A Privacy-Preserving Access Control Scheme with Verifiable and Outsourcing Capabilities in Fog-Cloud Computing

Zhen Cheng[1], Jiale Zhang[1], Hongyan Qian[1], Mingrong Xiang[2], and Di Wu[3]

[1] College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China
{czheng, jlzhang, qhy98}@nuaa.edu.cn

[2] Faculty of Science, Engineering and Built Environment, Deakin University, Melbourne, VIC 3216, Australia
mxiang@deakin.edu.au

[3] School of Computer Science, Centre for Artifical Intelligence, University of Technology Sydney, Sydney, NSW 2007, Australia
Di.Wu@uts.edu.au

Abstract. Fog computing is a distribution system architecture which uses edge devices to provide computation, storage, and sharing at the edge of the network as an extension of cloud computing architecture, where the potential network traffic jams can be resolved. Whereas, the untrustworthy edge devices which contribute the computing resources may lead to data security and privacy-preserving issues. To address security issues and achieve fine-grained access control to protect privacy of users, ciphertext-policy attribute-based encryption (CP-ABE) mechanism has been well-explored, where data owners obtain flexible access policy to share data between users. However, the major drawback of CP-ABE system is heavy computational cost due to the complicated cryptographic operations. To tackle this problem, we propose a privacy-preserving access control (PPAC) scheme and the contributions are tri-folded: 1) we introduce outsourcing capability in fog-cloud computing (FCC) environment; 2) the outsource verification mechanism has been considered to guarantee the third party execute the algorithm correctly; 3) we design a partiality hidden method to protect the privacy information embedded in the access structures. The experimental results show that our proposed PPAC is efficient, economical and suitable for mobile devices with limited resources.

Keywords: Fog-cloud computing · Attribute-based encryption · Privacy-preserving · Access Control.

1 Introduction

With the high demand of data storage and sharing in the Internet of Things (IoT), cloud computing has become solution regard to its ability to make all of

the connected devices work together and store the unprecedented amount of data [1]. However, transferring huge amount of data from IoT devices to the cloud server is not only adding latency of the data transportation, but also consumes constrained network resources. For example, vehicles in the autonomous vehicle network might produce gigabytes data in one second, where shorter response time could help the vehicle to avoid the accident [2]. In order to transport large amount of data and reduce the response time, fog computing has been proposed as a promising solution to mitigate the limitation of IoT devices with constrained resources. As an extension of cloud computing, fog computing offloads the communication and computation burden of the network by processing data near the sources of data [3]. However, the security and privacy issues still present practical concerns for fog computing, where the cloud server can not be fully trusted and the edge devices are untrustworthy as well [4]. Therefore, the purpose of this work is to achieve lightweight and privacy-preserving access control in fog-cloud computing (FCC) by outsourcing partial cryptosystem computations to the fog and cloud servers.

The notion of attribute-based encryption (ABE) was first introduced by Sahai and Waters based on fuzzy identity encryption [5]. Furthermore, Goyal et al. proposed the key-policy attribute-based encryption (KP-ABE) scheme [6] and Bethencourt et al. present ciphertext-policy attribute-based encryption (CP-APE) construction methods [7]. Subsequently, a large number of ABE schemes were proposed [8–12], which add different functions on original ABE, such as ciphertext auditing and privacy protecting. Due to the heavy computation cost for the encryption and decryption phases, a new method of outsourcing decryption of ABE was proposed by Green et al [13]. However, since the proxy server cannot be fully trusted, verifiable outsourced computation is required. The technique proposed in [14, 15] can be used to outsource the operations of decryption in ABE systems, because the verification mechanism is considered along with outsourcing ABE schemes. Recently, [16,17] demonstrate that the ABE scheme can be applied to fog computing environment to solve the secure storage problem. However, these solutions still have security and privacy problems: preventing to the untrustworthy third party from learning private information and ensuring the correctness of results returned by the untrustworthy third party.

To tackle the above problems, we propose a verifiable and outsourced ABE mechanism, which not only protects data privacy, fine-grained access control and brings verifiability to the outsourced computation, but also reduces the computation cost. The contributions can be summarized as follows:

- **A lightweight access control model in FCC**: We first propose a novel lightweight ABE system to outsource the computation of both encryption and decryption procedure. As a result, most of the complicated cryptographic operations can be outsourced to the fog or cloud servers.
- **A verifiable mechanism for outsourcing capability**: In order to avoid the incorrect results from cloud servers, we present a concrete verification mechanism for outsourced computation capability.

– **A partially hidden access structure**: To protect the private information embedded in the access structures, we further design a partially hidden method along with the outsourced ABE system. In the proposed system, the attributes are hidden from the users whose private key does not satisfy the access structure.

The rest of this paper are organized as follows. In Section 2, we briefly introduce the basic knowledge of attribute-based encryption method. The system model is presented in Section 3, and the construction of proposed privacy-preserving access control scheme is detailed in Section 4. Extensive experimental evaluation is conducted in Section 5. Finally, Section 6 gives the conclusion.

2 Preliminaries

In this section, we briefly revisit the basic definition of bilinear groups, CP-ABE scheme, access structure, and the linear secret sharing schemes (LSSS).

2.1 Bilinear Maps

Let an algorithm that inputs a security parameter λ and outputs a tuple $(q, \mathbb{G}, \mathbb{G}_T, e)$, where $\mathbb{G}$ and $\mathbb{G}_T$ are two multiplicative cyclic groups of prime order q. The bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ fulfills the following properties:

– Bilinearity: $e(g^a, g^b) = e(g, g)^{ab}$ for all $g \in \mathbb{G}$, $a, b \in \mathbb{Z}_q^*$.
– Non-degeneracy: There exist $e(g, g) \neq 1$ such that $g \in \mathbb{G}$.
– Computable: There is an efficient algorithm to compute e, for all $g \in \mathbb{G}$.

2.2 CP-ABE

CP-ABE scheme is consisted of the following four algorithms:

– **Setup** (λ, U): The setup algorithm inputs security parameter λ and universe description U. It outputs the public parameters PK and master key MSK.
– **KeyGen** (PK, MSK, S): The key generation algorithm takes as inputs the PK, the master key MSK, and a set of attributes S. It outputs the private key SK.
– **Encrypt** (PK, M, A): The encryption algorithm takes as input the PK, the message M, and an access structure A. It outputs the ciphertext CT.
– **Decrypt** (SK, CT): The decryption algorithm takes as inputs the private key SK with attribute set S. If S can satisfy the access structure A, it outputs the message M. Otherwise, it outputs $\perp$.

2.3 Access Structure

Let $\{P_1, P_2, \cdots, P_n\}$ is a collection of attributes, we say $A \in 2^{\{P_1, P_2, \cdots, P_n\}}$ is a monotone selection attribute set if $\forall B$: $B \in A$ and $B \subseteq C$ then $C \in A$. The monotone access structure A is a non-empty subsets of $\{P_1, P_2, \cdots, P_n\}$, i.e., $A \in 2^{\{P_1, P_2, \cdots, P_n\}} \setminus \{\emptyset\}$.

2.4 Linear Secret Sharing Schemes

A secret sharing scheme Π over a set of parties is called linear over $\mathbb{Z}_q$. The definitions are shown below:

- Assuming there are some shares among different parties from a vector over $\mathbb{Z}_q$.
- There exists a matrix M with ℓ rows and n columns called the share-generating matrix for Π. A function ρ which maps each row of the matrix to an associated party (for $i = 1, \cdots, \ell$), where $\rho(i)$ is the party associated with row i.
- When we consider the column vector $v = (s, r_2, \cdots, r_n)$, where $s \in \mathbb{Z}_q$ is the secret to be shared, and $(r_2, \cdots, r_n) \in \mathbb{Z}_q$ are randomly selected numbers, then M_v is the vector of ℓ shares of the secret s according to Π. Note that the share $(M_v)_i$ belongs to party $\rho(i)$.

3 System Model

3.1 Overview of Basic Outsourcing CP-ABE Scheme

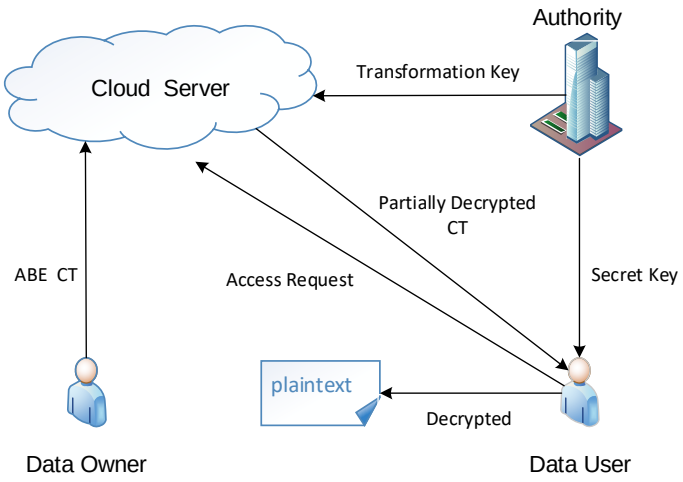


Fig. 1. The framework of ABE with outsourced decryption in cloud computing

In the conventional outsourced CP-ABE scheme [3], there exists a fully trusted authority to assign two types of keys for the data users and cloud server. One is the user's private key, called SK, which can be used to decrypt the ciphertext requested from the central server. Another is the corresponding transformation

key TK used to construct the outsourcing part of ciphertexts. The whole workflow of basic outsourced CP-ABE scheme in the cloud computing scenario is shown in Fig. 1. At first, the data owner encrypts his private data by using ABE scheme and sends the ciphertext to the cloud center for storage. Once receive the access request from the data user whose attributes satisfy the access policy, the cloud server converts the original ABE ciphertext into a simple ElGamal-form ciphertext by using TK. Then, this ElGamal-form ciphertext will be sent to the data user. Note that this ElGamal-form ciphertext is encrypted under the user's private key SK for the same message. At last, the data user can obtain the plaintext by a simple operation based on the secret key SK.

In our work, we mainly focus on the following issues over the basic outsourced CP-ABE scheme in the cloud computing scenario:

- Although conventional outsourced ABE scheme can shift part of decryption operations to the third party, the encryption step still brings the heavy computation costs, which will consume a lot of resources of IoT device and waste the network bandwidth.
- The correctness of the ciphertext which partially decrypted in the cloud computing center cannot be guaranteed. Some cloud servers may become "lazy" to save the overhead of bandwidth and resource, which will return "fake results" to the real ones.
- The access structures embedded in the ciphertext usually contain private information of data owner, that sensitive information could be disclosed by the untrusted data user.

3.2 Our Proposed PPAC Framework

To solve the aforementioned challenge, we take both verification and privacy into consideration and propose a privacy-preserving access control scheme with verifiable and outsourced capabilities in fog-cloud computing, named PPAC. Similar to the outsourced decryption mechanism, the data owner can also outsource the encryption to the fog server and verifies the correctness of the results returned by the fog server. Moreover, the embedded access structure is partially hidden from the data user whose private keys do not satisfy the access policy.

As depicted in Fig.2, the whole access control system consists of five entities: authority, cloud server, fog servers, data owners, and data users.

- **Authority**: The authority is a fully trusted entity whose duty is to bootstrap the whole system and generates key materials associated with access control policies.
- **Cloud**: The cloud is the central server for data storage and sharing. Its duty is to collect all the users' data from fog servers and further make some analytics.
- **Fog servers**: It mainly provides the data users with the computing and storing function on the edge network to reduce the delay and improve the service quality.

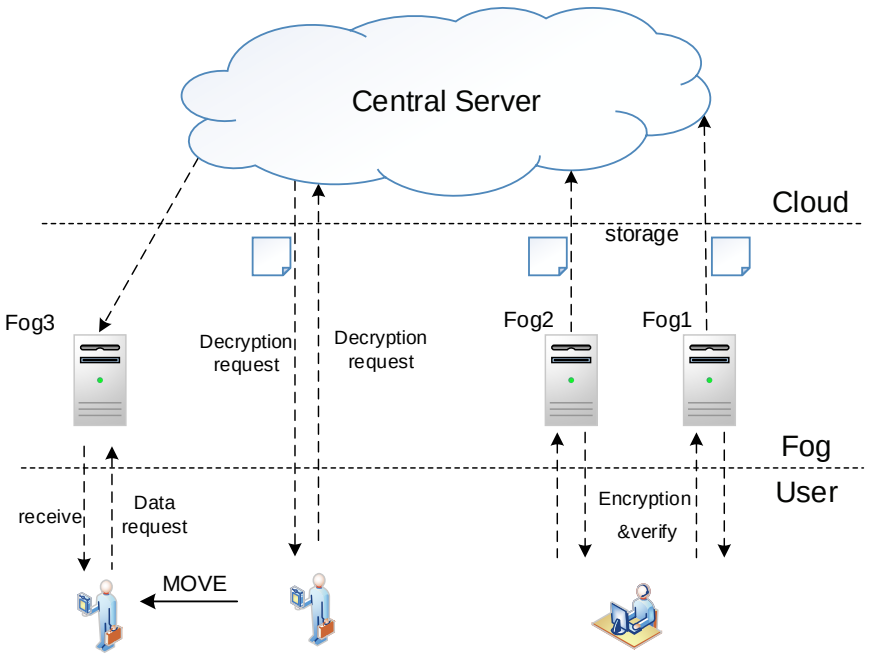


Fig. 2. Overview of proposed privacy-preserving access control system

- **Data owners**: In the fog-cloud model, IoT devices such as smart-phones, wearable devices, smart home appliances and etc., which gather data nearby and uploads them in an encrypted form. However, IoT devices with limited resources can only bear lightweight operations.
- **Data users**: Data users play a role with data consumers who can request private data stored in the cloud. They also cannot accept the heavy computation costs and a long-delayed response due to its resources-constraint.

4 Construction of Proposed PPAC

In this section, we present the concrete PPAC scheme with verifiable and outsourcing capabilities which consists of eight algorithms. Note that the algorithms of Setup, KeyGen, and GenTK.out are performed on a certain trusted authority.

4.1 Algorithms in PPAC Scheme

1. **Setup** $(\lambda, U) \rightarrow (PK, MSK)$: On input a security parameter and an attribute universe description $U = 1, 2....\ell$. It outputs public parameters PK and a master key MSK. The setup algorithm works as follows:

- Chooses a bilinear map $D = (p, \mathbb{G}, \mathbb{G}_T, e)$, where p is the prime order of the groups $\mathbb{G}$ and $\mathbb{G}_T$. Then, it chooses three random generators (g, h, u) and one exponent $\alpha \in \mathbb{Z}_p$.
- Chooses a collision-resistant hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$ and adopts the key derivation function (KDF1) [18], where the length of function output is defined as $\mathcal{L} = |key| + |p|$ and p is the prime order.
- Publishes the public parameters as $PK = \{D, e(g,g)^\alpha, g, h, u, H, \mathcal{L}, KDF_1\}$, and the corresponding master key can be formed as $MSK = (PK, \alpha)$.

2. **KeyGen** $(PK, MSK, S) \rightarrow (SK)$: On input public parameters PK, the master secret key MSK and a set of attributes $S = \{A_1, \cdots, A_n\}$. It outputs a private key SK associated with S. The algorithm works as follows:

- Generates a set of random values $(r, r_1, \cdots, r_k) \in \mathbb{Z}_p$.
- Computes $k_0 = g^\alpha u^r$, and $k_1 = g^r$.
- For all $i \in [1, k]$, calculates $k_{i,2} = g^{r_i}$ and $k_{i,3} = (uh^{A_i})^{r_i}$.
- Then, it sets the private key as $SK = (S, PK, k_0, k_1, \{k_{i,2}, k_{i,3}\}_{i\in[1,k]})$.
- Sends the private key SK to the data owners.

3. **GenTK.out** $(SK) \rightarrow (TK, RK)$: On input a private key SK, it outputs a transfer key TK and a corresponding retrieval key RK. The algorithm works as follows:

- Randomly chooses a number $\tau \in \mathbb{Z}_p^*$.
- Computes $k_0{}' = u^{\alpha/\tau} u^{r/\tau}$, $k_1{}' = g^{r/\tau}$, $k_{i,2}{}' = g^{r_i/\tau}$, and $k_{i,3}{}' = (uh^{A_i})^{r_i/\tau}$. Then, it sets the transfer key as $TK = (S, PK, k_0{}', k_1{}', \{k_{i,2}{}', k_{i,3}{}'\}_{i\in[1,k]})$ and the related retrieval key can be formed as $RK = (TK, \tau)$.
- Sends the retrieval key RK to the data users.

Note that, the aforementioned three algorithms (i.e., Setup, KeyGen, and GenTK.out) are all executed at the authority side.

4. **Encrypt.out1** $(PK, N) \rightarrow (IT_1)$: This algorithm is performed by the *fog1*. On input public parameters PK and maximum bound of N rows in any LSSS access structure, it outputs a intermediate ciphertext IT_1. The algorithm works as follows:

- Chooses three random numbers $(x_i{}', y_i{}', \lambda_i{}') \in \mathbb{Z}_p$, where $i \in [1, N]$ and computes $C_{i,1}{}' = u^{\lambda_i{}'}$ and $C_{i,2}{}' = (uh^{x_i{}'})^{y_i{}'}$.
- Randomly picks $s' \in \mathbb{Z}_p$ and computes $C_0{}' = g^{s'}$. Then, the intermediate ciphertext can be formed as $IT_1 = (s', C_0{}', \{x_i{}', y_i{}', \lambda_i{}', C_{i,1}{}', C_{i,2}{}'\}_{i\in[1,k]})$.
- Sends IT_1 to the data owner for consolidation and creates a replica for the storage in cloud center.

5. **Encrypt.out2** $(PK, N) \rightarrow (IT_2)$: This algorithm is executed by the *fog2*. On input public parameters PK and a maximum bound of N rows in any LSSS access structure, it outputs a intermediate ciphertext IT_2. The algorithm works as follows:

- Picks three random exponent $(x_i'', y_i'', \lambda_i'') \in \mathbb{Z}_p$, where $i \in [1, n]$ and calculates $C_{i,1}'' = u^{\lambda_i''}$ and $C_{i,2}'' = (uh^{x_i''})^{y_i''}$.
- Randomly picks $s'' \in \mathbb{Z}_p$ and computes $C_0'' = g^{s''}$. Then, the intermediate ciphertext is $IT_2 = (s'', C_0'', \{x_i'', y_i'', \lambda_i'', C_{i,1}'', C_{i,2}''\}_{i\in[1,k]})$.
- Returns IT_2 to the data owner for consolidation and send a copy to the cloud center for storage.

6. **Encrypt.owner** $(PK, IT_1, IT_2, A, \rho, \tau) \rightarrow (CT, key)$: It is performed by the data owners. On inputs public parameters PK and two intermediate ciphertexts IT_1, IT_2 and an access structure (A, ρ, τ), where A is an $\lambda \times n$ matrix, ρ is a map from each row A_i of A to an attribute name, and τ is a set consists of $\{\tau_{\rho(1)}, \cdots, \tau_{\rho(\lambda)}\}$ and $\tau_{\rho(i)}$ is the value of attribute $\rho(i)$ calculated from the access policy. At last, the algorithm outputs the ciphertext CT and keeps key locally. The algorithm works as follows:

- Adopts a collision-resistant hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$ and recomputes $s = s' + s''$ and $C_0 = C_0' \cdot C_0'' = g^{s'+s''}$, where $key = e(g,g)^{\alpha s}$.
- Recomputes the parameters $\lambda_i = \lambda_i' + \lambda_i''$, $y_i = y_i' + y_i''$, and $x_i = x_i' + x_i''$. Then, calculates $C_{i,1} = C_{i,1}' \cdot C_{i,1}'' = u^{\lambda_i}$ and $C_{i,2} = C_{i,2}' \cdot C_{i,2}'' = (uh^{x_i})^{y_i}$.
- Randomly chooses $v_2, \cdots, v_n \in \mathbb{Z}_p$, denotes a vector as $V = (s, v_2, \cdots, v_n)^T$ and computes another vector of shares of s as $(b_1, \cdots, b_\lambda) = M \cdot V$. For $i = 1$ to λ, it computes $C_{i,3} = b_i - \lambda_i$, $C_{i,4} = \rho(i)y_i - x_i y_i$, and $C_{i,5} = -y_i$.
- Generates a new key SSK with the key derivation function and computes $KDF1(key, L) = ssk||d$, $\hat{c} = u^{H(ssk)}v^{H(d)}$, where d is the length of SSK. Here, we add a Pedersen commitment $\hat{c}$ to achieve the verification of the outsourced computation.
- Randomly chooses $t' \in \mathbb{Z}_p$ and computes $F = e(g,g)^{\alpha t'}$.
- Picks a set of random values $v_2', \cdots, v_n' \in \mathbb{Z}_p$, denotes a vector $V' = (t', v_2', \cdots, v_n')$ and computes another vector of shares of t' as $b_1', \cdots, b_\lambda' = M \cdot V'$. Then, $C_{i,3}' = b_i' - \lambda_i$. At last, it sets the ciphertext as $CT_1 = (M, \rho, C_0, \hat{c}, \{C_{i,1}, C_{i,2}, C_{i,3}, C_{i,3}', C_{i,4}', C_{i,5}'\}_{i\in[1,\lambda]})$.

Note that each attribute consists of an attribute value and an attribute name. If the attribute set does not satisfy the access structure, the attribute value in the access structure is hidden, while the attribute name can be broadcast. The role of CT_2 is to help the user who satisfies the access structure to decide which attribute set satisfies the access structure and further prevent the cloud from getting too much information during decryption. The scheme can partially hide the access structure to the cloud center and fog server.

7. **Transform.out** $(TK, CT) \rightarrow CT'$: This algorithm is performed by the cloud server. On inputs a conversion key TK and the ciphertext CT, it outputs a transformed ciphertext CT'. We define L to be the smallest set of subsets that satisfy the access structure (M, ρ). When received a ciphertext CT and a conversion key TK for attribute sets S from the data users, the cloud server first checks whether the user's attributes meet the smallest set of subsets L. The algorithm works as follows:

– Checks whether there exists an $I \in L$ that satisfies:

$$F = \frac{e(C_0, k_0{}')}{e(u^{\sum_{i\in I} C'_{i,3} w_i}, k_1{}') \cdot \prod_{i\in I}(e(C_{i,1}, k_1{}')e(C_{i,2}h^{C_{i,4}}, k_1{}')e(g^{C_{i,5}}, k_{i,3}))^{w_i}} \tag{1}$$

where $\sum_{i\in I} w_i M_i = (1, 0, \cdots, 0)$. If there is no member of L that satisfies this equation, it outputs $\perp$. Otherwise, we calculate the encapsulated key as:

$$\begin{aligned} &\frac{e(C_0, k_0{}')}{e(u^{\sum_{i\in I} C_{i,3} w_i}, k_1{}') \cdot \prod_{i\in I}(e(C_{i,1}, k_1{}')e(C_{i,2}h^{C_{i,4}}, k_1{}')e(g^{C_{i,5}}, k_{i,3}))^{w_i}} \\ &= \frac{e(g^s, g^{\alpha/\tau})e(g^s, u^{r/\tau})}{e(u^{\sum_{i\in I}(b_i-\lambda_i)w_i}, g^{r/\tau})e(u^{\lambda_i}, g^{r/\tau})^{w_i}} \\ &= \frac{e(g^s, g^{\alpha/\tau})e(g^s, u^{r/\tau})}{e(u^s, g^{r/\tau})} = e(g, g)^{\alpha s/\tau} \end{aligned} \tag{2}$$

– It sets the transformed ciphertext as $CT' = (e(g, g)^{\alpha s/\tau}, T' = \hat{c}, M, \rho)$ and sends the CT' to the data users.

8. **Decrypt.user** $(RK, CT, CT') \to m$: This algorithm is executed by the data users. On input a retrieval key RK, a ciphertext CT and a transformed ciphertext CT', it outputs $\perp$ when $T' \neq \hat{c}$. Otherwise, it outputs $key = (key')^{\tau} = e(g, g)^{\alpha s}$. In this situation, it is necessary to check the correctness of the decryption results returned by the cloud server. We take the method of adding a verification message(Pedersen commitment $\hat{c}$) in the encryption process. When the data user requests data from the cloud server, if the attribute set meets the access structure, the cloud will send ciphertext to the data user, because the cloud does not need to do any calculation in this process, so there is no reason for cloud server sends a false ciphertext. Then, the data user requests the cloud server for the partially decrypt ciphertext. The Pedersen commitment $\hat{c}$ of the decrypted ciphertext is compared with the verification information T' in the original ciphertext by the user. If $T' = \hat{c}$, it computes $m = \frac{m \cdot e(g,g)^{\alpha s}}{e(g,g)^{\alpha s}}$.

4.2 Correctness of Batch Verification for Outsourced Encryption

Due to the complexity of fog computing, we consider a scenario where a fog server might perform only a portion of the computation and return incorrect results to save computing resources. According to [19], we can use a naive way to verify the modular exponentiation in some group, i,e,m given (g, x, y), check that whether $g^x = y$ is correct or not. In our work, there exists two types of ciphertexts: the ciphertext with the same one generator $(C_0{}' = g^{s'}, C_0{}'' = g^{s''}, C_{i,1}{}' = u^{\lambda_i{}'}, C_{i,1}{}'' = u^{\lambda_i{}''})$ and ciphertext with two different generators $(C_{i,2}{}' = (uh^{x_i{}'})^{y_i{}'}, C_{i,2}{}'' = (uh^{x_i{}''})^{y_i{}''})$. Thus, we give the correctness of two-types of ciphertexts as follows:

[**Type-1**]:

- Given: Let $P = |\mathbb{G}|$ be the order of the group $\mathbb{G}$ and g is a primitive element of $\mathbb{G}$. Defining $\{(x_1, y_1), \cdots, (x_n, y_n)\}$ with $x_i \in \mathbb{Z}_p$ and $y_i \in \mathbb{Z}_p$, as well as a security parameter k.
- Check: $\forall i \in [1, \cdots, n]$, $y_i = g^{x_i}$.
- Picks a random subset $S = (b_1, \cdots, b_n) \in \{0, 1\}^k$ and computes $x = \sum_{i=1}^{n} x_i b_i$ mod p, $y = \prod_{i=1}^{n} y^{b_i}$. If $g^x = y$, returns accept, else reject.

[**Type-2**]:

- Given: Let $P = |\mathbb{G}|$ be the order of the group $\mathbb{G}$ and g is a primitive element of $\mathbb{G}$. Defining $\{(x_1, y_1, z_1), \cdots, (x_n, y_n, z_n)\}$ with $(x_i, y_i) \in \mathbb{Z}_p$ and $z_i \in \mathbb{G}$, as well as a security parameter k.
- Check: $\forall i \in [1, \cdots, n]$, $z_i = u^{x_i} h^{y_i}$.
- Picks a random subset $S = (b_1, \cdots, b_n) \in \{0, 1\}^k$ and computes $x = \sum_{i=1}^{n} x_i b_i$ mod p, $y = \prod_{i=1}^{n} y^{b_i}$, and $z = \prod_{i=1}^{n} z_i b_i$. If $u^x h^y = z$, returns accept, else reject.

4.3 Security Analysis

Here, we give a brief discussion about the security properties of our PPAC scheme. Note that the cloud and the fog servers are assumed to be semi-trusted in our security model. In particular, we describe the security problems of the scheme from the following aspects.

Data confidentiality: First of all, we outsource the encryption to two fog servers, both of which are run by different operators. Note that there's no reason for these two fog servers to collude with each other and reveal the user's private key information. Besides, the cloud center cannot get any plaintext information about the user during the partial decryption process. Finally, when the data is requested, the data user cannot get any useless information about the plaintexts unless the attribute set satisfies the access structure.

Data validation: For the outsource verification, we adopt the batch verification mechanism to check the correctness of encryption result, which has been widely used in cryptographic to verify modular exponentiation operations. In general, the expected error of this method is within our acceptable range. For the outsourced decryption validation, the key derived function KDF and the anti-collision hash function have been used to hide the final decryption results in the ciphertexts, which can protect the privacy of original data. For the user side decryption phases, the data user can obtain the ciphertexts from the cloud center only if the user's attribute set satisfies the access structure. In this way, the user can get the verification information contained in the correct ciphertext to verify the results and further ensure the correctness of outsourcing decryption.

Privacy protection: In our proposed PPAC scheme, the attribute values of the access structure are hidden from the cloud and fog servers while only the attribute names of the access structure are public. Besides, when the cloud checks whether the attribute set of the data user satisfies the access structure, the data user is also unable to know the attribute value in this process, which means

the data owner's private information embedded in the access structures can be protected.

5 Performance Evaluation

5.1 Theoretical Analysis

Computation Cost Comparison : The computation cost of our scheme depend on the number of modular exponentiation and pairing operations. In Table 1, we compare the efficiency of our scheme with the original ABE [5] and the other two outsourced ABE schemes [13] and [16]. Note that, E_p and P denote the modular exponentiation and pairing operation, respectively. ω denotes the attribute set and λ represents the number of rows of the matrix M for LSSS. As shown in Table 1, our scheme only leaves three exponentiations and a pairing computation(for hiding access policy) during the encryption phase. The computation cost of our schemes is less than [5] and [13]. The ABE technique [16] has only three exponentiations, but the outsource server in [16] know the access structure which may contain sensitive information. Thus, our scheme still behaves better. During the decryption phase, the computational costs for the data user are three exponentiations which less than ABE schemes in [5] and [13]. Therefore, our scheme requires less computation costs compared with above schemes.

Table 1. Comparison of computation costs

Schemes	KeyGen	Enc.user	Dec.user	Enc.verify	Dec.verify	Hidding Policy
[3]	$2\|\omega\|E_p$	$2\|\lambda\|(E_p+P)$	$2\|\lambda\|(E_p+P)$	×	×	×
[4]	$2\|\omega\|E_p$	$(2+5\|\lambda\|)E_p$	E_p	×	✓	×
[19]	$2\|\omega\|E_p$	$3E_p$	$3E_p$	✓	✓	×
Our	$2\|\omega\|E_p$	$3E_p+P$	$3E_p$	✓	✓	✓

5.2 Experimental Evaluations

Here, we estimate the performance of our proposed PPAC scheme and compare the results with Green's work [13]. We implement the experiment by using pairing-based cryptography (PBC) library for a variety of cryptographic operations and Charm framework [20] for attribute-based encryption method. To achieve a high-security level, we choose the 224-bit MNT elliptic curve from the PBC library. The experiments are conducted on Intel(R) Core(TM) i5-8500 3.00GHz CPU with Ubuntu 12.04.1 LTS 64-bit enviroment and using Python 3.4 as programming language.

In order to evaluate the influence of access policies on performance, we set 30 different policies in the form of $(a_1, \cdots, a_n)$, where a_i represents the attributes

and $1 \leqq n \leqq 100$. As shown in Fig. 3, the running time for outsourced key generation is about 100-856 ms. Furthermore, from Fig. 4(a) and Fig. 4(b), we can see that the time for outsourced encryption and decryption are increased close to linearly with the growth of policy attributes. In Fig. 5(a) and Fig. 5(b), we further compared the complexity of encryption and decryption on user side. The results demonstrate that the user side computation time of our proposed PPAC scheme is a little higher than that of [13]. The reason for this situation is that the verification function will bring some extra computation cost.

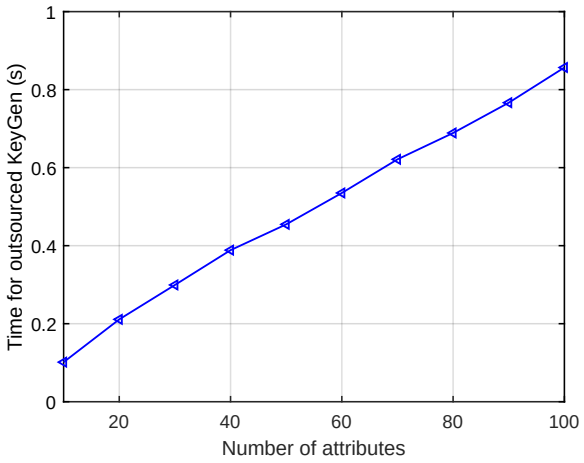
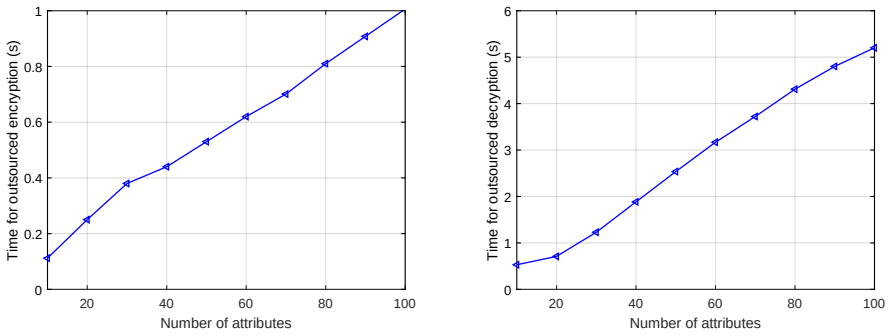
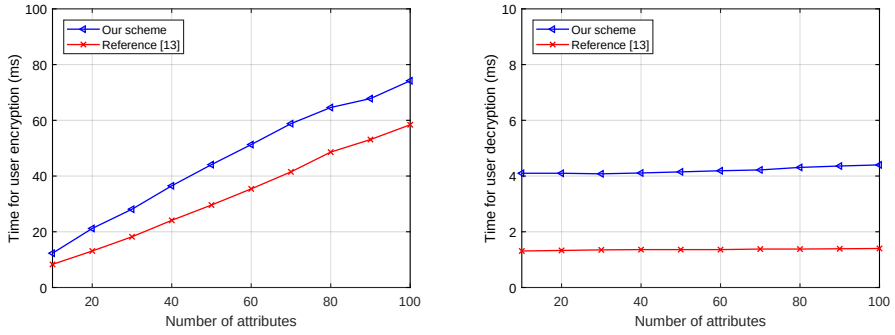


Fig. 3. Time for outsourced key generation



(a) (encryption) (b) (decryption)

Fig. 4. Time for outsourced encryption and decryption



(a) (1 attacker) (b) (3 attackers)

Fig. 5. Encryption and decryption time on user

6 Conclusion

Outsourced attributed-based encryption has performed its advantages on security data sharing in cloud computing. In this paper, we proposed a novel CP-ABE scheme in fog-cloud computing scenario, named PPAC, which contains the capabilities of verifiable outsourced encryption, decryption, and partially hidden access structure. The security analytics demonstrate that our presented PPAC scheme can achieve data confidentiality, data validation, and privacy protection simultaneously. Experimental results illustrate that PPAC can be effectively applied to the resource-constraint model devices. For the future, we will focus on integrating privacy-preserving techniques to machine learning services in the fog-cloud computing environment.

References

1. W Shi, J Cao, Q Zhang, Y Li, and L Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637-646. (2016)
2. K. Zhang, Y. Mao, S. Leng, Y, He, and Y. Zhang, "Mobile-Edge Computing for Vehicular Networks: A Promising Network Paradigm with Predictive Off-Loading," *IEEE Vehicular Technology Magazine*, vol. 12, no. 2, pp. 36-44. (2017)
3. F. Jalali, K. Hinton, R. Ayre, T. Alpcan, and R. S. Tucker, "Fog Computing May Help to Save Energy in Cloud Computing," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 5, pp. 1728-1739. (2016)
4. J. Zhang, B. Chen, Y. Zhao, X. Cheng and F. Hu, "Data Security and Privacy-Preserving in Edge Computing Paradigm: Survey and Open Issues," *IEEE Access*, vol. 6, pp. 18209-18237. (2018)
5. A. Sahai, B. Waters, "Fuzzy Identity-Based Encryption," in *Proc. Adv. Cryptol (CRYPTO 2005)*, pp. 457-473. (2005)

6. A. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters, "Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption," in *Proc. Adv. Cryptol*, pp. 6291. (2010)
7. J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symp. Security Privacy (SP 2007)*, pp. 321334. (2007)
8. K. Yang, X. Jia, "An Efficient and Secure Dynamic Auditing Protocol for Data Storage in Cloud Computing," in *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, pp. 1717-1726. (2013)
9. Z. Zhou, D. Huang, Z. Wang "Efficient Privacy-Preserving Ciphertext-Policy Attribute Based-Encryption and Broadcast Encryption," in *IEEE Transactions on Computers*, vol. 64, pp. 126-138. (2015)
10. S. Hohenberger and B. Waters, "Online/offline Attribute-based encryption," in *Proc. Public-Key Cryptography (PKC 2014)*, pp. 293310. (2014)
11. J. Zhou, Z. Cao, X. Dong, and X. Lin, "Tr-mabe: White-box traceable and revocable multi-authority attribute-based encryption and its applications to multi-level privacy-preserving e-healthcare cloud computing systems," in *IEEE Conference on Computer Communications (INFOCOM 2015)*, pp. 23982406. (2015)
12. Y. Rouselakis, B. Waters "Practical constructions and new proof methods for large universe attribute-based encryption," in *ACM SIGSAC Conference on Computer and Communications Security (CCS 2013)*, pp. 463474. (2013)
13. M. Green, S. Hohenberger, B. Waters, "Outsourcing the Decryption of ABE Ciphertexts," in *Proc. USENIX Security Symp (USENIX Security 2011)*. (2011)
14. J. Alderman, C. Janson, C. Cid, and J. Crampton, "Access control in publicly verifiable outsourced computation," in *10th ACM Symposium on Information, Computer and Communications Security (ASIACCS 2015)*, pp. 657-662. (2015)
15. J. Li, X. Huang, J. Li, X. Chen, and Y. Xiang, "Securely outsourcing attribute-based encryption with checkability," in *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 8, pp. 2201-2210. (2014)
16. H. Ma, R. Zhang, Z. Wan, Y. Lu, and S. Lin, "Verifiable and exculpable outsourced attribute-based encryption for access control in cloud computing," in *EEE Transactions on Dependable and Secure Computing*, vol. 14, no. 6, pp. 679-692. (2017)
17. Kaiping Xue, Jianan Hong, Yongjin Ma, David S. L. Wei, Peilin Hong, and Nenghai Yu, "Fog-Aided Verifable Privacy Preserving Access Control for Latency-Sensitive Data Sharing in Vehicular Cloud Computing," in *IEEE Network*, Volume: 32, pp. 7-13. (2018)
18. H. Krawczyk, "Cryptographic extraction and key derivation: The hkdf scheme," in *Proc. Adv. Cryptol (CRYPTO 2010)*, pp. 631-648. (2010)
19. M. Bellare, J. A. Garay, and T. Rabin, "Fast batch verification for modular exponentiation and digital signatures," in *Proc. Adv. Cryptol (CRYPTO 2007)*, pp. 74-90. (2007)
20. J. A. Akinyele, C. Garman, I. Miers, M. W. Pagano, M. Rushanan, M. Green, and A. D. Rubin, "Charm: A framework for rapidly prototyping cryptosystems," in *J. Cryptographic Eng.*, vol. 3, no. 2, pp. 111-128. (2013)