# REKS: Role-Based Encrypted Keyword Search With Enhanced Access Control for Outsourced Cloud Data

Yinbin Miao, *Member, IEEE*, Feng Li, Xiaohua Jia, *Fellow, IEEE*, Huaxiong Wang, Ximeng Liu, *Member, IEEE*, Kim-Kwang Raymond Choo, *Senior Member, IEEE*, and Robert H. Deng, *Fellow, IEEE*

***Abstract*—Keyword-based search over encrypted data is an important technique to achieve both data confidentiality and utilization in cloud outsourcing services. While commonly used access control mechanisms, such as identity-based encryption and attribute-based encryption, do not generally scale well for hierarchical access permissions. To solve this problem, we propose a Role-based Encrypted Keyword Search (REKS) scheme by using the role-based access control and broadcast encryption. Specifically, REKS allows owners to deploy hierarchical access control by allowing users with parent roles to have access permissions from child roles. Using REKS, we further facilitate token generation preprocessing and efficient user management, thereby significantly reducing the users' final token generation and index update overheads, respectively. Formal security analysis proves that REKS is secure against chosen keyword and internal keyword guessing attacks, and findings from the empirical evaluations demonstrate that REKS is efficient and practical.**

***Index Terms*—Access control, attribute-based encryption, identity-based encryption, keyword-based search, role-based access control.**

Manuscript received 17 February 2021; revised 18 September 2023; accepted 11 October 2023. Date of publication 16 October 2023; date of current version 11 July 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 62072361, in part by the Key Research and Development Program of Shaanxi under Grant 2022GY-019, in part by the Shaanxi Fundamental Science Research Project for Mathematics and Physics under Grant 22JSY019, in part by the National Natural Science Foundation of China under Grant 62125205, in part by the Fundamental Research Funds for the Central Universities under Grant QTZX23091, in part by Fellowship pf China Postdoctoral Science Foundation under Grant 2022T150507, and in part by the Opening Project of Intelligent Policing Key Laboratory of Sichuan Province under Grant ZNJW2023KFMS002. The work of Kim-Kwang Raymond Choo was supported by Cloud Technology Endowed Professorship. *(Corresponding author: Yinbin Miao.)*

Yinbin Miao and Feng Li are with the School of Cyber Engineering, Xidian University, Xi'an 710071, China (e-mail: ybmiao@xidian.edu.cn; feng.li@stu.xidian.edu.cn).

Xiaohua Jia is with the Department of Computer Science, City University of Hong Kong, Hong Kong 999077, China (e-mail: csjia@cityu.edu.hk).

Huaxiong Wang is with the Division of Mathematical Sciences, School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore 639798 (e-mail: hxwang@ntu.edu.sg).

Ximeng Liu is with the Key Laboratory of Information Security of Network Systems, College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350108, China (e-mail: snbnix@gmail.com).

Kim-Kwang Raymond Choo is with the Department of Information Systems and Cyber Security, University of Texas at San Antonio, San Antonio TX 78249 USA (e-mail: raymond.choo@fulbrightmail.org).

Robert H. Deng is with the School of Information Systems, Singapore Management University, Singapore 178902 (e-mail: robertdeng@smu.edu.sg).

Digital Object Identifier 10.1109/TDSC.2023.3324640

## I. Introduction

CLOUD computing services (e.g., data storage, outsourcing, and processing) are now deeply entrenched in our society. For example, resource-constrained clients often outsource computationally incentive operations to cloud servers for reducing local computation and storage burdens. Because clients lose physical control on remotely stored data, the potential data security and privacy concerns impede the widespread adoption of cloud computing. The data encryption based on cryptographic primitives is generally used to guarantee data confidentiality, but makes the search over encrypted data difficult with traditional plaintext retrieval techniques. To guarantee both data confidentiality and data searchability, the searchable encryption (SE) [1] technique, which provides an elegant solution to search encrypted files by keywords, was developed.

Existing work of SE ignored the access control, and incurred unauthorized access permissions when deployed in public cloud systems [2], [3]. Thus, it is important to combine access control with SE. Existing access control approaches can be classified into two categories, namely authentication-based approach and encryption-based approach. The authentication-based approach, such as password authentication, is a server-dominated access control mechanism and assumes that cloud servers are trusted, but could lead to security and privacy leakages once the servers are compromised. The encryption-based approach prevents unauthorized accesses through commonly used cryptographic primitives such as identity-based encryption (IBE) [4] or ciphertext-policy attribute-based encryption (CP-ABE) [5]. Encryption-based approach offers a promising direction for combining SE with access control. However, there are three issues to apply encryption-based approach to SE.

Let us first consider a cloud-based enterprise, where there are multiple roles such as R:chairman, R:manager, and R:staff (see Fig. 1). The roles are arranged in a hierarchical structure, and each role can be associated with many users. For example, a staff shares encrypted files with the sales manager, financial manager and chairman. The traditional encryption-based access control solutions applied to SE still incur three issues: 1) The IBE-based keyword search solutions [6], [7] achieve access control by pre-fixing user identities, in which one user corresponds to one identity, which cannot achieve hierarchical access permissions mentioned in the above example; 2) The ABE-based Keyword Search (ABKS) solutions provides fine-grained access control

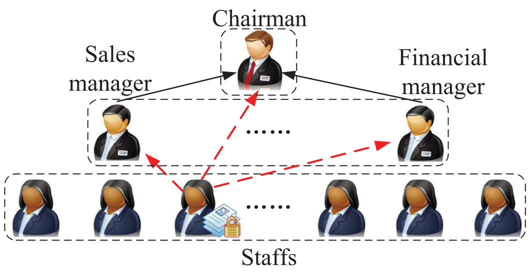


Fig. 1. Example of hierarchical structure.

compared with IBE-based keyword search schemes, but the index encryption and ciphertext search overheads in most of existing ABKS schemes [8], [9], [10], [11] linearly increase with the number of roles, and those of updated ciphertexts linearly increase with the number of users having the role, which incur high computation and storage burdens on both resource-constrained owners and users, respectively. Thus, ABE-based solution is difficult to deploy in practical applications due to high computation and storage costs; 3) The above two kinds of solutions have high index update overheads when updating identities/attributes [12] or access policies [13] in the dynamic setting.

From the above discussion, we can see that traditional broadcast encryption-based keyword search solutions [14], [15], [16] or ABKS [17] schemes can support the hierarchical role-based access control, but the costs associated with index constructions, ciphertext search or index updates are high. To address the challenging issues, we design a Role-based Encrypted Keyword Search with enhanced access control (REKS) scheme by combining role-based encryption (RBE) [18] and Identity-Based Broadcast Encryption (IBBE), which does not incur additional computational and storage overheads due to combination. Then, we extend it to support token generation preprocessing and efficient user management. The main contributions of our work are as follows:

- We propose a hierarchical role-based access control mechanism over encrypted data, in which the parent role has all the access permissions of child roles. We further provide efficient user management when the user set associated with the same role dynamically changes.
- We design an efficient index construction based on a specified role, in which the index construction costs are independent of the number of users and ancestor roles having the access permission of specified role. REKS only updates one index component when the user set updates.
- We extend REKS to support the token generation preprocessing, which shifts the computationally intensive operations to the preprocessed phase. Consequently, such a move significantly relieves users' token computations in the final phase.

The remainder of this paper is organized as follows. We first review the extant literature in Section II, before presenting the problem formulation (i.e., system model, problem definition, threat and security models) in Section III-B. We describe the construction of REKS in Section IV, and formally prove that REKS is secure against Chosen Keyword Attack (CKA) and Internal Keyword Guessing Attack (IKGA) in Section V. We evaluate the performance of the proposed REKS in Section VI. Finally, we conclude our work and discuss potential future research in Section VII.

## II. Related Literature

Since the proposal of public key encryption with keyword search [1] was proposed, a large number of extensions have been proposed, such as schemes seeking to improve search functionalities [19], [20], [21], [22], [23], efficiency [24], and security [9], [25]. To allow only authorized users having the right access permissions to encrypted data, either symmetric encryption-based [26] or asymmetric encryption-based access control mechanism [27], [28] can be employed. It is known that symmetric encryption-based access control approaches are vulnerable to key leakage threats; thus, we mainly focus on the asymmetric encryption-based access control mechanisms such as IBE-based keyword search and ABE-based keyword search in this paper.

*IBE-based keyword search:* In the IBE-based keyword search solution [7], the owner builds the indexes based on the public key of each user. When deployed in the multi-user setting, this mechanism dramatically expands the ciphertext storage space as it generates multiple ciphertexts for each keyword. The IBBE proposed by Cecile et al. [38] remedies the defect of IBE, and achieves the constant ciphertext size that is independent of the number of users. To allow the owner to flexibly grant access permission from one user to multiple users, Deng et al. formalized an IBE ciphertext transformation model [29] by combining two well-established IBE and IBBE solutions. While this scheme just provides flexible access control rather than keyword search. Thus, Jiang et al. integrated SE and IBBE to construct an efficient Identity-Based broadcast Encryption with Keyword Search scheme [15] in the static group, which resists IKGA. Zhang et al. designed the threshold broadcast encryption with keyword search scheme [30] for dynamic groups and flexible threshold values, but this scheme is vulnerable to CKA due to deterministic token generation algorithm. Mohamed et al. proposed the broadcast searchable keyword encryption scheme [31] which is secure against adaptive CKA.

*ABE-based keyword search:* In the dynamic group, the newly joined or revoked users could impact the flexibility and scalability of IBE-based keyword search schemes that often update indexes. ABE provides an expressive access control as it does not need to know the identities of users in advance. So far, many ABE-based keyword search solutions have been proposed for achieving both fine-grained access control and keyword search. For example, Miao et al. proposed a privacy-preserving ABKS scheme [32] in the shared multi-owner setting, but its encryption and decryption costs linearly increase with the number of attributes. Then, they reduced encryption cost by employing disjunctive normal form [39] rather than Boolean formula when specifying the access policy, and

TABLE I
COMPARATIVE SUMMARY BETWEEN OUR SCHEME AND PREVIOUS SCHEMES

| Schemes | [29] | [15] | [30] | [31] | [32] | [33] | [34] | [35] | [17] | [18] | [36] | [37] | REKS |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Access control | IBE | IBBE | IBBE | IBBE | ABE | ABE | ABE | IBBE | ABE | RBE | IBE | RBE | RBE |
| Hierarchical structure | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Search type | ✗ | Single | Single | Single | Single | Boolean | Single | ✗ | Conjunctive | ✗ | Single | Conjunctive | Single |
| Ciphertext size | Constant | Constant | Linear | Constant | Linear | Linear | Linear | Constant | Linear | Constant | Linear | Linear | Constant |
| User update | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✓ | ✗ | ✓ | ✓ |
| Security level | CPA | CKA/IKGA | — | CKA | KGA | CKA | CKA | CPA/CCA | CPA/CKA | CPA | CPA | CPA/CKA | CKA/IKGA |

— **Notes**. "CPA": Chosen Plaintext Attack; "CCA": Chosen Ciphertext Attack; "—": [30] is not secure against CKA as it claims.

achieved constant decryption cost through fast outsourced decryption. While above ABKS schemes just support conjunctive keyword search, which cannot satisfy more desired expressive search queries including Boolean keyword search [33] or comparable search [40]. To further prevent malicious cloud servers executing a fraction of search operations and returning incorrect search results, verifiable ABKS schemes have been put forward by using Merkle hash tree [34], Bloom filter [41] or digital signature [42].

When considering the hierarchical structure, the Hierarchical IBE/IBBE [35], [43] or Hierarchical ABE [44] can be used to support hierarchical access control, but the secret key of low level should be derived from that of high level and its size linearly increases with the depth of hierarchical tree. There are some state-of-art works focusing on achieving hierarchical access control and keyword search. For example, Miao et al. proposed ABKS scheme over hierarchical data [17] rather than hierarchical users. Sultan et al. constructed the role-based keyword search scheme [37], but needs to transform the indexes based on roles in the ciphertext search process. Li et al. proposed a hierarchical public key encryption with keyword search scheme [36] by utilizing a public key tree, but its index construction cost linearly grows with number of users. In addition, Liu et al. [45] adopted the structure of a public key tree to realize hierarchical access control, which allows users in different levels have different access permissions. Xiong et al. [46] proposed an authentication scheme with hierarchical access control based on self-certified public key cryptography and Chinese remainder theorem, which provides hierarchical access control, access without outside the scope of permission and efficient update user access privilege. Zhang et al. [47] proposed a new blockchain-based unified access control scheme in the form of ciphertext for the mobile cloud computing, which provides single registration, unified authentication, hierarchical access control, auditability and dynamic update. Riad et al. [48] proposed a new hierarchical access control scheme with dynamic revocation threshold vector for securing the cloud outsourced data, which can restrict the role of the central authority and revoke malicious users at different stages in the system, based on a dynamically changing revocation threshold vector. Compared with previous IBE-based or ABE-based keyword search schemes, REKS has following features in Table I.

## III. PROBLEM FORMULATION

In this section, we present the system model, problem definition, threat model and security model.

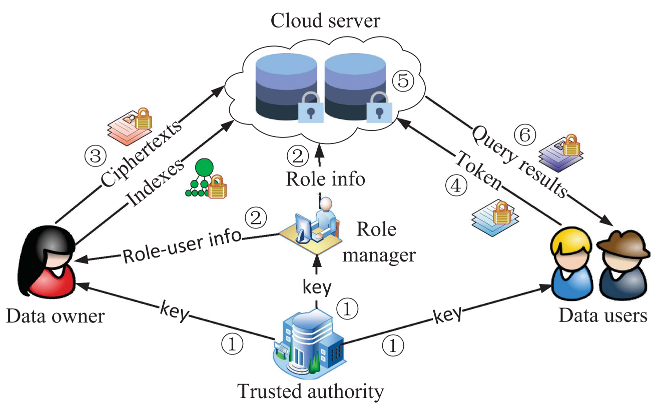


Fig. 2. System model of REKS.

### A. System Model

In this paper, we consider a cloud-based enterprise data sharing application, where the data users cannot share their secret keys with unauthorized ones. As depicted in Fig. 2, the system model of REKS consists of five entities, namely Trusted Authority (TA), Role Manager (RM), data owner, data users and Cloud Server (CS). The role of each entity is shown as follows:

- *Trusted authority:* TA is responsible for generating keys for owner, users and roles.
- *Role manager:* RM manages roles of users. RM sends the role information and role-user information to CS and owners, respectively.
- *Data owner:* The owner outsources encrypted files to CS. It builds search indexes based on the ancestor role set of specified role and uploads them to CS.
- *Data user:* An authorized user makes a search query by generating a token of the search keyword and submits the token to CS.
- *Cloud server:* CS stores ciphertexts and conducts search operations in respond to user's search queries, and returns the query results for search users.

In a cloud-based enterprise, TA distributes keys for owner, users and roles (Step ①) managed by RM. RM (e.g., enterprise administrator) manages users' access permissions and sends the role information, role-user information to CS and the owner respectively (Step ②). When the owner outsources its data to CS, it first encrypts files and builds corresponding indexes by using a specified role (Step ③) and secret key. The user generates the valid token based on the search keyword (Step ④) and secret

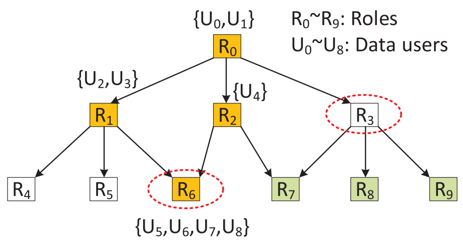


Fig. 3. Example of role-based access control.

key. CS matches the submitted token with indexes (Step ⑤) and returns relevant query results to the user (Step ⑥).

## B. Problem Definition

In this work, in order to divide the access permissions of user more precisely, we model the users' access permissions over encrypted files using the role-based access control. Each role corresponds to a certain access permission. Each user is associated with one or multiple roles. Roles are organized in a hierarchical structure, of which a parent role has all the access permissions of its child roles. Take Fig. 3 as an example, the parent role $\mathsf{R}_3$ has all access permissions of the child roles $\mathsf{R}_7, \mathsf{R}_8, \mathsf{R}_9$ (see the green nodes), and the ancestor role set having the access permission of role $\mathsf{R}_6$ is defined as $(\mathsf{R}_6, \mathsf{R}_2, \mathsf{R}_1, \mathsf{R}_0)$ (see the yellow nodes).

To facilitate keyword search over encrypted files in REKS, the owner needs to construct the searchable indexes by combining IBBE [38] with RBE [18]. Specifically, the owner first builds the inverted index $I_w$ for each keyword $w$ based on the ancestor role set of a specified role, then uploads encrypted indexes and file ciphertexts to CS. The user generates the token $T_{w'}$ based on its search keyword $w'$, role and identity, and then sends $T_{w'}$ to CS. If the token $T_{w'}$ matches with the index $I_w$, CS returns the query results to the user. In REKS, when the owner uses a role $\mathsf{R}_6$ to generate the index $I_w$, any user who has an ancestor role of $\mathsf{R}_6$ should be able to generate a valid $T_{w'}$ that can search $I_w$.

## C. Threat Model

In REKS, TA, RM, data owner and data users are fully trusted, note that there are no communications among data users, and they are trusted entities who do not sell or leak their secret keys and roles to unauthorized ones. For such reason, REKS is limited to specific application scenarios, where rules and regulations stipulate that leaking secret leaks will result in penalties. CS is honest-but-curious, which honestly conducts the search operations but is curious to infer some sensitive information about underlying keywords in accessible indexes and tokens. In addition to the keyword privacy leakage caused by the internal CS, the external attackers may try to distinguish indexes from those of their chosen keywords. Note that the external attackers only have access to indexes as tokens are generally transmitted via a secure channel. The potential keyword privacy threats incurred by the internal CS or external attackers are shown as follows:

- *Token linkability:* The internal CS can infer whether any two tokens are generated from the same keyword if the token generation algorithm is deterministic rather than non-deterministic.
- *Internal keyword guessing attack:* The internal CS can distinguish tokens from ciphertexts of its chosen keywords by using the public parameters and available ciphertexts.
- *Chosen keyword attack:* The internal CS or external attackers can distinguish indexes from ciphertexts of their chosen keywords with accessible information.

It is worth noting that the token linkability attack, internal keyword guessing attack and chosen keyword attack are all passive attacks, as the internal CS or external attackers only attempt to infer some sensitive information by observing accessible information without malicious actions (e.g., interfering with the search operations or modifying accessible information.)

## D. Security Model

In this work, we only consider the case the internal adversary is CS. To guarantee the keyword privacy, REKS should resist CKA and IKGA based on following two security games [15], [49], respectively.

*CKA game:* In this game, the adversary is allowed to make queries for some secret keys and token generations with some restrictions. Given a searchable index for the claimed user set having the role $\mathsf{R}$, this adversary cannot distinguish the index for keyword $w_0$ from that of keyword $w_1$, but the restriction is that the adversary cannot receive the corresponding token. The specific CKA game conducted between the adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ is shown as follows:

*Init:* The adversary $\mathcal{A}$ declares a challenging user set $UL_{\mathsf{R}}^* = \{ID_{\mathsf{U}_1}^*, \ldots, ID_{\mathsf{U}_n}^*\}$.

*Setup:* The challenger $\mathcal{C}$ first runs System Initialization and Key Generation for the owner, then sends the public parameters PP and the owner's public key $PK_{\mathsf{O}}$ to $\mathcal{A}$.

*Phase 1:* $\mathcal{A}$ conducts a polynomially bounded number of secret key queries and token generation queries as follows:

- *Secret key queries:* $\mathcal{A}$ issues a user's identity $ID_{\mathsf{U}_j} \notin UL_{\mathsf{R}}^*$ to $\mathcal{C}$. Then, $\mathcal{C}$ calls Key Generation to generate the user's secret key $SK_{\mathsf{U}_j}$ and sends it to $\mathcal{A}$.
- *Token generation queries:* $\mathcal{A}$ issues the user's identity $ID_{\mathsf{U}_j}$ and a queried keyword $w$ to $\mathcal{C}$. Then, $\mathcal{C}$ calls Token Generation to generate the token $T_w$ and sends it to $\mathcal{A}$.

*Challenge:* $\mathcal{A}$ submits two keywords $w_0, w_1$ with equal length and a challenging user set $UL_{\mathsf{R}}^*$ to $\mathcal{C}$, but it did not issue privates keys for the identity set $\{ID_{\mathsf{U}_1}^*, \ldots, ID_{\mathsf{U}_n}^*\}$ and token generation queries for keywords $w_0, w_1$ in *Phase 1*. Then, $\mathcal{C}$ chooses a random bit $b \in \{0, 1\}$ and calls Index Construction to generate the index $I_{w_b}$ for $\mathcal{A}$.

*Phase 2:* $\mathcal{A}$ continues to make the secret key or token generation query for $ID_{\mathsf{U}_j} \notin UL_{\mathsf{R}}^*$, or token generation query for $w \notin \{w_0, w_1\}$ on condition that $ID_{\mathsf{U}_j} \in UL_{\mathsf{R}}^*$. Then, $\mathcal{C}$ responds as *Phase 1*.

*Guess:* $\mathcal{A}$ outputs a guess bit $b' \in \{0,1\}$ and wins the above CKA game if $b' = b$.

*Definition 1:* We say that REKS is secure against CKA if for any polynomially bounded adversary $\mathcal{A}$, it has a negligible probability $\epsilon$ in breaking the above CKA game, namely $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{CKA}}(1^{\lambda}) = |\Pr[b' = b] - \frac{1}{2}| < \epsilon$.

*IKGA game:* In this game, the adversary $\mathcal{A}$ is permitted to make queries for secret keys and index constructions with some restrictions. Given the claimed identity $ID^{*}_{\mathsf{U}_j}$, $\mathcal{A}$ cannot distinguish the token for the keyword $w_0$ from that of keyword $w_1$, and it is not allowed to access the corresponding index. The specific IKGA game conducted between $\mathcal{A}$ and $\mathcal{C}$ is demonstrated as follows:

*Init:* $\mathcal{A}$ declares a challenging identity $ID^{*}_{\mathsf{U}_j}$ of user.

*Setup:* $\mathcal{C}$ calls System Initialization and Key Generation to generate public parameters PP and the owner's public key $PK_{\mathsf{O}}$ for $\mathcal{A}$.

*Phase 1:* $\mathcal{A}$ makes a polynomially bounded number of secret key queries and index construction queries as follows:

- *Secret key queries:* $\mathcal{A}$ issues a user's identity $ID_{\mathsf{U}_j} \neq ID^{*}_{\mathsf{U}_j}$ to $\mathcal{C}$. Then, $\mathcal{C}$ runs Key Generation to generate the secret key $SK_{\mathsf{U}_j}$ associated with the identity $ID_{\mathsf{U}_j}$ and sends it to $\mathcal{A}$.
- *Index generation queries:* $\mathcal{A}$ issues the identity set $\{ID_{\mathsf{U}_j}\}$ of users having the same role and the keyword $w$ to $\mathcal{C}$. Then, $\mathcal{C}$ responses it by calling Index Construction and sends the resulting index $I_w$ to $\mathcal{A}$.

*Challenge:* $\mathcal{A}$ submits two queried keywords $w_0, w_1$ with equal length and the user's identity $ID^{*}_{\mathsf{U}_j}$ to $\mathcal{C}$. While the restriction is that $\mathcal{A}$ did not make secret key query for $ID^{*}_{\mathsf{U}_j}$, or index construction queries for keywords $\{w_0, w_1\}$ based on the identity set $\{ID_{\mathsf{U}_j}\}$ in *Phase 1*. Then, $\mathcal{C}$ chooses a random bit $b \in \{0,1\}$ and calls Token Generation to generate the token $T_{w_b}$ for $\mathcal{A}$.

*Phase 2:* $\mathcal{A}$ continues to make the secret key query for $ID_{\mathsf{U}_j} \neq ID^{*}_{\mathsf{U}_j}$, index construction query for $\{ID_{\mathsf{U}_j}\}$ with $ID^{*}_{\mathsf{U}_j} \notin \{ID_{\mathsf{U}_j}\}$, or index construction query for the keyword $w \notin \{w_0, w_1\}$ and $\{ID_{\mathsf{U}_j}\}$ with $ID^{*}_{\mathsf{U}_j} \in \{ID_{\mathsf{U}_j}\}$.

*Guess:* $\mathcal{A}$ outputs a guess bit $b' \in \{0,1\}$ and wins the above IKGA game if $b' = b$.

*Definition 2:* We say that REKS is secure against IKGA if for any polynomially bounded adversary $\mathcal{A}$, it just has a negligible advantage $\epsilon$ in breaking the above IKGA game, namely $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{IKGA}}(1^{\lambda}) = |\Pr[b' = b] - \frac{1}{2}| < \epsilon$.

## IV. Proposed Role-Based Encrypted Keyword Search Scheme

We first show the technical overview of REKS before presenting its construction, then extend it to support token generation preprocessing and efficient user management.

### A. Technical Overview

In a role-based access control application shown in Fig. 4, each role is associated with many users. Aiming at achieving privacy-preserving keyword search over encrypted data, a naive solution is to build multiple indexes using IBE in the first phase, and then transform each index into another form generated by IBBE in the second phase, note that each role is analogous to an identity. Specifically, for the same keyword $w$, the owner first specifies a role $\mathsf{R}_1$ and generates IBE-based indexes $I_w^{\mathrm{IBE},1}, I_w^{\mathrm{IBE},2}, \ldots, I_w^{\mathrm{IBE},m}$ based on different ancestor roles $\mathsf{R}_1, \mathsf{R}_2, \ldots, \mathsf{R}_m$ respectively, then outputs the IBBE-based indexes $I_w^{\mathrm{IBBE},1}, I_w^{\mathrm{IBBE},2}, \ldots, I_w^{\mathrm{IBBE},m}$ using the delegated keys corresponding to the ancestor role set $PRL_{\mathsf{R}_1} = \{\mathsf{R}_1, \mathsf{R}_2, \ldots, \mathsf{R}_m\}$ [29]. The size of each ciphertext $I_w^{\mathrm{IBBE},i}$ does not increase with the number of users having the role $\mathsf{R}_i$, but this solution generates a re-encryption key for each role so that a proxy server can transform each ciphertext $I_w^{\mathrm{IBE},i}$ into another ciphertext $I_w^{\mathrm{IBBE},i}$, which brings high ciphertext transformation burdens.

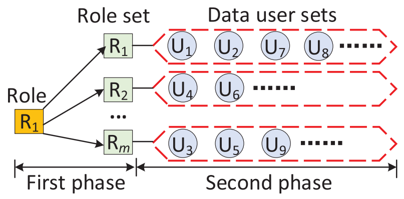


Fig. 4. Relationship between roles and users.

Another alternative way is to generate the same index $I_w^{\mathrm{ABE}}$ for the ancestor role set $PRL_{\mathsf{R}_1}$ using CP-ABE, note that the access policy P is determined based on $PRL_{\mathsf{R}_1}$ in the first phase and each role is analogous to an attribute. Then, $I_w^{\mathrm{ABE}}$ is re-encrypted to multiple indexes $\{I_w^{\mathrm{ABE},1}, I_w^{\mathrm{ABE},2}, \ldots, I_w^{\mathrm{ABE},m}\}$ with $m$ updated access policies respectively [50]. Each updated access policy $\mathsf{P}_i$ should be determined based on the corresponding user set. Not only the computation and storage costs of $I_w^{\mathrm{ABE}}$ linearly grow with the number of roles in $PRL_{\mathsf{R}_1}$ but also those of updated ciphertexts $\{I_w^{\mathrm{ABE},i}\}$ linearly increase with the number of users having the role $\mathsf{R}_i$. Thus, the above two naive solutions cannot be deployed in practical applications due to high computation and storage costs.

Our proposed scheme is as below: TA publishes a tuple $(A_{\mathsf{R}_1}, B_{\mathsf{R}_1})$ for the ancestor role set $PRL_{\mathsf{R}_1} = (\mathsf{R}_1, \mathsf{R}_2, \ldots, \mathsf{R}_m)$ by (1), where

$$A_{\mathsf{R}_1} = g^{\prod_{i=1}^{m}(\alpha + H_1(ID_{\mathsf{R}_i}))}, B_{\mathsf{R}_1} = A_{\mathsf{R}_1}^{\beta}, \tag{1}$$

$PRL_{\mathsf{R}_1}$ is the ancestor role set having the access permission of role $\mathsf{R}_1$, $g, h$ are two generators of group $\mathbb{G}$, $\alpha, \beta$ are two random elements and $H_1(\cdot)$ is a collision-proof hash function. Let $UL_{\mathsf{R}_1} = \{ID_{\mathsf{U}_1}, \ldots, ID_{\mathsf{U}_n}\}$ be a set of users having the role $\mathsf{R}_1$, RM generates the role-user information $N_{\mathsf{R}_1}$ for the owner by (2),

$$N_{\mathsf{R}_1} = h^{\alpha \prod_{j=1}^{n}(\alpha + H_1(ID_{\mathsf{U}_j}))}, \tag{2}$$

where $\mathsf{R}_1$ is specified by the owner in the process of index construction. The owner embeds $A_{\mathsf{R}_1}, B_{\mathsf{R}_1}, N_{\mathsf{R}_1}$ into the index $I_w$ for each keyword $w$, note that $I_w$ can be generated by using IBBE and SE. Thus, the index construction costs of our proposed scheme are independent of the number of roles in $PRL_{\mathsf{R}_1}$ or the number of users in $UL_{\mathsf{R}_1}$. RM also generates the role information $(UL_{\mathsf{R}_1}, M_{\mathsf{R}_1}, S_{\mathsf{R}_1})$ for CS, where $M_{\mathsf{R}_1}, S_{\mathsf{R}_1}$ are generated

TABLE II
NOTATION DEFINITIONS

| Notations | Definitions |
|---|---|
| $[x] = [1, 2, \cdots, x]$ | A set of integers |
| $PRL_{\mathsf{R}} = \{\mathsf{R}_1, \cdots, \mathsf{R}_m\}$ | Ancestor role set for role R |
| $\{\mathsf{U}_1, \cdots, \mathsf{U}_n\}$ | Data users having the same role |
| $\{ID_{\mathsf{R}_1}, \cdots, ID_{\mathsf{R}_m}\}$ | Identities for role set $PRL_{\mathsf{R}}$ |
| $UL_{\mathsf{R}} = \{ID_{\mathsf{U}_1}, \cdots, ID_{\mathsf{U}_n}\}$ | Identities of users having R |
| $\mathsf{F} = \{f\}, \mathsf{W} = \{w\}$ | File/keyword set |
| $SK_{\mathsf{R}}, SK_{\mathsf{U}}, SK_{\mathsf{O}}$ | Secret keys of role, user and owner |
| $\mathcal{I} = (I_1, I_2, I_3, I_4, I_5, I_6, \{I_w\})$ | Indexes for all keywords |
| $T_{w'} = (T_1, T_2, \{T_{3,j}, T_{4,j}\})$ | Token for the queried keyword $w'$ |
| $T_p = (T_p^1, T_p^2)$ | Preprocessed token |
| $T_f = (T_1, T_2, T_3, T_4)$ | Final token |

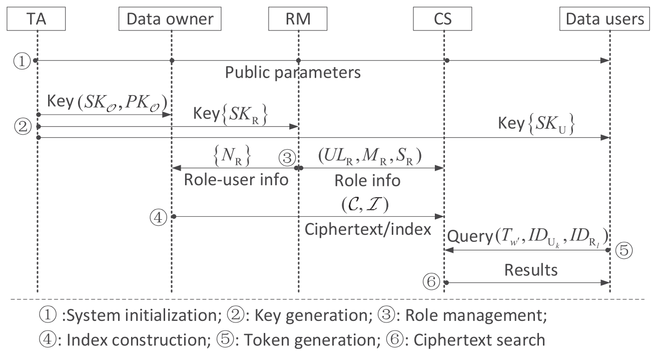

TA
Data owner
RM
CS
Data users
①
Public parameters
②
Key $(SK_{\mathcal{O}}, PK_{\mathcal{O}})$
Key$\{SK_{\mathrm{R}}\}$
Key$\{SK_{\mathrm{U}}\}$
③
$\{N_{\mathrm{R}}\}$
Role-user info
$(UL_{\mathrm{R}}, M_{\mathrm{R}}, S_{\mathrm{R}})$
Role info
④
$(\mathcal{C}, \mathcal{I})$
Ciphertext/index
⑤
Query$(T_{w'}, ID_{\mathrm{U}_k}, ID_{\mathrm{R}_l})$
⑥
Results
① :System initialization; ②: Key generation; ③: Role management;
④: Index construction; ⑤: Token generation; ⑥: Ciphertext search

Fig. 5. Algorithm flow of REKS.

based on the specified role $\mathsf{R}_1$. The role-user information and role information jointly guarantee that each user with a valid role $\mathsf{R}_i (1 \leq i \leq m)$ and identity $ID_{\mathsf{U}_j} (1 \leq j \leq n)$ can successfully make search queries, which satisfying $\mathsf{R}_i \in PRL_{\mathsf{R}_1}$ and $ID_{\mathsf{U}_j} \in UL_{\mathsf{R}_1}$.

## B. Construction of REKS

We use the following Bilinear pairings as the basis for constructing REKS. Let $\mathbb{G}, \mathbb{G}_T$ be two cyclic groups of order $p$ and $\mathbb{Z}_p$ be a field of order $p$. There exits an efficient computable bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ that satisfies $e(P^a, Q^b) = e(P, Q)^{ab}$ for all $P, Q \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p^*$. Before presenting the construction of REKS, we first define some notations in Table II.

The concrete construction of REKS consists of six algorithms, namely system initialization, key generation, role management, index construction, token generation and ciphertext search. We present REKS's algorithm flow in Fig. 5 and its construction as follows:

System Initialization $(1^\lambda)$: TA runs this algorithm to output global public parameters and master keys. The global public parameters are used in remaining algorithms and the master keys kept by TA are used to generate keys for the owner, roles and users. Given the security parameter $\lambda$, TA first generates the Bilinear pairing parameters $(\mathbb{G}, \mathbb{G}_T, e, p)$, then chooses two generators $g, h$ from the group $\mathbb{G}$. TA chooses two random elements $\alpha, \beta \in \mathbb{Z}_p^*$ and a hash function $H_1 : \{0, 1\}^* \rightarrow \mathbb{Z}_p$. Finally, TA generates the master keys MSK and public parameters PP as

$$\mathsf{MSK} = (g, \alpha, \beta),$$

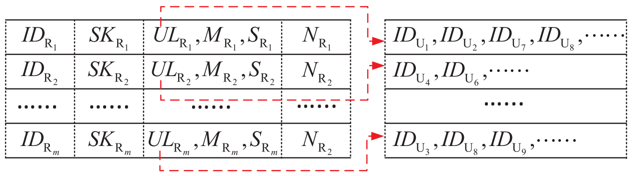

$ID_{\mathrm{R}_1}$
$SK_{\mathrm{R}_1}$
$UL_{\mathrm{R}_1}, M_{\mathrm{R}_1}, S_{\mathrm{R}_1}$
$N_{\mathrm{R}_1}$
$ID_{\mathrm{U}_1}, ID_{\mathrm{U}_2}, ID_{\mathrm{U}_7}, ID_{\mathrm{U}_8}, \cdots\cdots$
$ID_{\mathrm{R}_2}$
$SK_{\mathrm{R}_2}$
$UL_{\mathrm{R}_2}, M_{\mathrm{R}_2}, S_{\mathrm{R}_2}$
$N_{\mathrm{R}_2}$
$ID_{\mathrm{U}_4}, ID_{\mathrm{U}_6}, \cdots\cdots$
……
$ID_{\mathrm{R}_m}$
$SK_{\mathrm{R}_m}$
$UL_{\mathrm{R}_m}, M_{\mathrm{R}_m}, S_{\mathrm{R}_m}$
$N_{\mathrm{R}_2}$
$ID_{\mathrm{U}_3}, ID_{\mathrm{U}_8}, ID_{\mathrm{U}_9}, \cdots\cdots$

Fig. 6. The content stored by RM.

$$\mathsf{PP} = (H_1, v, v^*, g^\alpha, h^\beta, h, h^\alpha, \ldots, h^{\alpha^{q+1}}), \tag{3}$$

where $v = e(g, h)$, $v^* = e(h, h)$ and $q$ is the maximum number of users having the same role.

Key Generation $(\mathsf{MSK}, \mathsf{PP})$: With the public parameters PP and master keys MSK, TA generates keys for each role, user and owner as follows:

- For each role R with an identity $ID_{\mathsf{R}}$, TA generates its secret key $SK_{\mathsf{R}} = h^{1/(\alpha + H_1(ID_{\mathsf{R}}))}$ and sends $SK_{\mathsf{R}}$ to RM. Let the symbol $PRL_{\mathsf{R}}$ be the ancestor role set $\{\mathsf{R}_1, \ldots, \mathsf{R}_m\}$ having the access permission of role R, TA calls (1) to publish the tuple $(A_{\mathsf{R}} = g^{\prod_{i=1}^{m}(\alpha + H_1(ID_{\mathsf{R}_i}))}$, $B_{\mathsf{R}} = A_{\mathsf{R}}^\beta$ for R, where $ID_{\mathsf{R}_i}$ is the identity of role $\mathsf{R}_i \in PRL_{\mathsf{R}}$, $m$ is the size of $PRL_{\mathsf{R}}$. The tuple $(A_{\mathsf{R}}, B_{\mathsf{R}})$ is used to hide keywords in Index Construction.
- For each user U with an identity $ID_{\mathsf{U}}$, TA generates its secret key $SK_{\mathsf{U}} = g^{1/(\alpha + H_1(ID_{\mathsf{U}}))}$ so that it can generate valid tokens of queried keywords.
- For the owner O with an identity $\mathsf{ID_O}$, TA first chooses a random element $\gamma \in \mathbb{Z}_p^*$ and computes $g^{1/(\alpha+\gamma)}, g^{\alpha\gamma}, h^{\alpha\gamma}$, then outputs the secret key $SK_{\mathsf{O}}$ and public key $PK_{\mathsf{O}}$ by (4).

$$SK_{\mathsf{O}} = (\gamma, g^{1/(\alpha+\gamma)}), PK_{\mathsf{O}} = (g^{\alpha\gamma}, h^{\alpha\gamma}). \tag{4}$$

Role Management $(SK_{\mathsf{R}}, \mathsf{PP})$: In this algorithm, RM is responsible for managing roles, users and corresponding relationships between them when TA is offline, and generates the role-user information and role information. Fig. 6 gives the content stored by RM, note that RM just has access to the secret key of each role, which guarantees the role anonymity. Given the user set $UL_{\mathsf{R}} = \{ID_{\mathsf{U}_1}, \ldots, ID_{\mathsf{U}_n}\}$ having the role R, RM first chooses a random element $t_{\mathsf{R}} \in \mathbb{Z}_p^*$ and computes $M_{\mathsf{R}}, S_{\mathsf{R}}$ by (5), then sends the role information $(UL_{\mathsf{R}}, M_{\mathsf{R}}, S_{\mathsf{R}})$ to CS. RM calls (2) to compute the role-user information $N_{\mathsf{R}} = h^{\alpha \prod_{j=1}^{n}(\alpha + H_1(ID_{\mathsf{U}_j}))}$ and sends it to the owner, where $N_{\mathsf{R}}$ is derived from public parameters $h, h^\alpha, \ldots, h^{\alpha^{q+1}}$. The above two pieces of role information and role-user information can be used to the check the legitimacy of each user in Ciphertext Search.

$$M_{\mathsf{R}} = h^{-t_{\mathsf{R}}}, S_{\mathsf{R}} = SK_{\mathsf{R}} \cdot (h^\beta)^{t_{\mathsf{R}}}. \tag{5}$$

Index Construction $(\mathsf{PP}, \mathsf{F}, \mathsf{W}, SK_{\mathsf{O}}, PK_{\mathsf{O}}, A_{\mathsf{R}}, B_{\mathsf{R}}, N_{\mathsf{R}})$. The owner runs this algorithm to encrypt files and build indexes for facilitating privacy-preserving keyword search. Given the file set F and keyword set W, the owner first specifies a role R, then chooses two random elements $z_{\mathsf{R}}, s_{\mathsf{R}} \in \mathbb{Z}_p^*$ to generate the file ciphertexts and indexes as follows:

For each file $f \in \mathsf{F}$, the owner uses the traditional symmetric encryption algorithm $\mathsf{Enc}(\cdot)$ (e.g., AES) to encrypt it as $c_f = \mathsf{Enc}(f)$. For each keyword $w \in \mathsf{W}$, the owner uses $A_\mathsf{R}, B_\mathsf{R}, N_\mathsf{R}$ and its secret/public key pair $(SK_\mathsf{O}, PK_\mathsf{O})$ to compute $I_1, I_2, I_3, I_4, I_5, I_w, I_6$ by (6). In this way, $A_\mathsf{R}, B_\mathsf{R}, N_\mathsf{R}$ are embedded in each index, and the index size is independent of the number of roles in $PRL_\mathsf{R}$ and users in $UL_\mathsf{R}$.

$$I_1 = (g^\alpha)^{-z_\mathsf{R}}, I_2 = A_\mathsf{R}^{z_\mathsf{R}}, I_3 = B_\mathsf{R}^{z_\mathsf{R}}, I_4 = v^{z_\mathsf{R}};$$
$$I_5 = (g^{1/(\alpha+\gamma)})^{s_\mathsf{R}}, I_w = N_\mathsf{R}^{s_\mathsf{R}} N_\mathsf{R}^{H_1(w)\gamma}, I_6 = (g^\alpha)^{-s_\mathsf{R}}. \tag{6}$$

Finally, the owner sends the file ciphertexts $\mathcal{C}$ and encrypted indexes $\mathcal{I}$ to CS, where $\mathcal{C} = \{c_f\}$, $\mathcal{I} = (I_1, I_2, I_3, I_4, I_5, I_6, \{I_w\})$.

$\mathsf{Token\ Generation}$ $(SK_{\mathsf{U}_k}, \mathsf{PP}, PK_\mathsf{O}, w')$. If the identity $ID_{\mathsf{U}_k}$ of user with the role $\mathsf{R}_l$ belongs to the set $UL_{\mathsf{R}_l} = (ID_{\mathsf{U}_1}, \ldots, ID_{\mathsf{U}_n})$, the user is allowed to makes search queries, where $\mathsf{R}_l \in PRL_\mathsf{R} = (\mathsf{R}_1, \ldots, \mathsf{R}_m)$. Given the queried keyword $w' \in \mathsf{W}$, the user first chooses a random element $u \in \mathbb{Z}_p^*$ and uses its secret key $SK_{\mathsf{U}_k}$ to compute $T_1, T_2, T_{3,j}, T_{4,j}$ by (7), where $j \in [q]$. Then, the user sends the token $T_{w'} = (T_1, T_2, \{T_{3,j}, T_{4,j}\})$ and its identity and role information $(ID_{\mathsf{U}_k}, ID_{\mathsf{R}_l})$ to CS.

$$T_1 = (h^{\alpha^2+\alpha\gamma})^{-u}, T_2 = g^{u/(\alpha+H_1(ID_{\mathsf{U}_k}))};$$
$$T_{3,j} = (h^{\alpha^j})^u, T_{4,j} = e(g^{\alpha\gamma}, (h^{\alpha^j})^{H_1(w')})^u. \tag{7}$$

$\mathsf{Ciphertext\ Search}$ $(\mathsf{PP}, \mathcal{I}, ID_{\mathsf{U}_k}, ID_{\mathsf{R}_l}, T_{w'}, M_{\mathsf{R}_l}, S_{\mathsf{R}_l})$. CS runs this algorithm to find relevant results by matching the token and each index. When receiving the token submitted by the user that has the identity $ID_{\mathsf{U}_k}$ and the role $ID_{\mathsf{R}_l}$, CS first computes the intermediate variables $(Aux_1, Aux_2, Aux_3, Aux_4)$ by (8), where $Aux_3$ is computed based on public parameters $h^1, \ldots, h^{m-1}$.

$$Aux_1 = \prod_{i=1, i\neq l}^{m} H_1(ID_{\mathsf{R}_i}), Aux_2 = \prod_{j=1, j\neq k}^{n} H_1(ID_{\mathsf{U}_j});$$
$$Aux_3 = h^{\frac{1}{\alpha}(\prod_{i=1,i\neq l}^{m}(\alpha+H_1(ID_{\mathsf{R}_i}))-Aux_1)};$$
$$Aux_4 = e(M_\mathsf{R}, I_3). \tag{8}$$

Then, CS matches the user's token $T_{w'}$ with each index by (9), where $F_2 = \prod_{j=1,j\neq k}^{n}(\alpha + H_1(ID_{\mathsf{U}_j}))$, $F_1 = \frac{F_2 - Aux_2}{\alpha}$. $h^{u\alpha F_1}$ is computed by using a set of terms $\{T_{3,j}\}$, and $e(g^{\alpha\gamma}, h^{H_1(w')})^{uF_2}$ is calculated based on terms $\{T_{4,j}\}$. If (9) holds on condition that $w = w'$ and $ID_{\mathsf{U}_k} \in UL_{\mathsf{R}_l}$, CS returns relevant query results $\mathsf{Result}$ to the user.

$$e(I_5, T_1)^{Aux_2} \cdot e(I_w, T_2) \cdot e(I_6, h^{u\alpha F_1}) \cdot I_4 \overset{?}{=}$$
$$e(g^{\alpha\gamma}, h^{H_1(w')})^{uF_2} \cdot (e(Aux_3, I_1) \cdot e(S_\mathsf{R}, I_2) \cdot Aux_4)^{\frac{1}{Aux_1}}. \tag{9}$$

*Remark:* REKS not only allows the owner to build indexes based on a specified role, but also enables authorized users having any role in $PRL_\mathsf{R}$ to make valid search queries. In traditional SE solutions, the owner only uses public keys to build indexes. While in REKS, the owner uses its both public key and secret key to build indexes. In this way, REKS can resist IKGA as CS cannot generate valid indexes without the owner's secret key, thereby preventing CS from matching each token with forged indexes. In contrast to state-of-art works [37], [49], REKS achieves the hierarchical access control without re-encrypting or updating indexes based on roles, and incurs a less key management overhead. REKS has constant index construction and ciphertext search costs that do not increase with the number of roles in $PRL_\mathsf{R}$ or users in $UL_\mathsf{R}$.

However, there are two issues yet to be solved with REKS in various applications. First, the token generation process is affected by the maximum number $(q)$ of users having the same role, which has heavy computation burdens on resource-constrained users (e.g., mobile terminals, sensor nodes). Second, in the dynamic setting, the user set having the same role may frequently change, such as user enrolment and user revocation. If not handled properly, these two issues inevitably affect the scalability and practicability of REKS in practice. For these concerns, we extend REKS to support token generation preprocessing and efficient user management.

## C. Token Generation

Motivated by the offline/online encryption mechanism [51], we improve REKS to support token generation preprocessing. In the preprocessed phase, most of heavy operations are conducted without knowing the queried keyword in advance. When the queried keyword is available, the final phase only performs lightweight operations to complete the final token generation, which makes the computation cost of final token generation independent of the value of $q$. The token generation preprocessing mechanism is very useful for resource-limited mobile terminals or sensor nodes in practical applications.

In $\mathsf{Token\ Generation}$, as each user with an identity $ID_{\mathsf{U}_k}$ and role $\mathsf{R}_l$ does not know how many users having the role $\mathsf{R}_l$, it has to compute the maximum index components $\{T_{4,1}, \ldots, T_{4,q}\}$ for each queried keyword $w$. For achieving constant token computation and storage costs, we split $\mathsf{Token\ Generation}$ into two phases, namely preprocessed token generation and final token generation.

1) $\mathsf{Preprocessed\ Token}(ID_{\mathsf{U}_k}, ID_{\mathsf{R}_l}, \mathsf{PP}, PK_\mathsf{O})$. First, the user with an identity $ID_{\mathsf{U}_k}$ and role $\mathsf{R}_l$ interacts with RM by sending $h^{H_1(\mathsf{R}_l)}$, RM checks whether the equation $e(SK_{\mathsf{R}_l}, h^\alpha h^{H_1(\mathsf{R}_l)}) \overset{?}{=} v^*$. If it is true, RM sends $UL_{\mathsf{R}_l}$ to the user. Then, the user computes $T_p^1 = h^{\alpha F_1} = h^{F_2} h^{Aux_2}$, $T_p^2 = e(g^{\alpha\gamma}, h^{F_2})$, where $h^{\alpha F_1}, h^{F_2}$ is computed based on public parameters $h, h^\alpha, \ldots, h^{\alpha^n}$. The preprocessed token is defined as $T_p = (T_p^1, T_p^2)$.
2) $\mathsf{Final\ Token}(T_p, ID_{\mathsf{U}_k}, SK_\mathsf{U}, \mathsf{PP}, w')$: The user first chooses a random element $u \in \mathbb{Z}_p^*$ and computes $T_3 = (T_p^1)^u$, $T_4 = (T_p^2)^{H_1(w')u}$. Then, the user sends the final token $T_f = (T_1, T_2, T_3, T_4)$ to CS.

Upon receiving the final token $T_f$, CS checks whether $T_f$ matches with indexes $\mathcal{I}$ by (10).

$$e(I_5, T_1)^{Aux_2} \cdot e(I_w, T_2) \cdot e(I_6, T_3) \cdot I_4 \overset{?}{=} T_4 \cdot (e(Aux_3, I_1) \cdot e(S_{\mathsf{R}}, I_2) \cdot Aux_4)^{\frac{1}{Aux_1}}. \tag{10}$$

In the token generation preprocessing mechanism, each user interacts with RM to get the user list information $UL_{\mathsf{R}_l}$. If not, the user has to compute $q$ pairing operations $e(g^{\alpha\gamma}, h^{\alpha^j})$ in the preprocessed phase, which incurs high computation and storage costs. REKS significantly relieves the computation and storage costs of preprocessed token generation by sacrificing a small amount of communication overhead. In the preprocessed phase, the user does not leak the role identity to RM, which guarantees the role anonymity. The role anonymity is valuable and has wide applications. For example, in the healthcare, the patients' privacy and data protection are crucial. By anonymizing roles of patients, medical institutions can use anonymous data in data analysis and research to protect patients' personal identity information, while achieving the goals of medical research and data analysis. In the final phase, the user only computes four modular exponentiation operations. Thus, with the token generation preprocessing mechanism, REKS can be widely deployed in resource-limited applications. In addition, our scheme can be flexibly extended to support multi-owner scenario by generating multiple tokens via the idea of following remark. However, REKS still has some shortcomings, i.e., REKS only supports single keyword search, which has certain limitations for users. Our current solution focuses on reducing user computing costs, multi-keyword search will be solved as an important issue in future work.

*Remark:* In this paper, we consider the single owner multi-user setting, in which the single data owner shares his/her data among multiple data users. In addition, our scheme can be extended to support multi-owner scenario. The multi-owner scenario refers to the sharing of data from multiple owners $\{\mathsf{O}_1, \ldots, \mathsf{O}_m\}$ to multiple users $\{\mathsf{U}_1, \ldots, \mathsf{U}_n\}$. If a user $\mathsf{U}_{i|1\leq i\leq n}$ wants to query the data of $\{\mathsf{O}_1, \ldots, \mathsf{O}_m\}$, he first generates $m$ preprocessed tokens $\{T_{p,1}, \ldots, T_{p,m}\}$ using algorithm $\mathsf{Preprocessed\ Token}$, then uses $\{T_{p,1}, \ldots, T_{p,m}\}$ and $\mathsf{Final\ Token}$ to generate $m$ final tokens $\{T_{f,1}, \ldots, T_{f,m}\}$ sent to $CS$, which are shown by (11). It is worth noticing that the above mechanism is just a naive solution to support multi-owner multi-user setting. As part of our future work, we will aim to achieve hierarchical access control in multi-owner setting without incurring linear encryption costs.

$$\begin{cases} T_{p,1} = \mathsf{PreprocessedToken}(ID_{\mathsf{U}_i}, ID_{\mathsf{R}_l}, \mathsf{PP}, PK_{\mathsf{O}_1}); \\ \cdots \\ T_{p,m} = \mathsf{PreprocessedToken}(ID_{\mathsf{U}_i}, ID_{\mathsf{R}_l}, \mathsf{PP}, PK_{\mathsf{O}_m}). \\ T_{f,1} = \mathsf{FinalToken}(T_{p,1}, ID_{\mathsf{U}_i}, SK_{\mathsf{U}_i}, \mathsf{PP}, w'); \\ \cdots \\ T_{f,m} = \mathsf{FinalToken}(T_{p,m}, ID_{\mathsf{U}_i}, SK_{\mathsf{U}_i}, \mathsf{PP}, w'). \end{cases} \tag{11}$$

### D. Efficient User Management

In the dynamic setting, the access control policy regarding roles is relatively stable, but the user set having the same role may often be changed. Assume that a user with an identity $ID_{\mathsf{U}_k}$ and role $ID_{\mathsf{R}'_l}$ wants to register this system, it first sends the request $h^{H_1(\mathsf{R}'_l)}$ to RM. Then, RM checks its legitimacy by using the equation $e(SK_{\mathsf{R}_l}, h^{\alpha}h^{H_1(\mathsf{R}'_l)}) \overset{?}{=} v^*$. If this is true, RM adds the user's identity $ID_{\mathsf{U}_k}$ into the original user list $UL_{\mathsf{R}_l}$, and sends the updated user list $UL^*_{\mathsf{R}_l}$ to CS, where $n+1 \leq q$. RM re-computes $N^*_{\mathsf{R}} = h^{\alpha \prod_{j=1}^{n+1}(\alpha + H_1(ID_{\mathsf{U}_j}))}$ and sends it to the owner. It is worth noticing that the above equation just checks whether the role is legitimate. In fact, the user with certain role only has access permissions to files embedded with same role or child roles, and cannot access files embedded with high-level role or non-child roles. Thus, the user with certain role just has permitted rights rather than all non-permitted rights. In addition, each role has multiple user identities. In the search phase, our scheme still needs to check whether the identity $ID_{\mathsf{U}_k}$ belongs to the user set corresponding to role $ID_{\mathsf{R}'_l}$. There are two cases in the following processes:

- REKS does not support token generation preprocessing. The owner first updates the original index component $I_w$ as $I^*_w = (N^*_{\mathsf{R}})^{s_{\mathsf{R}}}(N^*_{\mathsf{R}})^{H_1(w)\gamma}$, then sends $I^*_w$ to CS. When conducting the ciphertext search, CS uses the updated information $Aux^*_2$, $e(I_6, h^{u\alpha F^*_1})$, $e(g^{\alpha\gamma}, h^{H_1(w')})^{uF^*_2}$ to verify (9). The terms $Aux^*_2, F^*_1, F^*_2$ are updated based on the auxiliary information $UL^*_{\mathsf{R}_l}$, but incur less computation overheads on CS. In this case, the users have to compute $q$ token components $\{T_{3,j}, T_{4,j}\}(j \in [q])$. Thus, the user enrolment does not affect each user's token generation process.
- REKS supports token generation preprocessing. In contrast to above case, each user generates the preprocessed token $T^*_p$ by using the updated information $Aux^*_2, F^*_1, F^*_2$. Note that the user enrolment incurs a less computation overhead in the preprocessed phase.

If a user with an identity $ID_{\mathsf{U}_k}$ and role $ID_{\mathsf{R}'_l}$ is revoked form the user list $UL_{\mathsf{R}'_l}$, REKS executes the same process as user enrolment. The user updates including user enrolment and user revocation just bring $m$ pairing operations and one modular exponentiation operation for RM, and $2\mathcal{W}$ modular exponentiation operations for the owner, but incur less computation burdens on CS and users. Thus, the user management mechanism provided in REKS is efficient and acceptable in the dynamic setting.

## V. Security Analysis

In this section, we formally prove that REKS guarantees the token unlinkability and is secure against CKA and IGKA under the security models defined in Section III-D. In $\mathsf{Token\ Generation}$, each user makes the keyword randomized by using a random element $u \in \mathcal{Z}_p$ when computing the token component $T_{4,j}$. Thus, REKS achieves the token unlinkability and makes it impossible for CS to guess whether two tokens are derived from the same keyword. According to the analysis presented in [52], the proofs of CKA security and IGKA security

ARE reduced to two Multi-Sequence of Exponents Decisional Diffie-Hellman (MSE-DDH) problems [15], [53], respectively.

$(n, \alpha, 1)$-MSE-DDH problem: Let $(\mathbb{G}, \mathbb{G}_T, e, p)$ be the bilinear parameters and $n$ be an integer. Given two random generators $g_0, h_0$ of the group $\mathbb{G}$, three co-prime polynomials $f_1, f_2, f_3$ in $\alpha \in \mathbb{Z}_p^*$ with respective orders $\mathtt{def}(f_1) = 1$, $\mathtt{def}(f_2) = \mathtt{def}(f_3) = n$, and a set $\Gamma$ of group elements:

$$\begin{cases} g_0^{f_1}, \ldots, g_0^{\alpha^{n-1} f_1}, g_0^{\alpha f_1 f_3}, g_0^{\gamma \alpha f_1 f_3}, g_0^{b f_1 f_3}, g_0^{ca f_3}, g_0^{c\alpha f_1 f_3}, \\ h_0^{\alpha}, \ldots, h_0^{\alpha^{n+1}}, h_0^{a\alpha}, \ldots, h_0^{a\alpha^{n-1}}, h_0^{\gamma\alpha}, h_0^{b\alpha f_1}, h_0^{b\alpha^2 f_1}, \\ h_0^{ba\alpha}, \ldots, h_0^{ba\alpha^n}, y, \ldots, y^{\alpha^{n+1}}, y^t, \ldots, y^{t\alpha^{n+1}}, y^b, \ldots, y^{b\alpha^n} \end{cases}$$

it is difficult to distinguish whether $Z$ is equal to $h_0^{ca\alpha f_2} y^{t\gamma\alpha f_2}$ or a random element in the group $\mathbb{G}$, where $a, b, c, t, \gamma \in \mathbb{Z}_p^*$ and $y \in \mathbb{G}$.

*Theorem 1.* REKS is secure against CKA on condition that $(n, \alpha, 1)$-MSE-DDH problem is intractable for any polynomial adversary.

*Proof.* Assume that there exists an adversary $\mathcal{A}$ that can break REKS with an advantage $\epsilon$, we build a simulator $\mathcal{S}$ that can simulate the challenger $\mathcal{C}$ and has an advantage $\epsilon / e q_T$ in breaking the $(n, \alpha, 1)$-MSE-DDH problem in the CKA game, where $e$ is the natural base and $q_T$ is the number of token queries $\mathcal{A}$ makes at most. Note that $\mathcal{S}$'s running time is approximately the same as that of $\mathcal{A}$.

*Init:* $\mathcal{A}$ declares the challenging user set $\{UL_{\mathsf{R}}^*\} = \{ID_{\mathsf{U}_j}^*\}$ and a user set $\{ID_{\mathsf{U}_j}\}$.

*Setup:* Given the bilinear parameters $(\mathbb{G}, \mathbb{G}_T, e, p)$ and the $(n, \alpha, 1)$-MSE-DDH instance $\Gamma$, the goal of $\mathcal{S}$ is to distinguish $h_0^{ca\alpha f_2} y^{t\gamma\alpha f_2}$ from a random element in the group $\mathbb{G}$ by using $\mathcal{A}$ as a subroutine. If $Z = h_0^{ca\alpha f_2} y^{t\gamma\alpha f_2}$, $\mathcal{S}$ outputs 1; if $Z$ is a random element in $\mathbb{G}$, $\mathcal{S}$ outputs 0. $\mathcal{S}$ selects random elements $\{a_j^*\}, \{a_j\} \in \mathbb{Z}_p^*$ and implicitly defines

$$f_1 = \alpha + \gamma, f_2 = \prod_{j=1}^{n} (\alpha + a_j^*), f_3 = \prod_{j=1}^{n} (\alpha + a_j),$$

where $f_1, f_2, f_3$ are unitary polynomials and are co-prime. We define $f_{3,j} = f_3 / (\alpha + a_j)$ for all $j \in [n]$. To generate the public parameters, $\mathcal{S}$ first sets $g = g_0^{f_1 f_3}$, $h = h_0$ and computes $g^{\alpha} = g_0^{\alpha f_1 f_3}$, $h^{\alpha^j} = h_0^{\alpha^j}$. Then, $\mathcal{S}$ specifies the keyword $w_b$ and sets $h^{H_1(w)}, h^{H_1(w)\alpha^j}$ by (12), (13), respectively.

$$h^{H_1(w)} = \begin{cases} y^t, & if\ w = w_b; \\ y, & if\ w \neq w_b; \end{cases} \tag{12}$$

$$h^{H_1(w)\alpha^j} = \begin{cases} y^{t\alpha^j}, & if\ w = w_b; \\ y^{\alpha^j}, & if\ w \neq w_b. \end{cases} \tag{13}$$

Finally, the simulator $\mathcal{S}$ sends the public parameters $\mathsf{PP} = (g^{\alpha}, \{h^{\alpha^j}, h^{H_1(w)\alpha^j}\})$ and the owner's public key $PK_{\mathsf{O}} = (g^{\alpha\gamma}, h^{\alpha\gamma})$ to $\mathcal{A}$, where $g^{\alpha\gamma} = g_0^{\alpha\gamma f_1 f_3}$, $h^{\alpha\gamma} = h_0^{\alpha\gamma}$.

$\mathcal{S}$ maintains a hash list $\mathcal{L}(ID_{\mathsf{U}_j}, x^j)$, which is initially empty. If $\mathcal{A}$ makes a hash query for the identity $ID_{\mathsf{U}_j}$, $\mathcal{S}$ first checks $\mathcal{L}$. If $ID_{\mathsf{U}_j} \in \mathcal{L}$, $\mathcal{S}$ picks the corresponding $x^j$ for $\mathcal{A}$; otherwise, $\mathcal{S}$ computes $x^j$ by (14), and adds $(ID_{\mathsf{U}_j}, x^j)$ to $\mathcal{L}$ before sending $x^j$ to $\mathcal{A}$.

$$x^j = \begin{cases} a_j^*, & if\ ID_{\mathsf{U}_j} \in \{ID_{\mathsf{U}_j}^*\}; \\ a_j, & if\ ID_{\mathsf{U}_j} \notin \{ID_{\mathsf{U}_j}^*\}. \end{cases} \tag{14}$$

*Phase 1:* $\mathcal{A}$ makes the secret key queries and token generation queries as follows:

- *Secret key queries:* $\mathcal{A}$ makes the secret key query for the user's identity $ID_{\mathsf{U}_j} \notin \{ID_{\mathsf{U}_j}^*\}$, and $\mathcal{S}$ calls Key Generation to generate $SK_{ID_{\mathsf{U}_j}} = g^{1/(\alpha + H_1(ID_{\mathsf{U}_j}))} = g_0^{f_{3,j} f_1}$, note that $g_0^{f_{3,j} f_1}$ can be derived from $g_0^{f_1}, \ldots, g_0^{\alpha^{n-1} f_1}$ in the above $(n, \alpha, 1)$-MSE-DDH instance.
- *Token generation query:* $\mathcal{A}$ makes the token generation query for identity $ID_{\mathsf{U}_j}$ and the keyword $w$:
  - If $ID_{\mathsf{U}_j} \notin \{ID_{\mathsf{U}_j}^*\}$, $\mathcal{S}$ gains the secret key by calling Key Generation and generates the token for $\mathcal{A}$ by running Token Generation.
  - If $ID_{\mathsf{U}_j} \in \{ID_{\mathsf{U}_j}^*\}$, $\mathcal{S}$ calls Token Generation in following two cases: 1) if $w = w^*$, $\mathcal{S}$ aborts this process; 2) if $w \neq w^*$, we have $x^j = H_1(ID_{\mathsf{U}_j}) = a_j^*$ and $h^{H_1(w)\alpha^j} = y^{\alpha^j}$. $\mathcal{S}$ chooses a random element $u^* \in \mathbb{Z}_p$ and computes the token $T_w$ by (15). Note that $T_1, T_2, T_{3,j}, T_{4,j}$ are derived from terms $g_0^{b f_1 f_3}$, $g_0^{\alpha\gamma f_1 f_3}$, $h_0^{b\alpha f_1}$, $h_0^{b\alpha^2 f_1}$, $h_0^{ba\alpha}, \ldots, h_0^{ba\alpha^n}$, $y^b, \ldots, y^{b\alpha^n}$ in the above $(n, \alpha, 1)$-MSE-DDH instance.

$$I_w = \begin{cases} T_1 = h_0^{-u^* b\alpha f_1 (\alpha + a_j^*)}; \\ T_2 = g_0^{u^* b f_1 f_3}; \\ T_{3,j} = h_0^{u^* ba(\alpha + a_j^*)\alpha^j}; \\ T_{4,j} = e(g_0^{\alpha\gamma f_1 f_3}, y^{u^* b\alpha^j (\alpha + a_j^*)}). \end{cases} \tag{15}$$

*Challenge:* $\mathcal{A}$ submits two challenging keywords $w_0, w_1$ and sends $(w_0, w_1, ID_{\mathsf{U}_j}^*)$ to $\mathcal{S}$. While the restriction is that $\mathcal{A}$ did not make secret key query for $ID_{\mathsf{U}_j}^*$ or token generation query for $ID_{\mathsf{U}_j}^*$ and $w \in \{w_0, w_1\}$. $\mathcal{S}$ responses to this query as follows:

- If $w_b \notin \{w_0, w_1\}$, $\mathcal{S}$ aborts this process.
- If $w_b \in \{w_0, w_1\}$, we have $(x^*)^j = H_1(ID_{\mathsf{U}_j}^*) = a_j^*$ and $h_0^{H_1(w_b)} = y^t$ based on the tuple $(ID_{\mathsf{U}_j}^*, (x^*)^j)$ in the hash list. Then, $\mathcal{S}$ selects a random bit $b \in \{0, 1\}$ and computes the challenging index $I_5 = g_0^{acf_3}, I_{w_b} = Z, I_6 = g_0^{-c\alpha f_1 f_3}$. Finally, $\mathcal{S}$ sends $(I_5, I_{w_b}, I_6)$ to $\mathcal{A}$.

If $Z = h_0^{ca\alpha f_2} y^{t\gamma\alpha f_2}$, one can compute above $(I_5, I_{w_b}, I_6)$ by implicitly setting $s_{\mathsf{R}} = ac$. If $Z$ is a random element in the group $\mathbb{G}$, the challenging index $I_{w_b}$ is random from the $\mathcal{A}$'s view and does not contain the term $w^*$.

*Phase 2:* $\mathcal{A}$ continues to make the secret key query for $ID_{\mathsf{U}_j} \notin \{ID_{\mathsf{U}_j}^*\}$, and the token generation query for $ID_{\mathsf{U}_j} \notin \{ID_{\mathsf{U}_j}^*\}$ or the token generation for $w \notin \{w_0, w_1\}$ on condition that $ID_{\mathsf{U}_j} \in \{ID_{\mathsf{U}_j}^*\}$. $\mathcal{S}$ responses to these queries as *Phase 1.*

*Guess:* $\mathcal{A}$ outputs a guess bit $b' \in \{0, 1\}$. $\mathcal{S}$ returns 1 if $b' = b$. Otherwise, $\mathcal{S}$ returns 0.

In the case that $\mathcal{S}$ does not abort in the CKA simulation: **1)** If $Z$ is the value in the above $(n, \alpha, 1)$-MSE-DDH instance, namely $Z = h_0^{ca\alpha f_2} y^{t\gamma\alpha f_2}$, $\mathcal{S}$ returns the correct form of challenging index. Thus, $\mathcal{S}$'s simulation is indistinguishable from the actual attack, and $\mathcal{A}$'s advantage in guessing the correct value on $b'$ is defined as $\Pr[b' = b | Z = h_0^{ca\alpha f_2} y^{t\gamma\alpha f_2}] = \epsilon$. **2)** If $Z$ is a random element in $\mathbb{G}$, it can be rewritten as $Z = h_0^{ca\alpha f_2} y^{t\gamma\alpha f_2} y^{z^*\gamma\alpha f_2}$, then we have $I_{w_b} = h^{s_{\mathsf{R}}\alpha \prod_{j=1}^{n}(\alpha + H_1(ID^*_{\mathsf{U}_j}))} (h^{H_1(w_b)} h^{H_1(w) z^*})^{\gamma\alpha \prod_{j=1}^{n}(\alpha + H_1(ID^*_{\mathsf{U}_j}))}$, where $z^* \in \mathbb{Z}_p$. Thus, $(I_5, I_{w_b}, I_6)$ is correct index form. As $z^*$ is a random element, $H_1(w_b) H_1(w)^{z^*}$ does not leak the information on $w_b$ from $\mathcal{A}$'s view. $\mathcal{A}$'s advantage in guessing the correct value $b'$ is defined as $\Pr[b' = b | Z \in \mathbb{G}] = \frac{1}{2}$.

According to the above analysis, we have $|\Pr[b' = b] - \frac{1}{2}| \geq \epsilon$ if $\mathcal{S}$ does not abort in above simulation, note that this probability is over random bits $b, b'$ guessed by $\mathcal{S}, \mathcal{A}$, respectively. Assume that $\mathcal{A}$ only makes the token generation query for each keyword one time, $\mathcal{S}$'s probability in aborting is $1/q_T$, but $\mathcal{S}$ does not abort when $\mathcal{A}$ makes the secret key queries. Thus, the probability that $\mathcal{S}$ does not abort during $\mathcal{A}$'s $q_T$ token generation queries is $(1 - \frac{1}{1+q_T})^{q_T} \geq \frac{1}{e}$ in *Phase 1* or *Phase 2*. In the *Challenge* phase, $\mathcal{S}$ aborts when $\mathcal{A}$ submits keywords $w_0, w_1$ satisfying $w_b \notin \{w_0, w_1\}$. As $\mathcal{A}$ did not make token generation queries for $w_0, w_1$ and $ID^*_{\mathsf{U}_j}$, the choices of keywords $w_0, w_1$ are independent of $\mathcal{A}$'s view, and we can have $\Pr[w_b = w_0 \; or \; w_1] = \frac{1}{q_T+1}$ and $\Pr[w_b \neq w_0, w_1] = (1 - \frac{1}{1+q_T})^2$. Thus, the probability that $\mathcal{S}$ does not abort in *Challenge* is $1/q_T$. As $\mathcal{A}$'s probability in breaking REKS is $\epsilon$, the probability that $\mathcal{S}$ does not abort in *Phase 1*, *Phase 2* and *Challenge* is $\epsilon \cdot \frac{1}{e} \cdot \frac{1}{q_T} = \frac{\epsilon}{eq_T}$, which conflicts with the assumption that the $(n, \alpha, 1)$-MSE-DDH problem is intractable. This completes the proof of Theorem 1. $\square$

$(n, \alpha, 2)$-**MSE-DDH problem:** Let $(\mathbb{G}, \mathbb{G}_T, e, p)$ be the bilinear parameters and $n$ be an integer. Given two random generators $g_0, h_0$ of the group $\mathbb{G}$, three co-prime polynomials $f_1, f_2, f_3$ in $\alpha \in \mathbb{Z}_p^*$ with respective orders $\mathtt{def}(f_1) = \mathtt{def}(f_2) = 1$, $\mathtt{def}(f_3) = n$, and a set $\Gamma$ of group elements:

$$\begin{cases} g_0^{f_1}, \ldots, g_0^{\alpha^{n-1} f_1}, g_0^{\alpha f_1 f_3}, g_0^{z\alpha f_1 f_3}, g_0^{\gamma\alpha f_1 f_3}, g_0^{ac}, g_0^{az f_3}, g_0^{c\alpha f_1}, \\ g_0^{b f_1 f_3}, h_0^{\alpha}, \ldots, h_0^{\alpha^{n+1}}, h_0^{a\alpha}, \ldots, h_0^{a\alpha^{n-1}}, h_0^{\gamma\alpha}, h_0^{b\alpha f_1 f_2}, \\ h_0^{ba\alpha f_2}, \ldots, h_0^{ba\alpha^{n-1} f_2}, h_0^{az\alpha f_2}, \ldots, h_0^{az\alpha^{n} f_2}, y, \ldots, y^{\alpha^{n+1}}, \\ y^t, \ldots, y^{t\alpha^{n+1}}, y^{\gamma\alpha f_2}, \ldots, y^{\gamma\alpha^n f_2}, h_0^{ac\alpha} y^{\gamma\alpha f_3}, h_0^{ac\alpha} y^{t\gamma\alpha f_3} \end{cases}$$

it is difficult to distinguish whether $Z$ is equal to $e(g_0, y^{tb\gamma\alpha f_1 f_2 f_3})$ or a random element in the group $\mathbb{G}_T$, where $a, b, c, z, t, \gamma \in \mathbb{Z}_p^*$ and $y \in \mathbb{G}$.

*Theorem 2.* REKS is secure against IGKA on condition that $(n, \alpha, 2)$-MSE-DDH problem is intractable for any polynomial adversary.

*Proof.* Assume that there exists an adversary $\mathcal{A}$ that can break REKS with an advantage $\epsilon$ in the IGKA game, we then build a simulator $\mathcal{S}$ that can simulate the challenger $\mathcal{C}$ and has an advantage $\epsilon/eq_T$ in solving the $(n, \alpha, 2)$-MSE-DDH problem. $\mathcal{S}$'s running time is approximately similar to that of $\mathcal{A}$.

*Init:* The adversary $\mathcal{A}$ declares a challenging identity of user $ID^*_{\mathsf{U}_j}$ and users's identity set $\{ID_{\mathsf{U}_j}\}$.

*Setup:* Given the public bilinear parameters $(\mathbb{G}, \mathbb{G}_T, e, p)$ and the $(n, \alpha, 2)$-MSE-DDH instance $\Gamma$, $\mathcal{S}$ utilizes $\mathcal{A}$ as a subroutine to decide whether $Z$ is equal to $e(g_0, y^{tb\gamma\alpha f_1 f_2 f_3})$ or a random element in $\mathbb{G}_T$. If $Z = e(g_0, y^{tb\gamma\alpha f_1 f_2 f_3})$, $\mathcal{S}$ outputs 1; otherwise, $\mathcal{S}$ outputs 0. First, $\mathcal{S}$ chooses a set of random elements $a^*, \{a_j\} \in \mathbb{Z}_p$ and sets

$$f_1 = \alpha + \gamma, f_2 = \alpha + a^*, f_3 = \prod_{j=1}^{n} (\alpha + a_j),$$

where the polynomials $f_2, f_3$ are co-prime. In addition, $\mathcal{S}$ defines $f_{3,j} = f_3/(\alpha + a_j)$ for all $j \in [n]$. Then, $\mathcal{S}$ sets $g = g_0^{f_1 f_3}$, $h = h_0$ and computes $g^{\alpha} = g_0^{\alpha f_1 f_3}$, $h^{\alpha^j} = h_0^{\alpha^j}$. $\mathcal{S}$ chooses the keyword $w_b$ and computes $h^{H_1(w)}$, $h^{H_1(w)^{\alpha^j}}$ by calling (12), (13), respectively. Finally, $\mathcal{S}$ responses to hash queries by maintaining a hash list $\mathcal{L}(ID_{\mathsf{U}_j}, x^j)$, which is similar to (14). The public parameters $\mathsf{PP} = (g^{\alpha}, \{h^{\alpha^j}, h^{H_1(w)^{\alpha^j}}\})$ and the owner's public key $PK_{\mathsf{O}} = (g^{\alpha\gamma}, h^{\alpha\gamma})$ are returned to $\mathcal{A}$.

*Phase 1:* $\mathcal{A}$ makes the secret key query and index construction query as follows:

- *Secret key query:* $\mathcal{A}$ makes the secret key query for the identity $ID_{\mathsf{U}_j} \neq ID^*_{\mathsf{U}_j}$. Then, $\mathcal{S}$ finds the term $H_1(ID_{\mathsf{U}_j}) = x^j = a_j$ from the hash list $\mathcal{L}(ID_{\mathsf{U}_j}, x^j)$. Finally, $\mathcal{S}$ calls Key Generation to generate the secret key $SK_{ID_{\mathsf{U}_j}} = g_0^{f_{3,j} f_1}$ for $\mathcal{A}$.
- *Index generation query:* $\mathcal{A}$ makes the index construction query for the users' identity set $\{ID_{\mathsf{U}_j}\}$ and the keyword $w$:
  - If $ID^*_{\mathsf{U}_j} \in \{ID_{\mathsf{U}_j}\}$, $\mathcal{S}$ calls Index Construction and responses in two cases: 1) If $w = w_b$, $\mathcal{S}$ aborts this process; 2) If $w \neq w_b$, $\mathcal{S}$ picks the element $x^j = H_1(ID_{\mathsf{U}_j})$ from $\mathcal{L}$, chooses a random element $s^*_{\mathsf{R}} \in \mathbb{Z}_p$ and generates the index for $\mathcal{A}$ by (16),
$$\begin{cases} I_5 = g_0^{s^*_{\mathsf{R}} az f_3}; \\ I_w = (h_0^{s^*_{\mathsf{R}} az\alpha f_2} y^{\gamma\alpha f_2})^{\prod_{a_j \neq a^*}(\alpha + a_j)}; \\ I_6 = g_0^{-s^*_{\mathsf{R}} z\alpha f_1 f_3}, \end{cases} \quad (16)$$
which can be derived from the elements $g_0^{az f_3}, g_0^{z\alpha f_1 f_3}$, $h_0^{az\alpha f_2}, \ldots, h_0^{az\alpha^n f_2}, y^{\gamma\alpha f_2}, \ldots, y^{\gamma\alpha^n f_2}$ in the $(n, \alpha, 2)$-MSE-DDH instance. One can verify the index form by implicitly setting $s^*_{\mathsf{R}} = s_{\mathsf{R}}/az$
  - If $ID^*_{\mathsf{U}_j} \notin \{ID_{\mathsf{U}_j}\}$, $\mathcal{S}$ calls Index Construction and responses in two cases: 1) If $w = w_b$, we pick the element $x^j = H_1(ID_{\mathsf{U}_j}) = a_j$ from $\mathcal{L}$ and have $H_1(w)^{\alpha^j} = y^{t\alpha^j}$. Then, $\mathcal{S}$ chooses a random element $s^*_{\mathsf{R}} \in \mathbb{Z}_p$ and sends the index $I_w = (I_5, I_w, I_6)$ to $\mathcal{A}$, where $I_5 = g_0^{s^*_{\mathsf{R}} ac}$, $I_w = h_0^{s^*_{\mathsf{R}} ac\alpha}$, $I_6 = g_0^{-s^*_{\mathsf{R}} c\alpha f_1}$. Note that $I_5, I_w, I_6$ can be derived from the elements $g_0^{ac}, g_0^{c\alpha f_1}, h_0^{ac\alpha} y^{t\gamma\alpha f_3}$ in the $(n, \alpha, 2)$-MSE-DDH instance, and its correctness can be verified by implicitly setting $s^*_{\mathsf{R}} = s_{\mathsf{R}} f_3/ac$. 2) If $w \neq w_b$, $\mathcal{S}$ outputs the correct index for $\mathcal{A}$, which is similar to above case.

*Challenge:* $\mathcal{A}$ submits two challenging keywords $w_0, w_1$ and the user's identity $ID^*_{\mathsf{U}_j}$. However, the restriction is that $\mathcal{A}$ did not make secret key query for the identity $ID^*_{\mathsf{U}_j}$, or the index construction query for $\{ID_{\mathsf{U}_j}\}$ and any keyword $w \in \{w_0, w_1\}$

TABLE III
THEORETICAL COMPUTATION AND STORAGE COSTS: A COMPARATIVE SUMMARY

| Algorithms | | KeyGen | Index Construction | Token Generation | Ciphertext Search |
|---|---|---|---|---|---|
| | | KeyGen | Ciphertext Generation | Content Request | Ciphertext Decryption |
| HPEKS [36] | Computation costs | $(2\mathcal{N}+1)\mathsf{E}+\mathcal{N}\mathsf{H}$ | $(\mathsf{E}+\mathsf{P})\mathcal{W}+5\mathsf{E}+2\mathsf{H}$ | $4\mathsf{E}+\mathsf{H}$ | $(\mathcal{N}+1)\mathsf{P}+\mathsf{E}$ |
| | Storage costs | $2\mathcal{N}\|\mathbb{G}\|+\|\mathbb{G}_T\|+\|\mathbb{Z}_p\|$ | $\mathcal{W}\|\mathbb{G}_T\|+4\|\mathbb{G}\|$ | $3\mathbb{G}\|$ | $(\mathcal{N}+1)\|\mathbb{G}_T\|$ |
| AACHC [54] | Computation costs | $\mathcal{N}(3\mathsf{E}+3\mathsf{E}_T+\mathsf{H}+\mathsf{P})$ | $3\mathsf{E}_T+m\mathsf{E}$ | $3\mathsf{P}+3\mathsf{E}+2\mathsf{E}_T+\mathsf{H}$ | $m(\mathsf{P}+\mathsf{E}+2\mathsf{E}_T)$ |
| | Storage costs | $3\mathcal{N}(\|\mathbb{G}_T\|+\|\mathbb{G}\|)$ | $3\|\mathbb{G}_T\|+m\|\mathbb{G}\|$ | $4\|\mathbb{G}_T\|+\|\mathbb{G}\|+\|\mathbb{Z}_p\|$ | $m(4\|\mathbb{G}_T\|+\|\mathbb{G}\|)$ |
| REKS | Computation costs | $3m\mathsf{E}+nm\mathsf{E}+3\mathsf{E}$ | $\mathcal{W}\mathsf{E}+7\mathsf{E}$ | $2\mathsf{E}+q(\mathsf{E}+\mathsf{P}+\mathsf{E}_T)$ | $7\mathsf{P}+3\mathsf{E}_T+\mathsf{E}$ |
| | Storage costs | $(3+n)m\|\mathbb{G}\|+3\|\mathbb{G}\|+\|\mathbb{Z}_p\|$ | $(\mathcal{W}+6)\|\mathbb{G}\|$ | $2\|\mathbb{G}\|+q(\|\mathbb{G}\|+\|\mathbb{G}_T\|)$ | $7\|\mathbb{G}_T\|+\|\mathbb{G}\|+2\|\mathbb{Z}_p\|$ |

**Notes**. $\mathcal{N}=mn$: the number of total users or attributes, $m$ is the number of roles, $n$ is the number of users having the same role, $\mathcal{W}$ is the number of keywords in the keyword set W.
CP-ABKS, HPEKS and REKS include KeyGen, Index Construction, Token Generation and Ciphertext Search; while AACHC includes KeyGen, Cipgertext Generation, Content Request and Ciphertext Decryption.

in the case $ID^*_{\mathsf{U}_j} \in \{ID_{\mathsf{U}_j}\}$. Then, $\mathcal{S}$ responses this query as follows:

- If $w_b \notin \{w_0, w_1\}$, $\mathcal{S}$ aborts this process.
- If $w_b \in \{w_0, w_1\}$, we pick the term $x^* = H_1(ID^*_{\mathsf{U}_j}) = a^*$ from the hash list $\mathcal{L}$ and have $H_1(w_b) = y^t$. Then, $\mathcal{S}$ chooses a random element $u^* \in \mathbb{Z}_p$ and calls Token Generation to generate the challenging token $T_{w_b} = (T_1, T_2, T_{3,j}, T_{4,j})$ for $\mathcal{A}$, where $T_1 = h_0^{-u^* b\alpha f_1 f_2}$, $T_2 = g_0^{u^* b f_1 f_3}$, $T_{3,j} = h_0^{u^* b a\alpha^j f_2}$, $T_{4,j} = Z^{u^* \alpha^j}$. Note that $T_1, T_2, T_{3,j}$ can be derived from the elements in the $(n, \alpha, 2)$-MSE-DDH instance.

If $Z = e(g_0, y)^{tb\gamma\alpha f_1 f_2 f_3}$, we can check the correctness of token when implicitly setting $u = u^* b f_2$. If $Z$ is a random element in $\mathbb{G}_T$, the challenging token $T_{w_b}$ is independent of $\mathcal{A}$'s view and contains no information about the keyword $w_b$.

*Phase 2:* $\mathcal{A}$ continues to make secret key query for the user's identity $ID_{\mathsf{U}_j} \neq ID^*_{\mathsf{U}_j}$, the index construction query for the identity set $\{ID_{\mathsf{U}_j}\}$ and $ID^*_{\mathsf{U}_j} \notin \{ID_{\mathsf{U}_j}\}$, or index construction query for $\{ID_{\mathsf{U}_j}\}$ and the keyword $w \notin \{w_0, w_1\}$ in the case $ID^*_{\mathsf{U}_j} \notin \{ID_{\mathsf{U}_j}\}$. The response of $\mathcal{S}$ is similar to that of *Phase 1*.

*Guess:* $\mathcal{A}$ outputs a guess bit $b' \in \{0, 1\}$. $\mathcal{S}$ outputs 1 when $b' = b$; otherwise, $\mathcal{S}$ outputs 0.

The analysis of $\mathcal{S}$'s probability in simulating the IGKA game is similar to that of Theorem 1. If $Z = e(g_0, y)^{tb\gamma\alpha f_1 f_2 f_3}$ is the element in the $(n, \alpha, 2)$-MSE-DDH instance, $\mathcal{S}$ returns the correct challenging token and its simulation in the above IGKA game is indistinguishable from the actual attack. In this case, $\mathcal{A}$'s probability in guessing the right bit on $b'$ is defined as $\Pr[b' = b | Z = e(g_0, y)^{tb\gamma\alpha f_1 f_2 f_3}] = \epsilon$. If $Z$ is a random element, the challenging token returned by $\mathcal{S}$ is random and independent of $\mathcal{A}$'s view, and $\mathcal{A}$'s probability in guessing the right bit on $b'$ is defined as $\Pr[b' = b | Z\, is\, random] = \frac{1}{2}$. According to the similar analysis in Theorem 1, the $\mathcal{S}$'s probability that it does not abort is at least $\epsilon / eq_T$, which conflicts with the $(n, \alpha, 2)$-MSE-DDH problem. This completes the proof of Theorem 2.

*Remark:* Since internal attackers are stronger than external attackers and our scheme is proved to resist internal keyword guessing attack, i.e., it can resist keyword guessing attack. In addition, keyword guessing attack can be divided into offline keyword guessing attack and online keyword guessing attack based on the attack method. Therefore, our scheme can resist offline keyword guessing attack and online keyword guessing attack. The difference between offline and online keyword guessing attack is the capability of attacker. In offline keyword guessing attack, the attacker can enumerate all possibilities, while in online keyword guessing attack, the attacker can just guess all possibilities. □

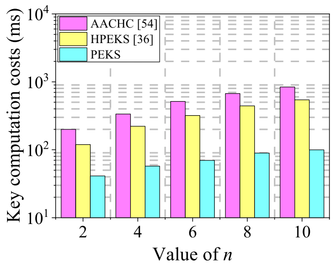
## VI. PERFORMANCE ANALYSIS

In this section, we evaluate the performance of REKS in terms of theoretical complexities and experimental tests using real-world dataset.

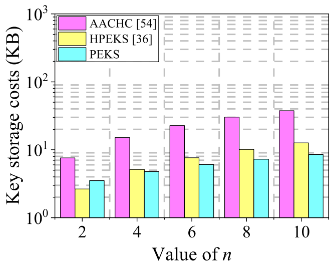
### A. Theoretical Complexities

To evaluate the feasibility and efficiency of REKS, we compare it with state-of-art schemes [36], [54]. The Hierarchical Public key Encryption with Keyword Search (HPEKS) scheme [36] achieved the hierarchical keyword search by using the public key tree, and the Accountable Access Control for Hierarchical Content (AACHC) scheme [54] achieved the hierarchical access control. For the theoretical complexities, we mainly focus on several time-consuming operations such as Pairing operation $\mathsf{P}$, hash operation $\mathsf{H}$ which maps the string of any length to the element of group $\mathbb{G}$ and modular exponentiation operation $\mathsf{E}$ (or $\mathsf{E}_T$) in group $\mathbb{G}$ or $\mathbb{G}_T$. Let $|\mathbb{G}_T|, |\mathbb{G}|, |\mathbb{Z}_p|$ be the length of elements in $\mathbb{G}_T, \mathbb{G}, \mathbb{Z}_p$ respectively, we show the theoretical complexities of REKS, HPEKS and AACHC in Table III.

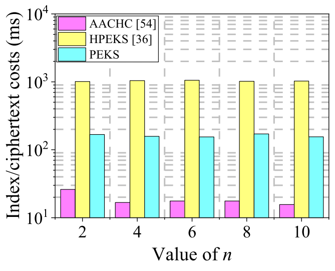
In Key Generation, REKS is efficient than other three schemes as it conducts about $\mathcal{N}\mathsf{E} + 3m\mathsf{E}$ for generating keys for roles and users, while HPEKS and AACHC conduct $2\mathcal{N}\mathsf{E} + \mathcal{N}\mathsf{H}$, $3\mathcal{N}(\mathsf{E} + \mathsf{E}_T) + \mathcal{N}(\mathsf{H} + \mathsf{P})$ for generating all users' keys respectively. With the same reason, the storage cost of key generation process of REKS is less than those of HPEKS and AACHC. In Index Construction or Ciphertext Generation, HPEKS and REKS supporting keyword search build indexes for $\mathcal{W}$ keywords by conducting about $\mathcal{W}(\mathsf{E} + \mathsf{P})$ and $\mathcal{W}\mathsf{E}$ operations. Since $\mathsf{P}$ is much efficient than $\mathsf{E}$, REKS has less computation cost than HPEKS in the index construction process, but these two schemes have approximately equal storage costs. While AACHC does not support keyword search, and it just generates each file ciphertext based on roles. Thus, AACHC is much more efficient than other schemes due to $m \ll \mathcal{W}$.



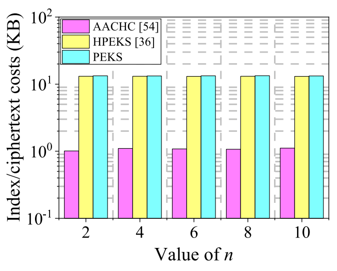
(a) $m = 5$



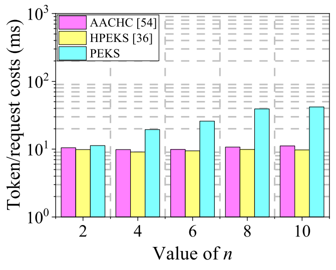
(b) $m = 5$



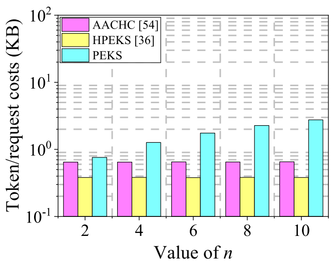
(c) $m = 5, \mathcal{W} = 100$



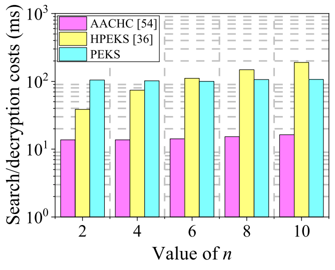
(d) $m = 5, \mathcal{W} = 100$



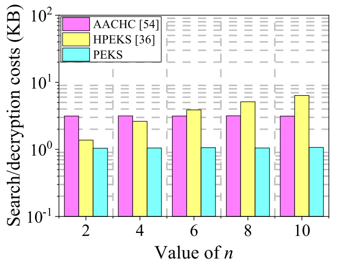
(e) $m = 5$



(f) $m = 5$



(g) $m = 5, \mathcal{W} = 100$



(h) $m = 5, \mathcal{W} = 100$

Fig. 7. Experimental tests of various schemes.

In Token Generation or Content Request, HPEKS and REKS conduct $4\mathsf{E} + \mathsf{H}$, $2\mathsf{E} + q(\mathsf{E} + \mathsf{P} + \mathsf{E}_T)$ operations to generate a token, respectively, and AACHC executes $\mathsf{P} + 3\mathsf{E} + 2\mathsf{E}_T + \mathsf{H}$ for generating a valid content request. Apparently, REKS has higher computation and storage costs due to $q\mathsf{E}$ extra operations. When being equipped with token generation preprocessing mechanism, REKS has constant computation and storage costs in the final phase, which is more efficient than HPEKS. In Ciphertext Search or Ciphertext Decryption, the computation and storage costs of HPEKS and AACHC are affected by the number of users and roles, respectively, while those of our proposed RBSKS are independent of above two variables $\mathcal{N}, m$. Thus, REKS outperforms HPEKS in the ciphertext search process. Since AACHC only decrypts one encrypted file, its computation costs in ciphertext decryption process are less than those of HPEKS and REKS in the ciphertext search process, but its storage cost in Ciphertext Decryption is more than that of REKS in Ciphertext Search.

## B. Experimental Tests

To evaluate the actual performance of REKS, AACHC and HPEKS schemes, we perform extensive experiments on macOS Catalina 10.15.4 with Intel Core i7 CPU 2.9 GHz by using Python3 language and Paring Based Cryptography (PBC) Library. The real-world dataset used in these empirical experiments comes from the public Enron Email Dataset,[1] and has a size of 422 MB emails distributed in 3500 folders. From Enron Email Dataset, we extract 100 high-frequency keywords using TF-IDF algorithm. Our paper focuses on the number $\mathcal{W}$ of keywords because our scheme is related to $\mathcal{W}$ and independent of the number of (document, keyword) pairs. Although the Enron Email dataset is large, we can control the number of keywords extracted. If we increase the number of extracted keywords, the number of (document, keyword) pairs will correspondingly increase. The common Type A curve $E(F_q) : y^2 = x^3 + x$ selected has 80-bit security level.[2] In practical, to ensure the security of the scheme, the higher security level, such as 128-bit, can be selected., and $\mathbb{G}, \mathbb{G}_T$ of order $p$ are treated as subgroups of $E(F_q)$, where the lengths of $p$ and $q$ are 160 bits and 512 bits, respectively. Thus, the size of each element in groups $\mathbb{G}_T, \mathbb{G}$ is 1024 bits, and that of element in the file $\mathbb{Z}_p$ is 160 bits. For simplicity, we set the number $m$ of roles as 5, the number $n$ of users as [10], and the number $\mathcal{W}$ of keywords as 100.

In Fig. 7(a), (b), we notice that the computation and storage costs of key generation in the above three schemes approximately increase with the variable $n$. When generating keys for $\mathcal{N} = mn$ users, AACHC conducts $\mathcal{N}(3\mathsf{E} + 3\mathsf{E}_T + \mathsf{H} + \mathsf{P})$ operations, while HPEKS and REKS just perform $\mathcal{N}(2\mathsf{E} + \mathsf{H})$, $\mathcal{N}\mathsf{E}$ operations, respectively. Apart from generating keys for users, REKS generates keys for the owner and $m = 5$ roles, but is more efficient than the other two schemes in the key generation process. For example, when the value of $n$ is set as 6, the computation and storage costs of AACHC, HPEKS and REKS are (514 ms, 22.70 KB), (320 ms, 7.65 KB), (70 ms, 6.07 KB) respectively.

In Fig. 7(c), (d), we demonstrate the ciphertext generation cost of AACHC and index construction costs of HPEKSA and REKS. The results show that ciphertext generation or index construction costs are independent of the variable $n$. In fact, the ciphertext generation costs (including computation and storage costs) of AACHC are affected by the number $m = 5$ of roles, and the index construction costs of HPEKS and REKS are related to

[1]http://www.cs.cmu.edu/~enron/

[2]For the convenience of testing, we select the elliptic curve with 80-bit security level

TABLE IV
PREPROCESSED/FINAL TOKEN GENERATION COSTS

| value of $n$ | 2 | 4 | 6 | 8 | 10 |
|---|---|---|---|---|---|
| Preprocessed token time (ms) | 39.29 | 39.28 | 39.60 | 43.95 | 39.48 |
| Final token time (ms) | 4.85 | 4.78 | 4.78 | 5.18 | 4.83 |
| Preprocessed token size (KB) | 1.91 | 1.92 | 1.89 | 1.88 | 1.91 |
| Final token size (KB) | 0.51 | 0.53 | 0.51 | 0.52 | 0.54 |

**Notes**: $m = 5$.

TABLE V
USER MANAGEMENT COSTS

| value of $n$ | 2 | 4 | 6 | 8 | 10 |
|---|---|---|---|---|---|
| RM's computation time (ms) | 66.76 | 66.48 | 67.31 | 73.75 | 68.86 |
| Owner's computation time (ms) | 152.49 | 145.81 | 148.19 | 159.04 | 13.27 |
| RM's storage size (KB) | 0.76 | 0.77 | 0.75 | 0.76 | 0.75 |
| Owner's storage size (KB) | 12.62 | 12.69 | 12.71 | 12.59 | 12.64 |

**Notes**: $m = 5, \mathcal{W} = 100$.

the number $\mathcal{W} = 100$ of keywords. Thus, the computation and storage costs of the above three schemes remain nearly constant when the variable $n$ varies. In addition, HPEKS and REKS have much more computation and storage costs than AACHC due to $\mathcal{W} \gg m$, and AACHC cannot facilitate keyword search. Compared with HPEKS that supports keyword search, REKS has less computation cost than HPEKS as it does not perform time-consuming operations such as hash operation $\mathsf{H}$ and pairing operation $\mathsf{P}$, but its storage cost is approximately equal to that of HPEKS. When setting $n = 10$, REKS's computation and storage costs are 155 ms and 13.28 KB respectively, which is acceptable in practical applications.

From Fig. 7(e), (f), we observe that the computation and storage costs of content request generation in AACHC and token generation in HPEKS keep almost unchanged when the variable $n$ varies, but those of REKS linearly grow with the increase of $n$. HPEKS is much more efficient than REKS. The main reason is that REKS conducts extra operations $q(\mathsf{E}_T + \mathsf{E} + \mathsf{P})$, and HPEKS just performs fixed $(4\mathsf{E} + \mathsf{H})$ operations. AACHC does not support keyword search, but it generates the authenticated content request with conducting $(3\mathsf{P} + 2\mathsf{E} + 2\mathsf{E}_T + \mathsf{H})$ operations. AACHC is efficient than REKS as its computation and storage costs are not affected by the value of $n$. For example, when the value of $n$ is set as 10, the computation and storage costs of token generation of REKS and HPEKS are (41.88 ms, 2.76 KB), (9.74 ms, 0.38 KB) respectively.

To overcome this defect, REKS is extended to support token generation preprocessing, and its preprocessed and final token generation costs are shown in Table IV. We fix the value of $m$ as 5 and vary the value of $n$. The preprocessed and final token generation time and size are not affected by the variable $n$, which remain constant. This is because the user in the preprocessed phase just conducts $3\mathsf{E} + 2\mathsf{P}$ operations based on the aggregation information $F_1, F_2$, and it only performs $4\mathsf{E}$ operations in the final phase. Thus, the final token generation process is more efficient than that of HPEKS. For instance, when setting $n = 10$, HPEKS takes 9.74 ms to generate the token, while REKS just takes 4.83 ms to generate the final token.

In Fig. 7(g), AACHS just decrypts each file ciphertext by using the secret key of a certain user, thus its decryption time is independent of variable $n$. When conducting search operation in Cipherext Search, REKS matches each index with the user's token, which makes its ciphertext time change linearly with the number $\mathcal{W}$ of keywords. The variables $m, n$ have almost no effect on REKS's ciphertext search time, but REKS has high ciphertext search time due to the effect of variable $\mathcal{W} = 100$. The ciphertext search time of HPEKS is affected by two variables $\mathcal{W}, \mathcal{N} = mn$. When the value of $n$ varies, HPEKS generates more indexes, which significantly increases the matching time between indexes and token. With the same reasons shown in Fig. 7(g), the storage cost of ciphertext search process in HPEKS linearly increases with the variable $n$, and is much higher than that of REKS. The storage cost of ciphertext decryption in AACHC is higher than that of REKS's search process.

When employed in the dynamic setting, the user management costs of REKS are shown in Table V. When a certain user is enrolled or revoked in the REKS system, RM first checks whether the user has a valid role by conducting $m$ pairing operations, then updates the user list $UL_{\mathsf{R}}$ and aggregation information $N_{\mathsf{R}}$ associated with the user's role $\mathsf{R}$. RM's computation and storage costs vary with the variable $m$, but are not affected by the variable $n$. Upon receiving the updated aggregation information $N^*_{\mathsf{R}}$, the owner only updates the index component $I_w$ whose computation and storage costs depend on the variable $\mathcal{W}$. The update costs of RM and the owner are small and are almost impervious to the variable $n$, thus the user management mechanism in REKS scales well in practice.

As a summary, the actual performance tests are consistent with the theoretical complexities. AACHC achieves the role-based access control, but cannot provide keyword search. Our proposed REKS is much more efficient in key generation, index construction, final token generation and ciphertext search when compared with HPEKS. Compared with ACCHC, REKS does not degrade the performance of key generation, final token generation and ciphertext search. REKS can be extended to provide lightweight token generation preprocessing and efficient user management. Thus, REKS is efficient in practice.

## VII. CONCLUSION

In this paper, we proposed an efficient role-based authorized keyword search scheme for managing hierarchical access permissions. Using this as a building block, we extended it to support token generation preprocessing and efficient user management. Finally, we formally proved our proposed approach resists both CKA and IGKA, and performed empirical experiments to demonstrate that its efficiency and feasibility in practical applications. Future extensions including exploring more efficient and expressive multi-keyword searches with hierarchical access control, and implementing a prototype of the proposed approach (or its future extension) in a real-world setting for evaluation.

## References

[1] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Proc. Int. Conf. Theory Appl. Cryptogr. Techn.*, 2004, pp. 506–522.

[2] K. Xue, W. Chen, W. Li, J. Hong, and P. Hong, "Combining data owner-side and cloud-side access control for encrypted cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 8, pp. 2062–2074, Aug. 2018.

[3] Y. Xue, K. Xue, N. Gai, J. Hong, D. S. Wei, and P. Hong, "An attribute-based controlled collaborative access control scheme for public cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 11, pp. 2927–2942, Nov. 2019.

[4] B. Waters, "Efficient identity-based encryption without random oracles," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, Springer, 2005, pp. 114–127.

[5] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symp. Secur. Privacy*, 2007, pp. 321–334.

[6] H. Li, Q. Huang, J. Shen, G. Yang, and W. Susilo, "Designated-server identity-based authenticated encryption with keyword search for encrypted emails," *Inf. Sci.*, vol. 481, pp. 330–343, 2019.

[7] L. Wu, Y. Zhang, K.-K. R. Choo, and D. He, "Efficient identity-based encryption scheme with equality test in smart city," *IEEE Trans. Sustain. Comput.*, vol. 3, no. 1, pp. 44–55, First Quarter 2017.

[8] Q. Zheng, S. Xu, and G. Ateniese, "Vabks: Verifiable attribute-based keyword search over outsourced encrypted data," in *Proc. IEEE Conf. Comput. Commun.*, 2014, pp. 522–530.

[9] W. Sun, S. Yu, W. Lou, Y. T. Hou, and H. Li, "Protecting your right: Attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud," in *Proc. IEEE Conf. Comput. Commun.*, 2014, pp. 226–234.

[10] J. Li, X. Lin, Y. Zhang, and J. Han, "KSF-OABE: Outsourced attribute-based encryption with keyword search function for cloud storage," *IEEE Trans. Services Comput.*, vol. 10, no. 5, pp. 715–725, Sep./Oct. 2016.

[11] Q. Xu, C. Tan, W. Zhu, Y. Xiao, Z. Fan, and F. Cheng, "Decentralized attribute-based conjunctive keyword search scheme with online/offline encryption and outsource decryption for cloud computing," *Future Gener. Comput. Syst.*, vol. 97, pp. 306–326, 2019.

[12] H. Deng, Z. Qin, Q. Wu, Z. Guan, and H. Yin, "Revocable attribute-based data storage in mobile clouds," *IEEE Trans. Services Comput.*, vol. 15, no. 2, pp. 1130–1142, Mar./Apr. 2022, doi: 10.1109/TSC.2020.2984757.

[13] J. Li et al., "An efficient attribute-based encryption scheme with policy update and file update in cloud computing," *IEEE Trans. Ind. Inform.*, vol. 15, no. 12, pp. 6500–6509, Dec. 2019.

[14] N. Attrapadung, J. Furukawa, and H. Imai, "Forward-secure and searchable broadcast encryption with short ciphertexts and private keys," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, Springer, 2006, pp. 161–177.

[15] P. Jiang, F. Guo, and Y. Mu, "Efficient identity-based broadcast encryption with keyword search against insider attacks for database systems," *Theor. Comput. Sci.*, vol. 767, pp. 51–72, 2019.

[16] Y. Yang, S.-L. Yang, F.-H. Wang, and J. Sun, "Post-quantum secure public key broadcast encryption with keyword search," *J. Inf. Sci. Eng.*, vol. 33, no. 2, pp. 485–497, 2017.

[17] Y. Miao, J. Ma, X. Liu, X. Li, Q. Jiang, and J. Zhang, "Attribute-based keyword search over hierarchical data in cloud computing," *IEEE Trans. Services Comput.*, vol. 13, no. 6, pp. 985–998, Nov./Dec. 2020.

[18] L. Zhou, V. Varadharajan, and M. Hitchens, "Achieving secure role-based access control on encrypted data in cloud storage," *IEEE Trans. Inf. Forensics Secur.*, vol. 8, no. 12, pp. 1947–1960, Dec. 2013.

[19] Y. Zhang and S. Lu, "Poster: Efficient method for disjunctive and conjunctive keyword search over encrypted data," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2014, pp. 1535–1537.

[20] Q. Liu, Y. Tian, J. Wu, T. Peng, and G. Wang, "Enabling verifiable and dynamic ranked search over outsourced data," *IEEE Trans. Services Comput.*, vol. 15, no. 1, pp. 69–82, Jan./Feb. 2022, doi: 10.1109/TSC.2019.2922177.

[21] H. T. Poon and A. Miri, "Fast phrase search for encrypted cloud storage," *IEEE Trans. Cloud Comput.*, vol. 7, no. 4, pp. 1002–1012, Fourth Quarter 2019, doi: 10.1109/TCC.2017.2709316.

[22] L. Xu, W. Li, F. Zhang, R. Cheng, and S. Tang, "Authorized keyword searches on public key encrypted data with time controlled keyword privacy," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 2096–2109, 2019.

[23] Y. Miao, R. Deng, K.-K. R. Choo, X. Liu, and H. Li, "Threshold multi-keyword search for cloud-based group data sharing," *IEEE Trans. Cloud Comput.*, vol. 10, no. 3, pp. 2146–2162, Third Quarter 2022, doi: 10.1109/TCC.2020.2999775.

[24] P. Xu, H. Jin, Q. Wu, and W. Wang, "Public-key encryption with fuzzy keyword search: A provably secure scheme under keyword guessing attack," *IEEE Trans. Comput.*, vol. 62, no. 11, pp. 2266–2277, Nov. 2012.

[25] H. Cui, Z. Wan, R. H. Deng, G. Wang, and Y. Li, "Efficient and expressive keyword search over encrypted data in cloud," *IEEE Trans. Dependable Secure Comput.*, vol. 15, no. 3, pp. 409–422, May/Jun. 2018.

[26] J. Alderman, K. M. Martin, and S. L. Renwick, "Multi-level access in searchable symmetric encryption," in *Proc. Int. Conf. Financial cryptogr. Data Secur.*, Springer, 2017, pp. 35–52.

[27] X. Zhang, Y. Tang, H. Wang, C. Xu, Y. Miao, and H. Cheng, "Lattice-based proxy-oriented identity-based encryption with keyword search for cloud storage," *Inf. Sci.*, vol. 494, pp. 193–207, 2019.

[28] M. H. Ameri, M. Delavar, J. Mohajeri, and M. Salmasizadeh, "A key-policy attribute-based temporary keyword search scheme for secure cloud storage," *IEEE Trans. Cloud Comput.*, vol. 8, no. 3, pp. 660–671, Third Quarter 2018, doi: 10.1109/TCC.2018.2825983.

[29] H. Deng et al., "Identity-based encryption transformation for flexible sharing of encrypted data in public cloud," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 3168–3180, 2020.

[30] S. Zhang, Y. Mu, and G. Yang, "Threshold broadcast encryption with keyword search," in *Proc. Int. Conf. Inf. Secur. Cryptol.*, Springer, 2015, pp. 322–337.

[31] M. Ali, H. Ali, T. Zhong, F. Li, Z. Qin, and A. A. AA, "Broadcast searchable keyword encryption," in *Proc. IEEE Int. Conf. Comput. Sci. Eng.*, 2014, pp. 1010–1016.

[32] Y. Miao et al., "Privacy-preserving attribute-based keyword search in shared multi-owner setting," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 3, pp. 1080–1094, May/Jun. 2021, doi: 10.1109/TDSC.2019.2897675.

[33] K. He, J. Guo, J. Weng, J. Weng, J. K. Liu, and X. Yi, "Attribute-based hybrid Boolean keyword search over outsourced encrypted data," *IEEE Trans. Dependable Secure Comput.*, vol. 17, no. 6, pp. 1207–1217, Nov./Dec. 2020, doi: 10.1109/TDSC.2018.2864186.

[34] Z. Chen et al., "Verifiable keyword search for secure Big Data-based mobile healthcare networks with fine-grained authorization control," *Future Gener. Comput. Syst.*, vol. 87, pp. 712–724, 2018.

[35] W. Liu, J. Liu, Q. Wu, and B. Qin, "Hierarchical identity-based broadcast encryption," in *Proc. Australas. Conf. Inf. Secur. Privacy*, Springer, 2014, pp. 242–257.

[36] H. Li, Q. Huang, and W. Susilo, "A secure cloud data sharing protocol for enterprise supporting hierarchical keyword search," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 3, pp. 1532–1543, May/Jun. 2022, doi: 10.1109/TDSC.2020.3027611.

[37] N. H. Sultan, M. Laurent, and V. Varadharajan, "Securing organization's data: A role-based authorized keyword search scheme with efficient decryption," 2020, *arXiv: 2004.10952*.

[38] C. Delerablée, "Identity-based broadcast encryption with constant size ciphertexts and private keys," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, Springer, 2007, pp. 200–215.

[39] Y. Miao, R. Deng, K.-K. R. Choo, X. Liu, J. Ning, and H. Li, "Optimized verifiable fine-grained keyword search in dynamic multi-owner settings," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 4, pp. 1804–1820, Jul./Aug. 2021, doi: 10.1109/TDSC.2019.2940573.

[40] Y. Miao, J. Ma, X. Liu, X. Li, Z. Liu, and H. Li, "Practical attribute-based multi-keyword search scheme in mobile crowdsourcing," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 3008–3018, Aug. 2018.

[41] W. Sun, S. Yu, W. Lou, Y. T. Hou, and H. Li, "Protecting your right: Verifiable attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 4, pp. 1187–1198, Apr. 2016.

[42] M. H. Ameri, M. R. Asaar, J. Mohajeri, and M. Salmasizadeh, "A generic construction for verifiable attribute-based keyword search schemes," *IACR Cryptol. ePrint Arch.*, vol. 2015, 2015, Art. no. 915.

[43] D. Boneh, X. Boyen, and E.-J. Goh, "Hierarchical identity based encryption with constant size ciphertext," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, Springer, 2005, pp. 440–456.

[44] G. Wang, Q. Liu, and J. Wu, "Hierarchical attribute-based encryption for fine-grained access control in cloud storage services," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2010, pp. 735–737.

[45] T. Liu et al., "Time-controlled hierarchical multikeyword search over encrypted data in cloud-assisted IoT," *IEEE Internet Things J.*, vol. 9, no. 13, pp. 11017–11029, Jul. 2022.

[46] L. Xiong, F. Li, M. He, Z. Liu, and T. Peng, "An efficient privacy-aware authentication scheme with hierarchical access control for mobile cloud computing services," *IEEE Trans. Cloud Comput.*, vol. 10, no. 4, pp. 2309–2323, 2022.

[47] Y. Zhang, L. Xiong, F. Li, X. Niu, and H. Wu, "A blockchain-based privacy-preserving auditable authentication scheme with hierarchical access control for mobile cloud computing," *J. Syst. Archit.*, vol. 142, 2023, Art. no. 102949.
[48] K. Riad, T. Huang, and L. Ke, "A dynamic and hierarchical access control for IoT in multi-authority cloud storage," *J. Netw. Comput. Appl.*, vol. 160, 2020, Art. no. 102633.
[49] H. S. Rhee, J. H. Park, W. Susilo, and D. H. Lee, "Improved searchable public key encryption with designated tester," in *Proc. Int. Symp. Inf., Comput., Commun. Secur.*, 2009, pp. 376–379.
[50] U. S. Varri, S. K. Pasupuleti, and K. Kadambari, "Key-escrow free attribute-based multi-keyword search with dynamic policy update in cloud computing," in *Proc. IEEE/ACM Int. Symp. Cluster, Cloud Internet Comput.*, 2020, pp. 450–458.
[51] Y. Miao, Q. Tong, K.-K. R. Choo, X. Liu, R. H. Deng, and H. Li, "Secure online/offline data sharing framework for cloud-assisted industrial Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8681–8691, Oct. 2019.
[52] C. Delerablée and D. Pointcheval, "Dynamic threshold public-key encryption," in *Proc. Annu. Int. Cryptol. Conf.*, Springer, 2008, pp. 317–334.
[53] R. Zhou, X. Zhang, X. Du, X. Wang, G. Yang, and M. Guizani, "File-centric multi-key aggregate keyword searchable encryption for industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 14, no. 8, pp. 3648–3658, Aug. 2018.
[54] N. H. Sultan, V. Varadharajan, S. Camtepe, and S. Nepal, "An accountable access control scheme for hierarchical content in named data networks with revocation," in *Proc. Eur. Symp. Res. Comput. Secur.*, Springer, 2020, pp. 569–590.

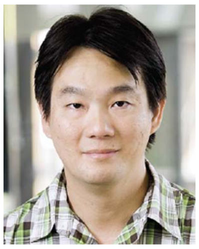
**Yinbin Miao** (Member, IEEE) received the BE degree from the Department of Telecommunication Engineering, Jilin University, Changchun, China, in 2011, and the PhD degree from the Department of Telecommunication Engineering, Xidian University, Xi'an, China, in 2016. He is also a postdoctor with Nanyang Technological University from 2018 to 2019, and a postdoctor with the City University of Hong Kong from 2019 to 2021. He is currently a professor with the Department of Cyber Engineering in Xidian University, Xi'an, China. His research interests include information security and applied cryptography.

**Feng Li** received the BE degree from the School of Cyberspace Security, Hangzhou Dianzi University, Hangzhou, China, in 2018. He is currently working toward the PhD degree with the School of Cyber Engineering, Xidian University. His current research interests include cloud computing security.

**Xiaohua Jia** (Fellow, IEEE) received the BSc and MEng degrees from the University of Science and Technology of China, in 1984 and 1987, respectively, and the DSc degree in information science from the University of Tokyo, in 1991. He is currently a chair professor with the Department of Computer Science, City University of Hong Kong. His research interests include cloud computing and distributed systems, computer networks and mobile computing. He is an editor of *IEEE Internet of Things*, *IEEE Transactions on Parallel and Distributed Systems* (2006-2009), *Wireless Networks*, *Journal of World Wide Web*, *Journal of Combinatorial Optimization*, etc. He is the general chair of ACM MobiHoc 2008, TPC Co chair of IEEE GlobeCom 2010 Ad Hoc and Sensor Networking Symposium, area-chair of IEEE INFOCOM 2010 and 2015.

**Huaxiong Wang** received the PhD degree in mathematics from the University of Haifa, Israel, in 1996, and the PhD degree in computer science from the University of Wollongong, Australia, in 2001. Since 2006, he has been an associate professor with the Division of Mathematical Sciences, School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore. His research interests include cryptography, information security, coding theory, and theoretical computer science.

**Ximeng Liu** (Member, IEEE) received the PhD degree from the Department of Telecommunication Engineering, Xidian University, Xi'an, China, in 2015. He is a professor with the Key Laboratory of Information Security of Network Systems, College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China. His research interests include applied cryptography and Big Data security.


**Kim-Kwang Raymond Choo** (Senior Member, IEEE) received the PhD degree in information security from the Queensland University of Technology, Australia, in 2006. He currently holds the Cloud Technology Endowed Professorship with the University of Texas at San Antonio. He is the founding co-editor-in-chief of ACM Distributed Ledger Technologies: Research & Practice, and the founding chair of IEEE Technology and Engineering Management Society Technical Committee (TC) on Blockchain and Distributed Ledger Technologies. He is the recipient of the 2022 IEEE Hyper-Intelligence TC Award for Excellence in Hyper-Intelligence Systems (Technical Achievement award), the 2022 IEEE TC on Homeland Security Research and Innovation Award, the 2022 IEEE TC on Secure and Dependable Measurement Mid-Career Award, and the 2019 IEEE TC on Scalable Computing Award for Excellence in Scalable Computing (Middle Career Researcher).

**Robert H. Deng** (Fellow, IEEE) is an AXA chair professor of cybersecurity and a professor of information systems with the School of Information Systems, Singapore Management University since 2004. His research interests include data security and privacy, multimedia security, network and system security. He served/is serving on the editorial boards of many international journals, including *IEEE Transactions on Information Forensics and Security*, *IEEE Transactions on Dependable and Secure Computing*. He has received the Distinguished Paper Award (NDSS 2012), Best Paper Award (CMS 2012), Best Journal Paper Award (IEEE Communications Society 2017).