieces and give it to N-number of participants. Out of N-number of participants, K-number of participants combine together to reconstruct the secret. The secret share generation steps are as follows.

1. Choose the secret value S.
2. Choose K−1 random coefficients and generate the polynomial f of K − 1 degree. Let $f(x) = S + f_1 x^1 + \cdots + f_{K-1} x^{K-1}$, where f(0) = S.
3. Compute and give the shares f(x) to parties where x = 1 to N

The share reconstruction is possible only if K-participants combine their shares to obtain the secret using the Lagrange interpolation formula. The formula is as follows.

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

$$f_j(x) = y_j \times \prod_{i=1, i\neq j}^{k} \frac{x - x_i}{x_j - x_i}$$

In CP-ABE, the participants role is taken by the attributes. In our scheme, we use Shamir secret sharing technique for providing access control.

3.4 Access structure

Definition 4 Let $P_1, P_2, \ldots, P_n$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{P_1, P_2, \ldots, P_n}$ is monotone if $\forall B, C$: if $B \in \mathbb{A}$ and $B \subseteq C$, then $C \in \mathbb{A}$. An access structure (monotone access structure) is a collection (monotone collection) $\mathbb{A}$ of non-empty subsets of $P_1, P_2, \ldots, P_n$, that is $\mathbb{A} \subseteq 2^{P_1, P_2, \ldots, P_n} \setminus \{\phi\}$. The sets in $\mathbb{A}$ are called the authorized sets, and the sets that are not in $\mathbb{A}$ are called the unauthorized sets.

In CP-ABE context, the role of the parties is taken by the attributes. Thus, the access structure $\mathbb{A}$ will contain the authorized sets of attributes.

3.4.1 Access tree

We employ the access tree as an access structure in the proposed work which consists of a set of nodes in which all the attributes are stored in leaf-nodes and the threshold gates (AND, OR) are stored in the non-leaf nodes. All the child nodes are assigned an integer number serially starting from 1 to cn_x. Let cn_x be the number of children for the node x. Let t_x be the threshold value of the node x. The threshold value should be between 1 and cn_x. For OR gate, it is one and for AND gate it is cn_x. We define few functions which helps us to facilitate working with access tree. The parent(x) function gives the parent node of the node x. The index(x) function returns the integer number assigned to the node x and att(x) function gives us the attribute name of the node x.

Definition 5 *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 δ satisfies the access tree T_x, then it denotes $T_x(\delta) = 1$. We compute $T_x(\delta)$ recursively as follows. If x is a non-leaf node, evaluate $T_x^{'}(\delta)$ for all children $x^{'}$ of node x. $T_x(\delta)$ returns 1 if and only if at least t_x children return 1. If x is a leaf node, then $T_x(\delta)$ returns 1 if and only if att(x) $\in \delta$.

3.5 CP-ABE framework

The CP-ABE framework consists of four algorithms such as Setup, Encrypt, KeyGen and Decrypt.

- *Setup* This algorithm takes the security parameter d as input and outputs the public key PK and master secret key MSK.
- *Encrypt* This algorithm takes the public key PK, the message M and the access structure τ as inputs and outputs the ciphertext C.
- *KeyGen* The set of user attributes δ, the master secret key MSK are the inputs for this algorithm and it outputs the user secret key S.
- *Decrypt* This algorithm returns message M or $\perp$. It returns M if the user attributes satisfies the access structure otherwise it returns $\perp$. It takes the input of public key PK, ciphertext C and user secret key S.

3.6 Security model

In security model, we define the following security game between challenger and adversary to prove the security of our scheme against chosen plain-text attack. This game captures the indistinguishability of messages and the collusion-resistance of user secret keys. There are six steps involved in the game as listed below:

Initializing game The adversary sends the challenge access structure to the challenger and requests for public key.

Setup phase The challenger generates the public key, master secret key using SystemSetup algorithm and sends the public key to the adversary.

Query phase 1 The adversary sends the set of attributes to the challenger and requests for the secret key. The challenger generates the secret key and sends the same to the adversary.

Challenge The adversary sends two equal size messages M_0 and M_1 to the challenger and requests for the cipher-text. The challenger receives both the messages and encrypts one message based on the random bit value $\sigma \in \{0, 1\}$.The challenger then returns the cipher-text to adversary.

Query phase 2 The adversary may request the different queries similar to the query phase 1 and the challenger also responses back for the queries to adversary as same like in query phase 1.

Guess The adversary submits the guess σ' for σ whether it is 0 or 1. The adversary wins the game if the guess σ' is equal to σ. The probabilistic advantage of adversary winning the game is $\Pr[\sigma = \sigma'] - \frac{1}{2}$.

Definition 6 The proposed NVO-CP-ABE scheme is said to be secure against CPA if no probabilistic polynomial time adversaries have non-negligible advantage in the above game.

4 NVO-CP-ABE system model

In this section, we present the NVO-CP-ABE system model, framework, security threats, and objectives of the proposed work.

4.1 System model

The proposed NVO-CP-ABE system consists of six entities such as cloud server, proxy server, trusted authority, data owner, data users, and third party auditor as shown in Fig. 1.

Cloud server (CS) The cloud server has a large volume of storage capacity, and provides the storage as the service to the data owner to store the data and gives the data access to the proxy server. The CS is untrusted in our scheme.

Proxy server (PS) The semi-trusted proxy server performs the outsourced encryption and outsourced decryption computations for data owner and data users respectively. PS stores the user secret key and it also verifies the user data access limit before performing the outsourced decryption computations.

Trusted authority (TA) The trusted authority performs the following tasks (1) generates the public parameter such as public key and master secret key (2) registers the new user, that is generates user id and assigns the number of permissible data access (3) generates the user global secret key and user secret key.

Data owner (DO) The fully trusted data owner defines the access structure and generates the part of cipher-text. After generating the cipher-text, DO uploads the cipher-text into the CS.

Data users (DU) The data users can decrypt the data only if their attributes satisfies the access structure and data access limit is within the permissible access limit. The data users are untrusted in our scheme.

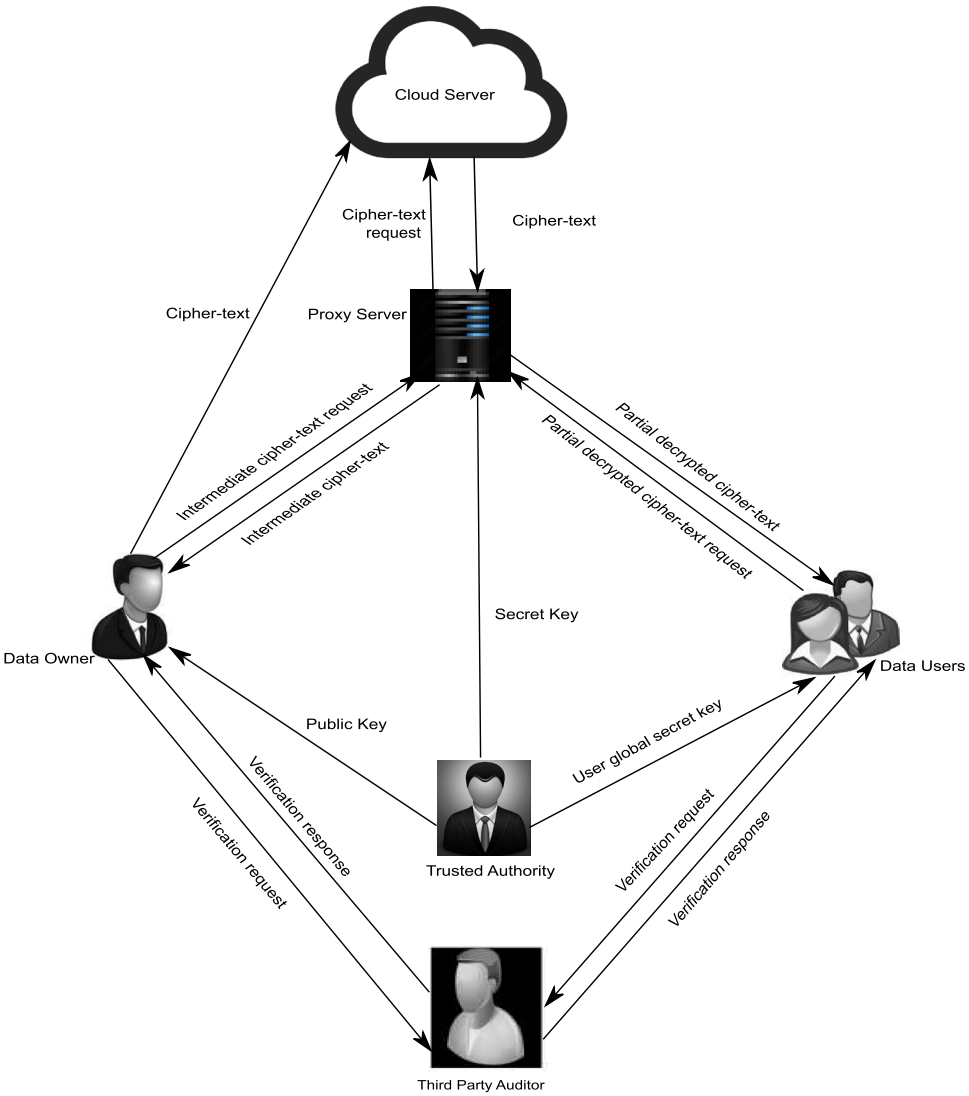

Fig. 1 Architecture of our NVO-CP-ABE scheme

Third party auditor (TPA) The fully trusted TPA verifies the correctness of the message, outsourced encryption and decryption computations.

In our system model, TA initially generates the public key, master secret key and sends the public key to data owner to encrypt the message. After getting the public key, DO defines the access structure, and requests the intermediate cipher-text (outsource the heavy encryption computations) to PS. Once PS has generated the intermediate cipher-text, it sends to DO. DO sends the intermediate cipher-text to TPA for outsourced computation verification, and simultaneously, DO computes the other part of the cipher-text. If TPA sends the success response, then DO assembles the cipher-text and uploads the data in to the cloud. In order to preserve the data privacy, DO encrypts the data before storing it into the cloud.

Whenever, the DU wants to access the data, DU requests partially decrypted cipher-text from PS. PS first checks the corresponding user data access limit, if it is within the limit then it computes partially decrypted cipher-text and sends to the DU. After receiving the partially decrypted cipher-text, DU verifies the correctness of outsourced decryption and encrypted data with the help of TPA. TPA returns the exact key to DU if the encrypted data is not modified, outsourced decryption computations are correct, and the user attributes satisfies the access structure in the cipher-text. Finally, DU decrypts the data successfully.

4.2 Framework

The different algorithms involved in the NVO-CP-ABE scheme are defined as follows:

SystemSetup(d) → *(PK, MSK):* TA executes this algorithm, and it takes the security parameter d as input and produces the public key (PK) and master secret key (MSK) as an output.

UserSetup(UL) → *(UID, UL):* TA executes this algorithm with the input of user list and it outputs the UID and UL.

Encrypt_Preproc(τ) → (s_v, sh_v_x) : DO executes this algorithm. It takes the access tree (τ) as input and it outputs the secret value (s_v) and share value (sh_v_x) of all the attributes in the access tree.

Encrypt_outsource (PK, s_v, sh_v_x, τ) → *ICT* : The proxy server executes this algorithm. It takes the input of public key (PK), secret value (s_v), share value of each attributes in the access tree (sh_v_x), access tree (τ) and it returns the intermediate cipher-text (ICT).

Encrypt_owner(τ, *PK, M, ICT*) → C: The DO performs this algorithm. It takes the input of an access tree (τ), public key (PK), message (M), an intermediate cipher-text (ICT) and it returns the cipher-text (C).

Verify_outsource_enc(sh_v_x, *PK, ICT*) → flag (1 or 0): The TPA verifies whether the intermediate cipher-text (ICT) is correct or not by using the share value of each attributes (sh_v_x), public key (PK) and intermediate cipher-text (ICT) as inputs. If the outsourced computation is correct, then it returns the flag = 1, otherwise flag = 0.

KeyGeneration(UID, MSK, UL, δ) → *(S, UGSK)* or ⊥: First, TA checks if the user is a registered user or not. If registered user, then TA generates user global secret key (UGSK) and the secret key (S) for the given input of user id (UID), master secret key (MSK), user list (UL), set of user's attributes (δ). It returns S and UGSK for registered users otherwise it returns ⊥.

Decrypt_outsource(UL, C, S) → *(PDCT,* $E_{key}(M)$, CT3, VK) or ⊥: The proxy server executes this algorithm using user list (UL), cipher-text (C), user secret key (S) and produces the output which consist of partially decrypted cipher-text (PDCT), encrypted data ($E_{Key}(M)$), cipher-text component (CT3), and verification key (VK) or it returns ⊥ when the user data access limit exceeds.

Verify_outsource_dec(PDCT, $E_{key}(M)$, *CT3, VK, UGSK)* → *Key* or 0: This algorithm is executed by the TPA with the input of partial decrypted cipher-text (PCT), cipher-text parameter (CT3), encrypted message ($E_{key}(M)$), verification key (VK), and user global secret key (UGSK). It returns decryption *Key* or 0.

Decryption_user(flag) → (M or ⊥): The data user executes this algorithm. It takes the input of outsourced decryption verification *flag* and outputs the message M or ⊥.

4.3 Security threats

Here, we define the threats that breaches the security of our NVO-CP-ABE scheme.

1. *Chosen plain-text attack* The adversary trying to reveal entire or part of the data by getting cipher-text for his/her arbitrary plain-text.
2. *User collusion attack* The two unauthorized data users combine their secret key component and try to access the data.
3. *Proxy server attack* The proxy server could skip or not perform computation properly during outsourced encryption and decryption that causes the entire scheme to fail, and also tries to access the plain text using the cipher-text and stored user secret key.
4. *Proxy server and unauthorized user collusion attack* The unauthorized users can combine with proxy server and try to access the plain-text.

4.4 Objectives

The main purpose of the proposed scheme is to develop a new verifiable outsourced CP-ABE scheme for big data access control in the cloud with the following objectives.

Privacy Protect the personal or sensitive data from unauthorized users in the cloud.

Access control Allow only authorized users to access the data.

Limited data access Our scheme is flexible to allow the authorized users to access the data at any number of times or restrict the data access with fixed number of times, which improves the usability of this scheme in commercial applications.

Security Provide the security against chosen plain-text attack, user collusion attack, proxy malfunction attack, and proxy-user collusion attack.

Efficiency Reduce the computation overhead of encryption and decryption process by outsourcing heavy computations to the proxy server and also improve the efficiency of the outsourced encryption and decryption computation algorithms by designing short cipher-text and less number of pairing operations required.

Correctness Verify the correctness of the message and the outsourced encryption and decryption computations.

5 Construction of NVO-CP-ABE scheme

In this section, we present the algorithmic construction of our NVO-CP-ABE scheme. NVO-CP-ABE scheme consists of four different phases such as setup, data encryption, key generation, and data decryption. The detailed algorithmic construction of these phases is explained below.

5.1 Setup phase

The main purpose of this phase is to generate the public parameters and the user registration. This phase consists of two algorithms such as SystemSetup, and UserSetup.

5.1.1 SystemSetup

SystemSetup algorithm is used to generate the public parameters. Let U be the universal attribute set. Let UL be the user list in the system. Let G_s be additive cyclic group of prime order p and G_t be the multiplicative cyclic group of prime order p. Let $\mathfrak{g}$ be the generator of group G_s. Let $\varrho : G_s \times G_s \rightarrow G_t$, which denotes the bilinear map. Now, TA generates the public key and master secret key with the following steps.

1. Initialize UL $= \perp$
2. Select random value α from Z_p.
3. Choose random value 't_j' for each attributes in U from Z_p, and compute $T_j = \mathfrak{g}^{t_j}$, j = 1 to n; n is number of attributes in U.
4. Compute $Y = \varrho(\mathfrak{g}, \mathfrak{g})^{\alpha}$
5. Return PK = ($\mathfrak{g}$, Y, T_j: j=1, 2, ..., n), MSK = (α, (t1, t2, t3, ..., tn))

5.1.2 UserSetup

Whenever the user joins the system, UserSetup algorithm is used to register the users and set the permissible number of data access for the users.

1. Generates unique identification number (UID) from Z_p
2. Assign the maximum number of data access limit to *acccnt*. If the user is permitted to access any number of times without limit then assign $acccnt = -1$
3. Add UID and *acccnt* in UL
4. Return UID and UL

At the end, the TA sends the UID to the user.

5.2 Data encryption phase

In this phase, we explain how the cipher-text is generated and uploaded into the cloud. There are four algorithms involved in this encryption phase such as Encrypt_Preproc, Encrypt_outsource, Encrypt_owner, and Verify_outsource_enc.

5.2.1 Encrypt_Preproc

The purpose of this algorithm is to find the secret share value for each attributes in the access tree which is used to provide the access control. The data owner first defines the access structure (τ) and gives it as the input to this algorithm. It returns s_v, sh_v_x as output. The steps involved in the algorithm are as follows:

1. Choose a random number s_v from Z_p
2. Lets r0 be the root node of access tree and r_x be the threshold value of the node x. Perform the following steps for each node of the access tree starting from the root node. if x = r0 then

   (a) Polynomial q_{r0} is chosen randomly with the degree $r_{r0} - 1$ and $q_{r0}(0) = s\_v$

   (b) The secret s_v is divided into a number of shares according to the number of child nodes available for root node r0. Find out the shares s_v_y for each child node using secret sharing scheme $s\_v_y = q_{r0}(index(y))$, y denotes the child node attributes of r0.

   else

   (a) Polynomial q_x with the degree $r_x - 1$ is chosen randomly for every child node.

(b) Check if x is a leaf node or not, if found to be a leaf node then compute $q_x(0)$ as $q_x(0) = s\_v_x = q_{parents(x)}(index(x))$.
(c) For all non leaf nodes x, again s_v_x is further divided into shares s_v_y using secret sharing scheme and assign it to each child node. $s\_v_y = q_x(index(y))$.

3. Let LN denotes all the available leaf nodes in the access tree and for each leaf node x in LN, $sh\_v_x = s\_v_x$
4. Return (s_v, $\forall x \in LN : sh\_v_x$)

The DO sends s_v and $\forall x \in LN : sh\_v_x$ to proxy server to compute the intermediate cipher-text.

5.2.2 Encrypt_outsource

After receiving the share values and PK from DO, the PS generates the part of cipher-text with the following steps.

1. Compute CT1 = $\mathfrak{g}^{s\_v}$
2. Let LN denote all the available leaf nodes in the access tree, and for each leaf node j in LN, Compute $CT_j = T_j^{sh\_v_j} = \mathfrak{g}^{t_j q_j(0)}$
3. Returns ICT = (CT1, $\forall j \in LN : CT_j$)

Then proxy server sends the intermediate cipher-text ICT to the data owner.

5.2.3 Encrypt_owner

Once receiving the intermediate cipher-text ICT from proxy, the data owner sends the request to TPA for checking the correctness of proxy computations. Simultaneously, DO generates part of cipher-text and verification key (VK) for the message and decryption outsourcing computations using the collusion-resistance hash function. The following are the steps involved in generating the complete cipher-text.

1. Choose random number t $\in Z_p$
2. Compute CT2 = $\mathfrak{g}^t$
3. Choose random number $key \in Z_p$
4. Compute CT3 = $(key).Y^t = key.\varrho(\mathfrak{g}, \mathfrak{g})^{\alpha t}$
5. Encrypt the data using symmetric key encryption algorithm $E_{key}(M)$
6. Compute VK = H(Key, $E_{key}(M)$), where H is the collusion-resistance hash function.
7. Return C = $(\tau, E_{key}(M)$, CT1, $\forall j \in LN : CT_j$, CT2, CT3, VK)

After generating cipher-text and receiving the TPA verification *flag*, the DO uploads the C into the cloud server if the TPA returns flag = 1.

5.2.4 Verify_outsource_enc

Once receiving the verification request from DO, TPA verifies the correctness of proxy computations. If proxy outsourced encryption is correct, then it returns flag = 1, otherwise it returns the flag = 0. This algorithm takes the input sh_v_x, PK, ICT and performs the following verification step.

$\varrho(\mathfrak{g}, CT_j) = \varrho(\mathfrak{g}^{sh\_v_j}, T_j)\ \forall j \in LN$
$\varrho(\mathfrak{g}, \mathfrak{g}^{t_j \cdot q_j(0)}) = \varrho(\mathfrak{g}^{q_j(0)}, \mathfrak{g}^{t_j})\ \forall j \in LN$
$\varrho(\mathfrak{g}, \mathfrak{g})^{t_j \cdot q_j(0)} = \varrho(\mathfrak{g}, \mathfrak{g})^{t_j \cdot q_j(0)}\ \forall j \in LN$

5.3 User key generation phase

In this phase, the TA generates user secret key and user global secret key for all registered users of the system using KeyGeneration algorithm.

KeyGeneration The trusted authority generates the S and UGSK for every user as follows with the inputs of UID, MSK, UL, and the set of user's attributes δ.

1. If UID $\in$ UL, then performs the following otherwise return S = $\perp$
2. Choose random number k from Z_p and assigns UGSK = k
3. Choose random number $\gamma \in Z_p$.
4. Compute SK1 = $\mathfrak{g}^{(\alpha+\gamma)UGSK}$
5. Compute SK2 = $\mathfrak{g}^{\gamma}$
6. For each attribute a_j in δ, compute $SK_j = \mathfrak{g}^{\frac{\gamma}{t_j}}$
7. Return S = (SK1, SK2, SK_j), UGSK

After generating the secret key, TA sends the UGSK to the user and stores the secret key (S) in the proxy server. Even though the user secret key is stored in the proxy server, the PS is not able to decrypt the data because without UGSK it is not possible to decrypt the data successfully.

5.4 Data decryption phase

In this phase, we explain how the DU decrypts the data from cipher-text. This decryption phase consists of three algorithms such as Decyprt_outsource, Verify_outsource_dec, Decrypt_user.

5.4.1 Decrypt_outsource

Whenever the user wants to access the data, the user sends the request to the proxy server for partial decrypted cipher-text. The proxy receives the cipher-text and UL from CS and first verifies whether the user data access limit is exceeded or

not. If the limit is not exceeded then it computes and sends the PDCT to the user using secret key (S) which is available with PS itself.

First it reads the *acccnt* value from UL. If $acccnt = -1$ or $acccnt > 0$ then it proceeds the following steps to compute PDCT or else it returns $\perp$, that means the user data access limit is exceeded.

1. $acccnt = acccnt - 1$ and update the *acccnt* value in the UL.
2. Let's us find out the secret share value of all the nodes in the access tree. For each leaf nodes $x \in LN$, LN denotes the available leaf nodes in the access tree. Let a_j be the attribute related with leaf node x. If $a_j \notin \delta$, then it returns $\perp$, otherwise computes SV_x.

$$SV_x = \varrho(SK_j, CT_j) = \varrho(\mathfrak{g}^{\frac{\gamma}{t_j}}, \mathfrak{g}^{q_x(0)t_j})$$
$$= \varrho(\mathfrak{g}, \mathfrak{g})^{\frac{\gamma}{t_j} q_x(0) t_j} = \varrho(\mathfrak{g}, \mathfrak{g})^{\gamma q_x(0)}$$

3. 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 the all the child nodes of non-leaf node x and δ_x be the attributes of ρ. If all child nodes share value of node x is $\perp$, then $SV_x = \perp$, otherwise compute SV_x. Let i = index(ρ) and $\delta_x^{'}$) = index(ρ):$\rho \in \delta_x$

$$SV_x = \prod_{\rho \in \delta_x} (SV_\rho)^{\triangle_{i,\delta_x'}(0)}$$
$$= \prod_{\rho \in \delta_x} (\varrho(\mathfrak{g}, \mathfrak{g})^{\gamma q_\rho(0)})^{\triangle_{i,\delta_x'}(0)}$$
$$= \prod_{\rho \in \delta_x} (\varrho(\mathfrak{g}, \mathfrak{g})^{\gamma q_{parent(\rho)}(index(\rho))})^{\triangle_{i,\delta_x'}(0)}$$
$$= \prod_{\rho \in \delta_x} \varrho(\mathfrak{g}, g)^{\gamma q_\rho(i) \triangle_{i,\delta_x'}(0)}$$
$$= \varrho(\mathfrak{g}, \mathfrak{g})^{\gamma q_x(0)}$$

4. Let $A = SV_{r0} = \varrho(\mathfrak{g}, \mathfrak{g})^{\gamma q_{r0}(0)} = \varrho(\mathfrak{g}, \mathfrak{g})^{\gamma s}$
5. Compute B;

$$B = \varrho(SK2, CT1) = \varrho(\mathfrak{g}^{\gamma}, \mathfrak{g}^{s}) = \varrho(\mathfrak{g}, \mathfrak{g})^{\gamma s}$$

6. Compute C;

$$C = \varrho(SK1, CT2) = \varrho(\mathfrak{g}^{(\alpha+\gamma)UGSK}, \mathfrak{g}^{t})$$
$$= \varrho(\mathfrak{g}, \mathfrak{g})^{(\alpha+\gamma)UGSK.t}$$
$$= \varrho(\mathfrak{g}, \mathfrak{g})^{\alpha t UGSK} . \varrho(\mathfrak{g}, \mathfrak{g})^{\gamma t UGSK}$$

7. Compute D;

$$D = \varrho(SK2, CT2) = \varrho(\mathfrak{g}^{\gamma}, \mathfrak{g}^{t}) = \varrho(\mathfrak{g}, \mathfrak{g})^{\gamma t}$$

8. Compute PDCT1

$$PDCT1 = \frac{A}{B} \times D = \frac{\varrho(\mathfrak{g}, \mathfrak{g})^{\gamma s}}{\varrho(\mathfrak{g}, \mathfrak{g})^{\gamma s}} \times \varrho(\mathfrak{g}, \mathfrak{g})^{\gamma t} = \varrho(\mathfrak{g}, \mathfrak{g})^{\gamma t}$$

9. Returns PDCT = (PDCT1, PDCT2=C), CT3, VK, $E_{key}(M)$.

5.4.2 Verify_outsource_dec

Once the DU receives PDCT from PS, the DU sends the PDCT, CT3, VK, $E_{key}(M)$, UGSK as input to TPA for verifying the correctness of the message and proxy computations. The TPA checks the correctness using the Verify_outsource_dec algorithm, and it returns decryption *Key* if the message and outsource computations are correct, otherwise it returns 0 to the DU. If it returns 0, it denotes any one of the following (1) the encrypted message or data may be altered (2) outsourced decryption computation by PS is wrong (3) the DU's attributes are not satisfying the access structure in the cipher-text i.e. unauthorized user.The following are the verification steps:

1. Compute *Key*

$$Key = CT3/((PDCT2)^{1/UGSK}/PDCT1)$$
$$= CT3/\frac{(\varrho(\mathfrak{g}, \mathfrak{g})^{\alpha t UGSK} . \varrho(\mathfrak{g}, \mathfrak{g})^{\gamma t UGSK})^{1/UGSK}}{\varrho(\mathfrak{g}, \mathfrak{g})^{\gamma t}}$$
$$= key . \varrho(\mathfrak{g}, \mathfrak{g})^{\alpha t} / \varrho(\mathfrak{g}, \mathfrak{g})^{\alpha t}$$

2. Compute the hash value VK1 = H(*Key*, $E_{key}(M)$)
3. If VK = VK1, then returns *Key*, otherwise returns 0.

5.4.3 Decrypt_user

After getting the verification status from TPA, the DU performs the following:

1. if TPA returns *Key*, then
   (a) perform symmetric decryption $D_{key}(E_{key}(M))$
   (b) return M
2. if TPA returns 0, then return $\perp$

In case, Decrypt_outsource algorithm returns $\perp$, then Decrypt_user algorithm returns $\perp$, which means the data access limit of the user is exceeded.

6 Security analysis

In security analysis, we prove that NVO-CP-ABE scheme is secure against the attacks that mentioned in the Sect. 4.3.

Theorem 1 *NVO-CP-ABE scheme is selectively secure against CPA. The polynomial-time adversary* ($\mathcal{A}$) *cannot selectively break our NVO-CP-ABE scheme in the security game mentioned in the* Sect. 3.6 *if DBDH assumption holds.*

Proof In the proof, we have reduced security of the chosen plain-text attack into the DBDH assumption. It implies that if DBDH problem is solved, then the adversary can breach this scheme successfully. Construct the simulator (χ) which performs the challenger (ς) task. If $\mathcal{A}$ wins this selectively CPA secure game with a non-negligible advantage ϵ, then χ can solve the DBDH assumptions in $\frac{\epsilon}{2}$. The challenger creates a basic setup to play the game. Let G_s be additive cyclic group of prime order p and G_t be the multiplicative cyclic group of prime order p. Let $\mathfrak{g}$ be the generator of G_s. The bilinear mapping function defined as $\varrho : G_s \times G_s \rightarrow G_t$ and pick a, b, c $\in Z_p$ as random values. The ς tosses the coin $\sigma \in \{0, 1\}$ and defines $\mathfrak{Z} = \varrho(\mathfrak{g}, \mathfrak{g})^{abc}$ when $\sigma = 0$, otherwise defines $\mathfrak{Z} = \mathfrak{R}$, and let $\mathfrak{R}$ be the random value from G_t. Now the ς gives the basic setup to the simulator and the responsibility to play the game. The challenge given to the simulator is $(\mathfrak{g}, A, B, C, \mathfrak{Z}) = (\mathfrak{g}, \mathfrak{g}^a, \mathfrak{g}^b, \mathfrak{g}^c, \mathfrak{Z})$. Now the challenger for adversary is simulator, the simulator starts playing the game with the adversary.

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

Setup phase In this phase, the simulator generates public parameters such as public key and master secret key and sends the PK to $\mathcal{A}$. For generating PK, the simulator randomly chooses $\kappa \in Z_p$ and let $\alpha = ab + \kappa$. Then, it computes $Y = \varrho(\mathfrak{g}, \mathfrak{g})^{\alpha} = \varrho(\mathfrak{g}, \mathfrak{g})^{\mathfrak{a}\mathfrak{b}+\kappa} = \varrho(\mathfrak{g}, \mathfrak{g})^{\mathfrak{a}\mathfrak{b}}.\varrho(\mathfrak{g}, \mathfrak{g})^{\kappa}$.

Next it computes T_j, by choosing a random number $t_j \in Z_p$ for all attributes a_j in U.

$$T_j = \begin{cases} \mathfrak{g}^{b/t_j} = B^{1/t_j}, & \text{if } a_j \notin \tau^* \\ \mathfrak{g}^{t_j}, & \text{if } a_j \in \tau^* \end{cases}$$

Now the simulator sends the PK= ($\mathfrak{g}$, Y, T_j: j=1, 2, ..., n) similar to the original scheme to the adversary.

Query phase 1 In this phase, the adversary requests the simulator for the secret key for set of attributes $\delta^* = \{a_j | a_j \in U\}$ with the constraint $a_j \notin \tau^*$. First, select a random number and compute the as same as in the KeyGeneration algorithm. Choose the random number $\gamma' \in Z_p$ and compute the secret key as per the algorithm.

Compute $SK1^* = \mathfrak{g}^{(\kappa+\gamma' b)UGSK} = \mathfrak{g}^{(\alpha-ab+\gamma' b)UGSK} = \mathfrak{g}^{(\alpha+\gamma' b-ab)UGSK} = \mathfrak{g}^{(\alpha+(\gamma' b-ab))UGSK}$. Thus $\gamma = \gamma' b - ab$.

Compute $SK2^* = \mathfrak{g}^{\gamma} = \mathfrak{g}^{\gamma' b-ab}$

Compute $SK_j^* = \mathfrak{g}^{\frac{\gamma}{t_j}} = \mathfrak{g}^{\frac{\gamma' b-ab\times t_j}{b}} = \mathfrak{g}^{\frac{(\gamma'-a)b\times t_j}{b}} = \mathfrak{g}^{\gamma' t_j - a t_j} = A^{-t_j}.\mathfrak{g}^{\gamma' t_j}$.

After generating the secret key $S^* = (SK1^*, SK2^*, SK_j^*)$, UGSK, the simulator sends it to $\mathcal{A}$.

Challenge Adversary submits two identical size messages M_0 and M_1 to the simulator. The χ tosses a binary coin (σ), and encrypts the message based on the value σ. First, execute the Encrypt_Preproc algorithm to compute the secret share values for each attributes in the access tree and get the ICT = (CT1, $\forall j \in LN : CT_j$) value by executing the Encrypt_outsource algorithm. The cipher-text is computed as below. Choose a random number $c \in Z_p$. Choose a random symmetric encryption key, $key \in Z_p$ and encrypt the data using the key $E_{key}(M_\sigma)$.

$CT2^* = \mathfrak{g}^c$

$CT3^* = key.\varrho(\mathfrak{g}, \mathfrak{g})^{\alpha.c} = key.\varrho(\mathfrak{g}, \mathfrak{g})^{(ab+\kappa).c} = key.\varrho(\mathfrak{g}, \mathfrak{g})^{abc}.\varrho(\mathfrak{g}, \mathfrak{g})^{\kappa.c} = key.\mathfrak{Z}.\varrho(\mathfrak{g}, \mathfrak{g})^{\kappa.c}$.

Send the challenge cipher-text $C^* = (\tau^*, E_{key}(M_\sigma), CT1^*, CT2^*, CT3^*, \forall j \in LN : CT_j^*)$ to the adversary.

Query phase 2 Adversary may ask the secret key repeatedly same as in query phase 1 with same constraints. The simulator also does the same as in query phase 1 whenever there is a request from adversary.

Guess Adversary submits the guess $\sigma' \in \{0, 1\}$ to the simulator for σ. If $\sigma = \sigma'$ then simulator outputs that $\sigma = 0$, so $\mathfrak{Z} = \varrho(\mathfrak{g}, \mathfrak{g})^{abc}$ which is a well-founded cipher-text. Hence, the advantage of adversary is $Pr[\sigma = \sigma' | \mathfrak{Z} = \varrho(\mathfrak{g}, \mathfrak{g})^{abc}] = \frac{1}{2} + \epsilon$, otherwise the simulator outputs that $\sigma = 1$, so $\mathfrak{Z} = \mathfrak{R}$, which is mapped to random. So the disadvantage of the adversary which is not to get information is $Pr[\sigma \neq \sigma' | \mathfrak{Z} = \mathfrak{R}] = \frac{1}{2}$.

Henceforth, on the whole advantage of the simulator to solve the DBDH assumption is $\frac{1}{2}(Pr[\sigma = \sigma' | \sigma = 0] + \frac{1}{2} Pr[\sigma \neq \sigma' | \sigma = 1]) - \frac{1}{2} = \frac{\epsilon}{2}$, thus NVO-CP-ABE scheme is secure against chosen plain-text attack.

Theorem 2 *NVO-CP-ABE scheme is free from user collusion attack.*

Proof In our scheme, the secret key S = (SK1, SK2, SK_j) is stored in the proxy server and the USGK alone is given to the DU. Whenever the user wants to access the data, the user first sends the request to PS for PDCT. The PS computes PDCT using C = ($\tau, E_{key}(M)$, CT1, CT2, CT3, $\forall j \in LN : CT_j$), S and sends it to the DU. That means, the user won't have a chance to hold or store their secret key, so it is not possible to share the secret key component with other users. Thus NVO-CP-ABE scheme is free from user collusion attack.

Theorem 3 *NVO-CP-ABE scheme is secure against proxy malfunction and attack.*

Proof In our scheme, the semi-trusted proxy server is involved in three major tasks such as performing partial encryption task, partial decryption task and storing the user secret key on behalf of data users. Following are the three possible malfunctions and attacks that might be caused by a PS.

Case 1 Malfunction during partial encryption computation: While executing Encrypt_outsource algorithm by the PS, the PS may give the intermediate cipher-text to DO without computing all the steps. If that happens, then, it is not possible to reconstruct the exact secret value during decryption even for the authorized users, which means no one can access the data. To secure the system and to identify the malfunction of PS, DO verifies whether PS computed intermediate cipher-text ICT = (CT1, $\forall j \in LN : CT_j$) is correct or not with the help of TPA using Verify_outsource_enc algorithm. Thus, our scheme NVO-CP-ABE is secure from malfunction of PS during outsourcing encryption computation.

Case 2 Malfunction during partial decryption computation: While executing Decrypt_outsource algorithm by the PS, the PS may avoid or skip some pairing computations and gives the incorrect partially decrypted cipher-text to DU. This affects the smooth functioning of the system by restricting the data access even for the authorized users. To secure the system and to find out the malfunction of PS, DU verifies whether PS computed PDCT=(PDCT1, PDCT2) is correct or not with the help of TPA using Verify_outsource_dec algorithm. If Verify_outsource_dec algorithm returns the value zero means that PS has performed some malfunction. Thus our scheme NVO-CP-ABE is secure from malfunction of PS during outsourcing decryption computation.

Case 3 Proxy attack - PS tries to access the data: The entire user secret key is stored in PS, so PS may try to decrypt with the help of stored user's secret key. But PS is not able to decrypt the cipher-text because it requires UGSK, but UGSK is only available with the user. Even if PS is trying to obtain plain-text without UGSK, PS will not exactly get $\varrho(\mathfrak{g}, \mathfrak{g})^{\alpha t}$ value. Without $\varrho(\mathfrak{g}, \mathfrak{g})^{\alpha t}$, it unable to obtain the *key*. Thus our scheme is secure against proxy server attack.

Theorem 4 *NVO-CP-ABE scheme is secure against proxy and unauthorized user collusion attack.*

Proof In our scheme, the secret key and UGSK is generated for every registered user. The secret key is stored in the proxy server and UGSK is given to the user. Suppose, if the unauthorized user tries to collude with proxy server to access the data, then proxy server may give the authorized user's partially decrypted cipher-text to unauthorized user. Eventhough the unauthorized user gets the PDCT of authorized users still it is not possible for it to access the data because the decryption requires *Key*. The exact *Key* can be obtained only when the UGSK is bound within the PDCT and UGSK of the user is same. But in this case, PDCT binds with authorized user UGSK which is not same as the unauthorized user UGSK, so unauthorized users is not able to obtain the exact *Key*. Thus our scheme NVO-CP-ABE is secure against proxy and unauthorized user collusion attack.

Theorem 5 *NVO-CP-ABE scheme is to ensure the message correctness property.*

Proof In our scheme, the DO generates the verification key VK = H(*Key*, $E_{key}(M)$) using collusion resistance function to verify the correctness of the message and outsourced decryption computations. The DU verifies the correctness of the message and proxy computed decryption with the help of TPA. The TPA finds the hash value VK1= H(*Key*, $E_{key}(M)$) for the obtained key and encrypted message received from proxy server. If the computed hash value (VK1) is same as VK, then we conclude that the message is not modified and the proxy computation is also correct, otherwise we assume that the message is modified or proxy computation is wrong. Thus our scheme ensures the message correctness.

7 Performance analysis

In performance analysis, we evaluate the performance of NVO-CP-ABE scheme theoretically as well as experimentally. The different notations used in performance analysis are listed in Table 1.

7.1 Theoretical analysis

In theoretical analysis, we compare NVO-CP-ABE scheme with other existing schemes with respect to features, storage overheads, communication overheads, and computation overheads.

7.1.1 Feature analysis

In feature analysis, we compare the functionality of different outsourced CP-ABE schemes (Lin et al. 2015; Mao et al. 2016; Ning et al. 2018; Wang et al. 2017a) in the literature with NVO-CP-ABE scheme and it is tabulated in Table 2. We can see that Lin et al. (2015) and Mao et al. (2016) schemes outsources only the decryption computations and verifies the outsourced decryption computations. In addition, Ning et al. (2018) scheme achieves limited data access. Wang et al. (2017a) scheme achieves verifiable outsourced encryption and decryption, but this scheme is not verifying the correctness of the outsourced decryption computations and is not supporting the limited

Table 1 Notations used in performance analysis

Notation	Meaning
n_U	Number of universal attributes
n_δ	Number of user attributes
n_τ	Number of attributes in the access tree or number of rows in the matrix
n_{nl}	Number of non-leaf nodes in the access tree
t_p	Time for one pairing operation
t_e	Time for one exponentiation operation

access. Our scheme achieves the verifiable outsourced encryption and decryption along with the limited data access for a set of users altogether.

7.1.2 Storage overhead

Here, we consider the storage overheads of data owner and data user only. The storage overhead of DO and DU depends on the size of the public key and secret key respectively, because the data owner needs a public key for generating cipher-text and the data users needs a secret key for decrypting the cipher-text. The Table 3 gives the comparative study of storage required for DO and DU of NVO-CP-ABE scheme with existing schemes.

It is clearly noted from Table 3 that the storage overhead of DU in our scheme is highly reduced than other schemes, because the user global secret key is only stored with DU and the secret key is stored with PS. The storage overhead of the DO is also relatively better than all schemes expect Ning et al. (2018), but Ning et al. (2018) scheme suffers with DU's high storage overhead and high DO's computation overhead.

7.1.3 Communication overhead

Table 4 compares the communication overhead of our scheme and other existing schemes. The major communication overhead happens when uploading cipher-text into the cloud, sending secret key for outsourced computations and receiving the cipher-text as an input to the decryption function. Table 4 shows that the communication overhead of our scheme between DO and CS is less than other schemes because of short cipher-text. In addition, the communication cost of our scheme between DU and PS is highly reduced because there is no secret key communication between DU and PS in our scheme.

Table 3 Storage overhead comparison

Scheme	Data owner	Data users
Lin et al. (2015)	$(n_U + 1)G_s + G_t$	$(n_\delta + 2)G_s$
Mao et al. (2016)	$(n_U + 1)G_s + G_t$	$(n_\delta + 2)G_s$
Ning et al. (2018)	$G_s + G_t$	$(2n_\delta + 2)G_s$
Wang et al. (2017a)	$(n_U + 2)G_s + G_t$	$(n_\delta + 3)G_s$
Our scheme	$n_U G_s + G_t$	Z_p

Table 4 Communication overhead comparison

Scheme	DO and CS	DU and PS
Lin et al. (2015)	$(2n_\tau + 1)G_s + G_t$	$(n_\delta + 2n_\tau + 3)G_s + G_t$
Mao et al. (2016)	$(2n_\tau + 1)G_s + G_t$	$(n_\delta + 2n_\tau + 3)G_s + G_t$
Ning et al. (2018)	$(3n_\tau + 1)G_s + G_t$	$(2n_\delta + 3) + G_s + G_t$
Wang et al. (2017a)	$(3n_\tau + 2)G_s + G_t$	$(n_\delta + 3) + G_s + G_t$
Our scheme	$(n_\tau + 2)G_s + G_t$	$Z_p + G_t$

7.1.4 Computation overhead

In the computation overhead analysis, we compute the number of pairing and exponentiation operations required for owner encryption, outsourced encryption and decryption, user decryption are tabulated in Table 5. As shown in Table 5, during encryption phase, the DO in our scheme requires only 2 exponentiation operations which is less than Lin et al. (2015), Mao et al. (2016), Ning et al. (2018) and Wang et al. (2017a) schemes and the computation cost of outsourced encryption is $(n_\tau + 1)t_e$ which is better than Wang et al. (2017a) scheme because our scheme requires

Table 2 Features comparison

Scheme	Outsourcing encryption	Outsourcing encryption verification	Outsourcing decryption	Outsourcing decryption verification	Message correctness verification	Limited data access
Lin et al. (2015)	×	×	✓	✓	✓	×
Mao et al. (2016)	×	×	✓	✓	✓	×
Ning et al. (2018)	×	×	✓	✓	×	✓
Wang et al. (2017a)	✓	✓	✓	×	×	×
Our scheme	✓	✓	✓	✓	✓	✓

Table 5 Computation overhead comparison

Scheme	Encryption by owner	Outsourced encryption	Decryption by user	Outsourced decryption
Lin et al. (2015)	$(3n_\tau + 2)t_e$	×	1	$(2n_\delta + 1)t_p$
Mao et al. (2016)	$(3n_\tau + 2)t_e$	×	1	$(2n_\delta + 1)t_p$
Ning et al. (2018)	$(5n_\tau + 2)$	×	t_e	$(3n_\delta + 3)t_p + t_e$
Wang et al. (2017a)	$(2n_\tau + 3)t_e$	$(2n_\tau)t_e$	t_p	$(3n_\delta + 2)t_p + 2t_e$
Our scheme	$2t_e$	$(n_\tau + 1)t_e$	1	$(n_\delta + 3)t_p + n_{nl}t_e$

less number of exponentiation operations. During the decryption phase, no exponentiation and pairing operations are required for DU in our scheme and the computation cost of outsourced decryption is $(n_\delta + 3)t_p + n_{nl}t_e$ which is less than Lin et al. (2015), Mao et al. (2016), Ning et al. (2018) and Wang et al. (2017a) schemes because our scheme require less number of pairing operations. Thus, NVO-CP-ABE scheme need less computation time when compared to other schemes.

7.2 Experimental analysis

In experimental analysis, we compare the efficiency of our scheme with Wang et al. (2017a) scheme. We implement both schemes using JetBrains PyCharm tool with charm framework (Akinyele et al. 2013). We use 512-bit base field SS512 elliptic curve from pairing based charm crypto-0.42 library and executed on an Intel core i7 processor @ 2.50 GHz, 16 GB RAM running on a windows 8 operating system and python 3.2. The results are obtained from an average of ten executions for each algorithm. The experimental results for secret key size, cipher-text size, computation time of owner encryption, proxy performed encryption and decryption, and user decryption are obtained and plotted as graphs. Figure 2 shows the growth of secret key size against number of user attributes, and Fig. 3 shows the relation between cipher-text size with respect to the number of attributes in the access policy. From Figs. 2 and 3, it is clearly observed that NVO-CP-ABE scheme has

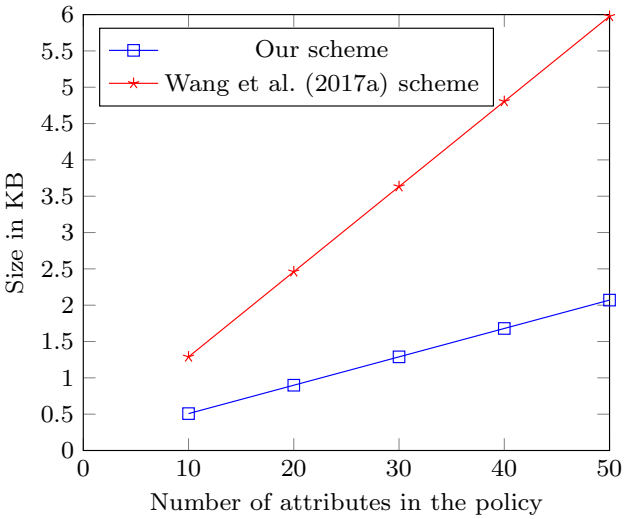

Fig. 3 Cipher-text size comparison

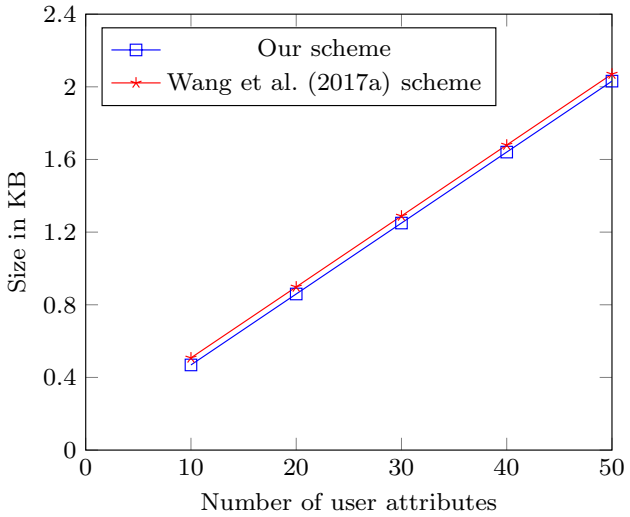

Fig. 2 Secret key size comparison

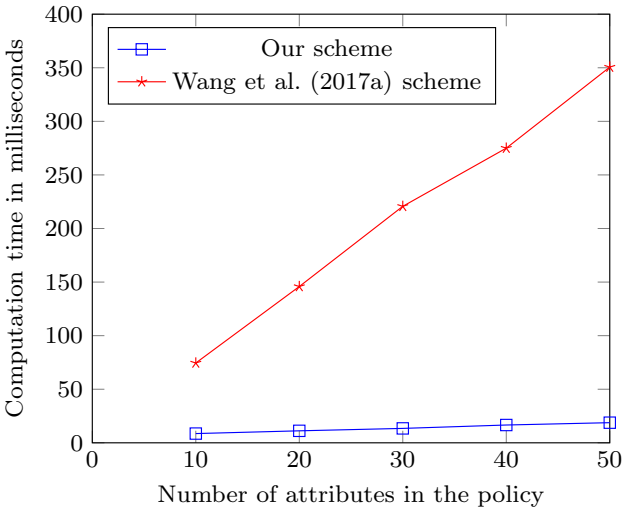

Fig. 4 Encryption computation time by data owner

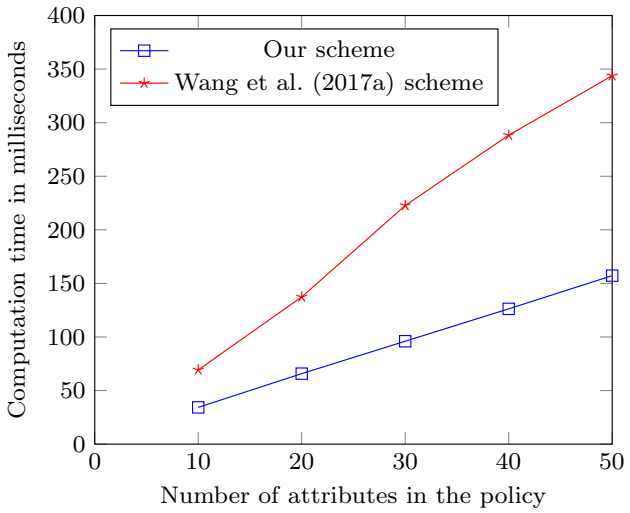

Fig. 5 Encryption computation time by proxy

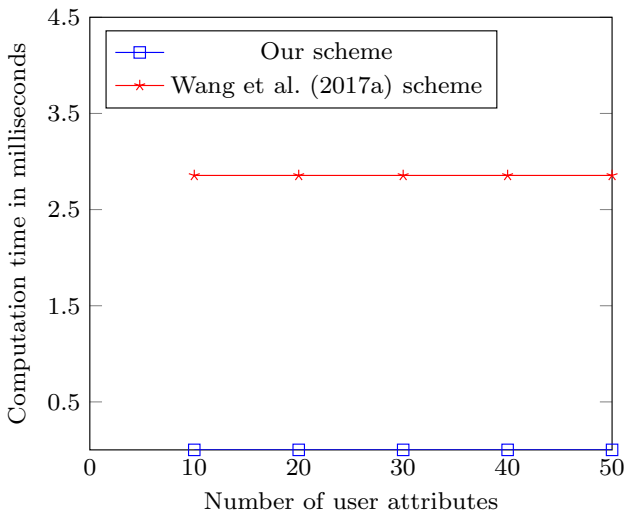

Fig. 6 Decryption computation time by data user

short secret key and short cipher-text than Wang et al. (2017a) scheme, which implies that NVO-CP-ABE scheme requires less communication overhead. Figure 4 shows the computation time of owner encryption against a number of attributes in the access policy. Figure 5 demonstrates the computation time of outsourced encryption against a number of attributes in access policy. Figures 4 and 5 clearly shows that computation cost of encryption phase is less than Wang et al. (2017a) scheme because of cipher-text size. The computation time of user decryption against number of user attributes is illustrated in Fig. 6. Figure 7 shows the computation time of outsourced decryption against number of user attributes. From Figs. 6 and 7, it is noted that our scheme requires less decryption phase cost than Wang et al. (2017a) scheme because of less number

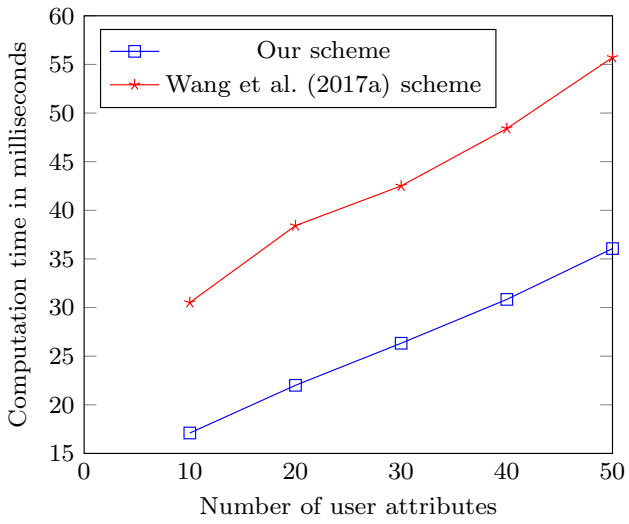

Fig. 7 Decryption computation time by proxy

of pairing operations required. Thus our scheme requires less computation time in all algorithms than Wang et al. (2017a) scheme.

8 Conclusion

Aiming to provide privacy and flexible access control for big data in cloud, we propose NVO-CP-ABE scheme. The NVO-CP-ABE scheme improves the computational efficiency of CP-ABE scheme for big data by outsourcing encryption and decryption computations to the proxy server. In addition, NVO-CP-ABE scheme checks the correctness of the encrypted message along with outsourced computations and supports limited data access for a set of users. The NVO-CP-ABE scheme is proved secure against CPA in the standard model, user collusion attack, and proxy attack. Theoretical analysis and experimental results show that the NVO-CP-ABE scheme is efficient in all aspects than existing schemes. Hence, NVO-CP-ABE scheme is more appropriate for big data privacy and access control in the cloud. In our future work, we will extent this work with revocation to provide the effective access control.

References

Akinyele JA, Garman C, Miers I, Pagano MW, Rushanan M, Green M (2013) Charm: a framework for rapidly prototyping cryptosystems. J Cryptogr Eng 3(2):111–128. https://doi.org/10.1007/s13389-013-0057-3

Bethencourt J, Sahai A, Waters B (2007) Ciphertext-policy attribute-based encryption. In: Security and privacy, 2007. SP'07. IEEE symposium, IEEE, pp 321–334. https://doi.org/10.1109/SP.2007.11

Cheung L, Newport C (2007) Provably secure ciphertext policy ABE. In: Proceedings of the 14th ACM conference on computer and

communications security. ACM, Alexandria, Virginia, pp 456–465. https://doi.org/10.1145/1315245.1315302
De SJ, Ruj S (2015) Decentralized access control on data in the cloud with fast encryption and outsourced decryption. In: 2015 IEEE global communications conference (GLOBECOM). IEEE, San Diego, CA, pp 1–6. https://doi.org/10.1109/GLOCOM.2015.7417639
Deng H, Wu Q, Qin B, Domingo-Ferrer J, Zhang L, Liu J (2014) Ciphertext-policy hierarchical attribute-based encryption with short ciphertexts. Inf Sci 275:370–384. https://doi.org/10.1016/j.ins.2014.01.035
Goyal V, Pandey O, Sahai A, Waters B (2006) Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of the 13th ACM conference on computer and communications security. ACM, Alexandria, Virginia, pp 89–98. https://doi.org/10.1145/1180405.1180418
Green M, Hohenberger S, Waters B (2011) Outsourcing the decryption of ABE ciphertexts. In: Proceedings of the 20th USENIX conference on security. USENIX Association, San Francisco, CA, p 34
Gupta B, Agrawal DP, Yamaguchi S (2016) Handbook of research on modern cryptographic solutions for computer and cyber security. IGI Global, Hershey, PA, pp 1–589. https://doi.org/10.4018/978-1-5225-0105-3
Jiang Y, Susilo W, Mu Y, Guo F (2017) Flexible ciphertext-policy attribute-based encryption supporting AND-gate and threshold with short ciphertexts. Int J Inf Secur. https://doi.org/10.1007/s10207-017-0376-y
Khan N, Yaqoob I, Hashem IAT, Inayat Z, Ali M, Kamaleldin W et al (2014) Big data: survey, technologies, opportunities, and challenges. Sci World J. https://doi.org/10.1155/2014/712826
Kumar PP, Kumar PS, Alphonse PJA (2018) Attribute based encryption in cloud computing: a survey, gap analysis, and future directions. J Netw Comput Appl 108:37–52. https://doi.org/10.1016/j.jnca.2018.02.009
Li J, Wang Y, Zhang Y, Han J (2017) Full verifiability for outsourced decryption in attribute based encryption. IEEE Trans Serv Comput. https://doi.org/10.1109/TSC.2017.2710190
Li L, Gu T, Chang L, Xu Z, Liu Y, Qian J (2017) A ciphertext-policy attribute-based encryption based on an ordered binary decision diagram. IEEE Access 5:1137–1145. https://doi.org/10.1109/ACCESS.2017.2651904
Li J, Chen X, Chow SS, Huang Q, Wong DS, Liu Z (2018) Multi-authority fine-grained access control with accountability and its application in cloud. J Netw Comput Appl 112:89–96. https://doi.org/10.1016/j.jnca.2018.03.006
Li J, Zhang Y, Chen X, Xiang Y (2018) Secure attribute-based data sharing for resource-limited users in cloud computing. Comput Secur 72:1–12. https://doi.org/10.1016/j.cose.2017.08.007
Li J, Li X, Wang L, He D, Ahmad H, Niu X (2018) Fuzzy encryption in cloud computation: efficient verifiable outsourced attribute-based encryption. Soft Comput 22(3):707–714. https://doi.org/10.1007/s00500-017-2482-1
Lin S, Zhang R, Ma H, Wang M (2015) Revisiting attribute-based encryption with verifiable outsourced decryption. IEEE Trans Inf Forensics Secur 10(10):2119–2130. https://doi.org/10.1109/TIFS.2015.2449264
Ma H, Zhang R, Wan Z, Lu Y, Lin S (2015) Verifiable and exculpable outsourced attribute-based encryption for access control in cloud computing. IEEE Trans Dependable Secure Comput. https://doi.org/10.1109/TDSC.2015.2499755
Ma X, Li J, Zhang F (2013) Outsourcing computation of modular exponentiations in cloud computing. Clust Comput 16(4):787–796. https://doi.org/10.1007/s10586-013-0252-0
Mao X, Lai J, Mei Q, Chen K, Weng J (2016) Generic and efficient constructions of attribute-based encryption with verifiable outsourced decryption. IEEE Trans Dependable Secure Comput 13(5):533–546. https://doi.org/10.1109/TDSC.2015.2423669
Ning J, Cao Z, Dong X, Liang K, Ma H, Wei L (2018) Auditable sigma-time outsourced attribute-based encryption for access control in cloud computing. IEEE Trans Inf Forensics Secur 13(1):94–105. https://doi.org/10.1109/TIFS.2017.2738601
Qin B, Deng RH, Liu S, Ma S (2015) Attribute-based encryption with efficient verifiable outsourced decryption. IEEE Trans Inf Forensics Secur 10(7):1384–1393. https://doi.org/10.1109/TIFS.2015.2410137
Sahai A, Waters B (2005) Fuzzy identity-based encryption. In: Annual international conference on the theory and applications of cryptographic techniques. Springer, Berlin, pp 457–473. https://doi.org/10.1007/11426639_27
Shamir A (1979) How to share a secret. Commun ACM 22(11):612–613. https://doi.org/10.1145/359168.359176
Takabi H, Joshi JB, Ahn GJ (2010) Security and privacy challenges in cloud computing environments. IEEE Secur Priv 8(6):24–31. https://doi.org/10.1109/MSP.2010.186
Voorsluys W, Broberg J, Buyya R (2011) Introduction to cloud computing. Cloud Comput Princ Paradig. https://doi.org/10.1002/9780470940105.ch1
Wang H, He D, Shen J, Zheng Z, Zhao C, Zhao M (2017) Verifiable outsourced ciphertext-policy attribute-based encryption in cloud computing. Soft Comput 21(24):7325–7335. https://doi.org/10.1007/s00500-016-2271-2
Wang H, He D, Han J (2017) VOD-ADAC: anonymous distributed fine-grained access control protocol with verifiable outsourced decryption in public cloud. IEEE Trans Serv Comput. https://doi.org/10.1109/TSC.2017.2687459
Waters B (2011) Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization. In: International workshop on public key cryptography, Springer, Berlin, pp 53–70. https://doi.org/10.1007/978-3-642-19379-8_4
Xiang C, Tang C (2015) Efficient outsourcing schemes of modular exponentiations with checkability for untrusted cloud server. J Ambient Intell Hum Comput 6(1):131–139. https://doi.org/10.1007/s12652-014-0254-7
Xiong H, Sun J (2017) Comments on verifiable and exculpable outsourced attribute-based encryption for access control in cloud computing. IEEE Trans Dependable Secure Comput 14(4):461–462. https://doi.org/10.1109/TDSC.2017.2710119
Ye H, Liu J, Wang W, Li P, Li T, Li J (2018) Secure and efficient outsourcing differential privacy data release scheme in cyber-physical system. Future Gener Comput Syst. https://doi.org/10.1016/j.future.2018.03.034
Zhang Y, Zheng D, Li Q, Li J, Li H (2016) Online/offline unbounded multi-authority attribute-based encryption for data sharing in mobile cloud computing. Secur Commun Netw 9(16):3688–3702. https://doi.org/10.1002/sec.1574
Zhang R, Ma H, Lu Y (2017) Fine-grained access control system based on fully outsourced attribute-based encryption. J Syst Softw 125:344–353. https://doi.org/10.1016/j.jss.2016.12.018
Zhang Y, Chen X, Li J, Wong DS, Li H, You I (2017) Ensuring attribute privacy protection and fast decryption for outsourced data security in mobile cloud computing. Inf Sci 379:42–61. https://doi.org/10.1016/j.ins.2016.04.015