A Password-Based Access Control Framework For Time-Sequence Aware Media Cloudization

Haiyan Wang, Penghui (Owen) Liu, Xiaoxiong Zhong, Weizhe Zhang

Abstract—The time sequence-based outsourcing makes new demands for related access control continue to grow increasingly in cloud computing. In this paper, we propose a practical password-based access control framework for such media cloudization relying on content control based on the time-sequence attribute, which is designed over prime-order groups. First, the scheme supports multi-keyword search simultaneously in any monotonic boolean formulas, and enables media owner to control content encryption key for different time-periods using an updatable password; Second, the scheme supports the key self-retrievability of content encryption key, which is more suitable for the cloud-based media applications with massive users. Then, we show that the proposed scheme is provably secure in the standard model. Finally, the detailed result of performance evaluation shows the proposed scheme is efficient and practical for cloud-based media applications.

Index Terms—Access control, password, time-sequence, content protection, searchable encryption, media cloudization.

1 Introduction

A class of important media applications based on time-sequence can only be accessed by authorized customers. For example, the digital video broadcasting (DVB) business in the cloud needs a content access control solution based on time-sequence in order to produce and stream the high value video content in a protection way, in which the feature that the high value video content produced or streamed during what time shall not be viewed by which class of people is a must. Besides, cloud-based chatroom or web conference system, the conference records can only be viewed by the authorized persons. With the continuous development of data sharing technology, cloud computing has become one of the most practical and efficient network computing models today [1]. Thus, large amounts of data (content) are outsourced to the cloud server. However, it directly makes content vendors lose control of their data. Meantime, the public cloud server cannot be assured to be fully trusted, the concern on both content security and user privacy of such applications arises inevitably.

One approach to solving the problem is to directly encrypt the sensitive content before outsoucing them to the public cloud, which can achieve content confidentiality for unauthorized users [2]. Searchable Encryption (SE) is a mainstream access control technology based on data encryption [3], [4], and it allows users to search for specific information (e.g., based on keywords) in lots of encrypted data directly, in which the cloud server cannot learn any sensitive information on the clear data. SE technology is divided into symmetric encryption (SSE) and public key encryption with keyword search (PEKS). SSE only allows the user having the secret symmetric key to encrypt the keywords and generate the keyword-related tokens, and PEKS allows anyone to encrypt the keywords using the public key of data user, in which the user having the private key can perform search on the encrypted data.

Many access control schemes supporting SE have been proposed for different scenarios over the past few decades. However, to the best of our knowlege, the research on access control based on time-sequence supporting SE over outsourced encrypted content for the cloud-based media applications is relatively inadequate (e.g., high computation time, high storage cost). In this paper, we propose an efficient password-based access control framework supporting expressive PEKS for media cloudization (PAC-MC). Our main contributions can be briefly summarized as follows:

1. The proposed scheme is the first one allowing media service provider to outsource the confidential content to the cloud, controlling content encryption key for different time periods by an updatable password. Meantime, the sevice provider can determine users' privilege of access outsourced data by the time-sequence attribute of content.
2. A new content encryption key generation scheme is proposed, in which users can self-retrieve content encryption key without frequently communicating with key management server.
3. A new PEKS built over the prime-order groups is proposed for cloud-based media applications, which enables the data user to perform multi-keyword search more efficiently over the outsourced content.
4. Security definition based on the standard model is formalized for our scheme, and then we formally prove it to be selectively secure under this model. The performance evaluation shows that our scheme is very efficient and practical.

The remainder of this paper is organized as follows: In Sections 2 and 3, related work and the preliminaries of our scheme are introduced, respectively. In Section 4, we present

- *H. Wang and P. Liu contributed equally to this work.*
- *H. Wang, P. Liu, X. Zhong, and W. Zhang are with the Department of New Networks, Peng Cheng Laboratory, Shenzhen, 518000, China.*
- *This work was supported by the Major Key Project of PCL (Nos. PCL2022A03, PCL2021A02, PCL2021A09), and Guangxi Natural Science Foundation (No. 2022GXNSFBA035650).*

the concrete construction of PAC-MC, including system architecture, security definitions and security proof. In Section 5, we briefly give a performance evaluation related to the scheme, and finally the conclusion is given in Section 6.

2 Related work

Most of the existing works on access control in modern cloud-based applications are designed based on a centralized architecture [5]. Moreover, in the cloud-based applications with access control supporting encrypted data searchability, the keyword-based SE is a mainstream technology.

Searchable symmetric encryption schemes (SSE). In 2000, Song *et al.* [1] firstly proposed the practical security concept of SSE, at the same time presented several SSE primitive constructions. In 2003, Goh *et al.* [6] introduced a bloom filter-based SSE scheme. Later, Chang *et al.* [7] also developed a scheme based on per-file index with higher efficiency than the one proposed by Goh at al., Regrettably, both two schemes cannot prevent the adaptive adversary from generating illegal queries by using the results of previous queries. Next, Curtmola *et al.* [8] introduced a new scheme using per-keyword technique, this approach is more efficient than the previous ones. After then, lots of solutions [9], [10], [11], [12], [13], [14] supporting various levels of security and efficiency have been proposed, most solutions are limited to single keyword search or single user search. Obviously, the design of single-keyword search and single-user search cannot satisfy modern cloud-based applications, which require more complex search. In addition, it is not difficult to see that all access controls based on searchable symmetric encryption scheme uses single key to encrypt and decrypt the shared data, the data owner is required to share a secret key used to generate trapdoors with other authorized users. Inevitably, this will result the risk that the secret-key could be easily leaked to the unauthorized user.

Public key encryption with keyword search (PEKS). In 2004, Boneh *et al.* [3] first proposed the concept of PEKS, developed a provably secure public key SE scheme using anonymous identity-based encryption technique. In 2007, Bellare *et al.* [15] developed a similar searchable public key encryption scheme, where a lot of deterministic searchable tags are defined, the server can sort these tags and match them within a minimum logarithmic time. Compared with previous ones, it is more efficient. However, this scheme only supports equality search, it is difficult to handle duplicate attribute values associated with the matched records. Moreover, some different records with duplicate attribute values will end with the same ciphertext, thus the appearance frequency of plaintext can be tracked statistically based on relevant attribute values. Subsequent works [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28] improved the search capability, support multiple, conjunctive or disjunctive keywords search. However, most works only focus on single user (data owner) search, or only aims to secure against the offline keyword dictionary guessing attacks, and the search number of keywords allowed in the searchable schemes are predefined in the setup phase while running the system. These don't fully satisfy the cloud-based applications that operate relying on the time-sequence attribute of content with multi-user search using any number of keywords. Meantime, these schemes designed over the composite-order groups require considerable computational complexity and mainly concerns the offline keyword dictionary guessing attacks.

Content access control based on time-sequence. The existing media access control schemes based on time-sequence attribute of the encrypted content, such as ITU-R Rec.810 [29] and solutions [30], [31], [32], [33], are mainly employed in the security systems of CAS (conditional access system) and DRM (digital rights management) for content encryption key management of streaming media service. Park *et al.* [30]proposed an efficient encryption and key management scheme for layered access control of H.264/Scalable Video Coding based on the hierarchical content access control of H.264 video. The scheme focuses on the feature extraction of low-level picture frame. However, it also cannot support user grouping and content search on the encrypted data for data users. Zhu *et al.* [31] proposed a hierarchical key distribution scheme for the conditional access system in DTV broadcasting, which requires an additional smart card and relies on the transport stream. Unfortunately, these schemes are only suitable for the video encryption for traditional cable broadcasting of digital TV service. Yang *et al.* [32] developed a simplified and secure conditional access system for the interactive TV service in converged network. this scheme [31] requires a smart card and severely relies on the transport stream, and an additional client access agent should be integrated into the device owned by data user. Feng *et al.* [33] proposed a content encryption key scheme for the multicast encryption and DRM (digital rights management), which requires to interact with the license server (content key management server) frequently when the DRM content needs to be decrypted. Moreover, the commnuication overhead will become excessive in order to obtain all encryption keys of the streaming media blocks, due to the decryption key for each content block is obtained by interacting with the license server one by one. In addition, this scheme mainly focuses on key management and access control for the recorded content, and each client (data user) needs to integrate an additional DRM agent as well, is only suitable for IPTV and not designed for cloud-based media application. Overall, the existing schemes designed for the access control do not support content search for users, and are not suitable for the modern cloud-based media applications.

3 Preliminaries and Background

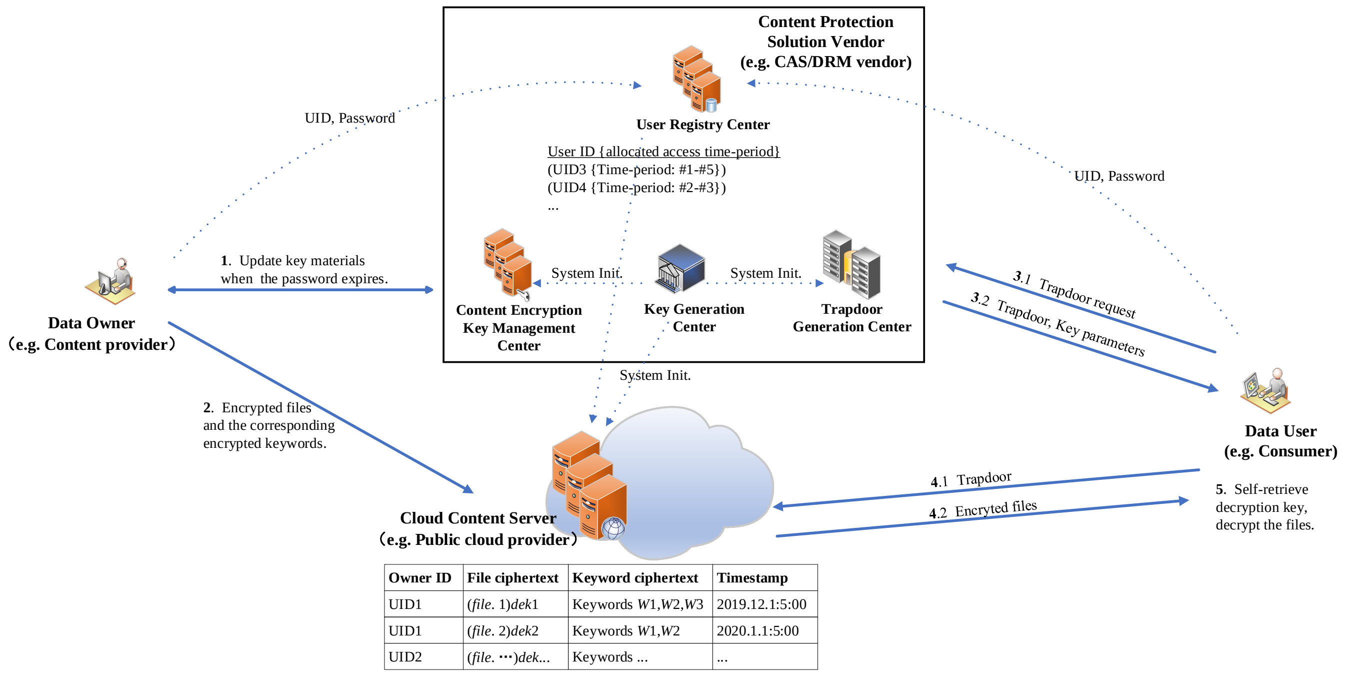
3.1 Bilinear Groups and Pairings

Bilinear Pairings [34]. Let $\mathcal{G}(1^\ell)$ be an algorithm taking a security parameter ℓ as input, it outputs the system parameters: $(p, \mathbb{G}, \mathbb{G}_T, \hat{e}) \leftarrow \mathcal{G}(1^\ell)$ where $\mathbb{G}, \mathbb{G}_T$ are both multiplicative cyclic groups with a prime order $p, \hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a bilinear pairing map function with the following properties:

1) Bilinearly:$\hat{e}(g^a, h^b) = \hat{e}(g, h)^{ab}$ for $g, h \in \mathbb{G}$, $a, b \in \mathbb{Z}_p$.

2) Non-degenerate:$\hat{e}(g, g) \neq 1$.

3) Computability: there exists an efficient algorithm to compute $\hat{e}(g, h)$ for all $g, h \in \mathbb{G}$.


Content Protection Solution Vendor (e.g. CAS/DRM vendor)
User Registry Center
User ID {allocated access time-period}
(UID3 {Time-period: #1-#5})
(UID4 {Time-period: #2-#3})
...
UID, Password
UID, Password
1. Update key materials when the password expires.
Data Owner (e.g. Content provider)
Content Encryption Key Management Center
System Init.
Key Generation Center
System Init.
Trapdoor Generation Center
3.1 Trapdoor request
3.2 Trapdoor, Key parameters
Data User (e.g. Consumer)
System Init.
2. Encrypted files and the corresponding encrypted keywords.
Cloud Content Server (e.g. Public cloud provider)
4.1 Trapdoor
4.2 Encryted files
5. Self-retrieve decryption key, decrypt the files.

Owner ID	File ciphertext	Keyword ciphertext	Timestamp
UID1	*(file. 1)dek*1	Keywords *W*1,*W*2,*W*3	2019.12.1:5:00
UID1	*(file. 2)dek*2	Keywords *W*1,*W*2	2020.1.1:5:00
UID2	*(file.* ⋯*)dek*...	Keywords ...	...

Fig. 1. Architecture of our proposed PAC-MC scheme.

3.2 One-way Hash Binary Tree(OHBT):

Let T be a OHBT [35], given an internal node in T, the value stored on its left child node is generated by hashing the value on its parent node with 0, and the value stored on its right child node is generated by hashing the value on its parent node with 1, i.e., Let $Left_T(x)$ and $Right_T(x)$ denote the left child and right child of node x in a given OHBT, respectively, let $value_T(x)$ be a value associated with node x, then $value_T(Left_T(x)) = HASH(value_T(x)||0)$, and $value_T(Right_T(x)) = HASH(value_T(x)||1)$.

3.3 Complexity Assumptions

Decisional q-2 Assumption [34]. For any probabilistic polynomial-time algorithm $\mathcal{A}$, given the above system parameters $(p, \mathbb{G}, \mathbb{G}_T, \hat{e})$ and following $\mathbb{D}$,

$$\mathbb{D} = (g, g^x, g^y, g^z, g^{xz^2},$$
$$g^{b_i}, g^{xzb_i}, g^{xz/b_i}, g^{x^2zb_i}, g^{y/b_i^2}, g^{y^2/b_i^2}, \forall i \in [q],$$
$$g^{xzb_i/b_j}, g^{yb_i/b_j^2}, g^{xyzb_i/b_j}, g^{(xy)^2 b_i/b_j}, \forall i, j \in [q], i \neq j),$$
$$[q] \overset{\text{def}}{=} \{1, 2, \ldots, q\},$$

where q is an integer. We assume it is impossible for $\mathcal{A}$ to distinguish $(\mathbb{D}, \hat{e}(g, g)^{xyz})$ from $(\mathbb{D}, Z)$, where the random number $Z \in \mathbb{G}_T, x, y, z, b_1, b_2 \ldots, b_q \in \mathbb{Z}_p$ are chosen randomly and independently.

Decisional Bilinear Diffie-Hellman Assumption [25] (DBDH). For any probabilistic polynomial-time algorithm $\mathcal{A}$, given the parameters: $\mathbb{D} = (g, g^x, g^y, g^z)$, we assume it is impossible for $\mathcal{A}$ to distinguish $(\mathbb{D}, \hat{e}(g, g)^{xyz})$ from $(\mathbb{D}, Z)$, where the ramdom number $Z \in \mathbb{G}_T, x, y, z \in \mathbb{Z}_p$ are chosen randomly and independently.

Decisional Diffie-Hellman Assumption [36] (DDH). Assume there are two big prime numbers: p', q', such that $q'|(p' - 1)$, and a group $\mathbb{G}_{q'}$, which is a subgroup of multiplicative group $\mathbb{Z}_{p'}^*$, g' is the generator of $\mathbb{G}_{q'}$. For any probabilistic polynomial-time algorithm $\mathcal{A}$, given the parameters: $\mathbb{D} = (g', g'^x, g'^y)$, we assume it is impossible for $\mathcal{A}$ to distinguish $(\mathbb{D}, g'^{xy})$ from $(\mathbb{D}, Z)$, where $Z \in \mathbb{G}_q$, and $x, y \in \mathbb{Z}_{q'}$, are chosen randomly and independently.

Pseudorandom Function Ensemble [37]. Let $\mathbb{F}$ be a function ensemble with functions in the set $\{0,1\}^n \to \{0,1\}^n$, $\mathbb{H}$ be the function ensemble distributed uniformly over all functions in the set $\{0,1\}^n \to \{0,1\}^n$, For any probabilistic polynomial-time algorithm $\mathcal{A}$, given the ability to query values on a function f', it is impossible for $\mathcal{A}$ to determine whether f' was drawn from $\mathbb{F}$ or from $\mathbb{H}$. Then we define $\mathbb{F}$ is a pseudorandom function ensemble.

3.4 Access Structures and Linear Secret Sharing

Access Structures [34], [38]. Let $\{P_1, \ldots, P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1,\ldots,P_n\}}$ is monotone if $\forall B, C$: if $B \in \mathbb{A}$ and $B \subseteq C$, then $C \subseteq \mathbb{A}$. An access structure (respectively, monotonic access structure) is a collection (respectively, a monotonic collection) $\mathbb{A}$ of non-empty subsets of $\{P_1, \ldots, P_n\}$, i.e.$\mathbb{A} \subseteq 2^{\{P_1,\ldots,P_n\}} \backslash \{\emptyset\}$. The sets in $\mathbb{A}$ are called authorized sets, and the sets not in $\mathbb{A}$ are called unauthorized sets. In this paper, we only focus on the monotone access structures. However, for non-monotone versions, as stated in [38], it is possible to implement the general access structures, where the non-monotone access structures can be described using monotonic access structures with inefficient techniques such as taking the negation of an attribute as a separate attribute.

Linear Secret Sharing schemes (LSSS) [34], [38]. A secret sharing scheme $\prod$ 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 $\mathbb{M}$ with ℓ rows and columns n columns called the share-generating matrix for $\prod$, For

all $i = 1, \ldots, \ell$, the i-th row of $\mathbb{M}$ is labeled by a party $\rho(i)$, (where $\rho(*)$ is a map function from $\{1, \ldots, \ell\}$ to P for labeling). When we consider the column vector $v = (s, r_2, \ldots, r_n)$, where $s \in Z_p$ is the secret to be shared, $r_2, \ldots, r_n \in \mathbb{Z}_p$, are randomly chosen, then $\mathbb{M}v$ is the vector of ℓ shares of the secret s according to $\prod$. The share $(\mathbb{M}v)_i$ belongs to party $\rho(i)$.

As noted in [34] that every LSSS enjoys the linear reconstruction property, which can be defined as follows: Suppose that $\prod$ is an LSSS for access structure $\mathbb{A}$. Let $\mathbb{S}$ be an authorized set chosen from $\mathbb{A}$, and let M_i denote the i-th row of $\mathbb{M}$, define $I \subseteq [\ell]$ as $I = \{i | \rho(i) \in \mathbb{S}\}$. Then there exist constants $\{\omega_i \in \mathbb{Z}_p\}_{i \in I}$ and a vector $w = (1, 0, 0, \ldots, 0)$, such that, for any valid shares $\{v_i\}$ of the secret s according to $\prod$, we have $\sum_{i \in I} \omega_i v_i = s$, and $\sum_{i \in I} \omega_i M_i = w$, where w is in the span of the rows of $\mathbb{M}$ indexed by I and constants $\{\omega_i \in \mathbb{Z}_p\}_{i \in I}$ can be found in the polynomial time in size of the share-generation matrix $\mathbb{M}$. However, for any unauthorized set $\mathbb{S}'$, no such constants $\{\omega_i\}$ satisfying the above definition exist. Moreover, in this case it is true, if $I' = \{i | i \in [\ell] \wedge \rho(i) \in \mathbb{S}'\}$, there exists a vector $\hat{\mathrm{w}}$ such that its first component $\hat{\mathrm{w}}_1$ is any no zero element in $\mathbb{Z}_p$ and $< M_i, \hat{\mathrm{w}} >= 0$ for all $i \in I'$.

4 Password-based access control scheme for media cloudization (PAC-MC)

We first describe the architecture of our scheme, then present the details on algorithm definitions, construction, threat model, as well as the relevant design goals.

4.1 System Architecture

The architecture of our access control system is shown in Fig. 1, which consists of four entities:

Data Owner: The data owner (media content and service provider) is responsible for content production, encryption and outsourcing, determines which users can search the outsourced content.

Content Protection Solution Vendor (CPSV)[1]**:** CPSV consists of four parts: Key generation center (KGC), User registry center (URC), Content encryption key management center (CEKMC) and Trapdoor generation center (TGC). Among which, KGC initializes and publishes the system parameters; URC manages all information on cloud users including access time-period and identity of each user; CEKMC assists data owner to update and manage all content encryption keys for different time-period if required; TGC issues trapdoor for content search for data user.

Data User: The data user shall request a trapdoor from TGC before using search service, if a data user is an authorized user, who will be privileged by data owner to search the outsourced content.

Cloud Content Server (CCS): CCS provides content storage and search services, performs the keyword search operations on behalf of authorized data user.

Assume that all entities have their own unique identity (UID), and all cloud users are authorized by using *separate password-based identity authentication approach* before providing service; The service lifetime of an application consists of many password lifetimes, which is divided into lots of constant time-periods, (The relationship among service lifetime, password lifetime and time-period is depicted in Fig. 2). The data owner uses different keys to encrypt the data outsourced to the public cloud in different time-periods based on time-sequence. This assumption is reasonable, due to considering the security of data and password, it will be better for data owner to change the data encryption key for the data outsourced in different time-period, and the cloud management system shall prompt user to periodically update the password.

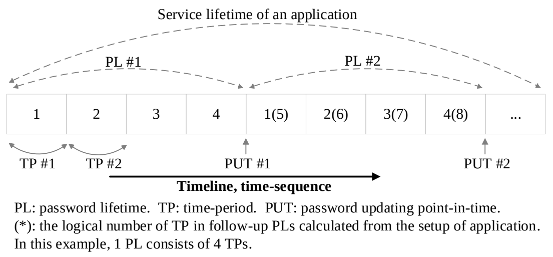

Service lifetime of an application
PL #1
PL #2
1
2
3
4
1(5)
2(6)
3(7)
4(8)
...
TP #1
TP #2
PUT #1
PUT #2
Timeline, time-sequence
PL: password lifetime. TP: time-period. PUT: password updating point-in-time.
(*): the logical number of TP in follow-up PLs calculated from the setup of application.
In this example, 1 PL consists of 4 TPs.

Fig. 2. The relationship among service lifetime, password lifetime and time-period.

We elaborate on the workflow of our scheme. Suppose there is a media application where a data owner is required to deliver some files (content) produced based on the time-sequence to other users by outsourcing the files to the cloud. Firstly, the data owner allocates access time-periods to each authorized user respectively, such as (UID_3, time-period:#1-#5), which means the user UID_3 can access files produced and outsourced in time-period: #1-#5, then the data owner provides the information on access time-period and identity of each authorized user to URC, who further hands the information over to CCS, then with the help of CEKMC, the data owner updates the key materials to generate a symmetric encryption key for file encryption in current time-period.

Next, the data owner extracts a set of keywords for each file, encrypts the files using the symmetric encryption key, whilst encrypts the keywords by using a PEKS to be described later, then uploads the encrypted files and keywords to CCS. To obtain the files of interest, the data user sends a trapdoor request with one keyword LSSS access structure to CPSV, where TGC issues a trapdoor and CEKMC generates a set of key parameters for the data user, then CPSV returns the trapdoor and key parameters to the data user after necessary *separate identity authentication*[2], further, the data user sends the trapdoor and a partial hidden access structure with keyword name only (without keyword value, see Fig. 3) to CCS for file (content) search.

Finally, CCS checks whether the timestamp attached to file satisfies the access time-period belongs to data user, and whether the trapdoor satisfies keyword ciphertext in database while searching files, returns the matched files,

1. Main CPSVs: NAGRA, Verimatrix, Irdeto, Conax, NDS, etc. Its functionality can be deployed in the domain of CPSVs geographically, or in the form of private cloud service of operator.

2. Here we only assume a *separate password-based identity authentication approach* is adopted and ignore the specific mechanism, which is out of the scope of this paper.

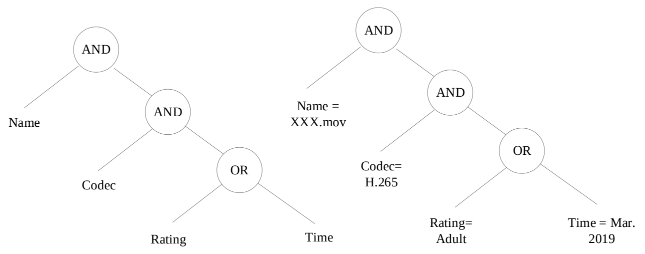


Fig. 3. (a) Partial hidden access structure. (b) Full access structure.

then the data user self-retrieves the content encryption key using key parameters returned by CPSV to decrypt the files.

Our solution aims to achieve the following goals for data access in media outsourcing service:

a. *Time-sequence based outsourced content key management.* The data owner can manage content encryption key for content produced and outsourced based on time-sequence. Moreover, the content encryption key corresponding to different time-period can be conveniently updated by data owner. The data owner can determine users's privilege to access outsourced data based on the allocated access time-periods.
b. *Efficient multi-keyword search on outsourced content.* The data user can efficiently perform multi-keyword search using a boolean formula expression consisting of "AND" and "OR" gates on the encrypted content.
c. *Password-driven access control.* Both the content encryption key and trapdoor can be generated under the control of password, more importantly, where the password can be updated as per the requirement.
d. *Encryption key self-retrievability/recoverability.* For cloud-based media application with massive users, the data user (consumer) can self-retrieve the encryption keys after obtaining the initial key materials, to a certain extent, the data user can recover temporarily lost encryption key that belongs to her or him without frequently communicating with key management server.
e. *Content keyword confidentiality.* Any polynomial-time adversary (such as the "curious" cloud server and outside attacker) cannot acquire the information on keyword values directly from the keyword ciphertext without its related trapdoor, the trapdoor shall also leak nothing on keyword values to any outside attacker without system private key or the cloud server's private key and the user's password.
f. *Collusion attack resistance.* The colluded data users cannot acquire the content encryption key corresponding to the access time-periods that not belongs to them via collusion attack. Additionally, any authorized user cannot obtain the content encryption key corresponding to the access time-periods that not belongs to her or him using the key matrials corresponding to two or more disjoint access time-periods that allocated to her or him.

4.2 System Algorithms and Security Model

In this section, we will introduce system algorithms and threat model, respectively.

4.2.1 System Algorithms

Our password-based access control system mainly consists of the following seven polynomial-time algorithms:

- Setup$(1^\ell) \rightarrow (pp, msk)$**:** For system initialization, CPSV runs the setup algorithm which takes a security parameter ℓ as input, outputs the system public parameter pp, and the master private parameter msk.
- KeyGen$(pp, msk, UID_i) \rightarrow (pk_i, sk_i, FHS_i, BHS_i)$**:** To initialize the public/private key pair and secret keys for CCS and cloud users, CPSV runs the key generation algorithm which takes the system public parameter pp, the master private parameter msk, and CCS or users' UID_i as input, outputs two hashing seeds (FHS_i, BHS_i) for each user, and a public/private key pair (pk_i, sk_i) for CCS and each user, respectively.
- KeyParUpdate$(pp, msk, pw_o, UID_{CPSV}, UID_o) \rightarrow (k_{seed},\ dek)$: With the help of CPSV, the data owner runs the key materials updating algorithm which takes the system public parameter pp, the master private parameter msk, the data owner's password pw_0, the CPSV's UID_{CPSV}, and the data owner's UID_o as input, outputs a updated seed k_{seed} of the key materials and a symmetric encryption key dek for content outsourced in the new time periods. Note that, this algorithm is not always running but executed by the data owner together with CPSV when password expires.
- Encrypt$(pp, file, dek, \mathbb{W}, sk_o) \rightarrow ((file)_{dek}, CT_{\mathbb{W}})$**:** To encrypt the file and its keywords for outsourcing, the data owner runs the encryption algorithm which takes the system public parameter pp, a file $file$, the encryption key dek for current time-period, the keyword set $\mathbb{W}$ of the file $file$, and the data owner's private key sk_o as input, outputs the encrypted file (file $)_{dek}$ and its keyword ciphertexts $CT_{\mathbb{W}}$.
- Trapdoor$(pp, msk, pk_s, \mathbb{U}, pw_u, UID_s, UID_u, UID_o,$ $LAS_{\mathbb{M},\rho,\mathbb{W}} = (\mathbb{M}, \rho, \{\mathbb{W}_{\rho(i)}\})) \rightarrow (Td_{\mathbb{W}}, kp)$**:** To generate a search trapdoor, CPSV runs the trapdoor generation algorithm which takes the system public parameter pp, the master private parameter msk, the public key pk_s of CCS, a collection of authorized users $\mathbb{U}$, the data user's password pw_u, CCS's UID_s, the data owner's UID_o, the data user's UID_u, and a keyword LSSS access structure $LAS_{\mathbb{M},\rho,\mathbb{W}} = (\mathbb{M}, \rho, \{\mathbb{W}_{\rho(i)}\})$ as input, outputs a trapdoor $Td_{\mathbb{W}}$ and key parameter kp.
- Check$(pp, sk_s, pk_o, pk_u, CT_{\mathbb{W}}, Td_{\mathbb{W}}) \rightarrow$ (**"Yes" and** $(file)_{dek}/$**"**$\perp$**")**: To search the file, CCS runs the check algorithm which takes the system public parameter pp, the private key sk_s of CCS, the data owner's public key pk_o, the data user's public key pk_u, the keyword ciphertexts $CT_{\mathbb{W}}$, and the trapdoor $Td_{\mathbb{W}}$ for the LSSS access structure $LAS_{\mathbb{M},\rho,\mathbb{W}}$ as input, outputs "Yes" and $(file)_{dek}$ if the keyword ciphertexts satisfy the trapdoor $Td_{\mathbb{W}}$, otherwise outputs "$\perp$".
- Decrypt$((file)_{dek}, kp) \rightarrow file$**:** To decrypt the files, the data user runs the decryption algorithm which takes the encrypted file $(file)_{dek}$ and key parameter kp as input, outputs the decrypted file $file$.

Correctness: A password-based access control scheme $\prod$ is correct, which means that, for any keyword-set $\mathbb{W}$ and a LSSS access structure $(\mathbb{M}, \rho, \{\mathbb{W}_{\rho(i)}\})$ such that $\mathbb{M}(\mathbb{W}) = 1$, if (pp, msk) $\leftarrow$Setup(1^{ℓ}), $(pk_i,\ sk_i,\ FHS_i,\ BHS_i)$ $\leftarrow$KeyGen$(pp,\ msk,\ UID_i)$, $(k_{seed},\ dek)$ $\leftarrow$KeyParUpdate$(pp,\ msk,\ pw_o,\ UID_{CPSV},\ UID_o)$, $((file)_{dek},\ CT_{\mathbb{W}})$ $\leftarrow$Encrypt$(pp,\ file,\ dek, \mathbb{W},\ sk_o)$, $(Td_{\mathbb{W}}, kp)$ $\leftarrow$Trapdoor$(pp,\ msk, pk_s,\ \mathbb{U},\ pw_u,\ UID_s,\ UID_u,\ UID_o,\ LAS_{\mathbb{M},\rho,\mathbb{W}} = (\mathbb{M},\ \rho,\ \{\mathbb{W}_{\rho(i)}\}))$, then Check$(pp, sk_s,\ pk_o,\ pk_u,\ CT_{\mathbb{W}},\ Td_{\mathbb{W}})$, outputs "Yes", and $file$ $\leftarrow$Decrypt $((file)_{dek}, kp)$ if the verification passes, otherwise outputs "$\perp$".

4.2.2 Security Model

In our work, the algorithm KeyParUpdate(*) is not always running but executed by data owner with the help of CPSV when the prescribed event occurs, such as password expires. We first define a security model for the sub-process KeyParUpdate(*) under standard model, then define a slightly weaker security model under the chosen keyword-set attacks for the whole PAC-MC scheme. It can be considered as a keyword SE in the sense of selectively semantic security, due to the adversary needs to commit two challenge keyword sets $\mathbb{W}_0^*$ and $\mathbb{W}_1^*$ as the attack target to the challenger at the beginning of the simulation. Thus, the proof is also divided into two parts later.

1) Security model for KeyParUpdate(*): We introduce a formal security model defined by Bellare et al. in [36]. Each of participants is modeled as a set of oracles, the oracle $\Pi_{i,j}^n$ denotes the n-th protocol instance between the participant i and j. In this model the adversary's capability is abstracted as unordered and adaptive query against the predefined oracles. The adversary $\mathcal{A}$ can make different queries to any oracle (it has an endless supply of oracles [26]), such as the Execute(*), Send(*), Reveal(*), Corrupt(*) and Test(*) queries.

Based on the model [14], [21], we define a security model in the sense of semantic security. The challenger $\mathcal{C}$ picks two big prime numbers: p', q', such that $q'|(p'-1)$, a group $\mathbb{G}_{q'}$ with a prime order q', a system secret key $\beta \in \mathbb{Z}_{q'}$, a system generator $g' \in \mathbb{G}_{q'}$, a pseudorandom function ensemble $\mathbb{F}$, and two one-way hash functions $H_0, f : \{0,1\}^* \rightarrow \mathbb{Z}_{q'}$, computes $v = g'^{\beta}$, and chooses a password pw shared by the protocol participants. Then the challenger $\mathcal{C}$ gives the public parameter $(H_0, p', q', \mathbb{G}_{q'}, g', v, f, \mathbb{F}, UID_i)$ to the adversary $\mathcal{A}$ and keeps the password pw by itself. Next, the adversary $\mathcal{A}$ makes the above queries to the challenger $\mathcal{C}$ in a ℓ-bounded polynomial-time and the challenger $\mathcal{C}$ responds to the adversary $\mathcal{A}$ in a manner prescribed by the algorithm KeyParUpdate(*). Finally, the adversary $\mathcal{A}$ makes a Test$(\Pi_{i,j}^n)$ query on a *fresh* oracle, if the adversary $\mathcal{A}$ successfully guesses the value b' of b, then we call the adversary wins the game. The advantage of the adversary $\mathcal{A}$ is defined as

$$Adv_{A,p',q',g'\cdot\mathbb{G}_1,\mathbb{F},\beta,pw}^{KeyParUpdate} \overset{\text{def}}{=} 2Pr\left[b = b'\right] - 1.$$

where $Adv_{A,p',q',g'\cdot\mathbb{G}_1,\mathbb{F},\beta,pw}^{KeyParUpdate}$ is the maximum value that all adversaries may achieve, and $Pr[b = b']$ is the success probability of the adversary $\mathcal{A}$. We call the algorithm KeyParUpdate(*) is secure if $Adv_{A,p',q',g'\cdot\mathbb{G}_1,\mathbb{F},\beta,pw}^{KeyParUpdate} \leq 2 \cdot C' \cdot q_{send}^{s'} + \varepsilon$, in which q_{send} is the number of the Send$(\Pi_{i,j}^n, M)$ queries performed by the adversary $\mathcal{A}$, and ε is a negligible variable for any probabilistic polynomial-time adversary $\mathcal{A}$. Here $C' \cdot q_{send}^{s'}$ limits a constant maximum number of passwords that an adversary $\mathcal{A}$ can guess and exclude via the Send$(\Pi_{i,j}^n, M)$ query in this game, where the password space follows the frequency distribution according to Zipf's law, and C' and s' are the Zipf regression parameters of dataset, considered as Zipf parameters [43].

2) Security model for the whole PAC-MC scheme related with PEKS: the security model defined by Boneh et. al in [3], in terms of the semantic security, and to ensure PAC-MC scheme does not leak any information on the keyword values embedded in ciphertext, we define a selective security model under chosen keyword-set attacks in terms of semantic security. In the game, the adversary $\mathcal{A}$ can acquire a polynomial number of trapdoors for any set of keywords. However, the game ensures it doesn't disclose any information on the keyword values embedded in the challenge ciphertext, if the adversary $\mathcal{A}$ has no any matching trapdoor or system private materials and password.

We introduce a formal security model according to [25], [34]. Two security cases may occur in our scheme are considered as follows:

(1) The first security case is defined as follows: (where the adversary $\mathcal{A}$ is assumed to be a malicious user, or an outside attacker who impersonating other user with $UID \in \mathbb{U}$ to forge the trapdoor with a set of guessed keyword values and try to determine whether a trapdoor matches the given keyword ciphertext.):

- **Setup:** The adversary $\mathcal{A}$ initially declares two distinct challenge keyword sets $\mathbb{W}_0^*$ and $\mathbb{W}_1^*$ of the same size with a file *file* to the challenger $\mathcal{C}$, and sends it to the challenger $\mathcal{C}$. The challenger $\mathcal{C}$ first runs Setup(1^{ℓ}) to obtain the system public parameter pp and the master private parameter msk, and provides the system public parameter pp to the adversary $\mathcal{A}$, and keeps the master private parameter msk by itself. Next, the challenger $\mathcal{C}$ runs KeyGen(pp, msk, UID_i) to generate two hashing seeds (FHS_i, BHS_i) for the user and the public/private key pair (pk_i, sk_i) for CCS and user, respectively, further chooses a password pw_i for each user, respectively, then it provides all public keys pk_i to the adversary $\mathcal{A}$, and keeps the private keys sk_i, the hashing seeds (FHS_i, BHS_i) and the password pw_i by itself. Finally, the challenger $\mathcal{C}$ runs KeyParUpdate$(pp, msk, pw_o, UID_{CPSV}, UID_o)$ to obtain the key materials and a symmetric encryption key dek for file encryption, then it keeps the key materials and the file encryption key dek by itself.
- **Query phase 1:** The adversary $\mathcal{A}$ adaptively issues a polynomial number of queries to the challenger $\mathcal{C}$ for the trapdoors with the keyword LSSS access structures $\{LAS_{\mathbb{M}_j,\rho_j,\mathbb{W}_j} = (\mathbb{M}_j, \rho_j, \{\mathbb{W}_{\rho_j(i)}\})|$ $j \in \{1, 2, \ldots, q_{\mathrm{T}}\}\}$ the challenger $\mathcal{C}$ directly runs Trapdoor$(pp, msk, pk_s, \mathbb{U}, pw_A, UID_S, UID_A, UID_o,$ $LAS_{\mathbb{M}_j,\rho_j,\mathbb{W}_j} = (\mathbb{M}_j, \rho_j, \{\mathbb{W}_{\rho_j(i)}\}))$ to generate the trapdoor $Td_{\mathbb{W}j}$ and key parameter kp_j, and sends the trapdoor $Td_{\mathbb{W}j}$ and key parameter kp_j to the adversary $\mathcal{A}$, where pw_A and UID_A are the impersonated user's password and unique ID,

respectively.

- **Challenge:** The adversary $\mathcal{A}$ submits the declared two distinct challenge keyword sets $\mathbb{W}_0^*$ and $\mathbb{W}_1^*$ of the same size with a file $file$ to the challenger $\mathcal{C}$, who chooses a random bit $b \in \{0,1\}$, and runs $\mathsf{Encrypt}(pp, file, dek, \mathbb{W}, sk_o)$ on $\mathbb{W}_b^*$ and $file$ to obtain the challenge keyword ciphertext $CT^*_{\mathbb{W}_b}$ and $(file)_{dek}$, then provides $CT^*_{\mathbb{W}_b}$ and $(file)_{dek}$ to the adversary $\mathcal{A}$, who runs $\mathsf{Check}(pp, sk_s, pk_o, pk_u, CT_{\mathbb{W}}, Td_{\mathbb{W}})$ to verify whether there is a trapdoor satisfying the ciphertext.
- **Query phase 2:** The adversary $\mathcal{A}$ continues to adaptively issue a polynomial number of queries to the challenger $\mathcal{C}$ for the trapdoors with the keyword LSSS access structures $\{LAS_{\mathbb{M}_j,\rho_j,\mathbb{W}_j} = (\mathbb{M}_j, \rho_j, \{\mathbb{W}_{\rho_j(i)}\}) | \ j \in \{q_{\mathrm{T}}+1, q_{\mathrm{T}}+2, \dots, q\}\}$ the challenger $\mathcal{C}$ directly runs $\mathsf{Trapdoor}(pp, msk, pk_s, \mathbb{U}, pw_A, UID_S, UID_A, UID_o, LAS_{\mathbb{M}_j,\rho_j,\mathbb{W}_j} = (\mathbb{M}_j, \rho_j, \{\mathbb{W}_{\rho_j(i)}\}))$ to generate the trapdoor $Td_{\mathbb{W}j}$ and key parameter kp_j, and sends the trapdoor $Td_{\mathbb{W}j}$ and key parameter kp_j to the adversary $\mathcal{A}$, where pw_A and UID_A are the impersonated user's password and unique ID, respectively.
- **Guess:** The adversary $\mathcal{A}$ outputs its guess $b' \in \{0,1\}$ and wins the game if $b' = b$.

(2) The second security case is defined as follows (where the adversary $\mathcal{A}$ is assumed to be the "curious" CCS who tries to learn the statistic information on the correlation between a set of keywords and a keyword ciphertext):

- **Setup:** Similar to **Setup** in the first case, it won't be repeat here.
- **Query phase 1:** Similar to **Query phase 1** in the first case, but the keyword LSSS access structures $\{LAS_{\mathbb{M}_j,\rho_j,\mathbb{W}_j} = (\mathbb{M}_j, \rho_j, \{\mathbb{W}_{\rho_j(i)}\}) | \ j \in \{1, 2, \dots, q_{\mathrm{T}}\}\}$ cannot be satisfied by neither the keyword set $\mathbb{W}_0^*$ nor $\mathbb{W}_1^*$, and also, pw_A and UID_A are the target user's password and unique ID, respectively.
- **Challenge:** Similar to **Challenge** in the first case, it won't be repeat here.
- **Query phase 2:** Similar to **Query phase 2** in the first case, but the keyword LSSS access structures $\{LAS_{\mathbb{M}_j,\rho_j,\mathbb{W}_j} = (\mathbb{M}_j, \rho_j, \{\mathbb{W}_{\rho_j(i)}\}) | \ j \in \{q_{\mathrm{T}}+1, q_{\mathrm{T}}+2, \dots, q\}\}$ cannot be satisfied by neither the keyword set $\mathbb{W}_0^*$ nor $\mathbb{W}_1^*$, and also, pw_A and UID_A are the target user's password and unique ID, respectively.
- **Guess:** The adversary $\mathcal{A}$ outputs its guess $b' \in \{0,1\}$ and wins the game if $b' = b$.

Note that, the parameters $(FHS_i, BHS_i), k_{seed}, dek, kp$, *file* and $(file)_{dek}$ in above games can be random data initiated by the challenger $\mathcal{C}$, and can be ignored by the adversary $\mathcal{A}$, here we reserve these parameters for keeping the workflow integrity only. We define the advantage of a probabilistic polynomial time adversary $\mathcal{A}$ in above two games as

$$Adv_{A,pw}^{\mathrm{PAC-EKS}} = |Pr\left[b' = b\right] - 1/2| \,.$$

Additionally, we call the PAC-MC scheme is selectively secure under chosen keyword-set attacks if the advantage $Adv_{A,pw}^{\mathrm{PAC-EKS}}$ and $Adv_{A,p',q',g'\cdot\mathbb{G}_1,\mathbb{F},\beta,pw}^{KeyParUpdate}$ in the above games are both negligible for any probabilistic polynomial time adversary.

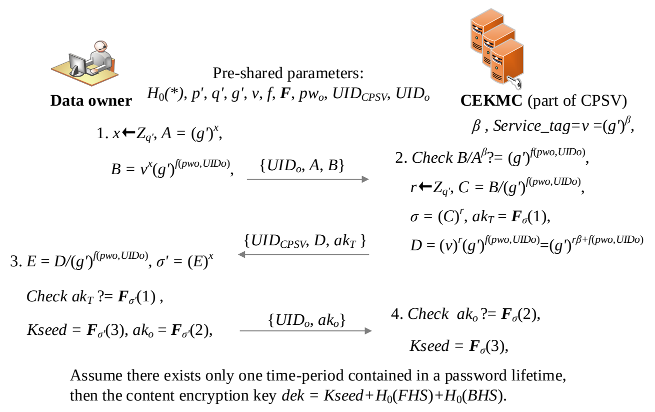

Data owner
Pre-shared parameters:
$H_0(*)$, p', q', g', v, f, $\boldsymbol{F}$, pw_o, UID_{CPSV}, UID_o
CEKMC (part of CPSV)
β , $Service\_tag = v = (g')^{\beta}$,
1. $x \leftarrow Z_{q'}$, $A = (g')^x$,
$B = v^x (g')^{f(pwo,UIDo)}$,
$\{UID_o, A, B\}$
2. $Check\ B/A^{\beta} ?= (g')^{f(pwo,UIDo)}$,
$r \leftarrow Z_{q'}$, $C = B/(g')^{f(pwo,UIDo)}$,
$\sigma = (C)^r$, $ak_T = \boldsymbol{F}_{\sigma}(1)$,
$D = (v)^r (g')^{f(pwo,UIDo)} = (g')^{r\beta + f(pwo,UIDo)}$
$\{UID_{CPSV}, D, ak_T\}$
3. $E = D/(g')^{f(pwo,UIDo)}$, $\sigma' = (E)^x$
$Check\ ak_T ?= \boldsymbol{F}_{\sigma'}(1)$,
$Kseed = \boldsymbol{F}_{\sigma'}(3)$, $ak_o = \boldsymbol{F}_{\sigma'}(2)$,
$\{UID_o, ak_o\}$
4. $Check\ ak_o ?= \boldsymbol{F}_{\sigma}(2)$,
$Kseed = \boldsymbol{F}_{\sigma}(3)$,
Assume there exists only one time-period contained in a password lifetime,
then the content encryption key $dek = Kseed + H_0(FHS) + H_0(BHS)$.

Fig. 4. Data owner (content provider) updates key materials and generates content encryption key with the help of CEKMC (part of CPSV).

4.3 Main Construction

In this section, we describe the concrete constructions of our PAC-MC, which consists of a new password-based content key management approach and a new PEKS based on knowledge of KP-ABE [34]. We assume all cloud users have been registered at URC with a password respectively, the access time-periods of outsourced files have been allocated for each data user by the data owner. Also, the information on the collection of authorized users and access time-periods for each user has been shared among the data owner, CPSV, and CCS.

1) ***System Initialization.*** KGC sets up the system by calling the Setup algorithm, and outputs a public key pp and a master secret key msk, where pp is publicized to a public board such that all entities have access to it, and msk is kept secret by CPSV. Meanwhile, for CCS and cloud users, KGC initializes them with secret keys by running the KeyGen algorithm after registration offline.

$\mathsf{Setup}(1^\ell)$: KGC first calls $\mathcal{G}(1^\ell)$ taking a secure parameter ℓ as input, generates the system parameters $\mathbb{D} = (p, \mathbb{G}, \mathbb{G}_T, \hat{e})$, where $\mathbb{G}, \mathbb{G}_T$ are both multiplicative cyclic groups with a prime order $p, \hat{e}$ is a bilinear map function $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, then picks a generator g of $\mathbb{G}$, the random terms $u, h, \omega \in \mathbb{G}$, and $\alpha \in \mathbb{Z}_p$. Next, it picks two big prime numbers: p', q', such that $q' | (p' - 1)$, chooses a group $\mathbb{G}_{q'}$ with a prime order q', a system secret key $\beta \in \mathbb{Z}_{q'}$, a system generator $g' \in \mathbb{G}_{q'}$, and a symmetric encryption algorithm $E(*)$ with appropriate integrity protection, constructs a pseudorandom function ensemble $\mathbb{F}$, three one-way hash functions $H_0 : \{0,1\}^* \to \mathbb{Z}_{q'}$, $f : \{0,1\}^* \to \mathbb{Z}_{q'}$, $H_1 : \{0,1\}^* \to \mathbb{Z}_p$, then computes $v = g'^{\beta}$. Finally, KGC outputs the system public parameter

$$pp = (\mathbb{D}, g, u, h, \omega, E, H_0, H_1, p', q', g', v, f, \mathbb{F}, \hat{e}(g,g)^{\alpha}),$$

the master private parameter $msk = (\alpha, \beta)$, and provides α to TGC, β to CEKMC, respectively.

$\mathsf{KeyGen}(pp, msk, UID_i)$: Given CCS or the cloud user's UID_i, KGC generates the private key $sk_i = H_1(\alpha \| UID_i)$, the public key $pk_i = g^{sk_i}$, further computes two hashing seeds: $FHS_i = H_0(sk_i), BHS_i = H_0(sk_i)^2 \in \mathbb{Z}_{q'}$ for each

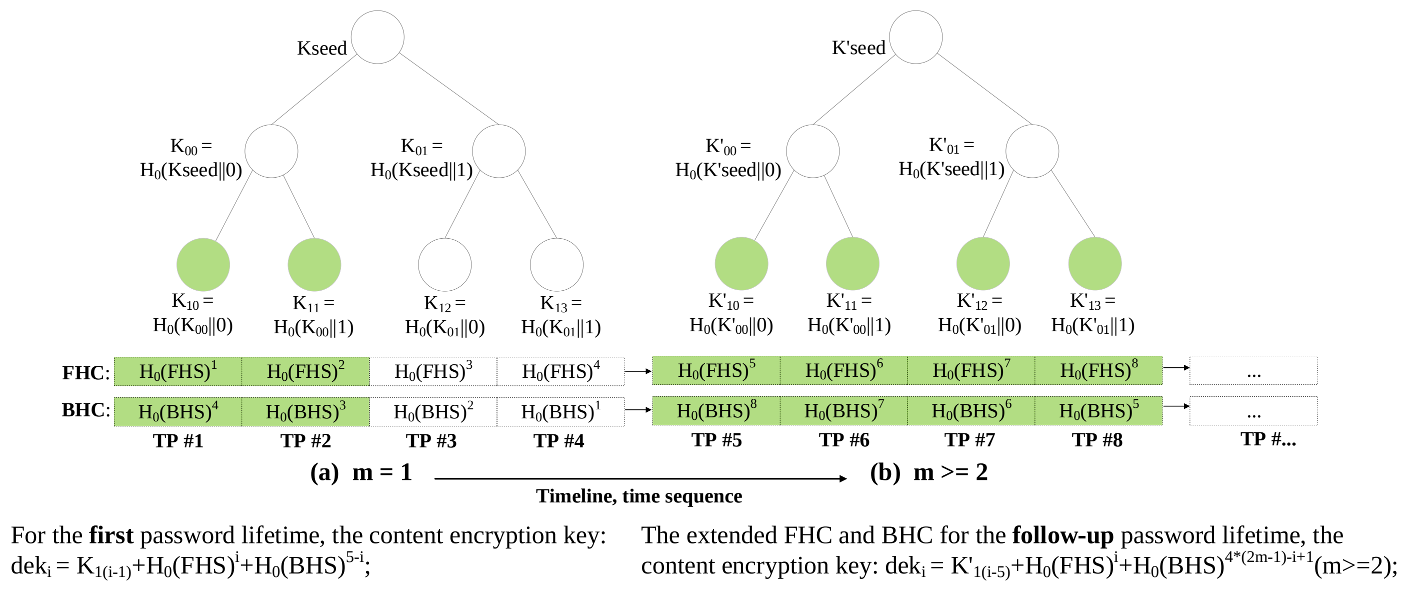

Kseed
K00 = H0(Kseed||0)
K01 = H0(Kseed||1)
K10 = H0(K00||0)
K11 = H0(K00||1)
K12 = H0(K01||0)
K13 = H0(K01||1)
K'seed
K'00 = H0(K'seed||0)
K'01 = H0(K'seed||1)
K'10 = H0(K'00||0)
K'11 = H0(K'00||1)
K'12 = H0(K'01||0)
K'13 = H0(K'01||1)
FHC: H0(FHS)^1 H0(FHS)^2 H0(FHS)^3 H0(FHS)^4 H0(FHS)^5 H0(FHS)^6 H0(FHS)^7 H0(FHS)^8 ...
BHC: H0(BHS)^4 H0(BHS)^3 H0(BHS)^2 H0(BHS)^1 H0(BHS)^8 H0(BHS)^7 H0(BHS)^6 H0(BHS)^5 ...
TP #1 TP #2 TP #3 TP #4 TP #5 TP #6 TP #7 TP #8 TP #...
(a) m = 1
Timeline, time sequence
(b) m >= 2
For the first password lifetime, the content encryption key: dek_i = K_1(i-1)+H0(FHS)^i+H0(BHS)^5-i;
The extended FHC and BHC for the follow-up password lifetime, the content encryption key: dek_i = K'_1(i-5)+H0(FHS)^i+H0(BHS)^4*(2m-1)-i+1(m>=2);

Fig. 5. (a) One-way hash binary tree for key materials management. (b) The extensions of FHC and BHC for new password lifetime.

user[3], then outputs the key pair (pk_i, sk_i) for CCS and each user, respectively, where $H_0(sk_i)^2$ denotes the algorithm iteratively calls $H_0(*)$ twice using sk_i, and "$\|$" denotes a concatenating operation.

2) ***Key Materials Updating.*** When initially using cloud service, or the password expires, CPSV assists data owner to update the key materials of content encryption key for those files outsourced in the coming new time-periods using a new password pw_o.

KeyParUpdate$(pp, msk, pw_o, UID_{CPSV}, UID_o)$: Given the data owner's password pw_o, the CPSV's UID_{CPSV}, and the data owner's UID_o, with the help of CPSV, the data owner executes the 3-round protocol as shown in Fig. 4 to generate a key seed k_{seed}. However, unlike the assumption stated in Fig. 4 that *"Assume there exists only one time-period contained in a password lifetime."*, for more practical scenarios where a password lifetime consists of multiple time-periods, the key materials k_{seed}, FHS, BHS, and content encryption key dek are further computed as follows:

We assume a password lifetime consists of $n (n = 4)$ time-periods, CEKMC (part of CPSV) builds an one-way hash binary tree (OHBT) by calling $H_0(*)$ using the key seed k_{seed} for content encryption key materials management, which is shown in Fig. 5(a), where the root node of OHBT is the key seed k_{seed}. Then it computes a forward hash chain (FHC) and a backward hash chain (BHC) as below, FHC and BHC are also shown at the bottom of Fig. 5(a).

$$FHC: H_0(FHS)^1, H_0(FHS)^2, H_0(FHS)^3, H_0(FHS)^4;$$
$$BHC: H_0(BHS)^4, H_0(BHS)^3, H_0(BHS)^2, H_0(BHS)^1;$$

We define the content encryption key dek_i for the i-th time-period contained in the *first* password lifetime as:

$$dek_i = K_{1(i-1)} + H_0(FHS)^i + H_0(BHS)^{n-i+1}, \quad (1)$$

where i is the logical number of current time-period calculated from the setup of application. For example, for the first time-period TP#1, the content encryption key $dek_1 = K_{10} + H_0(FHS)^1 + H_0(BHS)^4$, where K_{10} is a secret key located on the first leaf node of OHBT defined for TP#1; $H_0(FHS)^1$, $H_0(BHS)^4$ are the hash values of FHC and BHC defined for TP#1, respectively.

If KeyParUpdate(*) is performd when the prescribed event occurs, e.g. password expires, CEKMC builds another OHBT using the *new* generated key seed k'_{seed}[4], and extends FHC and BHC using their previous values respectively as shown in Fig. 5(b), we define the content encryption key $dek_i (i > 4)$ for the new password lifetimes as:

$$dek_i = K'_{1(i-n-1)} + H_0(FHS)^i + H_0(BHS)^{(2m-1)n-i+1}, \quad (2)$$

where $m (m >= 2)$ is the sequence number of current password calculated from the setup of application. For example, for the fifth time-period TP#5, the content encryption key $dek_5 = K'_{10} + H_0(FHS)^5 + H_0(BHS)^8$, where K'_{10} is a secret key located on the fifth leaf node of OHBT defined for TP#5; $H_0(FHS)^5$ and $H_0(BHS)^8$ are the hash values of FHC and BHC defined for TP#5, respectively.

After running KeyParUpdate(*), the data owner obtains the content encryption keys for outsouring occurs in current password lifetime as follows:

With the private key sk_o, the data owner computes hashing seeds $FHS_o = H_0(sk_o)$ and $BHS_o = H_0(sk_o)^2$, builds the OHBT using current key seed (k_{seed} or k'_{seed}), then retrieves the secret keys for all time-periods contained in current password lifetime from OHBT. In above example where n = 4, the secret keys for all time-periods that contained in the *first* password lifetime are $\{K_{10}, K_{11}, K_{12}, K_{13}\}$ located on the first 4 leaf nodes of OHBT, respectively.

3. Note that, CEKMC also keeps the hashing seeds by itself for other purposes, e.g. public enforcement audit etc.

4. Note that all the key seeds k_{seed} generated for the consecutive password lifetimes will be saved by CEKMC permanently until the relevant files are deleted from CCS, to ensure any user privileged after current time-period can access such files normally.

Next, using FHS_o and BHS_o, the data owner iteratively calls $H_0(*)$ to obtain the required hash value pairs:

$$\{H_0(FHS)^i, (H_0(BHS)^{n-i+1} or \\ H_0(BHS)^{(2m-1)n-i+1}) | i \in [n], m >= 2\},$$

for all time-periods contained in current password lifetime. Considering the above example where n = 4, the hash value pairs of FHC and BHC for all time-periods contained in the *first* password lifetime are

$$\{(H_0(FHS)^1, H_0(BHS)^4)\}, \{(H_0(FHS)^2, H_0(BHS)^3)\}, \\ \{(H_0(FHS)^3, H_0(BHS)^2)\}, \{(H_0(FHS)^4, H_0(BHS)^1]\}.$$

Finally, according to the equations (1) or (2), the data owner generates the content encryption keys $dek_i (i \in [n])$ for all time-periods that contained in current password lifetime with the corresponding secret key and hash value pairs of FHC and BHC, respectively.

3) ***Content Outsourcing.*** The data owner extracts a set of keywords $\mathbb{W}$ for each file $file$ produced and outsourced to the cloud based on time-sequence, encrypts the keywords $\mathbb{W}$, and encrypts $file$ itself by using symmetric algorithm with content encryption key dek corresponding to current time-period, uploads the encrypted file and its encrypted keywords to CCS.

$\mathsf{Encrypt}(pp, file, dek, \mathbb{W}, sk_o)$: Given a file $file$, the content encryption key dek for current time-period, the keyword set $\mathbb{W}$ of $file$ (where each keyword $\mathbb{W}_i$ consists of a generic name $\mathbb{W}_{iN}$ and a keyword value $\mathbb{W}_{iV} (i = 1, \ldots, m)$), and the data owner's private key sk_0, the data owner first randomly picks $s, r_1, r_2, \ldots, r_m \in \mathbb{Z}_p$, outputs the following keyword ciphertext for the file $file$. $CT_{\mathbb{W}} = (C, C_0, \{C_{i,1}, C_{i,2}, C_{i,3}\}_{i \in [m]})$, where

$$C = \hat{e}(g, g)^{\alpha s}, C_0 = g^s, C_{i,1} = g^{r_i}, \\ C_{i,2} = \omega^{-s}(u^{H_1(\mathbb{W}_{iV} \| sk_o)} h)^{r_i}, C_{i,3} = (C_{i,2})^{sk_o},$$

then outputs $(file)_{dek}$ by using $E(*)$ with the key dek. Finally appends $(file)_{dek}$ with a timestamp and $CT_{\mathbb{W}}$.

4) ***Trapdoor and Encryption Key Self-retrieving Parameters Generation.*** The data user sends a trapdoor request to TGC (part of CPSV), who generates a trapdoor $Td_{\mathbb{W}}$ and a set of key parameters kp after running separate identity authentication, and returns $Td_{\mathbb{W}}$ and kp to the data user.

$\mathsf{Trapdoor}(pp, msk, pk_s, \mathbb{U}, pw_u, UID_s, UID_u, UID_o, (LAS_{\mathbb{M},\rho,\mathbb{W}} = (\mathbb{M}, \rho, \{\mathbb{W}_{\rho(i)}\}))$: Given the public key pk_s of CCS, a collection of authorized users $\mathbb{U}$, the data user's password pw_u, the CCS's UID_s, the data user's UID_u, the data owner's UID_o, and a keyword LSSS access structure $LAS_{\mathbb{M},\rho,\mathbb{W}} = (\mathbb{M}, \rho, \{\mathbb{W}_{\rho(i)}\})$[5], TGC checks whether $UID_u \in \mathbb{U}$ and performs *the separated necessary identity authentication*, if approved, it picks a vector $\vec{y} = (\alpha, y_2, \ldots, y_n)^{\perp}$, where $y_2, \ldots, y_n \in \mathbb{Z}_p$, we assume the vector of the shares belongs to the master private key α is $\overline{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_l)^{\perp} = \mathbb{M}\vec{y}$, then it picks $t_0, t_1, t_2, \ldots, t_l \in \mathbb{Z}_p$, computes

$\Delta = \hat{e}(g^{H_1(H_1(\alpha \| UID_s)] | t_0 \| UID_u)}, pk_s)^{H_1(pw_u \| t_0 \| UID_u)}$ and outputs the following trapdoor

$Td_{\mathbb{W}} = (UID_u, (\mathbb{M}, \rho), \{T_{i,0}, T_{i,1}, T_{i,2}, T_{i,3}\}_{i \in [l]}, T_4, T_5)$, where

$$T_{i,0} = g^{\lambda_i} \omega^{t_i}, T_{i,1} = (u^{H_1(\mathbb{W}_{\rho(i)} \| H_1(\alpha \| UID_o)) \cdot} h)^{-t_i}, \\ T_{i,2} = g^{H_1(\Delta)} g^{t_i}, T_{i,3} = T_{i,1}^{H_1(\alpha \| UID_u)}, \\ T_4 = g^{H_1(pw_u \| t_0 \| UID_u)}, \\ T_5 = g^{H_1(H_1(\alpha \| UID_s) \| t_0 \| UID_u)}.$$

Next, CEKMC generates key parameters kp for content encryption key self-retrieving for the data user as follows:

- Firstly, it generates the key components $(K_{1(i-1)}, \ldots$ or $K'_{1(i-n-1)}, \ldots)$ by calculating *the lowest common ancestor nodes* using the secret keys that belongs to the data user from OHBT. For example, see Fig. 5, we assume there is an authorized user who has been allocated the access time-periods (TP): TP#1, #2, #5, #6, #7 and #8, which are also marked in green in Fig. 5, then it generates the key components (K_{00}, k'_{seed}) from OHBT, where the secret keys $K'_{10}, K'_{11}, K'_{12}, K'_{13}$ have the lowest common ancestor node k'_{seed}, the secret keys K_{10}, K_{11} have the lowest common ancestor node K_{00}. That is, *"the lowest common ancestor nodes"* means it tries to find the lowest common ancestor node for the successive leaf nodes holding secret keys that belongs to the data user.
- Secondly, it generates two starting hash values:
  $$\{H_0(FHS)^i, \\ H_0(BHS)^{n-i+1} or H_0(BHS)^{(2m-1)n-i+1}\},$$
  from FHC and BHC for the data user. Considering the example in Fig. 5, it computes the starting hash values $(H_0(FHS)^1, H_0(BHS)^3)$ from FHC and BHC for the data user, respectively.
- Finally, using the key components and starting hash values, it obtains the following set of key parameters kp, and forwards kp to TGC.
  $$kp = \{(K_{1(i-1)}, \ldots or K'_{1(i-n-1)}, \ldots), (H_0(FHS)^i, \\ H_0(BHS)^{n-i+1} or H_0(BHS)^{(2m-1)n-i+1})\}.$$

TGC outputs the trapdoor $Td_{\mathbb{W}}$, together with key parameters kp[6] to the data user. Considering the example in Fig. 5, $kp = \{(K_{00}, k'_{seed}), (H_0(FHS)^1, H_0(BHS)^3)\}$.

5) ***Content Searching.*** The data user sends the trapdoor $Td_{\mathbb{W}}$ and a partial hidden access structure to CCS, who matches the trapdoor $Td_{\mathbb{W}}$ with the keyword ciphertexts $CT_{\mathbb{W}}$ in database, returns the matched file to data user.

$\mathsf{Check}(pp, sk_s, pk_o, pk_u, CT_{\mathbb{W}}, Td_{\mathbb{W}})$: Given the private key sk_s of CCS, the data owner's public key pk_o, the data user's public key pk_u, the keyword ciphertexts $CT_{\mathbb{W}}$, and the trapdoor $Td_{\mathbb{W}}$ for the LSSS access structure $LAS_{\mathbb{M},\rho,\mathbb{W}} = (\mathbb{M}, \rho, \{\mathbb{W}_{\rho(i)}\})$, CCS checks whether the timestamp attached to file satisfies the access time-period of data user,

5. Here $\mathbb{M}$ is a $\ell \times n$ matrix over $\mathbb{Z}_p$, let M_i denotes the i-th row of $\mathbb{M}$, ρ is function mapping M_i to generic keyword name $\{\rho(i) | i \in [\ell]\}$, $\{\mathbb{W}_{\rho(i)}\}$ are corresponding keyword values, we denote the keyword values in trapdoor using $\mathbb{W}_{\rho(i)}$ to distinguish it from those keyword values $\mathbb{W}_{iV}$ in the keyword ciphertext.

6. We assume kp is encrypted by using $E(*)$ with appropriate integrity protection with password pw_u before being output to data user along with the trapdoor $Td_{\mathbb{W}}$. Here we ignore the specific approach, which is out of the scope of this paper.

if yes, generates a set $I_{\mathbb{M},\rho}$ from $(\mathbb{M},\rho)$, which is a set of the minimum subsets satisfying $(\mathbb{M},\rho)$, checks whether there exists an $\mathfrak{I} \in I_{\mathbb{M},\rho}$, which makes the constants $\{\omega_i \in \mathbb{Z}_p\}_{i\in\mathfrak{I}}$ such that $\sum_{i\in\mathfrak{I}} \omega_i M_i = (1,0,\ldots,0)$, and the following equations hold:

$$\hat{e}(g, C_{\tau,3}) \overset{?}{=} \hat{e}(pk_o, C_{\tau,2}), \hat{e}(g, T_{i,3}) \overset{?}{=} \hat{e}(pk_u, T_{i,1}),$$

$$\prod_{i\in\mathfrak{I}} \left(\hat{e}(C_0, T_{i,0})\,\hat{e}(C_{\tau,1}, T_{i,1})\,\hat{e}\left(C_{\tau,2}, T_{i,2}/g^{H_1(\hat{e}(T_4,T_5)^{sk_s})}\right)\right)^{\omega_i} \overset{?}{=} C,$$

where τ is the index of keyword $\rho(i)$ in $CT_{\mathbb{W}}$[7]. CCS outputs "$\perp$" , if the timestamp attached to file does not satisfy the access time-period that belongs to data user or there is no any $\mathfrak{I}$ in $I_{\mathbb{M},\rho}$ can satisfy the equations, otherwise outputs "Yes" and $(file)_{dek}$.

6) ***Key Retrieving and Content Decryption.*** The data user self-retrieves the content encryption key dek using kp, and decrypts the file $(file)_{dek}$ returned from CCS.

Decrypt$((file)_{dek}, kp)$: Given the key parameters kp, the data user self-retrieves dek_i, and decrypts $(file)_{dek}$ to get the plaintext of content in $file$. Considering the example in Fig. 5, after obtaining the key parameters $kp = \{(K_{00}, k'_{seed}), (H_0(FHS)^1, H_0(BHS)^3)\}$, the data user self-retrieves the secret keys: $K_{10}, K_{11}, K'_{10}, K'_{11}, K'_{12}, K'_{13}$ located on leaf nodes of OHBT using the key components (K_{00}, k'_{seed}) by iteratively calling $H_0(^*)$ following the structure of OHBT, and generates the following hash values:

$$\begin{aligned}\{H_0(FHS)^1,&H_0(FHS)^2, H_0(FHS)^5,\\ &H_0(FHS)^6, H_0(FHS)^7, H_0(FHS)^8\},\\ \{H_0(BHS)^3,&H_0(BHS)^4, H_0(BHS)^5,\\ &H_0(BHS)^6, H_0(BHS)^7, H_0(BHS)^8\},\end{aligned}$$

by iteratively calling $H_0(^*)$ using hash values $H_0(FHS)^1$ and $H_0(BHS)^3$ respectively, then computes the conent encryption key dek_i, if it is required to decrypt the files outsourced in the i-th time-period, where $i \in \{1,2,5,6,7,8\}$.

Correctness: According to the definition of access structures and LSSS, if there exists an $\mathfrak{I}$ in $I_{\mathbb{M},\rho}$ such that, the set of keywords encrypted in ciphertext $CT_{\mathbb{W}}$ satisfies the LSSS access structure used in the trapdoor $Td_{\mathbb{W}}$, then we have $\sum_{i\in\mathfrak{I}} \omega_i\lambda_i = \alpha$. Therefore

$$\hat{e}(g, C_{\tau,3}) = \hat{e}\left(g, (C_{\tau,2})^{sk_o}\right) = \hat{e}(pk_o, C_{\tau,2}).$$

$$\hat{e}(g, T_{i,3}) = \hat{e}\left(g, T_{i,1}^{H_1(\alpha||UID_u)}\right) = \hat{e}\left(g, T_{i,1}^{sk_u}\right) = \hat{e}(pk_u, T_{i,1}).$$

$$\begin{aligned}&\prod_{i\in\mathfrak{I}} \left(\hat{e}(C_0, T_{i,0})\,\hat{e}(C_{\tau,1}, T_{i,1})\,\hat{e}\left(C_{\tau,2}, T_{i,2}/g^{H_1(\hat{e}(T_4,T_5)^{sk_s})}\right)\right)^{\omega_i}\\ &= \prod_{i\in\mathfrak{I}} \hat{e}\left(g^s, g^{\lambda_i}\omega^{t_i}\right)^{\omega_i} \cdot \hat{e}\left(g^{r_\tau}, \left(u^{H_1(\mathbb{W}_{\rho(i)}\|sk_o)}h\right)^{-t_i}\right)^{\omega_i}\\ &\cdot \hat{e}\left(\omega^{-s}\left(u^{H_1(\mathbb{W}_{\rho(i)}\|sk_o)}h\right)^{r_\tau}, g^{t_i}\right)^{\omega_i}\\ &= \hat{e}(g,g)^{s\sum_{i\in\mathfrak{I}}\omega_i\lambda_i} = \hat{e}(g,g)^{\alpha s} = C.\end{aligned}$$

7. Here τ can be calculated by comparing the keyword name $\rho(i)$ with the generic name attached to the $CT_{\mathbb{W}}$ by traversal, depends on i.

4.4 Security Proof

Theorem 1. Suppose $H_0(^*), H_1(^*), f(^*)$ are three one-way hash functions, if the defined pseudorandom function ensemble $\mathbb{F}$ exists, then under the DDH assumption, the DBDH assumption and the decisional (q-2) assumption, our PAC-MC scheme is selectively secure under chosen keyword-set attacks for any adversary with polynomial-time computational capability.

Proof of Theorem 1: We first prove that the sub-process KeyParUpdate(*) is secure under the standard model, then prove that the whole PAC-MC scheme is with selectively secure under chosen keyword-set attacks.

Lemma 1. Let $\mathcal{T}$ be the algorithm KeyParUpdate(*), $\mathbb{F}$ is pseudorandom function ensemble, p', q' are two big prime numbers, such that $q'|(p'-1)$, and $G_{q'}$ is a q'-order subgroup of multiplicative group $\mathbb{Z}_p^*$, g' is the generator of $\mathbb{G}_{q'}$ and $v = g'^{\beta}$, where $\beta \in \mathbb{Z}_{q'}$ is a system secret key, C' and s' are the Zipf regression parameters, q_{send} is the number of the Send$(\Pi^n_{i,j}, M)$ queries, then under the DDH assumption, the algorithm KeyParUpdate(*) is secure for any probabilistic polynomial-time adversary $\mathcal{A}$, i.e. $Adv^{KeyParUpdate}_{A,p',q',g'\cdot\mathbb{G}_1,\mathbb{F},\beta,pw} \leq 2\cdot C'\cdot q^{s'}_{send} + \varepsilon$, where ε is a negligible variable in the security parameter.

Proof: We prove this lemma via a sequence of games $\mathcal{T}_0, \mathcal{T}_1, \ldots$, where the game $\mathcal{T}_0$ is the same as the original algorithm KeyParUpdate(*), then we gradually modify the oracle's reply to the adversary $\mathcal{A}$ in subsequent games and ensure that the adversary's success probability difference between two adjacent games is negligible. Note that, in all games, the oracle behind simulator replies to the adversary's query according to the description of the algorithm KeyParUpdate(*).We define the adversary's success probability in the game $\mathcal{T}_i$ as Pr $[\mathcal{T}_i]$.

Game $\mathcal{T}_0$: The game $\mathcal{T}_0$ is the same as the original algorithm, if the adversary $\mathcal{A}$ successfully guessed the value b' of b in the Test(*) query on a fresh oracle, then we have $Adv^{KeyParUpdate}_{A,p',q',g'\cdot\mathbb{G}_1,\mathbb{F},\beta,pw} = 2Pr[b=b'] - 1 = 2Pr[\mathcal{T}_0] - 1$.

Game $\mathcal{T}_1$: In the game $\mathcal{T}_1$, the simulator replaces B with a random number $r \in \mathbb{G}_{q'}$ in the message $\{UID_o, A, B\}$ when the adversary $\mathcal{A}$ performs the Execute(*) query. Under the DDH assumption, with the hybrid arguments technique, we can prove it is impossible for the adversary $\mathcal{A}$ to distinguish B from r as follows:

Let q_e be the total number of Execute(*) queries performed by the adversary $\mathcal{A}$, $\mathcal{T}_{1,\chi}$ denotes a game, in which the first χ Execute(*) queries are replied to the adversary $\mathcal{A}$ following the game $\mathcal{T}_1$, but the remaining $(q_e - \chi)$ Execute(*) queries are replied following the game $\mathcal{T}_0$ $(0 \leq \chi \leq q_e)$. Obviously, we have $\mathcal{T}_{1,0} = \mathcal{T}_0, \mathcal{T}_{1,q_e} = \mathcal{T}_1$, therefore $Pr[\mathcal{T}_1] - Pr[\mathcal{T}_0] = Pr[\mathcal{T}_{1,q_e}] - Pr[\mathcal{T}_{1,0}]$.

Suppose $v = g'^{\beta}$, $f(*)$ is an one-way hash function: $\{0,1\}^* \rightarrow \mathbb{Z}_{q'}$, then we can construct the algorithms $D_i(0 \leq i \leq q_e - 1)$ to distinguish $(g', g'^x, g'^{\beta}, g'^{x\beta})$ from $(g', g'^x, g'^{\beta}, r)$ in a simulated way as below, where r a random number chosen from $\mathbb{G}_{q'}$:

For each algorithm $D_i(0 \leq i \leq q_e - 1)$ with an input $(g', g'^x, g'^{\beta}, Z)$, where Z is either equal to $g'^{x\beta}$ or equal to r, the simulator randomly chooses a password pw_o for the data owner, then the simulator:

a) Chooses two random numbers: x' and $r' \in \mathbb{Z}_{q'}$, computes $A_j = g'^{x'}, B_j = r'g'^{f(pw_o,UID_o)}$ and replys to the adversary $\mathcal{A}$ with a message $\{UID_o, A_j, B_j\}$ when the adversary $\mathcal{A}$ performs the j-th($j < i$) Execute(*) query;

b) Computes $A_j = g'^{x}, B_j = Zg'^{f(pw_o,UID_o)}$ and replys to the adversary $\mathcal{A}$ with a message $\{UID_o, A_{i+1}, B_{i+1}\}$ when the adversary $\mathcal{A}$ performs the $(i+1)$-th Execute(*) query;

c) Chooses a random number $x' \in \mathbb{Z}_{q'}$, computes $A_j = g'^{x'}, B_j = v^{x'}g'^{f(pw_o,UID_o)}$ and replys to the adversary $\mathcal{A}$ with a message $\{UID_o, A_j, B_j\}$ when the adversary $\mathcal{A}$ performs the j-th $(j > i+1)$ Execute(*) query;

d) Outputs 1 if the adversary $\mathcal{A}$ wins in the simulation, otherwise outputs 0;

For each algorithm D_i, the simulator records all the simulated messages $\{UID_o, A_j, B_j\}$ in the $\mathcal{T}_{1,\chi}(0 \leq \chi \leq q_e)$. Obviously, during the simulation, the adversary $\mathcal{A}$ can also obtain q_e transcripts $Ts_\chi(0 \leq \chi \leq q_e - 1)$, each of which consists of a sequence of messages $\{UID_o, A_j, B_j\}$, Note that, the transcript Ts_i obtained by the adversary $\mathcal{A}$ is the same as the messages recorded in the game $\mathcal{T}_{1,i}$ when $Z = g'^{x\beta}$, whereas the transcript Ts_i is the same as the messages recorded in the game $\mathcal{T}_{1,i+1}$ when $Z = r(0 \leq i \leq q_e - 1)$. We can obtain

$$\begin{aligned}|Pr\left[\mathcal{T}_1\right] - Pr\left[\mathcal{T}_0\right]| &= \textstyle\sum_{i=0}^{q_e-1}\left(|Pr\left[\mathcal{T}_{1,i+1}\right] - Pr\left[\mathcal{T}_{1,i}\right]|\right)\\ &\leq q_e \cdot Adv^{ddh}_{g',\mathbb{G}}(*),\end{aligned}$$

where $Adv^{ddh}_{g',\mathbb{G}}(*)$ is the probability advandage of breaking the DDH assumption by the adversary $\mathcal{A}$. Thus, if the DDH assumption holds, $|Pr[\mathcal{T}_1] - Pr[\mathcal{T}_0]|$ is negligible. That is, it is impossible for the adversary $\mathcal{A}$ to distinguish B from r.

Game $\mathcal{T}_2$: Based on the game $\mathcal{T}_1$, the simulator replaces the generated seed k_{seed} with an equal-size random number r when the adversary $\mathcal{A}$ performs the Execute(*) query. If the pseudorandom function ensemble $\mathbb{F}$ exists, then we can prove that $|Pr[\mathcal{T}_1] - Pr[\mathcal{T}_0]|$ is also negligible as follows. If the adversary $\mathcal{A}$ don't perform the Reveal(*) query any more, then the information he obtained in the game $\mathcal{T}_2$ is the same as that obtained in the game $\mathcal{T}_1$, otherwise, the random number r is returned to the adversary $\mathcal{A}$. In the game $\mathcal{T}_1$, if the adversary $\mathcal{A}$ obtained the seed k_{seed}, which means he had performed a query on the pseudorandom function ensemble $\mathbb{F}$, whereas, in the game $\mathcal{T}_2$, if the adversary $\mathcal{A}$ obtained the seed r, which means he had performed a query on the uniform distribution function ensemble $\mathbb{H}$. According to the definition of pseudorandom function ensemble, we know that the adversary $\mathcal{A}$ is impossible to effectively distinguish the pseudorandom function ensemble $\mathbb{F}$ from the uniform distribution function ensemble $\mathbb{H}$. Thus, $|Pr[\mathcal{T}_2] - Pr[\mathcal{T}_1]| \leq \min\{q_e, q_r\} \cdot Adv^{\mathbb{F}}$, where q_r is the number of the Reveal(*) queries, and $Adv^{\mathbb{F}}$ is the probability advantage of breaking the pseudorandom function ensemble $\mathbb{F}$ by the adversary $\mathcal{A}$.

Game $\mathcal{T}_3$: Based on the game $\mathcal{T}_2$, the simulator replaces the parameters ak_T and ak_o with an equal-size random number r_1 and r_2 respectively when the adversary $\mathcal{A}$ performs the Execute(*) query. Because the simulator have replaced B with a random number in the game $\mathcal{T}_1$, the messages A and B cannot satisfy $B/A^\beta = g'^{f(pw_o,UID_o)}$ any more, we assume $B = v^x g'^R$ where $R \neq f(pw_o, UID_o)$, then the simulator (simulates the CPSV) can compute $\sigma = C^r = (v^x g'^{R-f(pw_o,UID_o)})^r$. We can see that, σ is also a uniform random number since the r is chosen randomly and independently. Similar to the analysis in the game $\mathcal{T}_2$, if the pseudorandom function ensemble $\mathbb{F}$ exists, with the hybrid arguments technique we can obtain $|Pr[\mathcal{T}_3] - Pr[\mathcal{T}_2]| \leq q_e \cdot Adv^{\mathbb{F}}$. Obviously, it is negligible.

Game $\mathcal{T}_4$: Based on the game $\mathcal{T}_3$, the simulator replaces the parameter C with an random number when the adversary $\mathcal{A}$ performs the Execute(*) query, in this case, the parameter E will be also a random number. Under the DDH assumption, we can obtain $|Pr[\mathcal{T}_4] - Pr[\mathcal{T}_3]| \leq q_e \cdot Adv^{ddh}_{g',\mathbb{G}}(*)$ by using the hybrid arguments technique.

Game $\mathcal{T}_5$: Based on the game $\mathcal{T}_4$, the simulator replaces the parameter D with an random number when the adversary $\mathcal{A}$ performs the Execute(*) query, in this case, the adversary $\mathcal{A}$ cannot distinguish any change since the message D itself is the product of a random number and $(g')^{f(pwo,UIDo)}$, we can obtain $|Pr[\mathcal{T}_5] = Pr[\mathcal{T}_4]|$.

Game $\mathcal{T}_6$: Based on the game $\mathcal{T}_5$, when the adversary $\mathcal{A}$ performs the Send(*, null) query, B is replaced with a random number r by the simulator. Similar to the analysis in the game $\mathcal{T}_1$, under the DDH assumption, with the hybrid arguments technique, we can obtain $|Pr[\mathcal{T}_5] - Pr[\mathcal{T}_4]| \leq q_{send} \cdot Adv^{ddh}_{g',\mathbb{G}}(*)$, where q_{send} is the number of the Send(*) queries.

Game $\mathcal{T}_7$: Based on the game $\mathcal{T}_6$, when the adversary $\mathcal{A}$ performs the Send(UID_o, A, B) query, if the equation $B = A^\beta\ (g')^{f(pwo,UID_o)}$ holds and checked by the simulator, then the adversary $\mathcal{A}$ wins the game, the algorithm exits, then

1) The adversary $\mathcal{A}$ may have acquired two messages A and B through the previous Execute(*) and Send(*) queries such that $B = A^\beta\ (g')^{f(pwo,UID_o)}$, the success probability in this case is $(q_{send} + q_e)/q$;

2) The adversary $\mathcal{A}$ may have guessed the correct password pw_o, the success probability in this case is $C' \cdot q^{s'}_{send}$;

3) The adversary $\mathcal{A}$ may have constructed two new messages A and B successfully through guessing a B to match the chosen A by himself, and $B = A^\beta\ (g')^{f(pwo,UID_o)}$ holds, the success probability in this case is q_{send}/q;

Then in this game, we have $|Pr[\mathcal{T}_7] - Pr[\mathcal{T}_6]| \leq C' \cdot q^{s'}_{send} + (2q_{send} + q_e)/q$.

Game $\mathcal{T}_8$: Based on the game $\mathcal{T}_7$, when the adversary $\mathcal{A}$ performs the Send(UID_{CPSV}, D, ak_T) query, if the messages satisfying $ak_T = \mathbb{F}_{\sigma'}(1)$ checked by the simulator, then the adversary $\mathcal{A}$ wins the game, the algorithm exits, then there exists 3 cases:

1) The adversary $\mathcal{A}$ may have got the effective message (UID_{CPSV}, D, ak_T) satisfying (UID_o, A, B) by replay attack, the success probability in this case is $(q_e + q_{send})/q$;

2) The adversary $\mathcal{A}$ may have guessed the correct password pw_o to obtain the parameters v^x and $\sigma = (g')^{xr\beta}$ satisfying (UID_{CPSV}, D, ak_T), the success probability in this case is $C' \cdot q^{s'}_{send}$;

3) The adversary $\mathcal{A}$ may have successfully computed a $ak_T = \mathbb{F}_{\sigma'}(1)$ without σ, the success probability in this case is $q_{send} \cdot Adv^{\mathbb{F}}$;

Then in this game, we have $|Pr[\mathcal{T}_8] - Pr[\mathcal{T}_7]| \leq C' \cdot q^{s'}_{send} + (q_e + q_{send})/q + q_{send} \cdot Adv^{\mathbb{F}}$.

Game $\mathcal{T}_9$: Based on the game $\mathcal{T}_8$, when the adversary $\mathcal{A}$ performs the Send(UID_o, ak_o) query, if the message satisfying $ak_o = \mathbb{F}_{\sigma'}(2)$ checked by the simulator, then the adversary $\mathcal{A}$ wins the game $\mathcal{T}_9$, the algorithm exits, if the adversary $\mathcal{A}$ has not been succeed in the game $\mathcal{T}_8$, then all messages must be generated by CPSV and Data owner, and the information on x, r and $(g')^{xr\beta}$ cannot be obtained by the adversary, if the pseudorandom function ensemble $\mathbb{F}$ exists, the probability that the adversary $\mathcal{A}$ can successfully guess the $\mathbb{F}_{\sigma'}(2)$ is $1/2^n$, we can obtain $|Pr[\mathcal{T}_9] - Pr[\mathcal{T}_8]| \leq q_{send}/2^n$.

Game $\mathcal{T}_{10}$: Based on the game $\mathcal{T}_9$, when the adversary $\mathcal{A}$ performs the Reveal(*) query, the parameter k_{seed} is replaced with a uniformly chosen random number by the simulator. If the adversary $\mathcal{A}$ doesnot perfrom any Reveal(*) query after Execute(*) and Send(*) query, then the information obtained by the adversary $\mathcal{A}$ in the game $\mathcal{T}_{10}$ is the same as in the game $\mathcal{T}_9$, the adversary $\mathcal{A}$ cannot obtain any information on σ in the game $\mathcal{T}_9$. Similar to the analysis in the game $\mathcal{T}_8$, if the pseudorandom function ensemble $\mathbb{F}$ exists, with the hybrid arguments technique we can obtain $|Pr[\mathcal{T}_{10}] - Pr[\mathcal{T}_9]| \leq \min\{(q_e + q_{send}), q_r\} \cdot Adv^{\mathbb{F}}$.

The adversary $\mathcal{A}$ cannot obtain any information on the correct password pw_o and also $(g')^{xr\beta}$ in the game $\mathcal{T}_{10}$. Unless the adversary $\mathcal{A}$ has previously queried the instance or its partner before this query, which is not allowed, therefore, the adversary $\mathcal{A}$ cannot distinguish k_{seed} and a random number in the final Test(*) query, we have $|Pr[\mathcal{T}_{10}]| = 1/2$.

We can see that the adversary $\mathcal{A}$ cannot obtain any information on the correct password pw_o in the game $\mathcal{T}_{10}$, due to all relevant parameters are replaced with random numbers. Also, the adversary $\mathcal{A}$ can obtain nothing on the parameter σ and σ'. Thus, it is impossible for the adversary $\mathcal{A}$ to distinguish k_{seed} from a random number, the probability that the adversary $\mathcal{A}$ can correctly guess the value of bit b is $1/2$ in the final Test(*) query. This also shows that the success probability of the adversary $\mathcal{A}$ depends on the guess of the password via the active Send(*) query, has nothing to do with queries such as Execute(*) and Reveal(*) query etc.

Combining the results in above games, we have

$$Adv^{KeyParUpdate}_{A,p',q',g' \cdot \mathbb{G}_1,\mathbb{F},\beta,pw} \leq 2 \cdot C' \cdot q^{s'}_{send} + \varepsilon,$$

which completes the proof of Lemma 1. □

Lemma 2. Suppose $H_1(*)$ is a one-way hash function, if both the DBDH assumption and the decisional (q-2) assumption hold, our PAC-MC scheme is semantically secure with selectively security under chosen keyword-set attacks.

Proof: There are two security cases.

1) Case 1, we prove that our scheme is secure via two games G_0 and G_1, where the game G_0 is the same as the original scheme with a real trapdoor. Furthermore, based on the game G_0, we replace the parameter used to generate trapdoor with a random number chosen from $\mathbb{G}_T$ to construct the game G_1. We can obtain, if the adversary $\mathcal{A}$ can distinguish the game G_0 from G_1, then the challenger $\mathcal{C}$ can also break the DBDH assumption. For simplicity, we omit the insignificant process and parameters in the following two games.

Game G_0: The same as what is performed in the original scheme in real-world.

- **Setup:** The adversary $\mathcal{A}$ initially declares two challenge keyword sets $\mathbb{W}_0^* = \{\mathbb{W}_{0,1}^*, \cdots, \mathbb{W}_{0,m}^*\}$ and $\mathbb{W}_1^* = \{\mathbb{W}_{1,1}^*, \ldots, \mathbb{W}_{1,m}^*\}$ of the same size with a file to the challenger $\mathcal{C}$. The challenger $\mathcal{C}$ first chooses a random bit $b \in \{0,1\}$, and runs Setup(1^ℓ) to obtain the system public parameter and the master private parameter. Next, the challenger $\mathcal{C}$ runs KeyGen(*) to generate a public/private key pair (pk_i, sk_i) and other parameters for the users, respectively, and sets $(pk_s = g^x, sk_s = x)$ to be the public/private key pair for CCS, where x is set using the setting in the DBDH assumption. Finally, it runs KeyParUpdate(*) to obtain other parameters as requried for keeping the game running.
- **Query phase 1:** The adversary $\mathcal{A}$ adaptively issues a polynomial number of queries to the challenger $\mathcal{C}$ for the trapdoors, since the challenger $\mathcal{C}$ knows the master private key, the shared password with user and all private keys for users and CCS, it can easily output the trapdoor as required, the challenger $\mathcal{C}$ runs Trapdoor(*) to generate a trapdoor $Td_{\mathbb{W}} = (UID_{A'}(\mathbb{M},\rho), \{T_{i,0}, T_{i,1}, T_{i,2}, T_{i,3}\}_{i\in[l]}, T_4, T_5)$, and sends it to the adversary $\mathcal{A}$. However, if the adaptive query with a LSSS access structure that can be satisfied by $\mathbb{W}_b^*$, then the challenger $\mathcal{C}$ implicitly set $Z = \Delta = \hat{e}(g^z, g^x)^y, T_4 = g^y, T_5 = g^z, T_{i,2} = g^{H_1(\Delta)} \cdot g^{t_i}$ when running Trapdoor(*), where x, y, z are set in the DBDH assumption.
- **Challenge:** The adversary $\mathcal{A}$ submits the declared keyword sets $\mathbb{W}_0^*$ and $\mathbb{W}_1^*$ with a file to the challenger $\mathcal{C}$, who runs Encrypt(*) on $\mathbb{W}_b^*$ to obtain the challenge ciphertext $CT^*_{\mathbb{W}_b}$, and provides it to the adversary $\mathcal{A}$ along with other parameters as requried, then the adversary $\mathcal{A}$ runs Check(*) to verify whether there exists a trapdoor satisfying the challenge ciphertext.
- **Query phase 2:** The same as what is performed in Query phase 1.
- **Guess:** The adversary $\mathcal{A}$ outputs its guess b' for b.

Game G_1: The same as the game G_0, except that the parameter Z is replaced with a uniformly chosen random number $r \in \mathbb{G}_T$ in Query phase 1 and 2.

Obviously, in the game G_0, $Z = \hat{e}(g^z, g^x)^y = \hat{e}(g,g)^{xyz}$, which is identical to the original scheme fom the perspective of the adversary $\mathcal{A}$, whereas Z is a random number chosen from $\mathbb{G}_T$ in the game G_1, thus, if the DBDH assumption holds, it is impossible for the adversary $\mathcal{A}$ to distinguish the game G_0 from G_1. Therefore, our scheme can resist the attacks may occur in the first security case.

2) Case 2, we also prove that our scheme is secure via two games G_0 and G_1, where the game G_0 is the same as the original scheme in real world with keyword ciphertext $CT_{0,W} = (C, C_0, \{C_{i,1}, C_{i,2}, C_{i,3}\}_{i\in[m]})$. Based on the game G_0, we replace a parameter of the keyword ciphertext with a uniformly chosen random number $Z \in \mathbb{G}_T$ to construct the game G_1, where the keyword ciphertext $CT_{1,W} = (Z, C_0, \{C_{i,1}, C_{i,2}, C_{i,3}\}_{i\in[m]})$. We can obtain, if the adversary $\mathcal{A}$ can distinguish the game G_0 from G_1, then the challenger $\mathcal{C}$ can also break the decisional (q-2) assumption. For simplicity, we omit the insignificant process and parameters in the following two games.

Game G_0: The same as what is performed in the original scheme in real-world.

- **Setup:** Similar to **Setup** in the first case, however, in the algorithm Setup(1^ℓ), the challenger $\mathcal{C}$ implicitly set $\alpha = xy$, where x and y are both set in the decisional (q-2) assumption and α is properly distributed, then it picks two random numbers $\tilde{u}, \tilde{h} \in \mathbb{Z}_p$, and provides the following system public parameter pp to the adversary $\mathcal{A}$,

$$pp = (\mathbb{D}, g = g, u = g^{\tilde{u}} \cdot \prod_{i\in[m]} g^{y/b_i^2},$$
$$h = g^{\tilde{h}} \cdot \prod_{i\in[m]} \left(g^{xz/b_i}\right) \cdot \prod_{i\in[m]} \left(g^{y/b_i^2}\right)^{-\mathbb{W}^*_{b,i}},$$
$$\omega = g^x, E(*), H_0(*), H_1(*), p', q', g, v, f(*), \mathbb{F},$$
$$\hat{e}(g,g)^{\alpha} = \hat{e}(g^x, g^y)),$$

where x multiplied by y in α is information-theoretically hidden from the perspective of the adversary $\mathcal{A}$, and u, ω, h are properly distributed. Note that, all these parameters can be computed by the challenger $\mathcal{C}$ using suitable parameters from the decisional (q-2) assumption and the challenge keyword provided by the adversary $\mathcal{A}$. Next, the challenger $\mathcal{C}$ runs KeyGen(*) to generate a public/private key pair and other parameters required for CCS and user, respectively. Finally, the challenger $\mathcal{C}$ runs KeyParUpdate(*) to obtain other parameters as required for keeping the game running.

- **Query phases 1:** The adversary $\mathcal{A}$ adaptively issues a polynomial number of queries to the challenger $\mathcal{C}$ for the trapdoors, the challenger $\mathcal{C}$ runs Trapdoor(*) with a keyword LSSS access structure $LAS_{\mathbb{M},\rho,\mathbb{W}} = \left(\mathbb{M}, \rho, \{\mathbb{W}_{\rho(i)}\}\right)$ to generate the required trapdoor, where $LAS_{\mathbb{M},\rho,\mathbb{W}}$ are not satisfied by the keyword sets $\mathbb{W}_0^*$ nor $\mathbb{W}_1^*$. Additionally, in the algorithm Trapdoor(*), due to $\mathbb{W}_b^*$ doesn't satisfy the keyword LSSS access structure $LAS_{\mathbb{M},\rho,\mathbb{W}}$, as mentioned in the access structure and LSSS definition, there exists a vector $\vec{w} = (w_1, w_2, \ldots, w_n)^{\perp} \in \mathbb{Z}_p^n$ such that its first component $w_1 = 1$ and $< M_i, \vec{w} >= 0$ for all $i \in [\ell]$ such that $\rho(i) \in \mathbb{W}_b^*$. The challenger $\mathcal{C}$ calculates the vector $\vec{w}$ using linear algebra, and the vector $\vec{y}$ that will be shared by the keywords is implicitly set as $\vec{y} = xy\vec{w} + (0, \tilde{y}_2, \tilde{y}_3, \ldots, \tilde{y}_n)^{\perp}$, where $\tilde{y}_2, \tilde{y}_3, \ldots, \tilde{y}_n \in \mathbb{Z}_p$, This vector $\vec{y}$ is properly distributed because its first element is set as $xy = \alpha$ and other elements are uniformly random number chosen from $\mathbb{Z}_p$.Thus for each row $i \in [\ell]$, the share is
$$\begin{aligned}\lambda_i &=< M_i, \vec{y} > \\ &= xy < M_i, \vec{w} > + < M_i, (0, \tilde{y}_2, \tilde{y}_3, \ldots, \tilde{y}_n)^{\perp} > \\ &= xy < M_i, \vec{w} > + \vec{\lambda}_i\end{aligned}$$
As we mentioned above, for each row i of $\mathbb{M}$, for which $\rho(i) \in \mathbb{W}_b^*$, we have $< M_i, \vec{w} >= 0$. Therefore, $\lambda_i = \vec{\lambda}_i =< M_i, (0, \tilde{y}_2, \tilde{y}_3, \ldots, \tilde{y}_n)^{\perp} >$, which is a value known to the challenger $\mathcal{C}$, who further randomly picks $t_i \in \mathbb{Z}_p$ and outputs a trapdoor $Td_{\mathbb{W}} = (UID_A, (\mathbb{M}, \rho), \{T_{i,0}, T_{i,1}, T_{i,2}, T_{i,3}\}_{i\in[l]}, T_4, T_5)$ as we described in the algorithm Trapdoor(*), however, for each row i of $\mathbb{M}$, for which such that $\rho(i) \notin \mathbb{W}_b^*$, the challenger $\mathcal{C}$ randomly picks $\tilde{t}_i \in \mathbb{Z}_p$ and set implicitly
$$t_i = -y < M_i, \vec{w} > + \sum_{j\in[m]} \frac{xzb_j<M_i,\vec{w}>}{H_1\left(\mathbb{W}_{\rho(i)} \| H_1(\alpha\|UID_o)\right) - H_1\left(\mathbb{W}^*_{b,j} \| H_1(\alpha\|UID_o)\right)} + \tilde{t}_i$$
Obviously, t_i is properly distributed due to $\tilde{t}_i$ is chosen randomly, and the intuition behind this choice is that, we can see the exponent y raises the power of ω to the secret $\alpha = xy$. However, this choice also results to the exponent xyz/b_i from h, which can be cancelled by the provided exponent xzb_i on the part y/b_i^2. Therefore, the challenger $\mathcal{C}$ can compute the following parameters of the trapdoor using the settings in the decisional (q-2) assumption. For the simplicity of equations, here we let $R(i_0, i_1) = H_1(\mathbb{W}_{\rho(i_0)} \| H_1(\alpha\|UID_o)) - H_1(\mathbb{W}^*_{b,i_1} \| H_1(\alpha\|UID_o))$.

$$\begin{aligned}T_{i,0} &= g^{\lambda_i}\omega^{t_i} \\ &= g^{xy<M_i,\vec{w}>+\vec{\lambda}_i} \\ &\quad \cdot g^{-xy<M_i,\vec{w}>+\sum_{j\in[m]}\left(x^2zb_j<M_i,\vec{w}>\right)/R(i,j)} \cdot \omega^{\tilde{t}_i} \\ &= g^{\vec{\lambda}_i} \cdot \prod_{j\in[m]} \left(g^{x^2zb_j}\right)^{<M_i,\overline{w}>/R(i,j)} \cdot \omega^{\tilde{t}_i},\end{aligned}$$

$$\begin{aligned}T_{i,1} &= \left(u^{H_1\left(\mathbb{W}_{\rho(i)}\|H_1(\alpha||UID_o)\right)}h\right)^{-t_i} \\ &= \left(g^{(H_1(\mathbb{W}_{\rho}(i)|H_1(\alpha\|UID_o))\tilde{u}+\tilde{h})} \cdot \prod_{j\in[m]} g^{xz/b_j}\right. \\ &\quad \left.\cdot \prod_{j\in[m]} g^{yR(i,j)/b_i^2}\right)^{y<M_i,\vec{w}>-\Sigma_{j\in[m]}(xzb_j<M_i,\vec{w}>)/R(i,j)} \\ &\quad \cdot \left(u^{H_1\left(\mathbb{W}_{\rho(i)}\|H_1(\alpha||UID_o)\right)}h\right)^{-\tilde{t}_i} \\ &= g^{y<M_i,\vec{w}>\left(\left(H_1\left(\mathbb{W}_{\rho(i)}\|H_1(\alpha||UID_o)\right)\right)\tilde{u}+\tilde{h}\right)} \\ &\quad \cdot \prod_{j\in[m]} g^{-xzb_j\left(\left(H_1\left(\mathbb{W}_{\rho(i)}\|H_1(\alpha||UID_o)\right)\right)\tilde{u}+\tilde{h}\right)<M_i,\vec{w}>/R(i,j)} \\ &\quad \cdot \prod_{j\in[m]} g^{xyz<M_i,\vec{w}>/b_j} \\ &\quad \cdot \prod_{j,k\in[m,m]} g^{-(xz)^2b_k<M_i,\vec{w}>/b_jR(i,k)} \\ &\quad \cdot \prod_{j\in[m]} g^{y^2<M_i,\vec{w}>R(i,j)/b_j^2} \\ &\quad \cdot \prod_{j,k\in[m,m]} g^{-xyz<M_i,\vec{w}>b_kR(i,j)/b_i^2R(i,k)} \\ &\quad \cdot \left(u^{H_1\left(\mathbb{W}_{\rho(i)}\|H_1(\alpha||UID_o)\right)}h\right)^{-\tilde{t}_i} \\ &= g^{y<M_i,\vec{w}>\left(\left(H_1\left(\mathbb{W}_{\rho(i)}\|H_1(\alpha||UID_o)\right)\right)\tilde{u}+\tilde{h}\right)} \\ &\cdot \prod_{j\in[m]} \left(g^{xzb_j}\right)^{-\left(\left(H_1\left(\mathbb{W}_{\rho(i)}\|H_1(\alpha\|UID_o)\right)\right)\tilde{u}+\tilde{h}\right)<M_i,\vec{w}>/R(i,j)} \\ &\cdot \prod_{j,k\in[m,m]} \left(g^{(xz)^2b_k/b_j}\right)^{-<M_i,\vec{w}>/R(i,k)} \\ &\cdot \prod_{j\in[m]} \left(g^{y^2/b_j^2}\right)^{<M_i,\vec{w}>/R(i,j)} \\ &\cdot \prod_{j,k\in[m,m],j\neq k} \left(g^{xyzb_k/b_j^2}\right)^{-<M_i\vec{w}>R(i,j)/b_i^2R(i,k)} \\ &\cdot \left(u^{H_1\left(\mathbb{W}_{\rho(i)}\|H_1(\alpha||UID_o)\right)}h\right)^{-\tilde{t}_i},\end{aligned}$$

$$\begin{aligned}T_{i,2} &= g^{H_1\left(\hat{e}\left(g^{H_1(H_1(\alpha||UID_S)\|t_0\|UID_A)}, pk_S\right)^{H_1\left(pw_A\|t_0\|UID_A\right)}\right)} g^{t_i} \\ &= g^{H_1\left(\hat{e}\left(g^{H_1(H_1(\alpha||UID_S)\|t_0\|UID_A)}, pk_S\right)^{H_1\left(pw_A\|t_0\|UID_A\right)}\right)} \\ &\quad \cdot (g^y)^{-<M_i,\vec{w}>} \\ &\quad \cdot \prod_{j\in[m]} \left(g^{xzb_j}\right)^{<M_i,\vec{w}>/R(i,j)} \cdot g^{-\tilde{t}_i}.,\end{aligned}$$

$$T_{i,3} = T_{i,1}^{H_1(\alpha||UID_A)},$$

$$T_4 = g^{H_1(pw_A\|t_0\|UID_A)},$$

$$T_5 = g^{H_1(H_1(\alpha\|UID_S)\|t_0\|UID_A)}.$$

Then the challenger $\mathcal{C}$ sends them to the adversary $\mathcal{A}$.

- **Challenge:** The adversary $\mathcal{A}$ submits the declared keyword sets $\mathbb{W}_0^*$ and $\mathbb{W}_1^*$ with a file to the challenger

$\mathcal{C}$, who runs Encrypt(*) on $\mathbb{W}_b^*$ to obtain the challenge ciphertext $CT^*_{\mathbb{W}_b}$, and then provides $CT^*_{\mathbb{W}_b}$ to the adversary $\mathcal{A}$ along with other parameters as required, the adversary $\mathcal{A}$ runs Check(*) to verify whether there exists a trapdoor satisfying the keyword ciphertext. However, in the algorithm Encrypt(*), the challenger $\mathcal{C}$ implicitly sets $s = z$ using the settings from the decisional (q-2) assumption, and further sets $r_i = b_i$ for every $i \in [m]$. Obviously, these parameters are all properly distributed since $z, b_1, \ldots, b_q$ are information theoretically hidden from the perspective of the adversary $\mathcal{A}$. Thus, the challenger $\mathcal{C}$ can compute the following parameters of the ciphertext $\mathbb{W}_b^*$ using the settings in the decisional (q-2) assumption:

$$\begin{aligned}
&C = \left(Z = \hat{e}(g,g)^{xyz}\right), C_0 = g^s = g^z,\\
&C_{i,1} = g^{r_i} = g^{b_i}\\
&C_{i,2} = \omega^{-s}\left(u^{H_1(\mathbb{W}_{iV}\|sk_o)}h\right)^{r_i}\\
&\quad = \left(g^{b_i}\right)^{\left(H_1\left(\mathbb{W}^*_{b,i}\|sk_o\right)\right)\tilde{u}+\tilde{h})} \cdot \prod\nolimits_{j\in[m]} g^{xzb_i/b_j}\\
&\quad\ \cdot \prod\nolimits_{j\in[m]} \left(g^{yb_i/b_j^2}\right)^{\mathbb{W}^*_{b,i}-\mathbb{W}^*_{b,j}} \cdot g^{-xz}\\
&\quad = \left(g^{b_i}\right)^{\left(H_1\left(W^*_{b,i}\|sk_o\right)\right)\tilde{u}+\tilde{h})} \cdot \prod\nolimits_{j\in[m],i\neq j} g^{xzb_i/b_j}\\
&\quad\ \cdot \prod\nolimits_{j\in[m],i\neq j} \left(g^{yb_i/b_j^2}\right)^{\mathbb{W}^*_{b,i}-\mathbb{W}^*_{b,j}}\\
&C_{i,3} = \left(C_{i,2}\right)^{sk_o}
\end{aligned}$$

Note that, since the challenger $\mathcal{C}$ sets $r_i = b_i$, one of the exponents xzb_i/b_j is raised to xz, which cancels $\omega^{-s} = g^{-xz}$ in above parameter $C_{i,2}$.

- **Query phase 2:** The same as what is performed in Query phase 1.
- **Guess:** The adversary $\mathcal{A}$ outputs its guess b' for b.

Game G_1: The same as the game G_0, except that the parameter Z is replaced with a uniformly chosen random number $r \in \mathbb{G}_T$, i.e. $C = (Z = r)$ in Query phase 1 and 2.

Obviously, in the game G_0, $Z = \hat{e}(g,g)^{xyz}$, which is identical to the original scheme fom the perspective of the adversary $\mathcal{A}$, whereas Z is a random number uniformly chosen from $\mathbb{G}_T$ in the game G_1, thus, if the decisional (q-2) assumption holds, it is impossible for the adversary $\mathcal{A}$ to distinguish the game G_0 from G_1. Therefore, our scheme can resist the attacks that my occur in the second security case. The proof for these two security cases completes the proof of Lemma 2. □

By combining the proof of Lemma 1 and Lemma 2, it completes the proof of Theorem 1. □

4.5 Discussion and Analysis

1. ***Keyword confidentiality.*** The efficient PEKS approach in our work is derived from the KP-ABE scheme (see Appendix C in [34]). However, a keyword value guessing attack may exist if we directly transform and integrate it into our scheme. For example, if we directly transform the KP-ABE scheme into a keyword SE scheme, assume there is a keyword ciphertext $CT_{\mathbb{W}} = (C, C_0, \{C_{i,1}, C_{i,2}\}_{i\in[m]})$, where

$$C = \hat{e}(g,g)^{\alpha s}, C_0 = g^s,$$
$$C_{i,1} = g^{r_i}, C_{i,2} = \omega^{-s}(u^{H_1(\mathbb{W}_{iV})}h)^{r_i}.$$

Then $\hat{e}(C_{i,2}, g) = \hat{e}(\omega^{-1}, C_0)\hat{e}(C_{i,1}, u^{H_1(\mathbb{W}_{iV})}h)$ holds, the "curios" CCS can guess the keyword value $\mathbb{W}_{iV}$ embedded in the ciphertext $CT_{\mathbb{W}}$ by checking whether the above equation holds with a guessed value and system public paramter. Whereas, in our work, given keyword ciphertext $CT_{\mathrm{W}} = (C, C_0, \{C_{i,1}, C_{i,2}, C_{i,3}\}_{i\in[m]})$, where

$$C = \hat{e}(g,g)^{\alpha s}, C_0 = g^s, C_{i,1} = g^{r_i},$$
$$C_{i,2} = \omega^{-s}(u^{H_1(\mathbb{W}_{iV}\|sk_o)}h)^{r_i}, C_{i,3} = (C_{i,2})^{sk_o}.$$

We also have the following equation holds,
$$\hat{e}(C_{i,2}, g) = \hat{e}(\omega^{-1}, C_0)\hat{e}(C_{i,1}, u^{H_1(\mathbb{W}_{iV}\|sk_o)}h),$$
however, the "curios" CCS shall have to first guess the private key sk_o of the data owner if it attempts to guess the keyword value $\mathbb{W}_{iV}$ by checking whether the above equation holds. Additionally, given a trapdoor
$$Td_{\mathbb{W}} = (UID_u, (\mathbb{M}, \rho), \{T_{i,0}, T_{i,1}, T_{i,2}, T_{i,3}\}_{i\in[\ell]}, T_4, T_5),$$
where

$$\begin{gathered}
T_{i,0} = g^{\lambda_i}\omega^{t_i},\\
T_{i,1} = (u^{H_1(\mathbb{W}_{\rho(i)}\|H_1(\alpha\|UID_o))\cdot}h)^{-t_i},\\
T_{i,2} = g^{H_1(\Delta)}g^{t_i},\\
T_{i,3} = T_{i,1}^{H_1(\alpha\|UID_u)},\\
T_4 = g^{H_1(pw_u\|t_0\|UID_u)},\\
T_5 = g^{H_1(H_1(\alpha\|UID_s)\|t_0\|UID_u)},\\
\Delta = \hat{e}(g^{H_1(H_1(\alpha\|UID_s)\|t_0\|UID_u)}, pk_s)^{H_1(pw_u\|t_0\|UID_u)}.
\end{gathered}$$

Obviously, the malicious attacker cannot forge a correct trapdoor with a set of guessed keyword values and determine whether a trapdoor matches the given keyword ciphertext without the master private parameter, the private key of CCS and the data user's password.

2. ***Key self-retrievability/recoverability.*** If cloud user can self-retrieve the subsequent content encryption keys after obtaining the initial key materials, which significantly reduces the unnecessary network overhead, and minimizes the security risks occurs in communication. If cloud user can recover the temporarily lost encryption key without frequently communicating with CEKMC, which further reduces the computational overhead of CEKMC.

In our work, after generating the initial key seed based on current password with the help of CEKMC, to reduce storage cost, the data owner can build parts of OHBT, FHC and BHC to temporarily derive the content encryption key for current time-period only, hands the burdensome content encryption key management over to CEKMC, who is responsible for storing all key materials including the history ones distributed to data users who are authorized to access the content outsourced by that data owner. See Fig. 5, according to our design of OHBT, FHC and BHC, CEKMC just generates key components (K_{00}, k'_{seed}) by calculating the lowest common ancestor nodes using retrieved secret keys belongs to that data user from OHBT, as well as the starting hash values: $(H_0(FHS)^1, H_0(BHS)^3)$ from FHC and BHC for that data user, who can self-recover the content encryption keys for all access time-periods using key parameters kp distributed by CEKMC. Additionally, the key parameters kp are distributed along with trapdoor, due to the authorized access time-periods may be dynamically allocated to each data user by data owner, and providing a certain amount of key redundancy to avoid that the data user may not receive the initial key materials if possible.

3. ***Collusion attack resistance.*** Suppose there are two malicious data users UID_a and UID_b without access time-period intersection, and two disjoint access time-period segments $[j,k]$ and $[\ell,v]$ such that $(1 \leq j < k < \ell < v)$ are allocated to them respectively, i.e. UID_a is an authorized user of the content outsourced during the time-periods: $\mathrm{TP}\#j, \#j+1, \ldots, \#k$, whilst UID_b is an authorized user of the content outsourced during the time-periods: $\mathrm{TP}\#\ell, \#\ell{+}1, \ldots, \#v$. In our work, CEKMC generates the relevant key components by calculating the lowest common ancestor nodes using those retrieved secret keys belongs to UID_a from OHBT for $\mathrm{TP}\#j, \#j+1, \ldots, \#k$. Using the key components, UID_a can derive all secret keys located on the leaf nodes of OHBT corresponding to the access time-period segments $[j,k]$. However, if the current time-period $\mathrm{TP}\#n$ satisfies $(n > k)$ or $(n < j)$, due to UID_a has no secret key corresponding to time-period $\mathrm{TP}\#n$, obviously UID_a cannot obtain the content encryption key dek_n. Similarly, UID_b cannot obtain the encryption key for the time-period $\mathrm{TP}\#n$ where $(n > v)$ or $(n < \ell)$. In addition, although UID_a and UID_b can obtain the hash values of FHC and BHC for the time-period segments $[k+1, \ell-1]$ via cooperation, due to UID_a and UID_b have no secret keys corresponding to the time-period segments $[k+1, \ell-1]$, thus the corresponding content encryption key cannot be obtained by them. Besides, our scheme can also prevent a single malicious user from using two or more disjoint allocated access time-period segments belongs to her or him to perform the above attack.

4. ***Efficient multi-keyword search on encrypted content.*** Our scheme derived from [34] essentially inherits the attribute such that the number of system public parameters has no relation with the size of the keyword sets. Meatime, the size of the keyword sets for the LSSS access structure of trapdoor can be unlimitedly large enough to satisfy any combination search containing any number of keywors in the form of a boolean formula expression consisting of "AND" and "OR" gates. Thus, Our scheme is more suitable for the real-world applications.

5 Performance Evaluation

In this section, we evaluate the performance of PAC-MC in terms of computation, communication and storage costs. As the process KeyParUpdate(*) is not always running but executed when password expires, it is a relatively independent sub-process of PAC-MC. In essence, PAC-MC can be viewed as an expressive PEKS solution, without considering the sub-process KeyParUpdate(*). For the proposed PEKS, we take it out independently to compare with other few schemes in TABLE 1, which describes the asymptotic computation, communication and storage overheads among our scheme [25], [38], [39]. Note that the schemes in [25], [38], [39] and our scheme, are all PEKS supporting multi-keyword search for outsourced data. To be in line with the representations in [25], we ignore all hash operation in all schemes. Here we define the following symbols: $|pp|$ denotes the size of system public parameter, $|msk|$ denotes the size of system master private parameter, $|CT|$ denotes the size of keyword ciphertext, $|T_{\mathbb{M},\rho(i)}|$ denotes the size of trapdoor, $|\mathbb{M}|$ denotes the size of access tree, k denotes the length of the vector for the keyword ciphertext in [39], l denotes the number of keywords embedded in the access tree, n denotes the maximum number of keywords for the system, m denotes the number of keywords in a keyword set related with the ciphertext, E denotes the exponentiation operation on the element of groups $(\mathbb{G}, \mathbb{G}_T)$, P denotes the bilinear pairing operation, $\mathcal{X}_1$ denotes the number of elements in $T_{\mathbb{M},\rho} = \{\mathfrak{T}_1, \mathfrak{T}_2, \ldots, \mathfrak{T}_{\mathcal{X}_1}\}$, $\mathcal{X}_2$ denotes $|\mathfrak{T}_1|+|\mathfrak{T}_2|+\cdots+|\mathfrak{T}_{\mathcal{X}_1}|$, and $\mathcal{X}_3$ denotes the number of primed keywords in a search predicate, as stated in [38], N is the number of attributes chosen by the data owner; $|S|$ denotes the number of the attributes appeared in each access tree [40]; Γ denotes set of all roles associated with a ciphertext [42], $\Gamma_\emptyset$ denotes system authorities associated with a ciphertext, S_{IDu} denotes set of roles associated with the user IDu [42] ; S_{IDi} denotes the attribute set possessed by the ith user, Γ denotes the access policy [41], n_{or} denotes number of "OR" gates in an access policy, K_c denotes the total number of keywords associated with an encrypted index; n_t denotes the number of attributes associated with a trapdoor [41].

We first analyze the asymptotic complexities including the computation, communication and storage overheads of our solution and other PEKS in [25], [38], [39], [40], [41], [42], as is shown in TABLE 1, where the PEKS schemes in [38], [39] are all developed based on composite-order groups, whilst our scheme and [25] are both solutions supporting unbounded number of keywords with bool formulas search based on prime order groups. In [25], the PEKS scheme designed with the prime order groups using pairing-friendly elliptic curve shall have a clear advantage over the one based on the composite order groups. Our scheme is more efficient than the one in [25], almost saves 50% of the cost in [25], in terms of the computation, communication and storage overheads, which is due to the scheme in [25] is designed using a "linear splitting" technique [25] on each keyword value-related element embedded in the ciphertext during keyword encryption, and also it re-randomizes the element on each keyword value in trapdoor during trapdoor generation. The computational overhead of the algorithms Trapdoor(*) and Encrypt(*) have a linear relation with the number of the keywords used during the trapdoor and ciphertext generation, respectively, similarly, the lengths of $|T_{\mathbb{M},\rho(i)}|$ and $|CT|$ are both showing such a relation as well. In [40], all overheads are determined by the attributes of the access tree of the user, the size of access tree is bigger than the number of keywords of the content, therefore, our scheme has relatively less overheads than the scheme in [40]. The storage and communication overhead [42] for ciphertext and trapdoor mainly depends on the set of all roles, the size of the secret key possessed by a user mainly depends on the number of SAs (i.e., number of organizations) and the number of roles associated with that user. Overall, the scheme in [42] has more overheads than ours in total due to its functionalities, however, the scheme in [42] has more features than ours, it is difficult to compare them directly since the design objectives and application scenarios of the two schemes are different. The storage cost of the scheme in [41] is similar to ours in complexity, if considering the number of "OR" gates in an access policy and total number

TABLE 1
Comparison of Computation, Storage and Communication Overhead among our new PEKS, [25], [38], [39], [40], [42] and [41]

	[25]	[38]	[39]	[40]	[42]	[41]	Ours
$\|pp\|$	9	n+4	2k+3	6	$2+k$	$6+2\|U_A\|$	5
$\|msk\|$	5	n+2	2k+4	2	4	$7+2\|U_A\|$	2
$\|T_{\mathbb{M},\rho}\|$	6l+2+$\|\mathbb{M}\|$	6k+1+$\|\mathbb{M}\|$	3l+$\|M\|$	3$\|S\|$ +1+$\|M\|$	$(3+2\|\Gamma\|)$	$\|S_{IDu}\|+4$	4l+2+$\|\mathbb{M}\|$
$\|CT\|$	5m+2	m+2	2k+1	2m+4	$2\|\Gamma\|+\|G_T\|$	K_c+2 $+n_{or}+\|\Gamma\|$	3m+2
KeyGen(*)	E	-	(6k)E	3$\|S\|E$	$(9k+1)E$	$7E$	E
Trapdoor(*)	(16l+1)E	(4l)E	(6k)E	(2$\|S\|$+l+1)E	$(3+2\|S_{IDu}\|)E$	User: $(2n_t+5)E$ $+2P$ CSP: $2E$	(6l+4)E+P
Encrypt(*)	(7m+2)E	(m+2)E+P	(4k+1)E	$(4+N)E$	(4+$\|\Gamma\|$+$\|\Gamma_\emptyset\|$+1)E	$(n_{or}+2K_c+3)E$ $+(n_{or}+1)P$	(4m+3)E
Test(*) or Check(*)	$\leq(\mathcal{X}_2+1)E$ $+(6\mathcal{X}_2+1)P$	$\leq(\mathcal{X}_2)E+$ $(2\mathcal{X}_3+2\mathcal{X}_2)P$	(2k+1)P	(2N+1)P+E	(2 +$\|\Gamma_\emptyset\|$)E+3P+ ($\|\Gamma\|$+1)E+ (2+$\|\Gamma\|$)P +($\|\Gamma\|$ +$\|\Gamma_\emptyset\|$)E+(1+$\|\Gamma\|$)P	VA: $(n_t+2)E$ $+2P$ CSP: $3E+4P$	$\leq(\mathcal{X}_2+1)E$ +$(3\mathcal{X}_2+5)P$
Group Order	Prime	Composite	Composite	Prime	Prime	Prime	Prime

of keywords associated with an encrypted index to be the same size of keywords in the ciphertext. In the Encryption phase, it shows that the encryption cost of the scheme mainly depends on the number of both "OR" gates in the access policy and keywords associated with an encrypted index. In the trapdoor generation phase, the scheme uses an interactive protocol between a user and the CSP to generate a trapdoor. Due to the remote interactive protocol, the scheme may have more communication overhead than others. In the Search phase, the computation cost is mainly incurred by the CSP and VA who has to perform cryptographic operations during keyword search operation over the encrypted indexes, it mainly depends on the number of attributes associated with a trapdoor. The communication cost in Encryption and Search phase, and the computation cost in all phases of this scheme is similar to ours in complexity, if considering all the number of "OR" gates in an access policy, the number of attributes associated with a trapdoor and the total number of keywords associated with an encrypted index to be the same size of keywords in the ciphertext, the number of keywords embedded in the access tree and X_2. Also, it is difficult to compare them directly since the design objectives and application scenarios of the two schemes are different.

The following evaluations are performed on the settings: CPU: Intel(R) Core(TM) i5-7200U@3.10 GHz max (2 core, 2.5GHz and 2.7GHz, the 7th generation), L3 Cache: 3MB, RAM: DDR3 8.0GB; OS: 64-bit Ubuntu 15.10 (Codename: wily), basic support software: pbc-0.5.14, gmp-6.1.2, m4-1.4.17, bison-3.0, flex-2.5.4, charm-0.43 and Python 3.4. For simplicity, we replace the pseudorandom function ensemble $\mathbb{F}$ with SHA256, the parameter p' and q' are chosen randomly such that $|p'| = 2048$, $|q'| = 256$. Additionally, we use the 160-bit elliptic curve groups based on the super singular curve $y^2 = x^3 + x$ over a 512-bit finite field.Note that, this curve is a super singular elliptic curve with the bilinear pairing being symmetric Type 1 pairing on it, thus here we have $\mathbb{G}_1 = \mathbb{G}_2$, but we still denote them with asymmetric

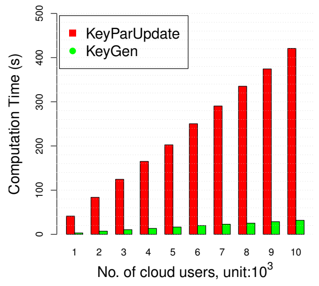


Fig. 6. Computational overheads of KeyParUpdate(*) and KeyGen(*) run by CPSV with respect to the number of cloud users (owners).

groups to meet the requirement of the Charm routines. By averaging the times of respective algorithms over 10 different instantiations of each basic operation, the basic performance parameters obtained on the test laptop computer are shown as follows: In PBC library (version 0.5.14), the pairing operation can be completed in approximately 4.5ms, the exponentiation operations in groups $\mathbb{G}_{q'}, \mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ can be completed in approximately 5.6ms, 3.0ms, 3.0ms and 0.4ms respectively, meanwhile, in our work, the randomly selecting elements (by reading data from Linux kernel's */dev/urandom*) are also significant operations costing CPU cycles, which require about 0.3ms for $\mathbb{Z}_p$, 4.5ms for $\mathbb{G}_1$ and 5.0ms for $\mathbb{G}_2$, respectively.

Assume that a password lifetime (PL) is 7 days (1 week) and a PL consists of 7 time-periods, i.e. 1 time-period equals 1 day. For data owner, the computational overheads for running the algorithm KeyParUpdate(*) is about 29.3ms. Without loss of generality, each cloud user may also play the role of a data owner, then CPSV shall be responsible for assisting all data owners to do most work of content encryption key management, the performance of the algorithms KeyParUpdate(*) and KeyGen(*) run by CPSV for those cases containing different number of cloud users are evaluated, which is shown in Fig. 6, where only one CPSV exists, the number of cloud users varies from 1000 to 10000 with step length 1000. Obviously, the computational overheads of the algorithms KeyParUpdate(*) and KeyGen(*) run by CPSV have an approximately linear relation with the number of cloud users, respectively. The computational overhead for running the algorithm KeyParUpdate(*) with 10000 cloud users is about 420.8s, and 31.7s for the algorithm KeyGen(*) running with 10000 cloud users during system initialization, which is quite suitable for the powerful server(s) that acting as CEKMC, due to the key materials updating time required by KeyParUpdate(*) is far less than 1 time-period (day). Actually, the size of time-period could be configured as per the requirement of application [8].

The performance of the algorithms Encrypt(*) and Trapdoor(*) for those cases containing different number of keywords are evaluated, which is shown in Fig. 7(a), where the number of keywords varies from 1 to 10 with step length 1, from which, we can see that, the computational overheads of the algorithms Encrypt(*) and Trapdoor(*) have an approximately linear relation with the number of keywords used during keyword ciphertext and trapdoor generation, respectively. The computation time taken to generate a trapdoor with 10 keywords is about 199.4ms, and 130.8ms for generating a keyword ciphertext with 10 keywords, which are both quite suitable for TGC and data owner, respectively. Additionally, the performance on the algorithm Trapdoor(*) run by TGC for a large number of cloud users concurrently is evaluated and shown in Fig. 7(b), 7(c), 7(d), 7(e) and 7(f), where the number of keywords varies from 1 to 10 with step length 1, and the number of cloud users varies from 1000 to 10000 with step length 1000, from which, we can see that, the computational overhead of the algorithm Trapdoor(*) run by TGC have an approximately linear relation with the number of cloud users. The computational overhead for running the algorithm Trapdoor(*) to generate 10000 trapdoors concurrently with 10 keywords is about 1748.2s, which can be further reduced vastly for the powerful server(s) supporting parallel computing technology and HSM that acting as TGC. Moreover, the number of keywords in a searching query is normally less than 10, according to the searching query logs of search engines [25].

The storage overheads of CEKMC with respect to different number of cloud users in consecutive 8 PLs are evaluated, respectively, which is shown in Fig. 8, where the number of cloud users varies from 1000 to 10000 with step length 1000 for each PL, the maximum storage cost for 10000 cloud users in PL$\#1 \sim \#8$ are approximately 2502.4KB, 2814.7KB, 3131.1KB, 3439.7KB, 3752.2KB, 4064.7KB, 4377.2KB and 4689.7KB, respectively, from which, it is clear that the storage overheads of CEKMC have an approximately linear relation between different PLs. Also in each PL, the storage overheads of CEKMC have an approximately linear relation with the number of cloud users as well, which can be concluded from Fig 8(a), (b), (c) and (d). Due to CEKMC stores the key seed only for all content encryption keys generated in one PL for each cloud user, the memory capacity growth rate of key storage for CEKMC is about 313KB per 10000 cloud users only as the number of PLs increase, which is quite suitable for the powerful server(s) that acting as CEKMC. In addition, the key storage cost of TGC and cloud user is approximately 2.5KB and 2.2KB, respectively.

We also evaluated the communication overheads of CEKMC and KGC with respect to the number of cloud users when running KeyParUpdate(*) and system initialization, as well as the communication overheads of CEKMC and TGC with respect to the number of cloud users during trapdoor and key parameter distribution in consecutive 8 PLs, which are shown in Fig. 9 and Fig. 10, where a PL consists of 7 time-periods and the key components generated by calculating the lowest common ancestor nodes using secret keys from OHBT is maximized in consecutive PLs for cloud users. Clearly, the communication costs of CEKMC and KGC when running KeyParUpdate(*) and system initialization have an approximately linear relation with the number of cloud users, the maximum communication cost for 10000 cloud users when simultaneously running KeyParUpdate(*) and system initialization are about 5625KB and 24267KB, respectively. In addition, the maximum communication cost for 10000 cloud users in PL$\#1 \sim \#8$ are approximately 54844KB, 56094KB, 57344KB, 58594KB, 59844KB, 61094KB, 62344KB and 63594KB, respectively, the communication costs of CEKMC and TGC have an approximately linear relation between different PLs during the trapdoor and key parameter distribution. Also in each PL, the communication costs of CEKMC and TGC have an approximately linear relation with the number of cloud users as well, the communication overhead growth rate for CEKMC and TGC is about 1250KB per 10000 cloud users only as the number of PLs increase, which is quite modest for cloud-based applications with powerful server(s) that acting as CEKMC and TGC.

The performance of the algorithm Check(*) for those cases where both the access structure of trapdoor and the ciphertext with different number of keywords are evaluated, which is shown in Fig. 11, where 10 different access structures and their related trapdoors are constructed using 1$\sim$10 keywords, respectively, and also 4 different ciphertexts are generated using 10, 20, 30 and 40 keywords, respectively, based on the 4 ciphertexts and by averaging the time for each trapdoor over 10 instantiations, the computational overheads of the algorithm Check(*) are evaluated for these 10 trapdoors, respectively. Given the number of keywords in the trapdoor and the related keyword names, we assign each keyword value with a different integer for the sake of simplicity, in addition, in order to assess the worst case, for each keyword access structure of trapdoor, it is constructed using a balanced binary tree such that, the total number of leaf nodes is equal to the number of keywords in the trapdoor, the interior nodes of the tree are assigned with "AND" and "OR" to maximize $\mathcal{X}_2 = |\mathfrak{T}_1| + |\mathfrak{T}_2| + \cdots + |\mathfrak{T}_{\mathcal{X}_1}|$, i.e. the total keyword matching times in one search.

For the case containing a trapdoor with 1 keyword and a ciphertext with 10 keywords, the average computation time of the algorithm Check(*) is approximate 40ms, and

8. As an option, for the time-constricted applications, the new password could be prepared ahead of time and update the key materials with CEKMC in a backup way, to generate the content encryption keys in advance for the coming new PL. Moreover, most of the modern CEKMCs are equipped with powerful HSM (hardware security module) to offload the cryptographic operations, to further reduce the time for content key generation.

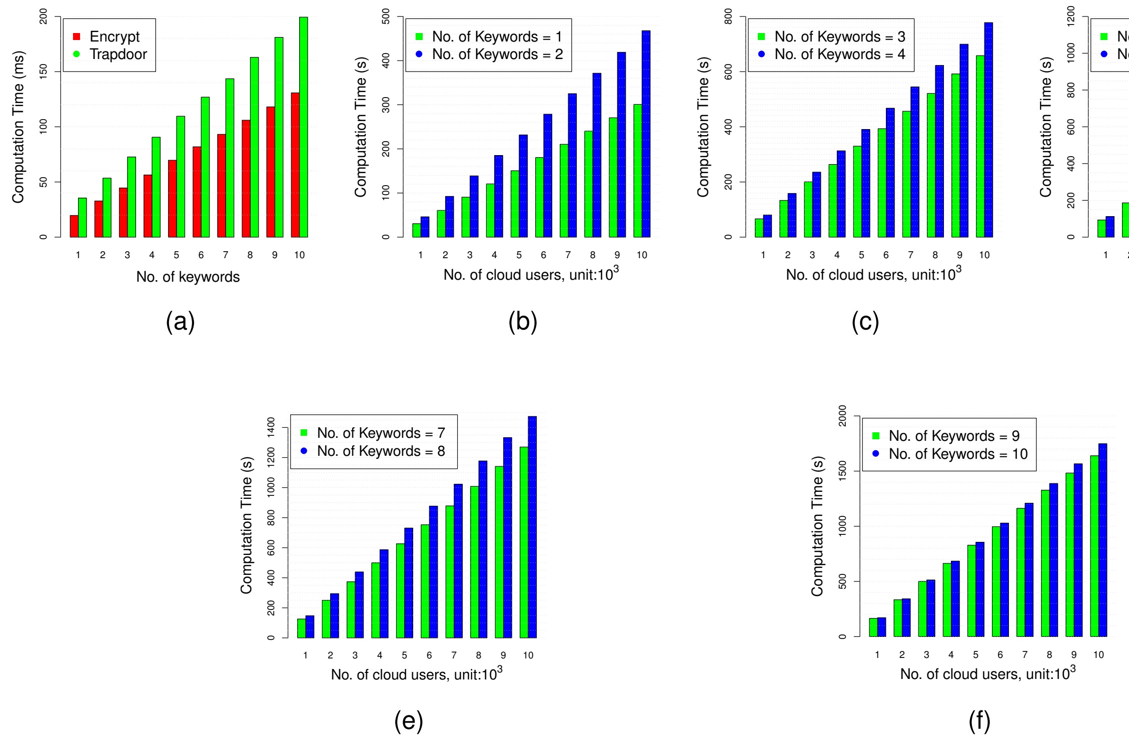


Fig. 7. (a) Computational overheads of Encrypt(*) and Trapdoor(*) run with respect to the number of keywords. (b) No. of keywords = 1 and 2, respectively. (c) No. of keywords = 3 and 4, respectively. (d) No. of keywords = 5 and 6, respectively. (e) No. of keywords = 7 and 8, respectively. (f) No. of keywords = 9 and 10, respectively.

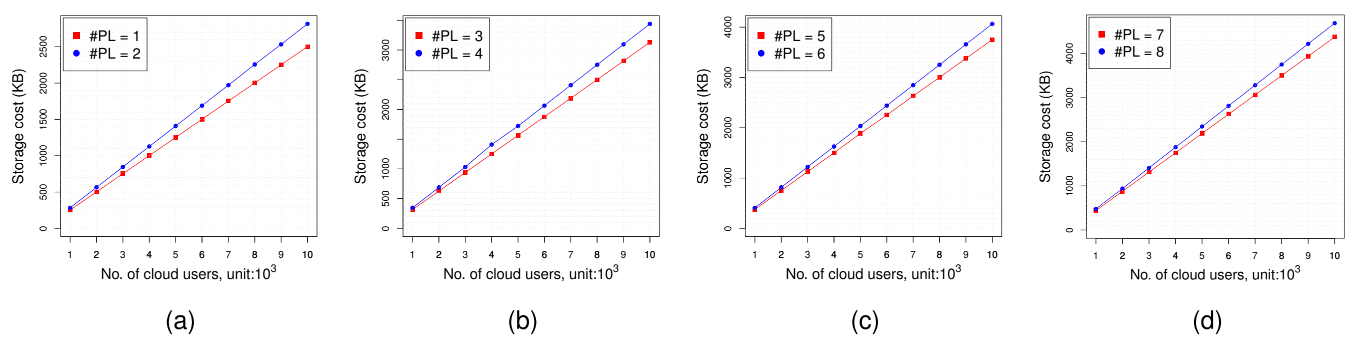


Fig. 8. Storage overheads of CEKMC with respect to the number of cloud users in consecutive 8 PLs. (a) $\#\mathrm{PL}$ = 1 and 2, respectively. (b) $\#\mathrm{PL}$ = 3 and 4, respectively. (c) $\#\mathrm{PL}$ = 5 and 6, respectively. (d) $\#\mathrm{PL}$ = 7 and 8, respectively.

3000ms for the case containing a trapdoor with 10 keywords and a ciphertext with 40 keywords, which can further be significantly reduced by the powerful server(s) acting as CCS. From Fig. 11, it is clear that, in the cases where the trapdoor contains 1$\sim$6 keywords, the computational overhead increases slowly as the number of keywords in ciphertext increases, however, in the cases where the trapdoor contains 7 or more keywords, the overhead increases dramatically as the number of keywords in ciphertext increases.

Based on the preliminary result mentioned above, we can draw the following trend that, as the number of keywords in the trapdoor increases, the computational overhead of the algorithm Check(*) will increase exponentially. This is due to the fact, in the algorithm Check(*), CCS needs to generate a set $I_{\mathbb{M},\rho}$ from $(\mathbb{M},\rho)$ of the trapdoor, where $(\mathbb{M},\rho)$ is a set of the minimum subsets satisfying $I_{\mathbb{M},\rho}$, and check whether there exists at least one subset $\mathfrak{T} \in I_{\mathbb{M},\rho}$ such that the trapdoor of each keyword in $\mathfrak{T}$ matches its keyword value in ciphertext after the corresponding keyword name being found in the keyword ciphertext of the file. To a large extent, the computational overhead of the algorithm Check(*) depends on both the number of keywords in the LSSS access structure of trapdoor and the number of keywords embedded in the ciphertext located on CCS, is mainly affected by the number of keywords in the trapdoor.

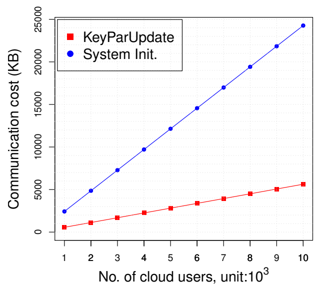


Fig. 9. Communication overheads of CEKMC and KGC with respect to the number of cloud users in KeyParUpdate(*) and System initialization.

6 Conclusion

In this paper, we proposed a new access control scheme PAC-MC, which is designed over the prime-order groups. First, the scheme can efficiently support multi-keyword search over encrypted data in any monotonic boolean formulas, and enable data owner to fully control the content encryption key using an updatable password based on the time-period. Meantime, the scheme supports the self-retrievability of content encryption key, which is suitable for the practical cloud-based media application with massive users. In addition, the authentication code or secret key generated based on the biological characteristics (e.g., Fingerprint, Iris, Face etc.) during authentication can also be processed similarly as password. By extending our scheme and combining the authentication based on biological characteristics, a new lightweight access control for cloud-based application in the internet of things (IOT) could be constructed in the future work.

References

[1] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in 2000 *IEEE Symposium on Security and Privacy, Berkeley, California, USA, May 14-17, 2000*,IEEE Computer Society, 2000, pp. 44–55.

[2] W. Sun, B. Wang, N. Cao, M. Li, W. Lou, T. Hou, and H. Li, "Privacy preserving multi-keyword text search in the cloud supporting similarity-based ranking," in *8th ACM ASIA Conference on Computer and Communications Security 2013*, May 2013.

[3] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *EUROCRYPTO 2004*, pp. 506–522, 2004.

[4] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," in *INFOCOM 2011*, pp. 829–837, 2011.

[5] Y. Hwang and P. Lee, "Public key encryption with conjunctive keyword search and its extension to a multi-user system," in *Pairing 2007*, pp. 2–22, 2007.

[6] E. Goh, "Secure indexes," IACR, *Cryptology ePrint Archive on October 7th*, pp. 1–18, 2003.

[7] Y. C. Chang and M. Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data," in *International Conference on Applied Cryptography and Network Security 2005*, vol. 3531, pp. 442–455.

[8] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions",in *ACM Conference on Computer and Communications Security 2006*, vol. 19, pp. 79–88, 2006.

[9] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren and W. Lou, "Fuzzy Keyword Search over Encrypted Data in Cloud Computing," in *2010 Proceedings IEEE INFOCOM, San Diego, CA*, 2010, pp. 1-5.

[10] M. Kuzu, M. S. Islam and M. Kantarcioglu, "Efficient Similarity Search over Encrypted Data," in *2012 IEEE 28th International Conference on Data Engineering, Washington, DC*, 2012, pp. 1156-1167.

[11] Y. Lu, "Privacy-preserving logarithmic-time search on encrypted data in cloud," in *Proceedings of 19th NDSS, San Diego, California, USA*, 2012.

[12] C. Orencik, E. Savas, "An efficient privacy-preserving multi-keyword search over encrypted cloud data with ranking," *J. Parallel and Distributed Databases*, 2014, vol. 32, no. 1, pp.119–60.

[13] C. Wang, N. Cao, K. Ren and W. Lou, "Enabling Secure and Efficient Ranked Keyword Search over Outsourced Cloud Data," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 8, pp. 1467-1479, Aug. 2012.

[14] N. Cao, C. Wang, M. Li, K. Ren and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," in *2011 Proceedings IEEE INFOCOM, Shanghai*, 2011, pp. 829-837.

[15] M. Bellare, A. Boldyreva, and A. O'Neill, "Deterministic and efficiently searchable encryption," in *Advances in Cryptology -CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007*, Proceedings, ser. Lecture Notes in Computer Science, vol. 4622. Springer, 2007, pp. 535–552.

[16] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," *J. Theory of Cryptography*, vol. 4392, pp. 535–554, 2007.

[17] N. Attrapadung, L. Benoît, "Functional Encryption for Public-Attribute Inner Products: Achieving Constant-Size Ciphertexts with Adaptive Security or Support for Negation," in *Public Key Cryptography - PKC 2010, 13th International Conference on Practice and Theory in Public Key Cryptography, Paris, France, May 26-28*, 2010.

[18] J. Katz, A. Sahai, B. Waters, "Predicate encryption supporting disjunctions, polynomial equations, and inner products," in *Proceedings of 27th annual international conference on the theory and applications of cryptographic techniques*, Berlin, Heidelberg, 2008, pp.146–62.

[19] E. Shi, J. Bethencourt, H. Chan, D. Song, A. Perrig, "Multi-dimensional range query over encrypted data," in *Proceedings of IEEE symposium on security and privacy. California*, 2007, pp.350–364.

[20] B. Waters, D. Balfanz, G. Durfee, D. Smetters, "Building an encrypted and searchable audit log," in *Proceedings of annual network and distributed security symposium, California*, 2004.

[21] A. Boldyreva, N. Chenette, Y. Lee, "Order-Preserving Symmetric Encryption," in *Proceedings of Advances in Cryptology, EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30*, 2009. Proceedings. DBLP, 2009.

[22] J. Yu, P. Lu, Y. Zhu, "Toward Secure Multi-Keyword Top-k Retrieval over Encrypted Cloud Data," *IEEE Transactions on Dependable and Secure Computing*, 2013, vol. 10, no. 4, pp.239-250.

[23] J. Shi, J. Lai, Y. Li, R. H. Deng, and J. Weng, "Authorized keyword search on encrypted data," in *Computer Security - ESORICS 2014 - 19th European Symposium on Research in Computer Security, Wroclaw, Poland, September 7-11, 2014. Proceedings, Part I, ser. Lecture Notes in Computer Science*, vol. 8712. Springer, 2014, pp. 419–435.

[24] Q. Zheng, S. Xu, and G. Ateniese, "VABKS: verifiable attribute-based keyword search over outsourced encrypted data," in *2014 IEEE Conference on Computer Communications, INFOCOM 2014, Toronto, Canada, April 27 - May 2*, 2014, pp. 522–530.

[25] H. Cui, Z. Wan, R. H. Deng, G. Wang and Y. Li, "Efficient and Expressive Keyword Search Over Encrypted Data in Cloud," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 3, pp. 409-422, 2016.

[26] K. He, J. Guo, J. Weng, J. Weng, J. K. Liu, X. Yi,"Attribute-Based Hybrid Boolean Keyword Search over Outsourced Encrypted Data" in *J. IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 1-12, 2018.

[27] K. Lee, S. Choi, D. Lee, J. Park, H. Park, and M. Yung,"Self-updatable encryption: Time constrained access control with hidden attributes and better efficiency", *Theoretical Computer Science*, vol. 667, pp. 51-92, 2017.

[28] K. Lee, D. Lee, J. Park, M. Yung, "CCA Security for Self-Updatable Encryption: Protecting Cloud Data When Clients Read/Write Ciphertexts", *The Computer Journal*, Vol. 62, no. 4, pp. 545-562, 2019.

[29] ITU-R Rec. 810: Conditional-Access Broadcasting Systems, (1992).

[30] S. W. Park, S. U. Shin. "An efficient encryption and key management scheme for layered access control of h.264/scalable video

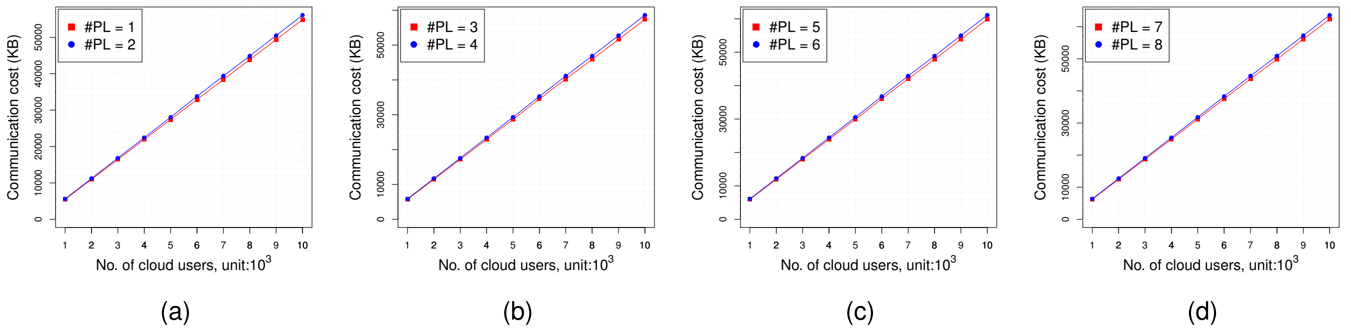


Fig. 10. Communication overheads of CEKMC and TGC with respect to the number of cloud users in consecutive 8 PLs. (a) $\#\text{PL}$ = 1 and 2, respectively. (b) $\#\text{PL}$ = 3 and 4, respectively. (c) $\#\text{PL}$ = 5 and 6, respectively. (d) $\#\text{PL}$ = 7 and 8, respectively.

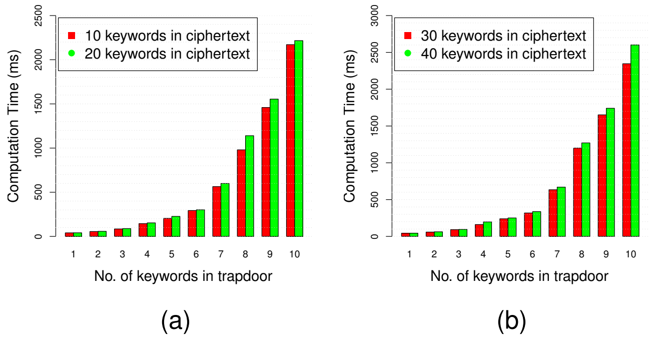


Fig. 11. Computational overheads of Check(*) with different number of keywords in the trapdoor $Td_{\mathbb{W}}$ and ciphertext $CT_{\mathbb{W}}$, where the number of keywords in $Td_{\mathbb{W}}$ varies from 1 to 10. (a) 10 and 20 keywords in $CT_{\mathbb{W}}$, respectively. (b) 30 and 40 keywords in $CT_{\mathbb{W}}$, respectively.

coding", *IEICE Transactions on Information and Systems*, 2009, 92-D(5), pp.851-858.

[31] M. Zhu, M. hang, X. Chen, D. Zhang, Z. Huang. "A Hierarchical Key Distribution Scheme for Conditional Access System in DTV Broadcasting", *Computational Intelligence and Security*, DBLP, 2007.

[32] C. Yang, J. Liu, Y. Zhang, J. Tian, L. Yang. "The Simplified and Secure Conditional Access for Interactive TV Service in Converged Network", *International Conference on Management and Service Science*, IEEE, 2009.

[33] M. Feng, B. Zhu. "When DRM Meets Restricted Multicast.: A Content Encryption Key Scheme for Multicast Encryption and DRM", *Consumer Communications and Networking Conference*, IEEE, 2007.

[34] Y. Rouselakis and B. Waters, "Practical constructions and new proof methods for large universe attribute-based encryption," in *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8*, 2013, ACM, 2013, pp. 463–474.

[35] K. Xiangying, C. Yanhui, Left Full Binary Hash Tree for Remote Attestation, 2017 IEEE 2nd International Conference on Signal and Image Processing.

[36] M. Bellare, D. Pointcheval and P. Rogaway, "Authenticated key exchange secure against dictionary attacks," in *Advances in Cryptology, EUROCRYPT 2000, LNCS 1807, Berlin, springer Verlag*, 2000, pp 139 -155.

[37] O. Goldreich, S. Goldwasser, S. Micali, "How to construct random functions" *J. of the ACM*, vol.33, no.4, pp. 792-807, 1986.

[38] Z. Lv, C. Hong, M. Zhang, and D. Feng, "Expressive and secure searchable encryption in the public key setting," in *Information Security - 17th International Conference, ISC 2014, Hong Kong, China, October 12-14, 2014. Proceedings, ser. Lecture Notes in Computer Science*, vol. 8783. Springer, 2014, pp. 364–376.

[39] J. Katz, A. Sahai, and B. Waters, "Predicate encryption supporting disjunctions, polynomial equations, and inner products," *J. Cryptology*, vol. 26, no. 2, pp. 191–224, 2013.

[40] M. Ameri, M. Delavar, J. Mohajeri, M. Salmasizadeh, "A Key-Policy Attribute-Based Temporary Keyword Search scheme for Secure Cloud Storage", *IEEE Transaction on Cloud Computing*, Vol. 8, no. 3, pp. 660-671, 2020.

[41] N. Sultan, N. Kaaniche, M. Laurent, F. Barbhuiya, "Authorized Keyword Search over Outsourced Encrypted Data in Cloud Environment", *IEEE Transaction on Cloud Computing*, DOI: 10.1109/TCC.2019.2931896, 2019.

[42] N. Sultan, M. Laurent, V. Varadharajan, "Securing Organization's Data: A Role-Based Authorized Keyword Search Scheme with Efficient Decryption", 2020, https://arxiv.org/abs/2004.10952.

[43] D. Wang, H. Cheng, P. Wang, X. Huang, G. Jian, "Zipf's Law in Passwords", *IEEE Transactions on Information Forensics and Security*, Vol. 12, no. 11, pp. 2776-2791, 2017.

Haiyan Wang obtained the Ph.D. degree in cryptography from the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China, in 2017. She is currently with the Department of New Networks, Peng Cheng Laboratory, Shenzhen, China. Her current research interests include cryptography and system theory. Dr. Wang is a recipient of the Director's Excellent Award of the Institute of Information Engineering, Chinese Academy of Sciences.

Penghui (Owen) Liu obtained his M.S. degree in the School of Computer and Communication, Lanzhou University of Technology in 2012, worked as a Security Technology Expert and Manager of DTV Chipset Security Technology Department at SMiT for about 10 years. Now he works in the Department of New Networks, Pengcheng laboratory, Shenzhen, China.

Xiaoxiong Zhong obtained his Ph.D. degree of Engineering in computer science and technology, Harbin Institute of Technology in 2015. Now he works in the Department of New Networks, Pengcheng laboratory, Shenzhen, China.

Weizhe Zhang obtained his Ph.D. degree of Engineering in computer science and technology, Harbin Institute of Technology(HIT) in 2006. Now he works as professor in HIT and the Department of New Networks.