


Article

An Improved Ciphertext-Policy Attribute-Based Encryption Scheme in Power Cloud Access Control

Yuancheng Li ^{1,*}, Pan Zhang ^{1,2} and Boyan Wang ¹ 

¹ School of Control and Computer Engineering, North China Electric Power University, Beijing 102206, China; zhangpan-23@126.com (P.Z.); 13261200907@163.com (B.W.)

² State Grid Information & Telecommunication Branch, Beijing 100761, China

* Correspondence: yuancheng@ncepu.edu.cn

Received: 13 September 2018; Accepted: 29 September 2018; Published: 6 October 2018



Abstract: In power cloud environment, the existing Ciphertext-Policy Attribute-Based Encryption (CP-ABE) access control schemes, do not consider the generation of access structure and the existence of malicious users. To tackle these problems, a power cloud access control (PCAC) scheme is proposed, which improves the traditional CP-ABE access control model. Considering the heavy time consumption of CP-ABE, PCAC encrypts the symmetric key, instead of the raw data. PCAC combines the access tree and linear secret-sharing scheme (LSSS) to achieve the automatic generation and efficient operation of access structure. Additionally, an action audit phase, based on zero-knowledge verification was designed to defend against malicious users. The experiments proved that PCAC meets the requirement of fine-grained access control, in a power cloud. Compared with existing CP-ABE schemes, the PCAC scheme reduced about half of the time consumption, in the action audit phase and costs about one-third the time, in the data obtainment stage.

Keywords: power cloud; access control; CP-ABE; action audit

1. Introduction

With the fast development of cloud computing, it is economic for individuals and companies to save their data in cloud. However, the safety of outsourced data, in cloud storage, has always been a big issue. Power cloud contains a large number of users with complex system structure and different data security requirements, which needs an automatic, fine-gained access control scheme. National Institute of Standards and Technology (NIST) specifies the role-based access control (RBAC) requirements for a smart grid access control [1]. In this study, we focus on the attribute-based access control (ABAC), which is more fine-gained than RBAC.

CP-ABE is an ABAC scheme and was first proposed by Beyhencourt [2], who defined the basic algorithms. In the model of CP-ABE, the ciphertext corresponds to access structure, while the key corresponds to the attribute set. If, and only if, the user's attribute set satisfies the access structure, the user can decrypt the data. This scheme can meet the fine-grained access control requirement of power clouds, but it is difficult to meet the security requirement. The recent research on CP-ABE focuses on the improvement of access structures [3–10] and the generation and distribution of the secret key [11–19]. Li [3] combined the access tree with an Ordered Binary Decision Diagram, and employed a Boolean function in the leaf node, to finish the generation of the secret key and the attribute-based encryption, with less time complexity, which reduced the computing cost. Wang [4] proposed a hierarchical access structure which was based on access trees, in order to save the storage space of the access tree. However, this scheme only achieved the deduplication of access structures, rather than improve the structure of access tree. Xue [5] solved the problem of a single-point performance bottleneck and the security problem of multi-agency access control. In their scheme, the verification

of the attribute set, and the access structure was shared by multiple semi-trusted third-part attribute authorities. They refined the steps of the generation and distribution of the secret key, confirmed, and traced malicious users by monitoring the key generation and the distribution process. Wang [6] improved the key generation phase and proposed a new scheme in which he changed the traditional single authority model. In this model he introduced the multi-agency idea where different agencies managed different trapdoors. Users could obtain the shared secret key from multiple agencies, so it can protect the user's identifier and attribute set, to achieve the purpose of privacy protection. Alrawais [7] applied a subset difference algorithm in the environment of fog computing, which is similar to cloud computing, in order to update the access tree after the attribute is revoked. Xue [8] proposed the generation and management of the sub-attributes with 0-encoding and 1-encoding, in order to achieve the attribute comparison. Zhang [9] proposed a secure scheme which achieve revocation and traceability requirement with the use of a subset cover algorithm. Zhou [10] proposed a Privacy Preserving Constant CP-ABE (PPC-CP-ABE) scheme that reduces the ciphertext to a constant size with any given number of attributes. PPCCP-ABE applies a hidden policy construction so that the recipients' privacy is preserved efficiently.

Many improvements have been made in key generation and distribution processes [11–19]. Yan [11] proposed a scheme based on trust evaluated by the data owner or reputations generated by a number of reputation centers through the CP-ABE and Proxy Re-Encryption. Waters [12] and Balu [19] proposed a new way using the linear secret-sharing scheme (LSSS). They described a specific access structure by constructing an LSSS matrix from which the secret key was generated. Only when all the attribute sets satisfy the requirement of the trap doors, the secret key can be generated correctly. Calculations of this scheme is simplified with the matrix operation. Han [13] proposed a privacy-preserving decentralized CP-ABE (PPDCP-ABE) to reduce the trust on the central authority and protect the users' privacy. Each authority can work independently without any collaboration to initial the system and issue secret keys to users. Users can obtain secret keys from multiple authorities without them knowing anything about himself/herself. Since CP-ABE uses a long-key and computational time for the encryption, and the decryption algorithm increases with the grow of attributes number, Guo [14] made improvements in the encryption algorithm. The algorithm they designed is independent of the number of attributes and the minimum secret key length. It is shortened to 672 bits, which greatly reduced the time for decryption. Lin [15] proposed a collaborative key management protocol in CP-ABE to realize the distributed generation, issue, and storage of private keys, without adding any extra infrastructure. Experiments prove that Lin's scheme has better performance in mobile devices. Ning [16] proposed an auditable σ -time outsourced CP-ABE, which provides σ -time fine-grained access control and audits the correctness of the operation. Liu [17] proposed a fine-grained two-factor authentication (2FA) access control system for web-based cloud computing services. The cloud server has no idea on the exact identity of the user, to protect the user's privacy. Li [18] proposed a new scheme called outsourcing attribute-based encryption with keywords search function(KSF-OABE). Cloud service privader perform search encrypted keywords instead of the keywords embedded in trapdoor.

However, the existing CP-ABE schemes have not considered the security issue of the data source and the generation of the access structure. On one hand, the data sources in power cloud are not only collectors but also sensors which cannot judge who has access to the data, so there should be an administrator to generate the access structure. If an agent is responsible for generating the access structure, it is difficult to adapt to the high-frequency data storage, and the administrator may make some mistakes. If an entity undertakes the task of a human administrator, some security issues may be discovered. On the other hand, there may be malicious users on both data source-side and user-side, who could make the power cloud create a wrong access structure, by forging or tampering identification. In order to meet the high-frequency data storage requirements of the power cloud and prevent possible attacks, the access control scheme must contain both the data collection stage and the data obtainment stage.

This paper proposes a Power Cloud Access Control (PCAC) scheme. PCAC improves the traditional CP-ABE system model, including outsourcing the generation of access structures and the verification of attribute sets to a third party, to reduce the computational complexity. Considering the heavy computation overhead in an access tree, PCAC generates an access structure with the access tree, and shares a secret key with the LSSS. In addition, our scheme achieves the action auditing of users and data sources. In previous research studies, private keys did not work for the decryption process, therefore, authentication is a must to protect data saved in the power cloud. The user's and the data source's action is audited with zero-knowledge proofs, which prevents malicious users from attacking the system from both the source side and the user side. The contribution of this paper can be summarized as follows:

- (1) A power cloud access control scheme based on a CP-ABE has been proposed. Considering the generation of the access structure and the heavy computation required for an access tree, PCAC achieves the automatic generation of access structures and effective sharing of the secret key.
- (2) In order to resist possible attacks that could come from the data source side and the user side, an action audit phase has been designed, in which the user's identity is verified without obtaining the user's private key.
- (3) The data confidentiality and the operation efficiency of our PCAC scheme have been analyzed with theory and experiments. The experiments show that our scheme satisfies the requirement of fine-grained access control.

The rest of this paper are organized as follow: In Section 2, some definitions in CP-ABE are presented. In Section 3, we build a system model of the program and make some security assumptions about the agencies in the model. In Section 4, the phases designed in the paper are described in detail. In Section 5, the security analysis and efficiency analysis of our PCAC scheme are given with theory. In Section 6, our scheme is compared with Waters' scheme [12] and the Robust and Auditable Access Control (RAAC) [5] scheme to test the time consumption and storage consumption of PCAC. Finally, in Section 7, we summarize the work of this paper.

2. Preliminaries and Definitions

2.1. Ciphertext-Policy Attribute-Based Encryption (CP-ABE)

CP-ABE, first proposed by Beyhencourt, is an access control scheme widely used in cloud storage. It mainly contains four algorithms:

Setup \rightarrow (PK, MSK): This algorithm is run by the authority in the system, to generate the system public key (PK) and the master secret key (MSK), with the system parameters.

Encrypt(M, T, PK) \rightarrow CT: This algorithm is executed by the owner of the data. The message M is encrypted by the system public key PK and the access structure T.

Key_Gen(MSK, S) \rightarrow SK: This algorithm is performed by the authority, to generate the secret key SK through the user's attribute set S and the master secret key MSK.

Decrypt(CT, SK) \rightarrow M': This algorithm is executed by the user, to decrypt the ciphertext CT by the secret key SK, and recover the message M'.

2.2. Attributes and Access Structures

Our access structure is similar to Waters [12]. Let $P = \{P_1, P_2, \dots, P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1, P_2, \dots, P_n\}}$ 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 $\{P_1, P_2, \dots, P_n\}$, i.e., $\mathbb{A} \subseteq 2^{\{P_1, P_2, \dots, P_n\}} \setminus \{\emptyset\}$. The sets in \mathbb{A} are called the authorized sets, and the sets not in \mathbb{A} are called the unauthorized sets.

In our PCAC scheme, the role of the parties is taken by the attributes. Different from most access control scheme, the access structure adopted in this paper is composed of two parts, one is the static general

access structure, denoted as T_1 , and the other is the dynamic restrictive access structure, denoted as T_2 , while the final access structure is $T = T_1$ and T_2 . A detailed description will be given in Section 4.

2.3. Bilinear Pairing

Let \mathbb{G} and \mathbb{G}_T be two multiplicative cyclic groups of prime order p . let g be a generator of \mathbb{G} and e be a bilinear map, $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$. The bilinear map e has the follow properties:

Bilinearity: for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$, we have $e(u, v)^{ab} = e(u, v)^{ab}$.

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

Computability: to compute the pairing e is efficient.

2.4. Linear Secret-Sharing Schemes

A secret-sharing scheme Π over a set of parties P is called linear (over \mathbb{Z}_p) if:

- (1) The shares for each party form a vector over \mathbb{Z}_p .
- (2) There exists a matrix M with l rows and n columns called the sharing-generating matrix for Π . For all $i = 1, 2, \dots, l$, the i 'th row of M , we let the function ρ define the party labeling row i as $\rho(i)$. When we consider the column vector $v = (s, r_2, \dots, r_n)$, where $s \in \mathbb{Z}_p$ is the secret to be shared, and $r_2, \dots, r_n \in \mathbb{Z}_p$ are randomly chosen, then $M \cdot v$ is the vector of l shares of the secret s according to Π . The share $(M \cdot v)_i$ belongs to party $\rho(i)$.

Every linear secret-sharing scheme based on the above definition enjoys the linear reconstruction property, defined as follow: Suppose that Π is an LSSS for access structure \mathbb{A} . Let $S \in \mathbb{A}$ be any authorized set, and let $i \subset \{1, 2, \dots, l\}$ be defined as $I = \{i : \rho(i) \in S\}$. Then, there exist constants $\{w_i \in \mathbb{Z}_p\}_{i \in I}$, if $\{\lambda_i\}$ are valid shares of any secret s according to Π , then $\sum_{i \in I} w_i \lambda_i = s$. These constants $\{w_i\}$ can be found in a time polynomial, in the size of the share-generating matrix M .

2.5. Access Tree Structure

The access tree is used to describe an access structure. Each leaf node in the tree represents an attribute. Each non-leaf node represents a relational function. The relational function can be AND (n of n), OR (1 of n), n of m ($m > n$) threshold, and so on. As can be seen in Figure 1.

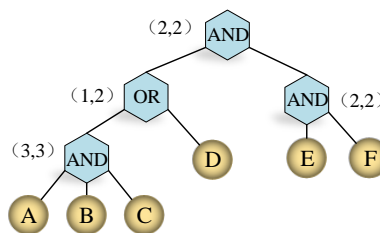


Figure 1. Access Tree Structure.

Our PCAC scheme achieves secret-sharing about “AND” door and “OR” door, where “AND” door is expressed as (n, n) , and “OR” door is expressed as $(1, n)$. Our scheme needs to achieve the transformation from the access tree structure to the LSSS sharing-generating matrix, and this step will be given in Section 4.

3. Our Proposed Scheme Model

3.1. System Model

Our PCAC scheme builds the system model which is shown in Figure 2. The model contains a total of five agencies: Power Cloud (PC), Central Key Authority (CKA), Key Share (KS), Data Source (DS), and User (U).

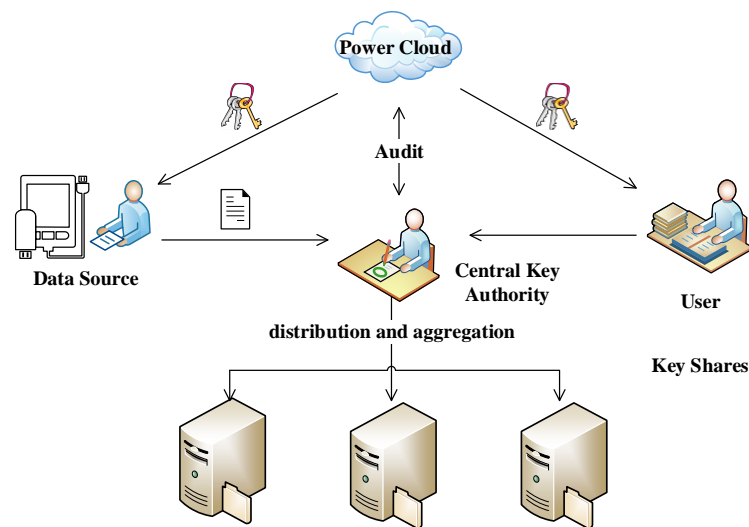


Figure 2. Power Cloud Access Control (PCAC) System Model.

PC: PC provides storage service to DSs and Us. A great deal of raw data is saved in the PC which needs fine-grained access control, like power transmission data, electricity data, account information, etc. PC is also an authority of the system. However, PC is not fully trusted, because there is a risk of being attacked. Therefore, the PC stores the ciphertext C_T of all collected data. PC also needs to save some of the ciphertext of the symmetric key k , which is distributed by CKA. PC is responsible for managing and coordinating the work of all agencies. When the system is initializing, the PC selects the system parameters k' . k' is used to generate the attribute sets S and the pair of public key and private key (Pk, Sk) , for each agency. The encrypt and decrypt algorithms are also constructed by these parameters. When the secret key SK is generated, PC needs to audit the user's action and prevent our system from giving access to malicious users.

CKA: CKA is responsible for the distribution and coordination of KSs' tasks. CKA shares the computation overhead of PC. CKA also stores the users' attribute sets and DSs' static general access structure which has been generated when PC initializes the system. When uploading the data, CKA is responsible for generating the LSSS sharing matrix M_{LSSS} through the access tree, and distributing the ciphertext CT to KSs, according to some policies. The ciphertext is part of the secret key SK and it will work in a decrypt algorithm, which is executed by U. During the user's action audit phase, CKA needs to interact with the user and send the result to the PC for audit. If the audit is successful, CKA will generate secret key SK and send it to the user.

KS: KS is responsible for storing a part of the ciphertext CT assigned by CKA, in order to prevent PC from being attacked and revealing all CT. For an access structure, there are multiple KSs involved in key sharing. Every KS does not know the other KSs who take part in the same access structure, and the data the KS saved does not contain other KSs' information. U cannot communicate with KSs directly. Only CKA is accessible to the data saved in KSs.

U: U is user of power cloud service, and U can't upload the data to PC. Each U has his/her own attribute set, and CKA will determine whether the user has the file access with it. When a U enters the system, the PC distributes a specific public-private key pair (Pk_U, Sk_U) to U, for identity verification, key generation, and file decryption. The request for data requires to be verified twice (attribute set verification led by CKA and action auditing performed by the PC). If U does not pass these verifications at the same time, U will not be able to gain access to the file.

DS: DSs are collectors of data. In the power cloud, the DSs may be smart meters, cameras, and other electronic equipment or some collectors. All the data C_T saved in PC comes from DSs, but DSs can't download data from the PC. The PC distributes an identity DS_{id} to each DS when the system is initialized. DS's action should be verified to prevent any attack which comes from the DS's side.

3.2. Security Assumption

In our PCAC scheme, the following assumptions are built: The network in a smart grid is a local area network with perfect security architecture, so the communication network is safe. PC, CKA, and KSs are always online after the system is initialized, and the determined system parameters do not change while the system is running. PC is the administrator in the system, managing and supervising the operation of the system. All the data stored in the PC comes from the DS, and all Us can obtain the ciphertext of the data in the way defined by the system. Our scheme assumes that CKA is “honest-and-curious”, which means that CKA will properly execute the task assigned to it, by following the rules. But CKA may be tricked by malicious users from either DS or U, so it is not fully trusted. Therefore, the PC is required to audit the action of DS and U to verify the identification of DS and U. Our PCAC scheme assumes that DS can only encrypt the data with a symmetric encryption algorithm, the data itself is secure, and there is no security risk. During the generation and validation of access structures, CKA is the distributor and collector of tasks and does not modify the data. Therefore, CKA can correctly reflect the operation of U and DS.

In PCAC model, there is a risk that a malicious user may attack while DSs upload data or Us request data. When DS uploads data, DS should send its public key to show its identification. However, a large number of smart meters are outdoor, and lack strong supervision. These meters' parameters may be modified with other DS's public key which is open to every agency. As a result, CKA will generate incorrect access structure, and the data has a risk of being revealed. Therefore, the attacker can download the data with his attribute sets. On the other hand, there also exists risk on the user side. When U requests data, he is asked to send his public key to show his identification. However, an attacker may send another user's public key to forge identification and get access to the data.

4. Our Proposed PCAC Scheme

4.1. Overview

Our paper proposes an automatic, fast, and effective access control model both in the data collection stage and the data obtainment stage. The flow chart of access control is shown in Figure 3. DS encrypts the collected data with symmetric encryption algorithm, and sends the ciphertext to PC for storage, while the symmetric key is sent to CKA for encryption in the encrypt phase. Since attribute-based encryption consumes a large amount of time, this encrypt phase reduces the computational burden by encrypting the symmetric key instead of the full file. CKA will distinguish each DS by the proof of identification, and then construct an interaction to generate evidence for an action audit, which will be executed by the PC. After the audit, CKA will generate the corresponding access structure, and exchange the access structure into the LSSS-shared matrix for an easy operation. When passing the audit, CKA will generate the DS's access structure and transform the access structure into the sharing-generating matrix, which is easy to calculate. After that, CKA will execute the encrypt algorithm and distribute the ciphertext to KSs and PC. This marks the end of the data collection stage. If U wants to get the data saved in PC, U needs to send his/her own identification proof to CKA, and CKA will find the attribute set of U and verify it with access structure. After passing the verification, CKA will construct an interaction to generate evidence for action audit, which is like the data collection stage. After the audit is passed, CKA will communicate with the KSs and the PC to obtain the secret key and send it to U. The ciphertext of the data saved in PC is accessible to every U. This marks the end of the data obtainment stage.

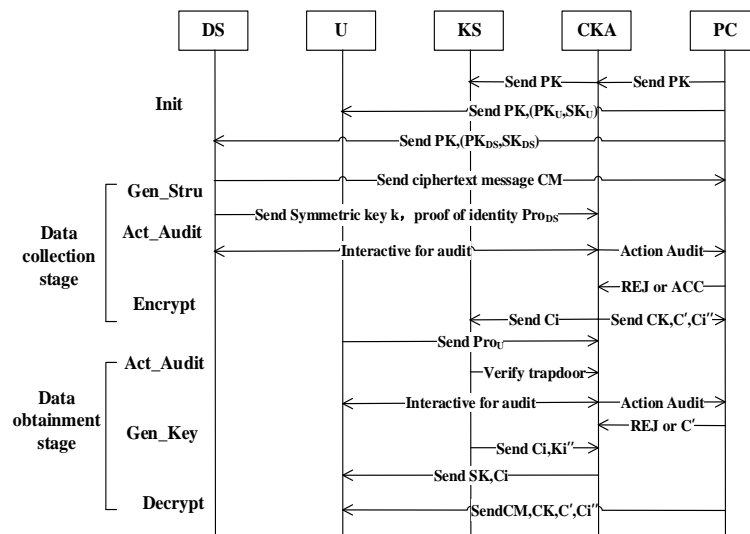


Figure 3. Flow Chart of Access Control.

4.2. Detail Procedures in PCAC

4.2.1. Init Phase

Init: The initialization phase of the system is executed by PC, where the public key PK and the master secret key MSK are obtained through the system parameter k' . Firstly, the PC determines a large prime p by the system parameter k' . Then, it chooses two linear groups G_0 and G_T , with prime order p and g as the generator of the linear group G_0 . Then, a pairing is chosen, $\hat{e} : G_0 \times G_0 \rightarrow G_T$, and a collection of attributes is defined, $S = \{s_1, s_2, \dots, s_m\} \in Z_p$, where all attributes existing in the system are contained by S . A unique public key is chosen $\{Sn_1, Sn_2, \dots, Sn_n\} \in G_0$ for each KS, which is used in secret-sharing. Finally, the PC chooses $\alpha, \beta \in Z_p$, randomly, to generate the PK and MSK. PK is:

$$PK = \{G_T, G_0, g, g^\beta, e(g, g)^\alpha, S, Sn_1, Sn_2, \dots, Sn_n\}$$

And MSK is:

$$MSK = \{\alpha, \beta\}$$

All agencies in the system can get the PK, but the MSK needs to be kept confidential to the other agencies in the system.

In the initialization phase, in order to distinguish between the different DSs, the PC needs to send a pair of public key and private key (Pk_{DS}, Sk_{DS}) to each DS. PC randomly chooses $DS_{id} \in Z_p$ as the identification of DS, and $Pk_{DS} = g^{DS_{id}}, Sk = DS_{id}$. All agencies in the PCAC model can achieve the DS's public key, and DS's private key is only owned by the DS itself, which is used to verify the DS's identity. Like DS, for each U, the PC generates a pair of public key and private key (Pk_U, Sk_U) , and $Pk_U = g^{U_{id}}, Sk = U_{id}$, where $U_{id} \in Z_p$ is randomly chosen. In general, during initialization phase, all DSs and Us can obtain a pair of public key and private key (Pk, Sk) from the PC, and the proof of identification of every DS and U is $Pro = g^{id}$.

4.2.2. Data Collection Stage

Data collection stage shows the flow when the DS uploads the collected data to the PC. There are three phases. First of all, DS,s identity should be audited by the PC. And then, the CKA generates the access structure for the data. At last, the collected data is encrypted with a symmetric encryption and the symmetric key is encrypted with CP-ABE.

Act_Audit: An action audit phase in the data collection stage is executed by three parties, the CKA, the DS, and the PC. CKA may be cheated with other DS's identity information. Our action audit phase

is aimed to ensure that the DS catches the private key associated with the identification that he/she has sent to the CKA. Different from the RAAC scheme [5], the phase in our PCAC scheme is executed before the Gen_key phase. If the audit fails, Gen_key phase is not executed and the computation overhead is reduced. Since the DS's private key must be kept absolutely secret, our action audit phase proves the identity of the DS with zero-knowledge proof, protecting the privacy of the DS. This phase is based on the chameleon hash, and a specific interaction is designed between the CKA and the DS. First of all, when the DS uploads data, *Pro* needs to be uploaded to show its identification. In addition, DS also needs to select a random number $r \in Z_p$ and upload $R = g^r$ to the CKA. So the message that the DS needs to send to the CKA, when the data is uploaded, is $\{k, Pro, R\}$ (k is a symmetric key chosen by DS, which is used in the encrypt phase). Then, the CKA randomly chooses r' and sends it to the DS. DS gets r' and calculates $m = \frac{r-r'}{DS_{id}}$. Since r is a random number for CKA (as it is hard to discover r with g^r), DS_{id} can be kept secret with r . DS generates the time stamp *TS* and sends a message, which contains m and *TS*, to CKA. CKA sends an audit message $AM = \{Pro, R, m, TS, r'\}$ to the PC for audit. The PC may set a transmission delay threshold ΔTS . Upon receiving *AM* sent by the CKA, the PC first determines whether the transmission delay is within this threshold. Suppose the time that PC receives *AM* is TS' . If $TS' - TS > \Delta TS$, PC will send CKA a *REJ* audit results. This means there may be some trouble with this DS, and it requires an overhaul. In addition, the audit is operated according to Equation (1):

$$e(g^m, Pro) = e(R/g^{r'}, g) \tag{1}$$

If the audit is passed, it means that DS catches the identification associated to itself. And then, the CKA generates the access structure for the DS.

Gen_Stru: In this phase, there are two tasks: generate the access tree and convert access tree to LSSS-sharing matrix.

The access structure generation phase generates a specific access structure based on the DS's ID and is executed by CKA. The static general access structure T_1 and the dynamic restrictive access structure T_2 are stored in the CKA, in the form of access trees. As the data's access structure may change according to different situations, the access structure is divided into two parts T_1 and T_2 . T_1 is static access structure that gives the limitation, which should be met in all situations. It means that any U with attribute sets that do not satisfy T_1 can't get access to the data, at all times. T_1 should be saved in the CKA, in advance, and does not change when the system is running. T_2 does change in different situations, it further restricts the attribute set. When T_2 changes, there should be a time limit $\{Time_1, Time_2 \mid Time_1 < Time_2\}$, and it means that when $Time_1 < Time < Time_2$, the T_2 changes. The access tree should combine the T_1 's limitations and T_2 's, so $T = T_1$ and T_2 .

After the access tree is generated, it is converted to an LSSS-sharing matrix. Every attribute a can be denoted as $M_a \in Z^{1 \times 1}$, and $M_a = [1]$. Every access structure A_i can be denoted as a matrix $M_i \in Z^{d_i \times e_i}$. C_i is denoted as the first column of M_i ; R_i is denoted as all other columns of M_i . To handle access structures with "OR" door, denoted as $A_c = A_b \vee A_c$, the generated matrix M_c is denoted as Equation (2):

$$M_c = \begin{matrix} C_a & R_a & 0 \\ C_b & 0 & R_b \end{matrix} \tag{2}$$

To handle access structure with "AND" door, denoted as $A_c = A_b \wedge A_c$, the generated matrix M_c is denoted as Equation (3):

$$M_c = \begin{matrix} C_a & C_a & R_a & 0 \\ 0 & C_b & 0 & R_b \end{matrix} \tag{3}$$

Therefore, in an LSSS share-generating matrix M_{LSSS} with $l \times n$, the number of specific attributes associated with the access structure, is l and n is a value related to the trapped function-parameters. Each row of M_{LSSS} is associated with every attribute in the access structure. Therefore, when T_2 changes, the rows associated to attributes, which exist in T_2 , must change, as shown in Figure 4.

$T_1 = \{\{A, B, C\}, D\}$ and $T_2 = \{E, F, G\}$, and the final access tree generated in this phase is $T = \{T_1, T_2\} = \{\{A, B, C\}, D, E, F, G\}$. Each row of matrix is associated to attribute $\{A, B, C, D, E, F, G\}$.

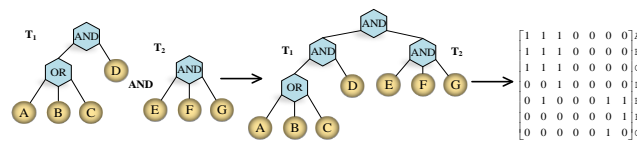


Figure 4. The Process of Generating the Access Tree.

Encrypt: Encryption algorithm is executed by CKA. Due to the heavy computation overhead of CP-ABE, our PCAC scheme uses symmetric encryption to encrypt the plaintext data M_T , and then uses CP-ABE to encrypt the key used to encrypt symmetrically. The DS randomly selects the symmetric key k and encrypts the data M_T with it, and then sends the ciphertext C_T and k to the PC and the CKA, respectively. CKA encrypts k with CP-ABE and shares the secret s with LSSS. An LSSS scheme can be expressed as (M_{LSSS}, ρ) , where ρ is a specific hash function and it maps each row in the M matrix to a specific KS, denoted as $\rho[i]$ (it can be understood that CKA distributes trapdoors to different KS, for storage by function ρ). CKA randomly selects a secret s for encrypting the symmetric key k , denoted as $C = ke(g, g)^{s\alpha}$. In order to hide and share the secret s , we randomly select $y_2, y_3, \dots, y_l \in Z_p$ and construct vector $\vec{v} = (s, y_2, y_3, \dots, y_l)$. Same as the commonly used LSSS scheme, we share the secret s and send them to specific KSs for storage, denoted as $\lambda_i = M_{LSSS, i} \cdot \vec{v}$ ($M_{LSSS, i}$ is the i^{th} row of sharing-generating M_{LSSS} and $i = 1, 2, \dots, l$). Finally, we add a random factor $r_1, r_2, \dots, r_l \in Z_p$ as part of the ciphertext. Therefore, our scheme finally calculates the ciphertext CT, as shown in Equation (4).

$$CT = \begin{cases} CK = k(e(g, g)^\alpha)^s \\ C' = g^s \\ C_i = (g^\beta)^{\lambda_i} \cdot Sn_{\rho_i}^{-r_i}, i = 1, 2, \dots, l \\ C_i'' = g^{r_i}, i = 1, 2, \dots, l \end{cases} \quad (4)$$

In Equation (4), CK is used to hide the symmetric key k , and C' , C_i , and C_i'' are used for decryption. C_i is distributed stored by KSs. After this algorithm, CKA does not hold any part of ciphertext CT, nor does it save the shared secret s . KS only stores part of C_i , and single KS cannot recover the secret s .

4.2.3. Data Obtainment Stage

The data obtainment stage shows the flow when U requests data from PC. There are three phases. First of all, U's identity should be audited by PC and then, CKA generates a secret key for U. At last, U can decrypt the data with a secret key.

Act_Audit: Action audit phase in the data obtainment stage is executed by three parties, CKA, U, and PC. This phase is similar to the Act_Audit phase in the data collection stage. First of all, U needs to upload Pro , to prove its identity. In addition, U also needs to select a random number $r \in Z_p$ and upload $R = g^r$ to CKA. And the message U needs to send to CKA is $\{Pro, R\}$. Then, CKA randomly chooses r' and sends it to U. U gets r' and calculates $m = \frac{r-r'}{U_{id}}$. U generates the time stamp TS and sends a message, which contains m and TS , to CKA. CKA sends an audit message $AM = \{Pro, R, m, TS, r'\}$ to PC for audit. PC may judge if the U's action is in overtime. In addition, the audit is operated according to Equation (1).

If the audit is passed, it means that U has caught the identification associated to himself/herself. Then, the CKA generates a secret key for U. The communication network between U and CKA is a wide area network, which has a risk of being wiretapped, so our Act_Audit phase contains randomness r' and U_{id} , which is difficult to be discovered by the listener, in multiple communications.

Gen_key: The secret key generation phase is executed by the CKA, aimed to generate the secret key, based on the CT. When U requests files, he/she needs to upload Pro_U , as long as some proof of his/her attribute set is obtained, in order to express their identity. CKA receives the message from U

and verifies the attribute set. If the verification is passed, CKA constructs an interaction to achieve specific message for Act_Audit phase. After passing the action audit, the PC calculates the SK with Equation (5), otherwise the PC sends REJ to the CKA. CKA makes further calculations to generate a secret key, which is used to decrypt the symmetric key. In order to resist replay attacks, the PCAC scheme introduces a random factor to further hide the shared secret s . CKA selects a random number $d \in Z$, and combines d with the time stamp TS and the user's Pro_U to generate a random factor A in the secret key, denoted as $A = H(d||TS||Pro_U)$. The secret key is calculated with Equation (5).

$$SK = \begin{cases} K = (g^\beta)^A \cdot g^\alpha \\ K' = g^A \\ K'' = Sn_{\rho_i}^A, i = 1, 2, \dots, l \end{cases} \tag{5}$$

After this, CKA asks the KSs and the PC for the CT. Then, CKA sends the SK and the CT to U for decryption.

Decrypt: The decrypt phase is executed by U. U is free to get the ciphertext from PC, and decrypt the symmetric key k with SK and CT, so that U is able to get the data he/she wants. If U's attribute set does not satisfy the access structure, U cannot get the secret key SK and cannot decrypt the data. For a U, holding an attribute set $S_U = \{S_{I_1}, S_{I_2}, S_{I_i} | I_i \in U\}$, if S_U satisfies the access structure, it is able to construct a matrix M_U (M_U is a sub-matrix of M_{LSSS} , and it is combined with the rows in M_{LSSS} , which is associated to the attributes that exist in S_U). Then, $\lambda_{U_i} = M_U \cdot \vec{v}^T$ and λ_U is calculated, which are sub-vectors of vector λ . There exists a vector $\vec{e} = (1, 0, \dots, 0)$, and a new vector $\vec{w} = (w_1, w_2, \dots, w_i) (w_i \in Z_p)$ can be constructed, which satisfies $\vec{e} = \vec{w} \cdot M_U$. The secret key s can be recovered from vector \vec{w} and λ_U , as shown in Equation (6).

$$\begin{aligned} s &= (1, 0, 0, \dots, 0) \cdot (s, y_1, y_2, \dots, y_i) \\ &= \vec{e} \cdot \vec{v}^T \\ &= \vec{w} \cdot M_U \cdot \vec{v}^T \\ &= \vec{w} \cdot \lambda_U \\ &= \sum_{i \in I} w_i \cdot \lambda_{U_i} \end{aligned} \tag{6}$$

Then, the further calculations are based on Equations (4)–(6).

$$\begin{aligned} C_k &= \frac{e(C', K)}{\prod_{i \in I} (e(C_i, K') \cdot e(C_i'', K_i''))^{w_i}} \\ &= \frac{e(g^s \cdot g^{\beta A + \alpha})}{\prod_{i \in I} (e(g^{\beta \lambda_i} \cdot Sn_{\rho_i}^{-r_i} \cdot g^A) \cdot e(g^{r_i}, Sn_{\rho_i}^A))^{w_i}} \\ &= \frac{e(g, g)^{s(\beta A + \alpha)}}{\prod_{i \in I} (e(g, g)^{\beta A \lambda_i} \cdot Sn_{\rho_i}^{-r_i A}) \cdot e(g, Sn_{\rho_i}^{r_i A})^{w_i}} \\ &= \frac{e(g, g)^{s(\beta A + \alpha)}}{e(g, g)^{\beta A \sum_{i \in I} w_i \cdot \lambda_i}} = e(g, g)^{\alpha s} \end{aligned} \tag{7}$$

Finally, U can get symmetric key k :

$$CK/C_k = ke(g, g)^{\alpha s} / e(g, g)^{\alpha s} = k \tag{8}$$

So, U can use the symmetric key k to decrypt the ciphertext that C_T got from PC.

5. Security Analysis and Performance Analysis

5.1. Data Confidentiality and Resist Collusion Attack

Our PCAC scheme is improved from the CP-ABE scheme, and also has CP-ABE’s security properties: data confidentiality and resist collusion attack.

Data confidentiality: File data must be kept confidential to users who do not have access. Additionally, our scheme must be able to defend Chose Plaintext Attack (CPA).

The proof of our PCAC scheme is similar to Waters [12]. The power cloud only saves ciphertext encrypted by the symmetric key and the encrypted symmetric key, in the encrypt phase. PC does not take part in generation of the secret key. It is hard to recover the plaintext without the secret key. It means that the data saved in power cloud is safe from an outside attacker.

Resist collusion attack: Multiple malicious users cannot combine with each other to decrypt the file ciphertext which a single user cannot decrypt.

Our PCAC scheme is able to effectively resist collusion attacks. In our scheme, the user does not need to upload his\her own attribute set and only needs to upload his own proof of identity. Therefore, the PC cannot be attacked by forging the attribute set. Additionally, in order to prevent to an attacked by forging proof of identity, our PCAC scheme designs Act_Audit phase based on zero-knowledge proof, and is executed before the Gen_key phase, so that malicious users cannot obtain the secret key generated by the CKA.

5.2. Storage Overhead

In this section, our PCAC scheme is compared with the scheme proposed by Waters [12] and the RAAC scheme [5], to test the storage overhead on each agency in our PCAC scheme. The comparison results are shown in Table 1. In the table, $|*|$ is the number of elements in the group $*$, L_p is the average element length in the linear group of order p . N_{KS} is the number of KSs, S is the collection of attributes, which is defined at initialization, S_U is the attribute set held by the user. To make the comparison fair, a little change needed to be made on the RAAC model. There exists several attribute authorities (AAs) in the model of RAAC, and we assumed that the number of AAs is the same as the number of KSs in our PCAC scheme.

Table 1. The Storage Overhead of Our PCAC, the Waters Scheme, and the RAAC model.

Agencies	PC	CKA	KS	U	DS/Owner
Waters [12]	N/A	$2L_p$	N/A	$(S_U + 3)L_p + L_{Zp}$	$(S + 3)L_p + L_{Zp}$
RAAC [5]	$(6 + 2 S)L_p + 4L_{Zp}$	N/A	N/A	$(S_U + 5)L_p + L_{Zp}$	$(S + 5)L_p + L_{Zp}$
PCAC	$(3 + N_{KS})L_p + 2L_{Zp}$	$3L_p + S L_p$	L_p	$4L_p + L_{Zp}$	$L_{Zp} + L_p$

As can be seen in Table 1, the PC only stores the symmetric encrypted data, while the symmetric key is encrypted by CKA and distributed stored by KSs. Besides, the PC is only responsible for initializing all the attributes in the system and does not participate in the validation of the attribute set, so the attribute data does not need to be stored in PC. Therefore, the data complexity of our PCAC scheme is less than the RAAC. Since CKA is not completely trusted, SK needs to be assigned to different KSs for distributed storage. At the same time, the storage overhead on CKA is reduced. U and DS in the PCAC scheme only saves a pair of public key and private keys, which are assigned by the PC. This reduces the storage of the attribute set.

5.3. Computation Overhead

In this section, our PCAC scheme is compared with the Waters’ scheme [12] and the RAAC scheme [5]. The computation overhead of them is analyzed. The result is shown in Table 2. In the table, N_U is the number of users, N_S is the number of attributes in the system, and N_{as} is the average number of attributes in the user’s attribute set (these attributes are also contained in the access structure). $O(1)$

means that the computation overhead is the same irrespective of the number of times the input data is increased.

Table 2. The Computation Overhead of Our PCAC, the Waters Scheme, and the RAAC model.

Phases	Init	Gen Stru		Encrypt		Gen_key		Decrypt		Act_Audit		
agencies	PC	CKA	CKA	CKA	DS	CKA	KS	PC	U	PC	CKA	U
Waters [12]	N/A	$O(N_U)$	N/A	N/A	$O(N_{SK})$	N/A	N/A	$O(N_{as})$	$O(N_{SK})$	N/A	N/A	N/A
RAAC [5]	$O(N_S + N_{DS} + N_U)$	N/A	N/A	N/A	$O(N_{SK})$	$O(N_{as})$	N/A	$O(N_{as})$	$O(N_{SK})$	$O(1)$	$O(1)$	$O(1)$
PCAC	$O(N_S + N_{DS} + N_U)$	N/A	$O(1)$	$O(N_{SK})$	$O(1)$	$O(N_{as})$	$O(1)$	$O(1)$	$O(N_{SK})$	$O(1)$	$O(1)$	$O(1)$

As can be seen in the table, the computation overhead of the encryption and decryption phases is the same as that in the Waters scheme and the RAAC scheme. The Gen_Stru phase consumes low computational complexity. The computation overhead of Act_Audit phase in the PCAC is the same as that in the RAAC. However, in RAAC it needs to be executed after the key-generation phase, so that if the audit fails, the computation overhead is wasted. Our audit phase is executed before the key-generation phase, so the PCAC can reduce some unnecessary computation overhead.

6. Experiment Analysis

In this section, our scheme has been experimentally compared with that of the RAAC, to test the operational efficiency of our PCAC scheme. The experiments were implemented in an environment of Windows 10 (in a PC with Core i7-7500U and 2.7 GHz) and programmed with JPBC [20]. The algorithm in the RAAC scheme was programmed by the description in the reference [5]. As the RAAC scheme was the first to propose the audit and tracking phase based on the CP-ABE, we chose it for the comparison experiments. Therefore, the computational efficiency of our Act_Audit phase could be tested as a comparison with the RAAC scheme.

6.1. Time Consumption in the Data Collection Stage and the Data Obtainment Stage

In this experiment, our PCAC scheme was compared with the RAAC when the DSs uploaded the data and Us requested the data. When the DSs uploaded the data, our PCAC scheme counted the Encrypt and Act_Audit phase, while RAAC counted the Encrypt phase. When requesting the data, PCAC counted the Act_Audit, the Gen_key, and the Decrypt phase, while RAAC counted the Gen_key, the Decrypt, and the Audit phase. As the Init phase was executed once the system was running, it has not been counted here. The RAAC scheme does not take the generation of the access structure into consideration, so the performance of Gen_Stru phase was not analyzed here. Instead, the time consumption of the Gen_Stru phase was counted alone, and it was found to be at least 2 ms. This experiment ignored the influence of network communication. The results are shown in Figures 5 and 6.

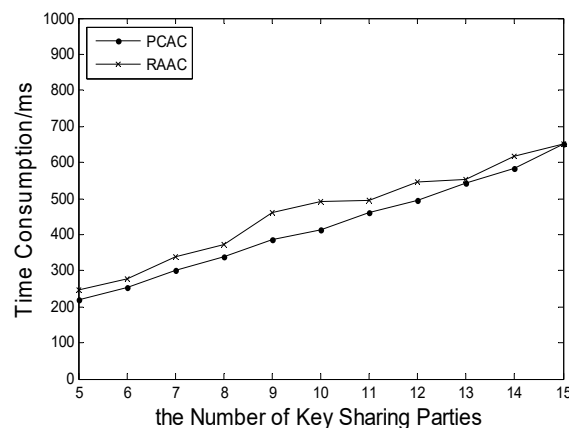


Figure 5. Time Consumption in the Data collection stage.

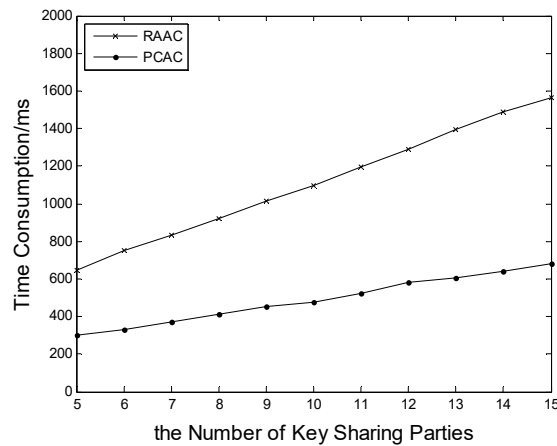


Figure 6. Time Consumption in the Data Obtainment Stage.

As can be seen from the comparison, the time consumption of our PCAC scheme increased with the increase of the number of secret-sharing parties (the number of KSs), in both stages. The number of secret-sharing parties was determined by the row number of the sharing matrix, and the row number of the matrix determined the number of C_i in CT. Therefore, the time consumption of the PCAC was proportional to the number of KSs. As can be seen from Figure 6, in the data obtainment stage, the time consumption in the PCAC was almost half of that in the RAAC.

6.2. Time Consumption of the Encrypt and the Decrypt Phase

In this section, the PCAC scheme has been compared with RAAC in the Encrypt and the Decrypt phase. The result is shown in Figure 7.

As can be seen in the figure, the time cost in the Encrypt and the Decrypt phase increased with the increase of the number of secret-sharing parties. In the Encrypt phase, the secret-sharing parties were mapped to the SKs directly, instead of using other elements to mark the KSs. Therefore, the Decrypt phase of the PCAC scheme consumed relatively less time.

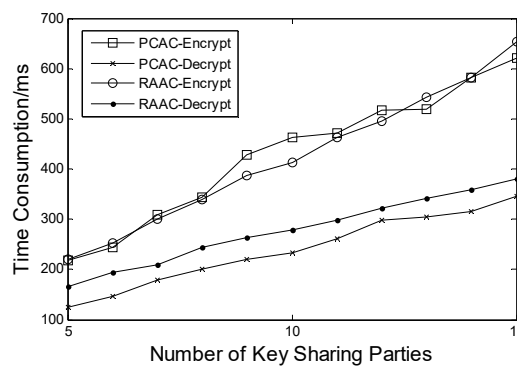


Figure 7. Time Consumption in the Encrypt and Decrypt Phase.

6.3. Time Consumption of the Act_Audit Phase

The comparison results of the Act_Audit phase is shown in Figure 8. In this phase, the RAAC scheme used zero-knowledge proof, on the basis of the chameleon hash algorithm to verify the user’s identity. To make the comparison fair, the time consumption of the trace phase in RAAC scheme was not counted. The Act_Audit phase in the PCAC was less burdensome than that in the RAAC. In the PCAC, the Act_Audit phase took about 30 ms, in average, while it cost more than 60 ms in the RAAC scheme.

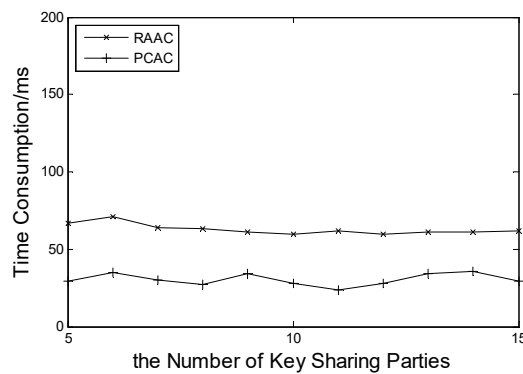


Figure 8. Time Consumption in the Act_Audit Phase.

6.4. Storage Consumption of CKA

In the data obtainment stage, U needed to send his/her Pk , and CKA was the first to verify the attribute set with his/her Pk , so the CKA needs to save $Us' Pk$ and Us' attribute sets. In this section, the experiment on the storage consumption of CKA has been discussed, and the results are shown in Table 3.

Table 3. Data Size Saved in the CKA.

Number of Attributes	Number of Us				
	100,000	300,000	500,000	1,000,000	
10	122 MB	366 MB	615 MB	1.18 GB	
15	168 MB	505 MB	842 MB	1.65 GB	
20	216 MB	644 MB	1.04 GB	2.07 GB	
25	260 MB	789 MB	1.28 GB	2.57 GB	
30	309 MB	930 MB	1.51 GB	2.99 GB	

As can be seen from the table, the data size saved in CKA was proportional to the number of U. When there were 30 attributes and 1 million Us, the data size CKA needed to be save was less than 3 GB. This was easy to meet with the storage technology today.

7. Conclusions

In this paper, we proposed a PCAC access control scheme that is suitable for power cloud environment. The PCAC scheme achieved access control from the data collection to data obtainment. A new action audit phase based on zero-knowledge proof, was also proposed, which verifies the user’s identity without infringing on the privacy of Us. Finally, the storage and computation overhead of the PCAC were analyzed through both theoretical and experimental analyses. Compared to the existing CP-ABE access control schemes, the length of the master secret key in our scheme was shorter and the storage pressure of the power cloud was lower, and the time occupied by action audit was relatively low. Therefore, the PCAC scheme could satisfy high-frequency access control requirement in the power cloud. However, there were still some shortcomings in our scheme. In the access structure generation phase, our trapdoor also had a lot of limitations. Our access structure could not achieve “NOT” door and comparable attributes. Therefore, our future work will focus on the construction of these access structure.

Author Contributions: Y.L. and P.Z. designed the framework and wrote the manuscript. Y.L. and P.Z. collected the data. Y.L. and B.W. mainly responsible for the formula analysis. B.W. verified the results of our work and conceived the experiments together. Y.L. and P.Z. discussed the results and contributed to the final manuscript.

Funding: This work was supported by the Fundamental Research Funds for the Central Universities (2018ZD06).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. US Department of Commerce, NIST. *NIST Framework and Roadmap for Smart Grid Interoperability Standards, Release 3.0*; US Department of Commerce: Washington, DC, USA, 2014.
2. Bethencourt, J.; Sahai, A.; Waters, B. Ciphertext-Policy Attribute-Based Encryption. In Proceedings of the 2007 IEEE Symposium on Security and Privacy (SP'07), Berkeley, CA, USA, 20–23 May 2007; pp. 321–334.
3. Li, L.; Gu, T.; Chang, L.; Xu, Z.; Liu, Y.; Qian, J. A Ciphertext-Policy Attribute-Based Encryption Based on an Ordered Binary Decision Diagram. *Proc. IEEE Access* **2017**, *5*, 1137–1145. [[CrossRef](#)]
4. Wang, S.; Zhou, J.; Liu, J.K.; Yu, J.; Chen, J.; Xie, W. An Efficient File Hierarchy Attribute-Based Encryption Scheme in Cloud Computing. *IEEE Trans. Inf. Forensics Secur.* **2016**, *11*, 1265–1277. [[CrossRef](#)]
5. Xue, K.; Xue, Y.; Hong, J.; Li, W.; Yue, H.; Wei, D.S.; Hong, P. RAAC: Robust and Auditable Access Control with Multiple Attribute Authorities for Public Cloud Storage. *IEEE Trans. Inf. Forensics Secur.* **2017**, *12*, 953–967. [[CrossRef](#)]
6. Wang, S.; Liang, K.; Liu, J.K.; Chen, J.; Yu, J.; Xie, W. Attribute-Based Data Sharing Scheme Revisited in Cloud Computing. *IEEE Trans. Inf. Forensics Secur.* **2016**, *11*, 1661–1673. [[CrossRef](#)]
7. Alrawais, A.; Alhothaily, A.; Hu, C.; Xing, X.; Cheng, X. An Attribute-Based Encryption Scheme to Secure Fog Communications. *IEEE Access* **2017**, *5*, 9131–9138. [[CrossRef](#)]
8. Xue, K.; Hong, J.; Xue, Y.; Wei, D.S.; Yu, N.; Hong, P. CABE: A New Comparable Attribute-Based Encryption Construction with 0-Encoding and 1-Encoding. *IEEE Trans. Comput.* **2017**, *66*, 1491–1503. [[CrossRef](#)]
9. Zhang, R.; Hui, L.; Yiu, S.; Yu, X.; Liu, Z.; Jiang, Z.L. A Traceable Outsourcing CP-ABE Scheme with Attribute Revocation. In Proceedings of the 2017 IEEE Trustcom/BigDataSE/ICSS, Sydney, NSW, Australia, 1–4 August 2017; pp. 363–370.
10. Zhou, Z.; Huang, D.; Wang, Z. Efficient Privacy-Preserving Ciphertext-Policy Attribute Based-Encryption and Broadcast Encryption. *IEEE Trans. Comput.* **2015**, *64*, 126–138. [[CrossRef](#)]
11. Yan, Z.; Li, X.; Wang, M.; Vasilakos, A.V. Flexible Data Access Control Based on Trust and Reputation in Cloud Computing. *IEEE Trans. Cloud Comput.* **2017**, *5*, 485–498. [[CrossRef](#)]
12. Waters, B. Ciphertext-Policy Attribute-Based Encryption: An Expressive, Efficient, and Provably Secure Realization. *Lect. Notes Comput. Sci.* **2011**, *2008*, 321–334.
13. Han, J.; Susilo, W.; Mu, Y.; Zhou, J.; Au, M.H.A. Improving Privacy and Security in Decentralized Ciphertext-Policy Attribute-Based Encryption. *IEEE Trans. Inf. Forensics Secur.* **2015**, *10*, 665–678.
14. Guo, F.; Mu, Y.; Susilo, W.; Wong, D.S.; Varadharajan, V. CP-ABE with Constant-Size Keys for Lightweight Devices. *IEEE Trans. Inf. Forensics Secur.* **2014**, *9*, 763–771.
15. Lin, G.; Hong, H.; Sun, Z. A Collaborative Key Management Protocol in Ciphertext Policy Attribute-Based Encryption for Cloud Data Sharing. *IEEE Access* **2017**, *5*, 9464–9475. [[CrossRef](#)]
16. Ning, J.; Cao, Z.; Dong, X.; Liang, K.; Ma, H.; Wei, L. Auditable sigma-Time Outsourced Attribute-Based Encryption for Access Control in Cloud Computing. *IEEE Trans. Inf. Forensics Secur.* **2018**, *13*, 94–105. [[CrossRef](#)]
17. Liu, J.K.; Au, M.H.; Huang, X.; Lu, R.; Li, J. Fine-Grained Two-Factor Access Control for Web-Based Cloud Computing Services. *IEEE Trans. Inf. Forensics Secur.* **2016**, *11*, 484–497. [[CrossRef](#)]
18. Li, J.; Lin, X.; Zhang, Y.; Han, J. KSF-OABE: Outsourced Attribute-Based Encryption with Keyword Search Function for Cloud Storage. *IEEE Trans. Serv. Comput.* **2017**, *10*, 715–725. [[CrossRef](#)]
19. Balu, A.; Kuppusamy, K. An expressive and provably secure Ciphertext-Policy Attribute-Based Encryption. *Inf. Sci.* **2014**, *276*, 354–362. [[CrossRef](#)]
20. De Caro, A.; Iovino, V. JPBC: Java pairing based cryptography. In Proceedings of the 2011 IEEE Symposium on Computers and Communications (ISCC), Kerkyra, Greece, 28 June–1 July 2011; pp. 850–855.

