Received August 19, 2019, accepted September 10, 2019, date of publication September 20, 2019, date of current version October 3, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2942649

Time and Attribute Based Dual Access Control and Data Integrity Verifiable Scheme in Cloud Computing Applications

QIAN ZHANG[1], SHANGPING WANG[2], DUO ZHANG[3], JIFANG WANG[3], AND YALING ZHANG[1]

[1]School of Computer Science and Engineering, Xi'an University of Technology, Xi'an 710048, China
[2]School of Science, Xi'an University of Technology, Xi'an 710054, China
[3]School of Automation and Information Engineering, Xi'an University of Technology, Xi'an 710048, China

Corresponding author: Shangping Wang (spwang@mail.xaut.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61572019, and in part by the Key Project of Natural Science Foundation of Shaanxi Province of China under Grant 2016JZ001.

ABSTRACT In the era of big data, cloud-based industrial applications can provide available and convenient data access for resource-constrained smart devices. Attribute-based encryption can be used to ensure data security while providing fine-grained data access. However, current attribute-based encryption schemes rarely consider the access control of time, and the integrity verification of data simultaneously. In response to the above two problems, we propose a time and attribute based dual access control and data integrity verifiable scheme in cloud computing applications (DCDV). Firstly, a hierarchical time tree is introduced in the attribute-based encryption technology by using of hierarchical identity-based encryption technology to set an effective access time period and a specified decryptable time period for the user's attributes key and encrypted data separately. The decryption operation can only be performed if the attribute set of user satisfies the data owner's access policy and the effective access time period of the user's attributes key completely covered the decryption time period set by the data owner. In this way, the data is dual controlled with time and attributes to solve the problem of privacy data leakage caused by private key leakage. Secondly, by using of the inverted index and Merkle hash tree, the data verification tree is designed. The data user can verify the integrity of the ciphertext data returned by the cloud server without decryption, which solves the problem that the cloud server may delete or modify the data. Finally, the security proof and efficiency analysis show that our scheme is secure and practical.

INDEX TERMS Attribute based encryption, dual access control, time specified encryption, data integrity verification.

I. INTRODUCTION

The cloud computing system [1], [2] generally comprises four basic parts: cloud platform, cloud storage, cloud terminal, and cloud security. It is an increase, use, and delivery mode of related services based on the Internet. Users access the data center through smart devices like laptops, mobile phones, etc, and perform calculations according to their own needs. Cloud computing is a pay-per-use model that provides usable, convenient, on-demand network access into a configurable pool of computing resources (including networks, servers, storage, applications, services). These resources can be quickly provided with little administrative effort or little interaction with service providers. The cloud platform [3] manages a large number of hardware resources such as CPUs, switches and memories as a basis for providing cloud computing services, which uses virtualization technology to integrate resources in a data center or multiple data centers, shields the differences between different underlying devices, and provides users with various services such as computing and storage in a transparent manner. Based on this, the cloud can transform a resource-constrained mobile device such as mobile phone, tablet, etc. into a supercomputer with a large amount of memory and a very fast central processing unit. As a result, more and more enterprises and users outsource data to cloud service providers, and many applications based

The associate editor coordinating the review of this manuscript and approving it for publication was Aniello Castiglione.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see http://creativecommons.org/licenses/by/4.0/

on cloud computing have emerged [4]–[7]. In cloud computing based applications, to prevent user sensitive data (such as medical data) from being tampered with or compromised by users or businesses, the privacy protection of data must be taken seriously. Based on this, allowing authorized users to access data for data sharing is worthy of being done.

Attribute-based encryption (ABE) [8]is a promising primitive for implementing user privacy protection and access control of data, which is an extension of public key encryption (PKE) [9] and identity-based encryption (IBE) [10] and provides more flexible encryption and decryption relationship than traditional cryptography. ABE belongs to the public key encryption mechanism, and its decryption object is a group rather than a single user. The key to realizing this feature is the introduction of the concept of attributes, which are information elements that describe the user group. For example, students in the campus network have attributes such as department, student category, grade, and major, teachers have attributes such as department, title, and teaching age. The group refers to a combination of certain attribute values of user collection. A graduate student in faculty of mathematics refers to a group of graduate students whose faculty attribute values are mathematics colleges and whose student category attribute values are graduate students. Therefore, in an attribute-based encryption mechanism, both the ciphertext and the private key are associated with a set of attributes describing the characteristics of the group, and the data owner can specify an access policy consisting of attributes [11]–[13] (CP-ABE), or the data has an access policy defined by data user [14]–[16] (KP-ABE) that he/she wants to access. Based on the encrypted content and the information about the recipient characteristics, the resulting ciphertext can only be decrypted by the user whose attributes satisfy the access policy. In generally, fine-grained non-interactive access control can be effectively implemented through ABE [17], which greatly enriches the flexibility of encryption policy and the descriptiveness of user rights. Attribute-based encryption has the characteristics of high efficiency, dynamic, flexible, privacy protection and fine-grained access control and cloud computing has a large amount of computing, storage and other resources, which can enable users of resource-constrained mobile terminals to perform secure data storage while achieving fine-grained access control of stored data by integrating of the two technologies mentioned above. There have resulted in a variety of industrial applications such as medical electronic health systems [18], [19], pay-TV systems [20], publish-subscribe systems [21], [22], Internet of Things [23], and smart cities [24], [25].

Although attribute-based encryption technology is very promising in cloud computing applications, one of its main problems is that once user's attributes satisfy the access policy, he/she can access data of the data owner's without time limit and unlimited number of times, which will disclose the user's private information to a certain extent, and it cannot be applied to specific application scenarios. In many information applications, time should be a very pivotal factor needing to be considered, and the time limit for access rights of data is realistic. In many cloud computing applications, data access is time-limited. For example, in a publish-subscribe system, a subscriber can only receive data that he/she accesses during the subscription time period, and cannot access data before or after the subscription period. At the same time, the publisher also wants to control the data that he/she publishes, that is, she/he can specify a time period to access his/her data subscribers, which requires the communication parties to set the access time. In the electronic health record designed in [26], the patient could delegate partial access rights to others, and the authorized user could automatically perform the data access of the authorized user within the specified time period, which limited the time that the authorized person searches, but didn't limit the time of the private key. The attribute-based encryption scheme with specified time period designed by the literature [27], [28] controlled the time of the encrypted data and the private key, but the two schemes expressed the time interval in the ciphertext or private key as one or more discrete time points, which was easily attacked by exhaustive methods.The literature [29] provided time-grained access control method, which limited the access rights of a specific group of users and could access the maximum number of times within a specified time. In this scheme, the scalability of the scheme to a certain extent was limited, and there was no valid time of the private key was limited. The literature [30] given a method of encrypting with time lock, that is, it could only be decrypted after a certain deadline, without any interaction with the sender, other receivers or trusted third parties. However, this scheme only controlled the decryption time, and didn't implement fine-grained control of access rights, and didn't applicable to specific data sharing applications.

Since the data is outsourced to a remote cloud server, the data owner hasn't control over the data itself. To prevent the cloud storage platform from modifying and deleting data due to saving storage space or interest, it is pivotal to verify the integrity of the data. Literature [31] proposed a new identity-based remote data integrity checking protocol, which used key homomorphic encryption primitive to reduce the complexity and cost of establishing and managing a public key authentication framework, and didn't leak information about the storage data, but the scheme was identity-based encryption and hadn't data sharing capabilities. Literature [32] introduced fuzzy identity-based auditing to solve the complex key management challenge in cloud data integrity checking, which used biometric-based identity as input. However, the verification process required multiple pairs of multiplication operations, which caused the computational burden of users. In literature [33], the cloud server could verify the ciphertext data instead of the user for data integrity verification. To reduce the search cost, the literature [34] used keywords instead of files to search and designed a privacy-protected searchable encryption scheme. Later, the literature [35] used searchable encryption technology to design a verifiable keyword search for secure big data-based mobile healthcare networks with fine-grained authorization control, then used a reversible bloom lookup table [36] and merkle hash tree [37] to implement verifiable of search results, but the result validation operation must be performed after the user performs the decryption operation.

In order to meet the requirements of fine-grained dual access control and data integrity verification in practical industrial applications, we propose a time and attribute based dual access control and data integrity verifiable scheme in cloud computing applications. Firstly, the hierarchical time tree is introduced in the attribute-based encryption technology by using hierarchical identity-based encryption technology [38], in which the time period is hierarchically controlled. The data user can process decryption calculate if attribute set of it satisfies the access policy of the data owner and the time period in private key completely covered the decryptable time period set by the data owner. which realizes the dual access control of the data with attributes and time; In addition, automatically destroys the private key after expiration which avoids the leakage of private data caused by the unrestricted use of the private key. After that, the inverted index [39] and the Merkle hash tree are used to design the data validation tree. In this way, the integrity of ciphertext data returned by the cloud server can be verified without decryption, and the decryption calculation is performed after the verification is passed, thereby avoiding unnecessary calculation overhead and tampering or deleting the data by malicious servers.

Authorized users can perform unlimited access in the existing attribute-based encryption scheme, and there may be a risk of leakage of private data due to leakage of the user's private key. Meanwhile, the cloud server may delete or modify stored data due to interest or saving memory, which may lead to the results that the retrieved data for data user is incomplete. To solve these problems, we propose a time and attribute based dual access control and data integrity verifiable scheme in cloud computing applications. The main contributions are as follows.

(1) A hierarchical time tree is designed with hierarchical identity-based encryption technology in the proposed scheme, in which a decryptable time period in the encrypted data and a valid access time period of the private key in the data user's private key are set respectively. Note that the time period can be a continuous time period or a discrete time period in encrypted data and private key of data users. The data user can perform the decryption calculation if he/she whose attribute set satisfies the access policy defined by the data owner and access time period of the private key completely covered the decryption time period set by the data owner. Our proposal implements dual access control of data with attributes and time, so that the encrypted data will no longer be decrypted after expiration of private key, that is, the automatic invalidation function can be achieved after the expiration of the private key of data user, which solves the problem of privacy data leakage caused by unlimited access by authorized data users.

(2) A data verification tree is generated by using the inverted index structure and Merkle tree, which can perform a fast integrity testing on the ciphertext data without decryption to prevent the malicious cloud server from tampering or deleting the encrypted data. Moreover, the use of inverted index technology improves the efficiency of data retrieval.

(3) Security analysis shows that the proposed scheme is secure under the selectively chosen plaintext attacks, and the security of the search token is proved. In addition, the resist multi-user collusion attacks and malicious server attacks can also be proved.

The rest of the paper is organized as follows. Section II review the preliminaries used throughout this paper. In Section III, the definition of system model and system security are described in detail. The concrete construction of scheme is presented in Section IV, and the security proof and performance and security analysis are discussed in Section V and Section VI, respectively. Finally, the conclusion is presented in Section VII.

II. PRELIMINARIES

A. BILINEAR MAPPING

Considering two multiplicative cyclic groups G and G_1 of prime order p, $G =< g >$, $e : G \times G \rightarrow G_1$ is a bilinear pairing [40] if it meets the properties:

a) **Bilinearity**: $\forall g \in G$ and $a, b \in Z_p^*$, we have $e(g^a, g^b) = e(g, g)^{ab}$;

b) **Non-degeneracy**: g is a generator of G,and $e(g, g) \neq 1$;

c) **Computability**: $e(g_1, g_2)$ is computable for all $g_1, g_2 \in G$;

B. ACCESS POLICY

1) ACCESS STRUCTURE

Let $\{P_1, \cdots, P_n\}$ is a collection of participants, we call a collection $A \subseteq 2^{\{P_1,\cdots,P_n\}}$ is monotonous, for $\forall B, C$, if $B \in A$ and $B \subseteq C$, there is $C \in A$. An access structure [41] $\{P_1, \cdots, P_n\}$ is a non-empty subset of a collection A, wherein $A \subseteq 2^{\{P_1,\cdots,P_n\}} \backslash \emptyset$. The collection in A is called the authorization set, and the collection in the absence A is called the unauthorized set. We noticed that any access structure can be converted to a Boolean function.

2) LINEAR SECRET SHARING SCHEME(LSSS)

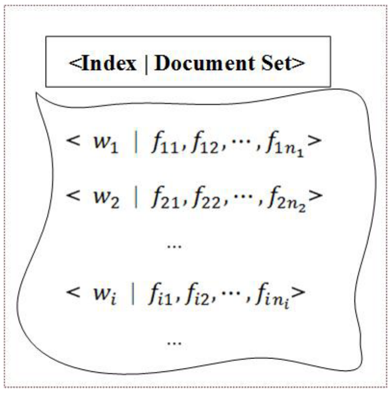
A linear secret sharing scheme [41] Π on a set of participants P is said to be linear in Z_p if :

a) The sharing of each participant forms a vector in Z_p;

b) A matrix $\mathbf{M}$ with l row and n column is called shared generation matrix. For all $i = 1, \cdots, l$, the function ρ defines the ith row of the participant's label as $\rho(i)$; we consider a column vector $\mathbf{v} = (s, r_2, \cdots, r_n) \in Z_p^n$, wherein $s \in Z_p$ is the shared secret and $r_2, \cdots, r_n \in Z_p^{(n-1)}$ are randomly chosen. $\mathbf{Mv}$ is shared vector calculated of the secret s according to Π and the shared share $(\mathbf{Mv})_i$ belongs to the participant $\rho(i)$.

C. INVERTED INDEX

The inverted index [39] is a kind of useful data structure, especially in the searchable encryption system, which maps



<Index | Document Set>
$< w_1 \mid f_{11}, f_{12}, \cdots, f_{1n_1} >$
$< w_2 \mid f_{21}, f_{22}, \cdots, f_{2n_2} >$
…
$< w_i \mid f_{i1}, f_{i2}, \cdots, f_{in_i} >$
…

FIGURE 1. Inverted index list.

a word or atomic search term to a set of documents or index units containing the word and its published content, thereby implementing the data quickly search function. Fig.1 shows an inverted index list instance. An index list consists of keyword w_i and all files $f_{i1}, f_{i2}, \cdots, f_{in_i}$ containing keyword w_i, wherein n_i is the number of files included keyword w_i.

D. MERKLE TREE

Merkle tree [37] is a kind of tree, most of which is binary tree, or multi-forked tree, which is also called Merkle hash tree. Leaf node value of Merkle tree is the unit data hash value of the data or data set. Starting from the hash value of the leaf node, the recursive operation is performed sequentially from the bottom layer to the upper layer, wherein the value of the parent node is obtained by hashing all the child node sets until the value of the root node is obtained, and this value is the basis of data verification. The Merkle tree can be used to verify the integrity of the data. As shown in Fig.2, data blocks are stored in the leaf nodes, and for other nodes, we can calculate $C = hash(d_1||d_2)$, $D = hash(d_3)$, $A = hash(C||D)$, $Root = hash(A||B)$ and so on. To verify the integrity of the data block d_3, we only need to know the hash value of the node C and node B. Comparing the calculated result with the value of the root node $Root$, we can determine whether the data in the data block is complete.

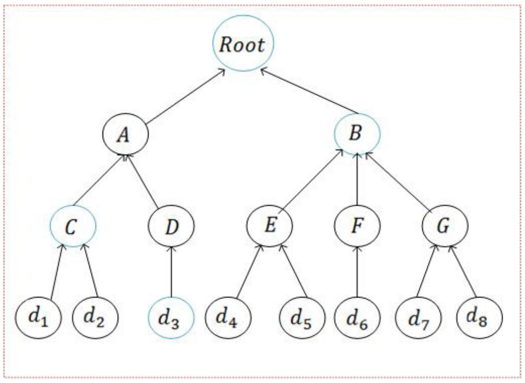


FIGURE 2. Merkel tree.

E. DATA VERIFICATION TREE

We use a binary Merkle tree (each node always contains data blocks of two adjacent nodes or their hash values), and the hash algorithm SHA-256. A data verification tree is constructed using a binary Merkle tree and an inverted index structure as shown in Fig.3. The inverted index data set $\{f_{i1}, f_{i2}, \cdots, f_{in_i}\}$ containing the keyword w_i is encrypted with the symmetric encryption key which is randomly selected to obtain a ciphertext data set L_{w_i}, and then a data verification tree IMT_{w_i} is generated for the ciphertext data set L_{w_i}, where L_{w_i} is leaf node sets of the tree IMT_{w_i}, σ_{w_i} is generated root of the tree IMT_{w_i}. In general, the algorithm complexity of confirming the integrity of any data block in the data verification tree composed of nodes N is $\log_2(N)$ only, which will greatly reduce the bandwidth and verification time required for the computer to run.

F. HIERARCHICAL TIME TREE

The idea of borrowing the hierarchical identity-based encryption (HIBE) algorithm is applied to our scheme for effective control of time. As shown in Fig.4, the root node of the hierarchical time tree (HTT) [42] is set to empty. Except for the root node, each node in the HTT represents a time period. The first layer represents the year, the second layer represents the month, and the third layer represents the day. Our scheme can also support more layers of time, such as hours, minutes, seconds, etc. Here we only describe the layer to the day. For a data owner, he/she can encrypt a message at a specified time period, which can be one year, one month or a specific day, and the data user's private key can be valid time period for one year, one month or a specific day. In the design of our scheme, the decryptable time period T_{enc} is set by the data owner, and only when the valid time period (the valid time period is recorded as T_{use}) of the authorized user who completely covered the time period defined by data owner can the data user decrypt data, that is $T_{enc} \subseteq T_{use}$ or $T_{enc} = T_{use}$. For example, the valid time period of data user's private key in the system is only May 31, 2018, and the decryptable time period defined by the data owner is May 2018. This is not a complete time period coverage, i.e. $T_{enc} \not\subset T_{use}$, then the user cannot decrypt the data encrypted by the data owner. On the contrary, if the user's validity time period of the private key is May 2018, and the decryptable time period set by the data owner is May 31, 2018, this is a complete time period coverage, namely $T_{enc} \subseteq T_{use}$, then the user can perform the decryption operation. If the encryption time of data owner is set for a certain month, then the authorized user who owns the private key for this month can decrypt the ciphertext; if the ciphertext specifies a certain day, the user can get the decryption key for this particular day from the corresponding year or month of the use HTT. For convenience of description, assuming that the depth of the HTT is $T \in Z_p^*$, each node has $\Gamma \in Z_p^*$ child node, and any time period can be represented as a string $\kappa = (\kappa_{\chi_1}, \kappa_{\chi_2}, \cdots, \kappa_{\chi_t})$ of a Γ elements, where $\chi_t < T$, and the depth of the time segment κ in the hierarchical time tree is represented as $\chi_t \in Z_p^*$.

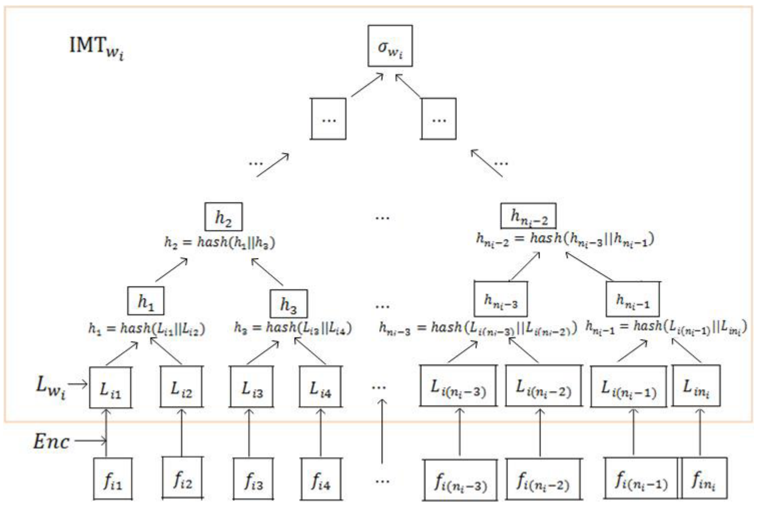

IMT_{w_i}
σ_{w_i}
$h_2 = hash(h_1||h_3)$
$h_{n_i-2} = hash(h_{n_i-3}||h_{n_i-1})$
$h_1 = hash(L_{i1}||L_{i2})$
$h_3 = hash(L_{i3}||L_{i4})$
$h_{n_i-3} = hash(L_{i(n_i-3)}||L_{i(n_i-2)})$
$h_{n_i-1} = hash(L_{i(n_i-1)}||L_{in_i})$
L_{w_i}
Enc

FIGURE 3. Data verification tree.

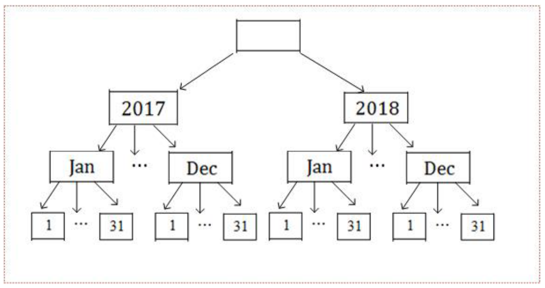


FIGURE 4. Hierarchical time tree.

G. MATHEMATICAL ASSUMPTION

1) DECISIONAL Q-BILINEAR DIFFIE-HELLMAN EXPONENTIAL ASSUMPTION(Q-BDHE)

Given $g, g^s, g^a, \cdots, g^{a^q}, g^{a^{q+2}}, \cdots, g^{a^{2q}}$, where $s, a \in Z_p$, $g \in G$, $TQ, R \in G_1$. Decisional q-BDHE assumption [42] is that no PPT algorithm P can distinguish the tuple $(g, g^s, g^a, \cdots, g^{a^q}, g^{a^{q+2}}, \cdots, g^{a^{2q}}, TQ = e(g, g)^{sa^{q+1}})$ from the tuple $(g, g^s, g^a, \cdots, g^{a^q}, g^{a^{q+2}}, \cdots, g^{a^{2q}}, TQ = R)$ with more than a negligible advantage $\varepsilon(k)$ in security parameter k. The advantage of P is defined as:

$$\begin{aligned} &Adv_P^{q-BDHE}(k) \\ &\quad = |Pr[P(g, g^s, g^a, \cdots, g^{a^q}, g^{a^{q+2}}, \cdots, g^{a^{2q}}, \\ &TQ = e(g, g)^{sa^{q+1}}) = 1] - Pr[P(g, g^s, g^a, \cdots, g^{a^q}, g^{a^{q+2}}, \\ &\qquad \cdots, g^{a^{2q}}, TQ = R) = 1]| \geqslant \varepsilon(k) \end{aligned}$$

III. DEFINITION OF SYSTEM MODEL AND SYSTEM SECURITY

A. SYSTEM MODEL

Firstly, we give some notations that will appear in our scheme as shown in Table 1.

The time and attribute based dual access control and data integrity verifiable scheme in cloud computing applications (the system model can be seen in Fig.5) contains the following four entities, and the meaning of brown abbreviations for specific operation of each entity in the Fig.5 is shown in Table 2.

TABLE 1. Notations table.

Symbol	Description
U	Universal attribute
T	The depth of HTT
GID	Global identity identifier
PP	Public parameters
MSK	Master secret key
SK	Private key of data user
T	Valid time period of SK
$\mathbb{T}$	Cover sets of T
TW	Search token
T_c	Decryptable time period of ciphertext
IMT_w	Data verification tree of w
L_w	The set of leaf node of tree IMT_w
σ_w	The root of IMT_w
I	Index of data
CT	Data ciphertext

TABLE 2. The description of abbreviation.

Abbreviation	Description
SS	System Setup
PKG	Private key Generation
TG	Token Generation
DE	Data Encryption
Up	Data uploads
DR	Data Requests
DS	Data Search
RB	Results Back
DD	Data Decryption

Attribute Authority (AA): AA is responsible for system establishment, and assigns private keys with access rights and access time period to users (including DOs and DUs) according to their corresponding attributes and identities. That is, the construction of the private key has two features of time and attribute, which is used to control the access of data.

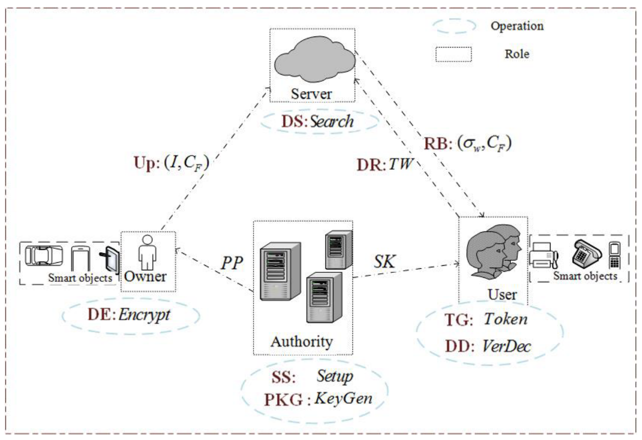


FIGURE 5. The system model of the DCDV.

Data Owner (DO): DO encrypts the uploaded data with its access policy and decryptable time period on its lightweight device side, and then generates data index by using of verification root generated by data verification tree, and uploads the ciphertext and index to CS for data sharing.

Cloud Server (CS): After receiving the search request of DU, CS provides a data retrieval service for the DU by matching the data index and the search token.

Data User (DU): DU generates a search token for sending a search request to the CS, and verifies whether the data has falsified by testing data verification root after obtaining the ciphertext data returned by the CS; if the verification passes, the decryption calculation can be performed to obtain the search data set; otherwise, the decryption calculation is terminated.

B. THE ALGORITHM DEFINITION OF THE SYSTEM

A time and attribute based dual access control and data integrity verifiable scheme in cloud computing applications is comprised of six algorithms.

Setup$(U, T, 1^k) \rightarrow (MSK, PP)$: The system setup algorithm is executed by the AA. It takes the universal attribute U, the depth T of the hierarchical time tree and the security parameter k as inputs. After the running of the algorithm, the master secret key MSK and public parameters PP of the system can be obtained.

KeyGen$(PP, MSK, S, GID, \mathrm{T}) \rightarrow SK$: The private key generation algorithm is run by the AA. It takes the public parameters PP, master secret key MSK, attribute set of DU, global identity identifier GID, and valid time period T as inputs. After the running of the algorithm, the secret key of DU can be obtained.

Token$(PP, SK, w') \rightarrow TW$: The search token generation algorithm is executed by the DU. When the input is public parameters PP, the private key SK of DU, and keyword $w' \in \{0, 1\}^*$, the algorithm generates a search token TW for the DU after the end of the algorithm.

Encrypt$(PP, T_c, w, F, (M, \rho)) \rightarrow (I, C_F)$: The data encryption algorithm is executed by the DO. When the input is public parameters PP, keyword $w \in \{0, 1\}^*$, data set F, decryptable time period C_F of data set F and access policy (M, ρ), the algorithm can obtain the data index I and ciphertext C_F of DO after the end of the algorithm.

Search$(TW, I) \rightarrow (\sigma_w, C_F)$: The data search algorithm is run by the CS. When the TW and I of DU and DO are respectively obtained, the CS performs the matching calculation of the token and the index, and returns the searched verification root σ_w and the matching ciphertext C_F to the DU.

VerDec$(C_F, SK, \sigma_w) \rightarrow For\perp$: The data decryption algorithm is executed by the DU. It takes the verification root σ_w, ciphertext C_F and private key SK of DU as inputs. Firstly, it is judged whether the ciphertext data is complete and not tampered with or deleted; If the valid access time period of the DU cannot completely cover the decryptable time period defined in the ciphertext, or the attribute set of the DU cannot satisfy the access policy set by the DO, then the algorithm terminates; otherwise, the DU performs decryption calculation to obtain search data set.

C. THE DEFINITION OF THE SYSTEM SECURITY

1) DEFINITION 1: SELECTIVELY CHOSEN PLAINTEXT ATTACKS GAME (SCPA)

SCPA game is interacted by an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. It consists of six steps: Initialization, System setup, Inquiry phase 1, Challenge, Inquiry phase 2, and Guess. The concrete steps are as follows.

Init: $\mathcal{A}$ submits firstly to $\mathcal{C}$ a challenge access policy (M^*, ρ^*), there are $n^* \leq q$ columns in M^*, a challenge time period T_c^* is expressed as $\kappa^* = (\kappa_1^*, \kappa_2^*, \cdots, \kappa_{\tau^*}^*)$, wherein $\tau^* \leqslant T$.

Setup: $\mathcal{C}$ runs the system setup algorithm, and sends the public parameters PP to $\mathcal{A}$.

*Phase*1: $\mathcal{A}$ carries out repeatedly the private key inquiry corresponding to the attribute set and the time tuple (S, T), wherein the tuple at least satisfies either of the following two cases.

(i) The access policy (M^*, ρ^*) cannot be satisfied by the attribute set S;

(ii) κ^* and all its prefixes are not in the cover set $\mathbb{T}$ of T.

Challenge: $\mathcal{A}$ submits two equal length messages m_0 and m_1 to $\mathcal{C}$. $\mathcal{C}$ throws a random coin $\beta \in \{0, 1\}$ and generates the ciphertext CT_β^* by encrypting the message m_β in the access policy (M^*, ρ^*) and time period T_c^*, and then sends generated ciphertext CT_β^* to the $\mathcal{A}$.

*Phase*2: The steps are the same as *Phase*1.

Guess: $\mathcal{A}$ final outputs β' as a guess for β.

The advantage of $\mathcal{A}$ in above SCPA game can be defined as:

$$Adv_A^{SCPA}(k) = |\Pr[\beta' = \beta] - 1/2|.$$

IV. THE CONCRETE CONSTRUCTION OF DCDV SCHEME

To solve the problem of disclosure of private information caused by unrestricted search by authorized users and the untrue data returned by cloud servers, we propose a time and attribute-based dual access control and data integrity verifiable scheme in cloud computing applications. Firstly, A hierarchical time tree is introduced in attribute based encryption technology, which controls the data search by using time and attributes, so that the leakage of private information caused by private key leakage is solved. After that, the Merkle tree and the inverted index technology are used to design a data verification tree, in which the integrity of the search results can be verified returned by the cloud server without decryption. In this way, the problem that the cloud server returns data is not true and incomplete is solved. The specific algorithm construction of the time and attribute based dual access control and data integrity verifiable scheme in cloud computing applications is explained as follows.

A. SYSTEM SETUP

Setup$(U, T, 1^k) \rightarrow (MSK, PP)$: The concrete steps of the system setup are shown in algorithm 1. The public parameters PP are public and available to users in the system, and the master key MSK is kept in AA itself for generating private keys of authorized users.

Algorithm 1 System setup

```
  Input: Security parameter $k$, universal attribute $U$,
         the depth $T$ of HTT.
  Output: Public parameters $PP$, master secret key $MSK$.
1 Selects multiplicative cyclic groups $G$ and $G_1$,
  $e : G \times G \rightarrow G_1$, wherein $G =< g >$, a time is
  expressed a string$\{1, \Gamma\}^{T-1}$ of $\Gamma$ elements;
2 Selects randomly $h_1, h_2, \cdots, h_U \in G$ and
  $V_0, V_1, \cdots, V_T \in G$;
3 Selects two hash functions $H : \{0, 1\}^* \rightarrow Z_p$,
  $H_1 : \{0, 1\}^{256} \times G_1 \rightarrow \{0, 1\}^{256}$;
4 Selects randomly $\alpha, a \in Z_p$;
5 Calculates $g^a, e(g, g)^\alpha$;
6 $\alpha \leftarrow MSK$;
7 $\{g, g^a, e(g, g)^\alpha, h_1, h_2, \cdots, h_U, V_0, V_1, \cdots, V_T\} \leftarrow$
  $PP$;
8 return $MSK, PP$.;
```

B. PRIVATE KEY GENERATION

KeyGen$(PP, MSK, S, GID, \mathrm{T}) \rightarrow SK$: The concrete steps of the private key generation as shown in algorithm 2. AA sends generated private key SK to data owner DU, and DU executes decryption computing by using it. For a specific user in the system, assuming that his/her private key is valid from October 20, 2017 to December 31, 2018, then he/she can get the private key on the nodes "2017-10-30," "2017-10-31," "2017-11," "2017-12," "2018." That is, the cover set is $\mathbb{T} = \{(2017, 10, 30), (2017, 10, 31), (2017, 11), (2017, 12), (2018)\}$.

C. SEARCH TOKEN GENERATION

Token$(PP, SK, w') \rightarrow TW$: The specific steps of the search token generation are shown in algorithm 3, which obtains

Algorithm 2 Private Key Generation

```
  Input: Attribute set $S$ of DU, Global identity identifier
         $GID$, valid time period T, $PP$, $MSK$.
  Output: Private key $SK$ of DU.
1 $\mathbb{T}$ is minimum cover set of T, $\mathbb{T}$ contains multiple time
  intervals expressed as $\kappa = (\kappa_{\chi_1}, \kappa_{\chi_2}, \cdots, \kappa_{\chi_\tau}) \in$
  $\{1, \Gamma\}^{\chi_\tau}$, For any $\kappa \in \mathbb{T}$, there is $\chi_\tau < T$, where
  $\chi_\tau \in Z_p^*$ is the depth of HTT;
2 Selects randomly $t, v_\kappa \in Z_p$;
3 Calculates $H(GID) \leftarrow u$, $g^t \leftarrow D_0$, $g^u \leftarrow D_1$,
  $g^{a/u} \leftarrow D_2$, $\{g^{v_\kappa} \leftarrow D_{0,\kappa}\}_{\kappa \in \mathbb{T}}$,
  $\{g^\alpha g^{at} g^u (V_0 \prod_{j=1}^{\chi_\tau} V_j^{\kappa_j})^{v_\kappa} \leftarrow D_{1,\kappa}\}_{\kappa \in \mathbb{T}}$,
  $\{V_j^{v_\kappa} \leftarrow R_{j,\kappa}\}_{j=\chi_\tau+1,\cdots,T,\kappa \in \mathbb{T}}$, $\{h_x^t \leftarrow K_x\}_{x \in S}$;
4 $\{D_0, D_1, D_2, \{D_{0,\kappa}, D_{1,\kappa}, R_{\chi_\tau+1,\kappa}, \cdots, R_{T,\kappa}\}_{\kappa \in \mathbb{T}},$
  $\{K_x\}_{x \in S}\} \leftarrow SK$;
5 return $SK$;
```

the search token TW of the DU. In the algorithm 3, the elements D_1, D_2 are components of the private key SK of the DU, which are confidential and are known to DU only. The keyword w' is hashed firstly, and then it is encrypted at the exponent position.

Algorithm 3 Search Token Generation

```
  Input: Public parameters $PP$, Private key $SK$ of DU,
         keyword $w'$.
  Output: Search token $TW$.
1 Calculates $H(w')$;
2 Calculates $D_1^{H(w')}$;
3 Computes $e(D_1^{H(w')}, D_2)$;
4 $e(D_1^{H(w')}, D_2) \leftarrow TW$;
5 return $TW$.
```

D. DATA ENCRYPTION

Encrypt$(PP, T_c, w, F, (M, \rho)) \rightarrow (I, C_F)$: The specific algorithm steps of the data encryption algorithm are shown in algorithm 4. The DO uploads the ciphertext C_F and the index I generated by the data verification root and keyword w to the CS for data sharing. In the encryption algorithm, the decryptable time period T_c of ciphertext is represented in the same way as the algorithm 2. The data verification tree generation process in step 4 is described in the part E of the section II.

E. DATA SEARCH

Search$(TW, I) \rightarrow (\sigma_w, C_F)$: The specific algorithm steps of the data search algorithm are shown in algorithm 5. After the running of the algorithm, CS sends the matching verification root σ_w and data ciphertext C_F to the DU.

Algorithm 4 Data Encryption

```
Input: Public parameters $PP$, access policy$(M_{l\times n}, \rho)$,
       keyword $w$, data set $F$, decryptable time $T_c$ of
       ciphertext.
Output: Index $I$, ciphertext $C_F$.
1  The time $T_c$ can be expressed as $\kappa_c = (\kappa_1, \kappa_2, \cdots, \kappa_\tau) \in \{1, \Gamma\}^\tau$, wherein $\tau < T$, $\tau \in Z_p^*$ is the depth of the time period $T_c$ in the hierarchical time tree.
2  Selects the symmetric encryption key $k_e \in G_T$ of AES;
3  Calculates $Enc_{k_e}(F) \leftarrow L_w$ using symmetric encryption algorithm AES;
4  Generates a data verification tree $IMT_w$ for the ciphertext data set $L_w$, where $L_w$ is the collection of leaf nodes of $IMT_w$, and $\sigma_w$ is the root of $IMT_w$;
5  Selects randomly a vector $v = (s, l_2, \cdots, l_n) \in Z_p^n$;
6  Calculates $\lambda_i = M_i \cdot v$, for $i = 1, \cdots, l$;
7  Selects randomly $r \in \{0, 1\}^{256}$;
8  Calculates $(r, \sigma_w \oplus H_1(r, e(g^{H(w)}, g^a)))$;
9  $(r, \sigma_w \oplus H_1(r, e(g^{H(w)}, g^a))) \leftarrow I$;
10 Computes $k_e \cdot e(g, g)^{\alpha s} \leftarrow C_0$, $g^s \leftarrow C_1$, $(V_0 \prod_{j=1}^{\tau} V_j^{\kappa_j})^s \leftarrow C_2$, $\{g^{a\lambda_i} h_{\rho(i)}^{-s} \leftarrow C'_i\}_{i=1,\cdots,l}$;
11 $\{C_0, C_1, C_2, \{C'_i\}_{i=1,\cdots,l}\} \leftarrow CT$;
12 $\{CT, IMT_w\} \leftarrow C_F$;
13 return $I$,$C_F$.
```

Algorithm 5 Data Search

```
Input: Index $I$, search token $TW$.
Output: $(\sigma_w, C_F)$ or $\perp$.
1  Parses $(r, \sigma_w \oplus H_1(r, e(g^{H(w)}, g^a))) \leftarrow I$, obtains $(r, \sigma_w \oplus val)$;
2  Judges $H_1(r, TW)$=$val$;
3  if $H_1(r, TW) \neq val$ then
4      The program is terminated, outputs $\perp$;
5  else
6      Computes $\sigma_w = H_1(r, TW) \oplus val$, and continues to step 8;
7  end
8  $H_1(r, TW) \oplus val \leftarrow \sigma_w$,  $(IMT_w, CT) \leftarrow C_F$;
9  return $\sigma_w$, $C_F$.
```

F. DATA DECRYPTION

$VerDec(C_F, SK, \sigma_w) \rightarrow F or \perp$: The specific algorithm steps of the data decryption are shown in algorithm 6. The DU firstly verifies the integrity of the data by using the ciphertext data set and the data verification root σ_w returned by the CS. After the verification is passed, the DU performs the decryption calculation, if the access time period of DU cannot completely cover the decryptable time period which is defined in the encrypted data or the attribute set of the DU cannot satisfy the access policy set by the DO, then algorithm is terminated and outputs $\perp$; otherwise, the DU calculates the decryption secret key k_e to decrypt the parsed ciphertext set L_w, and obtains effective plaintext data set F.

Algorithm 6 Data decryption

```
Input: Data verification root $\sigma_w$, ciphertext $C_F$, private
       key $SK$ of DU.
Output: Plaintext data set $F$ or $\perp$.
1  Parses from ciphertext $C_F$ to get $IMT_w$, obtains the ciphertext data set $L_w$;
2  Verifies the data in $L_w$ and get the root $\sigma'_w$;
3  Compares $\sigma'_w$ and $\sigma_w$;
4  if $\sigma'_w \neq \sigma_w$ then
5      The program is terminated, outputs $\perp$;
6  else
7      Continues to step 9;
8  end
9  Defined $I = \{i : \rho(i) \in S\}$, $I \subset \{1, 2, \cdots, l\}$;
10 if the access policy $(M, \rho)$ can't be satisfied by attribute set $S$ of DU then
11     The program is terminated, outputs $\perp$;
12 else
13     Continues to step 15;
14 end
15 Calculates $\{\omega_i \in Z_p\}_{i\in I}$;
16 Calculates $\prod_{i\in I} (e(C'_i, D_0) \cdot e(C_1, K_{\rho(i)}))^{\omega_i} \leftarrow \zeta$; (1)
17 if $\kappa_c = (\kappa_1, \kappa_2, \cdots, \kappa_\tau) \in \mathbb{T}$ then
18     $D_{1,\kappa_c} = D_{1,\kappa_c}$;
19 else
20     The prefix $\kappa'_c = (\kappa_1, \kappa_2, \cdots, \kappa_{\tau'}) \in \mathbb{T}$ of $\kappa_c$,
       wherein $\tau' < \tau$, and $\kappa_{c'} \in \mathbb{T}$;
21 end
22 Computes the private key $D_{1,\kappa_c} = D_{1,\kappa_{c'}} \prod_{j=\tau'+1}^{\tau} R_{j,\kappa'_c}^{\kappa_j}$ of $\kappa_{c'}$, and sets $\kappa_c = \kappa'_c$;
23 Computes $k_e = \frac{C_0 \cdot \zeta \cdot e(C_1, D_1) \cdot e(D_{0,\kappa_c}, C_2)}{e(D_{1,\kappa_c}, C_1)}$ ;(2)
24 $F \leftarrow Dec_{k_e}(L_w)$;
25 return $F$.
```

V. SECURITY PROOF

A. CORRECTNESS ANALYSIS

In equation (1) of the algorithm 6, if the attribute set S satisfies the access policy (M, ρ), a set of constants $\{\omega_i \in Z_p\}_{i\in I}$ can be calculated, so that $\sum_{i\in I} \omega_i\lambda_i = s$, then we can obtain:

$$\begin{aligned}\zeta &= \prod_{i\in I} (e(C'_i, D_0) \cdot e(C_1, K_{\rho(i)}))^{\omega_i} \\ &= \prod_{i\in I} (e(g^{a\lambda_i} h_{\rho(i)}^{-s}, g^t) \cdot e(g^s, h_{\rho(i)}^t))^{\omega_i} \\ &= e(g, g)^{ta \sum_{i\in I} \omega_i\lambda_i} \cdot e(g^s, h_{\rho(i)}^t) \cdot e(g^t, h_{\rho(i)}^{-s}) = e(g, g)^{tas}\end{aligned}$$

The equation (1) is proved; if the attribute set of DU cannot satisfy the access policy, then such a constant set cannot be found in the polynomial time, then the result $\zeta = e(g, g)^{tas}$

cannot be obtained, the matching fails; After that, the process of continuing to verify equation (2) is as follows:

$$
\begin{aligned}
k_e &= \frac{C_0 \cdot \zeta \cdot e(C_1, D_1) \cdot e(D_{0,\kappa_c}, C_2)}{e(D_{1,\kappa_c}, C_1)} \\
&= \frac{k_e \cdot e(g,g)^{\alpha s} \cdot e(g,g)^{tas} \cdot e(g^s, g^u) \cdot e(g^{v_{\kappa_c}}, (V_0 \prod_{j=1}^{\tau} V_j^{\kappa_j})^s)}{e(g^\alpha g^{at} g^u (V_0 \prod_{j=1}^{\tau} V_j^{\kappa_j})^{v_{\kappa_c}}, g^s)} \\
&= \frac{k_e \cdot e(g,g)^{\alpha s} \cdot e(g,g)^{tas} \cdot e(g,g)^{su} \cdot e(g^{v_{\kappa_c}}, (V_0 \prod_{j=1}^{\tau} V_j^{\kappa_j})^s)}{e(g^\alpha, g^s) \cdot e(g^{at}, g^s) \cdot e(g^u, g^s) \cdot e((V_0 \prod_{j=1}^{\tau} V_j^{\kappa_j})^{v_{\kappa_c}}, g^s)} \\
&= \frac{k_e \cdot e(g,g)^{\alpha s} \cdot e(g,g)^{tas} \cdot e(g,g)^{su} \cdot e(g^{v_{\kappa_c}}, (V_0 \prod_{j=1}^{\tau} V_j^{\kappa_j})^s)}{e(g,g)^{\alpha s} \cdot e(g,g)^{tas} \cdot e(g,g)^{su} \cdot e((V_0 \prod_{j=1}^{\tau} V_j^{\kappa_j})^{v_{\kappa_c}}, g^s)}
\end{aligned}
$$

In equation (2), if the valid time period of user's private key completely covers the decryptable time period set by the DO, we can calculate the decryption key k_e by equation (2); if the decryption time period set by the DO cannot be completely covered by the valid time period of user's private key, then the portion of the equation (2) regarding the time setting cannot be canceled, and the decryption key k_e cannot be obtained. Therefore, the correctness of the scheme can be proved by equations (1) and (2) in the algorithm 6.

B. SECURITY PROOF

The specific steps of the security proof of ABE are as follows: The challenger pre-sets a difficult problem, and the adversary attacks the scheme constructed by us. The challenger interacts with the adversary to conduct a secure game, and embeds mathematical difficulties into the solution; the adversary attacks the solution we construct, and the challenger solves the difficult problem set in advance. Since the pre-set math problem is intractable in PPT time, the security of our scheme can be derived from the concept of reduction to absurdity. In the DCDV scheme, we reduction its security to solve the difficult problem of q-BDHE.

Theorem 1: Given the size $l^* \times n^* (l^*, n^* \leq q)$ of the challenge matrix (M^*, ρ^*) and a Γ meta-representation $\kappa^* = (\kappa_1^*, \kappa_2^*, \cdots, \kappa_{\tau^*}^*)$ of the challenge time period $T_c^* (\tau^* < T, T \leq q)$, if the q-BDHE assumption holds, our DCDV scheme is secure against selectively chosen plaintext attacks.

Proof: Suppose that the PPT adversary $\mathcal{A}$ who has a non-negligible advantage $\varepsilon(k)$, then we can construct a PPT challenger $\mathcal{C}$ who can solve the q-BDHE problem with the advantage of $\varepsilon(k)/2$. The interaction between the adversary $\mathcal{A}$ and the challenger $\mathcal{C}$ is as follows.

Init: The challenger $\mathcal{C}$ obtains $y = (g, g^s, g^a, \cdots, g^{a^q}, g^{a^{q+2}}, \cdots, g^{a^{2q}}) \in G^{2q+1}, TQ \in G_1$, then the adversary $\mathcal{A}$ judges whether there is $TQ = e(g,g)^{sa^{q+1}}$. $\mathcal{A}$ submits an access policy (M^*, ρ^*) that it wants to challenge, there are $n^* \leq q$ columns in M^*, and submits a challenge time period T_c^* is expressed as $\kappa^* = (\kappa_1^*, \kappa_2^*, \cdots, \kappa_{\tau^*}^*)$ to $\mathcal{C}$ simultaneously, wherein $\tau^* \leq T$.

Setup: $\mathcal{C}$ selects randomly $\alpha', \delta_0, \xi_0, \xi_1, \cdots, \xi_T \in Z_p$, and sets implicitly $\alpha = \alpha' + \delta_0 a^{q+1}$, then computes $e(g,g)^\alpha = e(g^a, g^{a^q})^{\delta_0} e(g,g)^{\alpha'}$. $\mathcal{C}$ chooses randomly z_x for each $x (1 \leq x \leq U)$, and denotes the set of the index i as I, that is $\rho^*(i) = x$, and calculates h_x as $h_x = g^{z_x} g^{aM_{i,1}^*} \cdot g^{a^2 M_{i,2}^*} \cdots g^{a^{n^*} M_{i,n}^*}$. If $I = \varnothing$, we can obtain $h_x = g^{z_x}$. Then for any $j \in [1, T]$, $\mathcal{C}$ defines $V_j = g^{\xi_j a^{q-j+1}}$, $V_0 = \prod_{j=1}^{\tau^*} V_j^{-\kappa_j^*} g^{\xi_0}$.

*Phase*1: $\mathcal{A}$ carries out repeatedly the private key inquiry corresponding to the attribute set and the time tuple (S, T), wherein the tuple at least satisfies either of the following two cases.

*Case*1: The access policy (M^*, ρ^*) cannot be satisfied by the attribute set S;

*Case*2: κ^* and all its prefixes are not in the cover set $\mathbb{T}$ of T.

Then, the adversary $\mathcal{A}$ simulates the private key in two cases separately.

*Case*1: The access policy (M^*, ρ^*) cannot be satisfied by attribute set S;

$\mathcal{C}$ selects randomly $\varphi \in Z_p$, then computes a vector $\omega = (\omega_1, \cdots, \omega_{n^*}) \in Z_p^{n^*}$, wherein $\omega_1 = -1$ and $\omega \cdot M_i^* = 0$ for any i of $\rho^*(i) \in S$. $\mathcal{C}$ defines implicitly $t = \varphi + \delta_0(\omega_1 a^q + \omega_2 a^{q-1} + \cdots + \omega_{n^*} a^{q-n^*+1})$, and simulates D_0 as $D_0 = g^t = g^\varphi \prod_{i=1}^{n^*} (g^{a^{q-i+1}})^{\delta_0 \omega_i}$; Then it chooses randomly $u \in Z_p$, and sets $D_1 = g^u$,$D_2 = g^{a/u}$. For all $\kappa = (\kappa_{\chi_1}, \kappa_{\chi_2}, \cdots, \kappa_{\chi_\tau}) \in \mathbb{T}$, $\mathcal{C}$ selects randomly $v_\kappa \in Z_p$ to set $D_{0,\kappa} = g^{v_\kappa}$, and simulates $D_{1,\kappa}$ as:

$$
\begin{aligned}
D_{1,\kappa} &= g^{\alpha'} g^{a\varphi} \prod_{i=2}^{n^*} (g^{a^{q-i+2}})^{\delta_0 \omega_i} g^u (V_0 \prod_{j=1}^{\chi_\tau} V_j^{\kappa_j})^{v_\kappa} \\
&= g^{\alpha'} g^{\delta_0 a^{q+1}} g^{a\varphi} g^{-\delta_0 a^{q+1}} \prod_{i=2}^{n^*} (g^{a^{q-i+2}})^{\delta_0 \omega_i} \cdot \\
&\quad g^u (V_0 \prod_{j=1}^{\chi_\tau} V_j^{\kappa_j})^{v_\kappa} \\
&= g^\alpha g^{a\varphi} g^{\omega_1 \delta_0 a^{q+1}} \prod_{i=2}^{n^*} (g^{a^{q-i+2}})^{\delta_0 \omega_i} g^u (V_0 \prod_{j=1}^{\chi_\tau} V_j^{\kappa_j})^{v_\kappa} \\
&\quad (\because \omega_1 = -1) \\
&= g^\alpha g^{a\varphi} \prod_{i=1}^{n^*} (g^{a^{q-i+2}})^{\delta_0 \omega_i} g^u (V_0 \prod_{j=1}^{\chi_\tau} V_j^{\kappa_j})^{v_\kappa} \\
&= g^\alpha (g^\varphi \prod_{i=1}^{n^*} (g^{a^{q-i+1}})^{\delta_0 \omega_i})^a g^u (V_0 \prod_{j=1}^{\chi_\tau} V_j^{\kappa_j})^{v_\kappa} \\
&= g^\alpha g^{at} g^u (V_0 \prod_{j=1}^{\chi_\tau} V_j^{\kappa_j})^{v_\kappa}
\end{aligned}
$$

Finally, $\mathcal{C}$ simulates K_x for any $x \in S$ as follows. If there not exist i that has the situation $\rho^*(i) = x$, then lets $K_x = D_0^{z_x}$; Otherwise, it sets $K_x = D_0^{z_x} \cdot \prod_{j=1}^{n^*}(g^{a^j\varphi} \prod_{\mu=1,\mu\neq j}^{n^*} (g^{a^{q+1+j-\mu}})^{\omega_\mu \delta_0})^{M^*_{i,j}}$.

*Case*2: κ^* and all its prefixes are not in the cover set $\mathbb{T}$ of T.

For all $\kappa = (\kappa_{\chi_1}, \kappa_{\chi_2}, \cdots, \kappa_{\chi_\tau}) \in \mathbb{T}$, $\mathcal{C}$ defines firstly $\kappa_{\chi_\tau+1} = \cdots = \kappa_q = 0$ and $\kappa^*_{\tau^*} = \cdots = \kappa^*_q = 0$, then it can obtain the minimum index $\chi_{\tau'} \leq \chi^*_{\tau'}$ to make $\kappa_{\chi'_\tau} \neq \kappa^*_{\chi'_\tau}$. $\mathcal{C}$ chooses randomly $v_\kappa \in Z_p$, and calculates $D_{0,\kappa} = g^{\frac{\delta_0 a^{\tau'}}{\xi_{\tau'}(\kappa^*_{\tau'}-\kappa_{\tau'})}+v_\kappa}$ by setting implicitly $v_\kappa \in Z_p$. Then, challenger $\mathcal{C}$ sets $D_0 = g^t$,$D_1 = g^u$, $D_2 = g^{a/u}$ by selecting randomly $t, u \in Z_p$, and calculates $D_{1,\kappa}$ as:

$$\begin{aligned}
&D_{1,\kappa}\\
&= g^{\alpha'+at+u+v_\kappa\xi_0}(g^{a^{q-\tau'+1}})^{v_\kappa\xi_{\tau'}(\kappa_{\tau'}-\kappa^*_{\tau'})}(g^{a^{\tau'}})^{\frac{\xi_0\delta_0}{\xi_{\tau'}(\kappa^*_{\tau'}-\kappa_{\tau'})}} \cdot\\
&\quad \prod_{j=1}^{\chi_\tau-\tau'+1}(g^{a^{q-j+1}})^{\frac{\xi_{j+\tau'}\kappa^*_{j+\tau'}\delta_0}{\xi_{\tau'}(\kappa^*_{\tau'}-\kappa_{\tau'})}} \cdot \prod_{j=\tau'+1}^{\chi_\tau+1}(g^{a^{q-j+1}})^{\xi_j\kappa^*_j v_\kappa}\\
&= g^{\alpha'+\delta_0 a^{q+1}} g^{at} g^u g^{(\xi_0+\xi_{\tau'}a^{q-\tau'+1}(\kappa_{\tau'}-\kappa^*_{\tau'}))(\frac{\delta_0 a^{\tau'}}{\xi_{\tau'}(\kappa^*_{\tau'}-\kappa_{\tau'})}+v_\kappa)} \cdot\\
&\quad \prod_{j=\chi_{\tau'}+1}^{\chi_\tau+1} g^{\xi_j a^{q-j+1}\kappa^*_j(\frac{\delta_0 a^{\tau'}}{\xi_{\tau'}(\kappa^*_{\tau'}-\kappa_{\tau'})}+v_\kappa)}\\
&= g^\alpha g^{at} g^u g^{(\xi_0+\xi_{\tau'}a^{q-\tau'+1}(\kappa_{\tau'}-\kappa^*_{\tau'}))\tilde{v}_\kappa} \prod_{j=\chi_{\tau'}+1}^{\chi_\tau+1} g^{\xi_j a^{q-j+1}\kappa^*_j\tilde{v}_\kappa}\\
&= g^\alpha g^{at} g^u (V_0 \prod_{j=1}^{\chi_{\tau'}} V_j^{\kappa_j})^{\tilde{v}_\kappa} \prod_{j=\chi_{\tau'}+1}^{\chi_\tau+1} V_j^{\kappa_j\tilde{v}_\kappa}\\
&= g^\alpha g^{at} g^u (V_0 \prod_{j=1}^{\chi_\tau+1} V_j^{\kappa_j})^{\tilde{v}_\kappa}\\
&= g^\alpha g^{at} g^u (V_0 \prod_{j=1}^{\chi_\tau} V_j^{\kappa_j})^{\tilde{v}_\kappa} (\because \kappa_{\chi_\tau+1} = 0)
\end{aligned}$$

After that, $\mathcal{C}$ computes $\{K_x = h_x^t\}_{x\in S}$, and computes $R_{j,\kappa}$ for all $\chi_\tau + 1, \cdots, T, \kappa \in \mathbb{T}$ as:

$$\begin{aligned}
R_{j,\kappa} &= g^{\frac{\delta_0\xi_j a^{q+1+\tau'-j}}{\xi_{\tau'}(\kappa^*_{\tau'}-\kappa_{\tau'})}+v_\kappa\xi_j a^{q+1-j}}\\
&= (g^{\xi_j a^{q-j+1}})^{\frac{\delta_0 a^{\tau'}}{\xi_{\tau'}(\kappa^*_{\tau'}-\kappa_{\tau'})}+v_\kappa} = V_j^{\tilde{v}_\kappa}
\end{aligned}$$

Challenge: $\mathcal{A}$ submits two equal length messages m_0 and m_1 to $\mathcal{C}$. $\mathcal{C}$ throws a random coin $\beta \in \{0, 1\}$ and generates the ciphertext CT^*_β by encrypting the message m_β in the access policy (M^*, ρ^*) and time period T^*_c, then it computes $C_0 = m_\beta \cdot TQ^{\delta_0} \cdot e(g^s, g^{\alpha'})$, $C_1 = g^s$. Since the time period being challenge is $\kappa^* = (\kappa^*_1, \kappa^*_2, \cdots, \kappa^*_{\tau^*})$, the item g^{a^i} in the V_i can be eliminated. $\mathcal{C}$ sets $C_2 = (g^s)^{\xi_0}$. For C', $\mathcal{C}$ selects $l'_2, \cdots, l'_{n^*} \in Z_p$ and conducts secret sharing by using vector $v^* = (s, sa+l'_2, sa^2+l'_3, \cdots, sa^{n-1}+l'_{n^*}) \in Z_p^{n^*}$, it defines:

$\lambda^*_i =< v^*, M^*_i >= sM^*_{i,1}+(sa+l'_2)M^*_{i,2}+(sa^2+l'_3)M^*_{i,3}+ (sa^{n-1} + l'_{n^*})M^*_{i,n^*}$.

For $i = 1, \cdots, n^*$, $\mathcal{C}$ generates C'_i as:

$$\begin{aligned}
C'_i &= g^{a\lambda^*_i} h^{-s}_{\rho^*(i)}\\
&= (g^{asM^*_{i,1}} g^{(sa+l'_2)aM^*_{i,2}} g^{(sa^2+l'_3)aM^*_{i,3}} \cdots g^{(sa^{n-1}+l'_{n^*})aM^*_{i,n^*}})\\
&\quad \cdot(g^{-s})^{z_{\rho^*(i)}}(g^{-saM^*_{i,1}} \cdot g^{-sa^2M^*_{i,2}} \cdots g^{-sa^{n^*}M^*_{i,n}})\\
&= g^{aM^*_{i,2}l'_2} g^{aM^*_{i,3}l'_3} \cdots g^{aM^*_{i,n^*}l'_{n^*}} \cdot (g^{-s})^{z_{\rho^*(i)}}\\
&= (\prod_{j=2}^{n^*}(g^a)^{M^*_{i,j}l'_j})(g^s)^{-z_{\rho^*(i)}}
\end{aligned}$$

Hence, challenge ciphertext can be obtained, and it can be expressed as $CT^*_\beta = (C_0, C_1, C_2, \{C'_i\}_{i=1,\cdots,l})$, then it will be sent to the adversary $\mathcal{A}$.

*Phase*2: The steps are the same as *Phase*1.

Guess: $\mathcal{A}$ final outputs β' as a guess for β. If $\beta = \beta'$, then $\mathcal{C}$ outputs 1 indicates $TQ = e(g, g)^{sa^{q+1}}$; Otherwise it outputs 0 indicates that TQ is the random element in G_1. Therefore, the advantage of $\mathcal{C}$ in the q-BDHE game is as follows.

$$\begin{aligned}
&Adv_C^{q-BDHE}(k)\\
&\quad = |Pr[P(g, g^s, g^a, \cdots, g^{a^q}, g^{a^{q+2}}, \cdots, g^{a^{2q}},\\
&\qquad TQ = e(g, g)^{sa^{q+1}}) = 1] - Pr[P(g, g^s, g^a,\\
&\qquad \cdots, g^{a^q}, g^{a^{q+2}}, \cdots, g^{a^{2q}}, TQ = R) = 1]|\\
&\quad = \varepsilon(k) + \frac{1}{2}(1 - \varepsilon(k)) - \frac{1}{2} = \frac{\varepsilon(k)}{2}
\end{aligned}$$

Hence, the theorem 1 can be proved.

C. OTHER SECURITY

1) THE SECURITY OF SEARCH TOKEN

In the processing of search token generation, the elements D_1, D_2 are the components of the private key SK of the DU, which are confidential and are known to DU only. The keyword w' is first hashed, and then it is encrypted at the exponent position. Thus, if the attacker wants to get the information of the keyword w', it needs to calculate firstly the discrete logarithm difficulty problem, and then it needs to calculate a collision of the hash function; after that, it can get the information of the keyword. Since discrete logarithms and hash collision are known to be difficult problems, the possibility of solving them in polynomial time is negligible, so we can determine that our search token is secure.

2) AGAINST COLLUSION ATTACKS

The collusion attack means that two or more data users cannot decrypt the ciphertext separately, and if they collude with each other to combine the private keys, the decryption will succeed. In our design, each user's private key is associated with a globally unique identity identifier. At the same time, the effective access time peroid of different users' private keys is different, so the private keys of different users cannot

TABLE 3. Functional comparison of DCDV with the main references.

Scheme	Method	Access Policy	Time specified	Multi-file encryption	Searchable	Verification
[27]	KP-ABE	LSSS	✓ [1]	× [2]	×	×
[33]	CP-ABE	LSSS	×	×	×	✓
[35]	CP-ABE	Access Tree	×	✓	✓	✓
[42]	CP-ABE	LSSS	✓	×	×	×
DCDV	CP-ABE	LSSS	✓	✓	✓	✓

[1] There is corresponding function in the scheme.
[2] There is no corresponding function in the scheme.

TABLE 4. Computational complexity comparison of TAKPS with the main references.

Component	[27]	[33]	[35]	[42]	DCDV
Public parameters	$(2+t)e_G+e_T$	e_G+e_T	$(2+u)e_G+e_T$	$(1+t)e_G+e_T$	e_G+e_T
Private key	$(3+t+3n')e_G$ $+(3+t)c_M$	$(3+n')e_G$ $+(1+n')c_M$	$(n'+1)e_G$	$(t+n'+4)e_G$ $+(3+t)c_M$	$(3+t+n')e_G$ $+(2+t)c_M$
Encryption	$(2+l')e_G+2e_T+$ $(t+3)c_M$	$(4+4l')e_G+(2+2l')e_T+$ $(1+2l')c_M+c_E$	$(5+l')e_G+e_T$	$(2+2l')e_G+2e_T+$ $(t+l')c_M$	$(1+2l')e_G+2e_T+$ $(t+l')c_M$
Verification	$-$	$(6+2l')e_T+4c_E$	$4c_E$	$-$	$c_E+\log_2 N$
Decryption	$te_G+2ne_T+6c_E$	$2ne_T+2c_E$	$-$	$te_G+(n+7)e_T+7c_E$	$te_G+(n+5)e_T+6c_E$

u: The number of attribute in the system; n': The number of data user attribute; l': The number of attribute in the access policy; n: The number of attributes that satisfy the access policy; N: The number of ciphertext documents; t: The depth of hierarchal time tree; e_G: Exponential operation in group G; e_T: Exponential operation in group G_1; c_M: Point multiplication operation in group G; c_E: Pairing operation;

be successfully combined. Therefore, our scheme can avoid multi-user collusion attacks.

3) AGAINST MALICIOUS SERVER ATTACKS

Our proposal is to against malicious server attacks and the integrity of the data in the scheme can be verified. Each data file of the same keyword of each data owner is stored in the form of a Merkle verification tree, and then the data user can verify the data file returned by the cloud server. If a malicious cloud server wants to modify the data file after calculating the verification root, so as to hide the fact that the file is modified, it must find multiple hash collisions to reconstruct the data verification tree, which is not feasible in polynomial time. Therefore, our proposal guarantees the integrity of the data.

VI. PERFORMANCE ANALYSIS

A. FUNCTIONAL COMPARISON

In order to be applicable to specific cloud computing industrial applications, we first functionally compare our DCDV scheme with the literature [27], [33], [35], [42]. As shown in Table 3, we can clearly see that our scheme has a more comprehensive function than the comparative scheme, which can support the verification of search results and the time control of data access.

B. EFFICIENCY COMPARISON

To demonstrate the practicality of the DCDV scheme, we compare the computational complexity of each component in the DCDV scheme with the literature [27], [33], [35], [42]. The results are shown in Table 4.

In order to evaluate the computational overhead of the DCDV scheme, we simulate the computation time of each algorithm under Windows X64, Inter Core i5 CPU, 3.20 GHz and 4.00 GBRAM, and calculate runtime of each algorithm using the cryptography library PBC [43] in different numbers of attribute, different depths of hierarchical time trees, including private key generation algorithms, data encryption algorithms, data verification algorithms, data decryption algorithms. In contrast to the literature [27], [33], [35], [42], we can obtain an efficiency comparison figure for each algorithm.

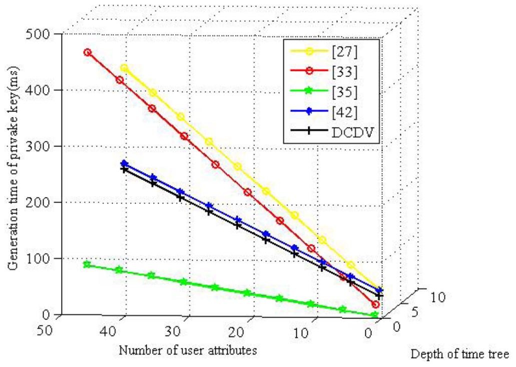


FIGURE 6. The generation time of private key.

Fig.6 shows the comparison between the private key generation algorithm of our scheme and other schemes in terms of time consumption in different numbers of attributes, different depths of hierarchical time tree. Obviously, after joining specified time period for access control, the private key generation time of our scheme is linear with the number of attributes and the depth of hierarchical time tree, and has less computational overhead than the literature [27], [42] that contain time in the private key generation. It can be concluded from Fig.7 that the ciphertext generation time of the DCDV scheme is linear with the number of attributes and the depth of hierarchical

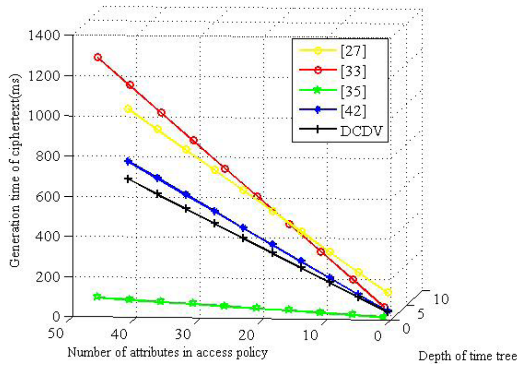


FIGURE 7. The generation time of ciphertext.

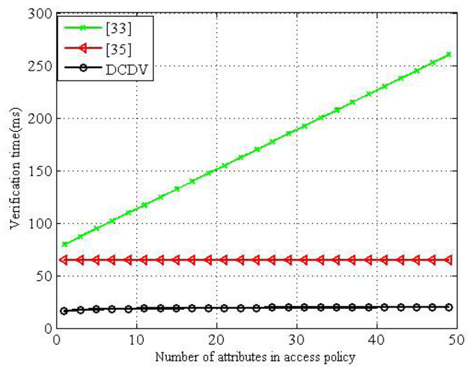


FIGURE 8. The runtime of data verification.

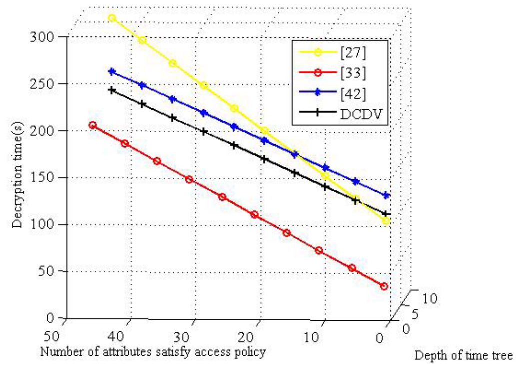


FIGURE 9. The runtime of decryption.

time tree. Compared with other literature [27], [42] containing ciphertext with time settings, and compared with the literature [33] containing only attributes in the ciphertext, the efficiency of the DCDV scheme is significantly improved. Fig.8 shows the comparison between the data verification algorithm of our DCDV scheme and the literature [33], [35] in terms of time consumption. It can be seen from the figure that the data verification time in our scheme has nothing to do with the attribute. In addition, the gradual increase in the number of ciphertext files we give in the experiment does not impose a burden on the verification time, and the runtime required for data verification in our scheme is relatively less than the literature [33], [35]. When comparing the decryption runtime with the literature [27], [33], [42], we can see that even if specified time period is added for access control, the decryption time required by our DCDV scheme is less than that of other schemes, and the result is shown in Fig.9.

VII. CONCLUSION

In order to solve the problem of private data leakage caused by unlimited access of authorized data users in cloud computing applications and the possibility that data in remote clouds may be deleted or modified, we propose a time and attribute based dual access control and data integrity verifiable scheme in cloud computing applications. The DCDV scheme constructs a hierarchical time tree, and sets a decryptable time period and an access time period in the encrypted data and data user private key separately, so that the data user can perform the decryption calculation only the attribute set satisfies the specified access policy of data owner and the effective access time period in the user private key completely cover the decryptable time period set by the data owner, which automatically realizes the automatic destruction after the expiration of the private key of the data owner, and solves the privacy data leakage problem caused by the private key leakage; afterward, the Merkle tree and inverted index technology are used to construct data verification tree, which can perform fast integrity verification on the ciphertext data returned by the cloud server. Finally, security analysis and efficiency analysis show that the DCDV scheme can be applied to the Internet of Things, electronic health records, and data publish-subscribe services and other industry applications based on cloud computing. In the future, we will improve our approach to research user tractable attribute-based encryption scheme, which enables attribute authority to revoke private keys of data users who disclose private keys to other users or enterprises for financial gain to deprive the right to access data.

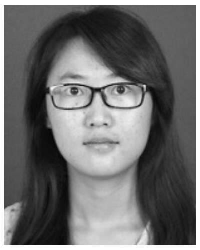
REFERENCES

[1] B. Hayes, "Cloud computing," *Commun. ACM*, vol. 51, no. 7, pp. 9–11, Jul. 2008.
[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, Apr. 2010. [Online]. Available: http://doi.acm.org/10.1145/1721654.1721672
[3] M. P. Rad, A. S. Badashian, G. Meydanipour, M. A. Delcheh, M. Alipour, and H. Afzali, "A survey of cloud platforms and their future," in *Proc. Int. Conf. Comput. Sci. Appl. (ICCSA)*. Berlin, Germany: Springer-Verlag, 2009, pp. 788–796. doi: 10.1007/978-3-642-02454-2_61.
[4] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Gener. Comput. Syst.*, vol. 25, no. 6, pp. 599–616, 2009. doi: 10.1016/j.future.2008.12.001.
[5] D. Giusto, A. Iera, G. Morabito, and L. Atzori, "The Internet Things," in *Proc. 20th Tyrrhenian Workshop Digit. Commun.* New York, NY, USA: Springer, 2010. doi: 10.1007/978-1-4419-1674-7.
[6] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. 1st Ed. MCC Workshop Mobile Cloud Comput. (MCC)*, New York, NY, USA, 2012, pp. 13–16. [Online]. Available: http://doi.acm.org/10.1145/2342509.2342513
[7] G. Xiong, T. Ji, X. Zhang, F. Zhu, and W. Liu, "Cloud operating system for industrial application," in *Proc. IEEE Int. Conf. Service Oper. Logistics, Inform. (SOLI)*, Nov. 2015, pp. 43–48.
[8] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Proc. 24th Annu. Int. Conf. Theory Appl. Cryptograph. Techn. (EUROCRYPT)*. Berlin, Germany: Springer-Verlag, 2005, pp. 457–473. doi: 10.1007/11426639_27.

[9] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Advances in Cryptology—EUROCRYPT*, C. Cachin and J. L. Camenisch, Eds. Berlin, Germany: Springer, 2004, pp. 506–522.
[10] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," *SIAM J. Comput.*, vol. 32, no. 3, pp. 586–615, Mar. 2003. doi: 10.1137/S0097539701398521.
[11] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symp. Secur. Privacy (SP)*, Washington, DC, USA, 2007, pp. 321–334. doi: 10.1109/SP.2007.11.
[12] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Proc. 14th Int. Workshop Public Key Cryptogr. (PKC)*, Berlin, Germany: Springer-Verlag, 2011, pp. 53–70. [Online]. Available: http://dl.acm.org/citation.cfm?id=1964658.1964664
[13] C. Laicheng, L. Yufei, D. Xiaoye, and G. Xian, "User privacy-preserving cloud storage scheme on CP-ABE," *J. Tsinghua Univ. (Sci. Technol.)*, vol. 58, no. 2, p. 150, 2018. [Online]. Available: http://jst.tsinghuajournals.com/EN/abstract/article_152222.shtml
[14] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. 13th ACM Conf. Comput. Commun. Secur. (CCS)*, New York, NY, USA, 2006, pp. 89–98. [Online]. Available: http://doi.acm.org/10.1145/1180405.1180418
[15] J. Kim, W. Susilo, F. Guo, M. H. Au, and S. Nepal, "An efficient KP-ABE with short ciphertexts in prime ordergroups under standard assumption," in *Proc. ACM Asia Conf. Comput. Commun. Secur. (ASIA CCS)*, New York, NY, USA, 2017, pp. 823–834. [Online]. Available: http://doi.acm.org/10.1145/3052973.3053003
[16] F. L. Ţiplea, C. C. Drağan, and A.-M. Nica, "Key-policy attribute-based encryption from bilinear maps," in *Proc. Int. Conf. Inf. Technol. Commun.*, Oct. 2017, pp. 28–42.
[17] Y. Zhou, J. Liu, H. Deng, B. Qin, and L. Zhang, "Non-interactive revocable identity-based access control over e-healthcare records," in *Information Security Practice and Experience*, J. Lopez and Y. Wu, Eds. Cham, Switzerland: Springer, 2015, pp. 485–498.
[18] H. Yan, J. Li, X. Li, G. Zhao, S.-L. Lee, and J. Shen, "Secure access control of E-health system with attribute-based encryption," *Intell. Automat. Soft Comput.*, vol. 22, pp. 345–352, Feb. 2016.
[19] M. H. Au, T. H. Yuen, J. K. Liu, W. Susilo, X. Huang, Y. Xiang, and Z. L. Jiang, "A general framework for secure sharing of personal health records in cloud system," *J. Comput. Syst. Sci.*, vol. 90, pp. 46–62, Dec. 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0022000017300296
[20] P. Junod and A. Karlov, "An efficient public-key attribute-based broadcast encryption scheme allowing arbitrary access policies," in *Proc. 10th Annu. ACM Workshop Digit. Rights Manage. (DRM)*, New York, NY, USA, 2010, pp. 13–24. [Online]. Available: http://doi.acm.org/10.1145/1866870.1866875
[21] M. Ion, G. Russello, and B. Crispo, "Design and implementation of a confidentiality and access control solution for publish/subscribe systems," *Comput. Netw.*, vol. 56, no. 7, pp. 2014–2037, May 2012. doi: 10.1016/j.comnet.2012.02.013.
[22] Y. Zhao and J. Wu, "Building a reliable and high-performance content-based publish/subscribe system," *J. Parallel Distrib. Comput.*, vol. 73, no. 4, pp. 371–382, Apr. 2013. doi: 10.1016/j.jpdc.2012.12.014.
[23] X. Yao, Z. Chen, and Y. Tian, "A lightweight attribute-based encryption scheme for the Internet of Things," *Future Gener. Comput. Syst.*, vol. 49, pp. 104–112, Aug. 2015. doi: 10.1016/j.future.2014.10.010.
[24] M. Rasori, P. Perazzo, and G. Dini, "ABE-cities: An attribute-based encryption system for smart cities," in *Proc. IEEE Int. Conf. Smart Comput. (SMARTCOMP)*, Jun. 2018, pp. 65–72.
[25] B. Tang, Z. Chen, G. Hefferman, S. Pei, T. Wei, H. He, and Q. Yang, "Incorporating intelligence in fog computing for big data analysis in smart cities," *IEEE Trans. Ind. Informat.*, vol. 13, no. 5, pp. 2140–2150, Oct. 2017.
[26] Y. Yang and M. Ma, "Conjunctive keyword search with designated tester and timing enabled proxy re-encryption function for e-health clouds," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 4, pp. 746–759, Apr. 2016.
[27] S. Ma, J. Lai, R. H. Deng, and X. Ding, "Adaptable key-policy attribute-based encryption with time interval," *Soft Comput.*, vol. 21, pp. 6191–6200, Oct. 2016.
[28] M. H. A. Ameri, M. Delavar, J. Mohajeri, and M. Salmasizadeh, "A key-policy attribute-based temporary keyword search scheme for secure cloud storage," *IEEE Trans. Cloud Comput.*, to be published.
[29] J. Ning, Z. Cao, X. Dong, H. Ma, L. Wei, and K. Liang, "Auditable σ-time outsourced attribute-based encryption for access control in cloud computing," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 1, pp. 94–105, Jan. 2018.
[30] J. Liu, T. Jager, S. A. Kakvi, and B. Warinschi, "How to build time-lock encryption," *Des., Codes Cryptogr.*, vol. 86, no. 11, pp. 2549–2586, Nov. 2018. doi: 10.1007/s10623-018-0461-x.
[31] Y. Yu, M. H. Au, G. Ateniese, X. Huang, W. Susilo, Y. Dai, and G. Min, "Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 4, pp. 767–778, Apr. 2017.
[32] Y. Li, Y. Yu, G. Min, W. Susilo, J. Ni, and K.-K. R. Choo, "Fuzzy identity-based data integrity auditing for reliable cloud storage systems," *IEEE Trans. Depend. Sec. Comput.*, vol. 16, no. 1, pp. 72–83, Jan./Feb. 2019.
[33] D. Li, J. Chen, J. Liu, Q. Wu, and W. Liu, "Efficient CCA2 secure revocable multi-authority large-universe attribute-based encryption," in *Cyberspace Safety and Security*, S. Wen, W. Wu, and A. Castiglione, Eds. Cham, Switzerland: Springer, 2017, pp. 103–118.
[34] H. Giang Do and W. K. Ng, "Blockchain-based system for secure data storage with private keyword search," in *Proc. IEEE World Congr. Services (SERVICES)*, Jun. 2017, pp. 90–93.
[35] Z. Chen, F. Zhang, P. Zhang, J. K. Liu, J. Huang, H. Zhao, and J. Shen, "Verifiable keyword search for secure big data-based mobile healthcare networks with fine-grained authorization control," *Future Gener. Comput. Syst.*, vol. 87, pp. 712–724, Oct. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X17300730
[36] M. T. Goodrich and M. Mitzenmacher, "Invertible bloom lookup tables," in *Proc. 49th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Sep. 2011, pp. 792–799.
[37] R. C. Merkle, "Protocols for public key cryptosystems," in *Proc. IEEE Symp. Secur. Privacy*, Apr. 1980, pp. 122–134.
[38] D. Boneh, X. Boyen, and E.-J. Goh, "Hierarchical identity based encryption with constant size ciphertext," in *Proc. 24th Annu. Int. Conf. Theory Appl. Cryptograph. Techn. (EUROCRYPT)*. Berlin, Germany: Springer-Verlag, 2005, pp. 440–456. doi: 10.1007/11426639_26.
[39] D. E. Knuth, *The Art of Computer Programming, Volume 3: Sorting and Searching* (Series in Computer Science and Information Processing), vol. 17. Boston, MA, USA: Addison-Wesley, 1998, p. 324.
[40] W. Guo, X. Dong, Z. Cao, and J. Shen, "Efficient attribute-based searchable encryption on cloud storage," *J. Phys., Conf. Ser.*, vol. 1087, Sep. 2018, Art. no. 052001.
[41] S. Belguith, N. Kaaniche, M. Laurent, A. Jemai, and R. Attia, "PHOABE: Securely outsourcing multi-authority attribute based encryption with policy hidden for cloud assisted iot," *Comput. Netw.*, vol. 133, pp. 141–156, Mar. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128618300495
[42] J. K. Liu, T. H. Yuen, P. Zhang, and K. Liang, "Time-based direct revocable ciphertext-policy attribute-based encryption with short revocation list," in *Applied Cryptography and Network Security*, B. Preneel and F. Vercauteren, Eds. Cham, Switzerland: Springer, 2018, pp. 516–534.
[43] N. P. Smart, "Access control using pairing based cryptography," in *Proc. RSA Conf. Cryptographers Track (CT-RSA)*. Berlin, Germany: Springer-Verlag, 2003, pp. 111–121. [Online]. Available: http://dl.acm.org/citation.cfm?id=1767011.1767023

QIAN ZHANG received the B.S. and M.S. degrees from the School of Science, Xi'an University of Technology, Xi'an, China, in 2016 and 2019, respectively, where she is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering. Her research interests include information security and cryptography.

SHANGPING WANG received the B.S. degree in mathematics from the Xi'an University of Technology, Xi'an, China, in 1982, the M.S. degree in applied mathematics from Xi'an Jiaotong University, Xi'an, in 1989, and the Ph.D. degree in cryptology from Xidian University, Xi'an, in 2003. He is currently a Professor with the Xi'an University of Technology. His current research interests are cryptography and information security.



DUO ZHANG received the B.S. and M.S. degrees from the School of Science, Xi'an University of Technology, Xi'an, China, in 2014 and 2017, respectively, where she is currently pursuing the Ph.D. degree with the School of Automation and Information Engineering. Her research interests include information security and modern cryptography.

JIFANG WANG received the B.S. degree in mathematics and the M.S. degree in applied mathematics from the Xi'an University of Technology, Xi'an, China, in 2009 and 2012, respectively, where she is currently pursuing the Ph.D. degree. Her research interests include cryptography, information security, and blockchain technology.

YALING ZHANG received the B.S. degree in computer science from Northwest University, Xi'an, China, in 1988, and the M.S. degree in computer science and the Ph.D. degree in mechanism electron engineering from the Xi'an University of Technology, Xi'an, in 2001 and 2008, respectively, where she is currently a Professor. Her current research interests include cryptography and network security.