

Practical Revocable Keyword Search Over Mobile Cloud-Assisted Internet of Things

Shuqin Liu, *Member, IEEE*, Yinbin Miao^{1b}, *Member, IEEE*, Feng Li^{1b}, Xinghua Li^{1b}, Kim-Kwang Raymond Choo^{1b}, *Senior Member, IEEE*, and Robert H. Deng^{1b}, *Fellow, IEEE*

Abstract—Searchable encryption (SE) can potentially be used to guarantee both data confidentiality and searchability over mobile cloud-assisted Internet of Things. However, existing SE solutions mainly focus on user revocation rather than keyword revocation. The keyword revocation may be required in certain situations. For example, patients do not allow their doctors to access records on some diseases such as syphilis. Hence, we propose a basic revocable keyword search (RKS) scheme over encrypted electronic medical records in the group setting, which supports keyword revocation (by using a revocation list) and authorized access permissions (via a group key exchange protocol). Then, we design an enhanced RKS (called RKS+) to significantly reduce the size of revoked keyword ciphertexts and the costs of token generation and ciphertext retrieval. Our schemes also support efficient user revocation by updating only one index component, and guarantee forward security. The formal security analysis proves that our schemes are secure against both chosen-keyword attacks and chosen-plaintext attacks, and findings from the empirical evaluations demonstrate that our schemes are efficient and practical.

Index Terms—Group key exchange protocol, keyword revocation, revocation list, searchable encryption (SE), user revocation.

I. INTRODUCTION

INTERNET of things can be found in a broad range of applications, including smart transportation and smart medical care. For example, medical record (EMR) system [1] enables

patients to share medical information with other stakeholders in healthcare settings and can potentially improve diagnostic accuracy and treatment efficiency. With the ever-increasing amount of EMRs being collected, shared, and stored, the potential of mobile cloud-based outsourcing services [2], [3], [4], [5] has been explored. To ensure security and privacy of data in such services, different cryptographic primitives, such as identity-based encryption (IBE) [6] and attribute-based encryption (ABE) [7], [8], [9], have been proposed. Searchable encryption (SE) [10], [11], another popular cryptographic primitive, can be leveraged to facilitate the searching of encrypted files based on specified keywords, without compromising either data confidentiality or searchability. In practice, users may be interested in documents that contain multiple keywords. SE based on conjunctive keyword search can return documents containing all query keywords, providing users with more practical search services.

While deleting the corresponding keyword index can achieve security once for all, it may not always be the preferred approach in real-world scenarios. In certain cases, data owners (DOs) may require the flexibility to temporarily revoke access to specific keywords for a certain period or for specific users, rather than permanently removing the keyword index. By maintaining the keyword ciphertexts and updating them as needed, our proposed revocable keyword search (RKS) and RKS+ schemes provide more versatile access control mechanisms that cater to various real-world requirements. A broad range of SE solutions can be deployed in the EMR system to facilitate coarse-grained or fine-grained keyword searches [12] by using IBE [13] or ABE [14], respectively. However, most of the existing SE schemes are designed to support user revocation rather than keyword revocation. Take Fig. 1 as an example, a particular patient does not authorize anyone, including the medical doctors, to access encrypted EMRs containing certain sensitive keywords (e.g., “Aids” and “Syphilis”). Medical data belongs to sensitive information, involving personal health status and privacy. Based on group setting, fine-grained control of medical data can be achieved, ensuring that only authorized users can access and operate specific data, thereby protecting data privacy and security. There are two solutions to achieve controllable keyword search, including indirect solution and direct solution. The indirect solution is to revoke the access rights of authorized doctors once and for all by revoking identities or attributes. However, doing so makes it impossible for them to search other unrevoked keywords. Such an approach prevents unauthorized

Manuscript received 6 March 2024; revised 17 April 2024; accepted 6 May 2024. Date of publication 9 May 2024; date of current version 8 August 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 62072361; 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 Opening Project of Intelligent Policing Key Laboratory of Sichuan Province under Grant ZNJW2023KFMS002; in part by the Open fund of Key Laboratory of Computing Power Network and Information Security under Grant 2023ZD020; and in part by the Natural Science Foundation of Shaanxi Province under Grant 2024JC-YBQN-0730. The work of Kim-Kwang Raymond Choo was supported only by the Cloud Technology Endowed Professorship. (Corresponding author: Yinbin Miao.)

Shuqin Liu is with the School of Computer Science & Technology, Xi'an University of Posts & Telecommunications, Xi'an 710061, China (e-mail: liushuqin@xupt.edu.cn).

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

Xinghua Li is with the State Key Laboratory of Integrated Services Networks and the School of Cyber Engineering, Xidian University, Xi'an 710071, Shaanxi, China (e-mail: xhli1@mail.xidian.edu.cn).

Kim-Kwang Raymond Choo is with the Department of Information Systems and Cyber Security, The 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/IIOT.2024.3398792

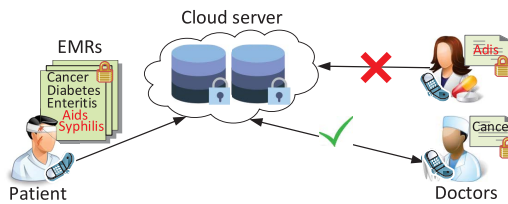


Fig. 1. Example for sharing EMRs.

access permissions by updating the secret keys [15] of unrevoked identities or attributes in the access policies, but incurs high-computation overheads when embedding users' identities or attributes into revoked keyword ciphertexts. Furthermore, such an approach does not invalidate the access permissions of previous secret keys. In other words, such an EMR system cannot efficiently prevent revoked doctors from making valid search queries over indexes by using old or outdated secret keys, thereby compromising the forward security [16].

The direct solution is to use time-specific encryption [17] or temporary access control [18], in which time intervals are encapsulated in the revoked keyword ciphertexts. When the specified time expires, the authorized doctors cannot make queries of these keywords. However, there are three challenges to be addressed in such an approach. First, this solution introduces a trusted time server to synchronize time on both owners and users, which incurs high-communication overheads. Besides, doing so also increases the potential for leakage of time information to semi-honest cloud servers (CSs). Second, all revoked keywords are encrypted with one or more time intervals, which incurs high-keyword encryption costs that linearly increase with the number of time intervals. Third, if the token is generated from the revoked keyword, this solution still requires CSs to match this particular token with all keyword ciphertexts, which results in significant matching complexity $\mathcal{O}(n)$, where n is the number of keywords in the dictionary.

One can observe from the above discussion that both approaches (i.e., indirect and direct solutions) cannot be deployed in the group setting to achieve efficient keyword revocation, and both approaches do not generally scale well in practical applications due to inflexible keyword revocations and high encryption and search overheads. To address these limitations, we first design a basic RKS over encrypted EMRs in a group setting, by combining a group key exchange protocol [19] and identity-based revocable key encapsulation mechanism [20]. In a typical EMR system, patients' medical records contain various types of information, such as personal details, medical history, diagnoses, treatments, and test results. These records are often associated with a wide range of keywords, including medical terms, symptoms, diseases, medications, and procedures. When sharing EMRs with healthcare providers or researchers, patients may want to selectively revoke access to certain sensitive keywords, such as those related to mental health, infectious diseases, or genetic disorders. The proposed RKS and RKS+ schemes enable patients to flexibly control access to their EMRs at the keyword level, ensuring that only authorized parties can search and retrieve the desired information. However, its public key

size and keyword revocation costs increase with the number of keywords in the dictionary. Then, we design an enhanced RKS (called RKS+) so that the public key size and key revocation costs are independent of the number of keywords. The contributions of our work are summarized as follows.

- 1) We propose a group sharing mechanism by using a secure and scalable group key exchange protocol. This mechanism allows group members to make search queries with the negotiated common key, and prevents revoked group members from accessing indexes with old or outdated tokens. The group member revocation only incurs minimal communication overheads and updates one index component.
- 2) We propose a RKS solution by specifying a keyword revocation list, which enables owners to flexibly revoke keywords. The size of revocable keyword ciphertexts in RKS or RKS+ is proportionate to the number of keywords in the dictionary or revoked list, respectively.
- 3) In RKS+, we achieve both short index size and token size, thereby incurring fewer storage costs on owners and users, respectively. We further achieve lightweight ciphertext search, whose computation and storage costs are independent of the number of keywords in the dictionary.
- 4) We conduct extensive experiments using a real-world data set to demonstrate the efficiency and feasibility of our schemes. Specifically, our enhanced scheme takes fewer computation and storage costs in terms of token generation and ciphertext search when compared with previous solutions, which provides users with good search experiences.

II. RELATED WORKS

On the basis of first public-key encryption with keyword search (PEKS) scheme [10], extensive SE solutions have been proposed, which focus on expressive search [21], user revocation [22] and securities [23]. For managing access permissions over users or keywords, asymmetric encryption-based access control mechanisms can be employed. Existing solutions for achieving authorized search include identity-based keyword search and attribute-based keyword search (ABKS) [24], [25], [26], [27]. *Identity-based keyword search.* When PEKS is deployed in the multiuser setting, its index size increases with the number of users. Thus, Bao et al. [28] presented a private query solution that allows each user to search all indexes submitted by multiple users using transformation keys. But the above scheme do not resist keyword guessing attacks as the keyword space is limited. To allowing users to freely join or leave, Shu et al. [29] put forward a proxy-free task matching solution that provides efficient user revocation without updating ciphertexts. The recent [30], [31] can be used to realize efficient user revocation, but do not support the keyword search. Besides, the revoked users may use old or outdated keys to make unauthorized queries and even lead to key exposure. Then, Zhang et al. [32] designed a forward secure PEKS based on lattice assumptions, which also resists the internal keyword guessing attack. Furthermore,

Miao et al. [33] used role-based access control to propose a role-based encrypted keyword search scheme, which can resist chosen keyword and internal keyword guessing attack. The identity-based keyword search avoids the certificate management burden, but only achieves coarse-grained access control and incurs high-ciphertext update overheads when revoking users.

ABKS: To support fine-grained access control, ABKS [14] was proposed by combining ABE with SE. To prevent malicious CSs from conducting a fraction of search operations and returning incorrect results, verifiable ABKS solutions have been constructed by using Bloom filter [34], bilinear-map accumulator [35] and digital signature [36]. When users are revoked from the ABKS system, Cui et al. [37] proposed the revocable ABKS solution that achieves the user revocation by only updating the secret data stored on the CS. Liu et al. [38] proposed a blockchain-based keyword search scheme in which the blockchain is used to update user revocation lists without updating ciphertexts and keys of unrevoked users. In practical applications, revoked users may reveal their secret keys to unauthorized users. To solve this key abuse issue, Yang et al. [39] proposed a traceable ABKS scheme that achieves outsourced decryption and efficient user revocation. However, the above ABKS scheme cannot guarantee the privacy of access policies that may contain sensitive information. Thus, the hidden policy ABKS [40] or dynamic policy update ABKS [41] have been proposed to solve the above issue.

The above identity-based or ABKS schemes provide the user revocation, but cannot achieve the controllable keyword revocation. A general way to restrict the searchability of some keywords is to build keyword indexes based on time intervals. For example, Fan and Huang [42] provided the effective revocation for search privileges by generating a time restrictive token and a predicate token. Miao et al. [43] designed a time-controllable keyword search scheme by using an attribute-based comparable access control. However, the above two schemes are only implemented in the symmetric encryption system. To overcome this defect, Yang et al. [44] proposed revocable public key encryption scheme with equality test, but this scheme requires both the owner and user to synchronize time. Other time-controlled keyword search schemes [45], [46] allow users to generate tokens based on their secret keys and time tokens distributed by the time server, but these schemes just achieve temporary access control rather than flexible access control specified by various owners. Aiming at achieving flexible and RKS in the multiuser setting, we propose practical RKS schemes in the group setting. Compared with previous schemes, our schemes have the following features shown in Table I.

III. PRELIMINARIES

A. Bilinear Pairing Map

For facilitating description, we first introduce the bilinear pairing [14]. Let $(\mathbb{G}, \mathbb{G}_T, e, p, g)$ be the bilinear pairing parameters, where $\mathbb{G}, \mathbb{G}_T$ are two cyclic groups of prime order p , e is the bilinear map $\mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ and g is the generator of $\mathbb{G}$. The bilinear pairing has the following three properties:

TABLE I
COMPARATIVE SUMMARY BETWEEN OUR SCHEMES AND PREVIOUS SCHEMES

Schemes	Search type	User revocation	Keyword revocation	Search complexity	Ciphertext size
[1]	Conjunctive	✗	✗	$\mathcal{O}(1)$	$\mathcal{O}(n)$
[18]	Single	✗	✗	$\mathcal{O}(\ln S)$	$\mathcal{O}(n Atts)$
[37]	Single	✓	✗	$\mathcal{O}(\ln)$	$\mathcal{O}(n Atts)$
[39]	Conjunctive	✓	✗	$\mathcal{O}(1)$	$\mathcal{O}(n + Atts)$
[47]	Single	✗	✓	$\mathcal{O}(l)$	$\mathcal{O}(n)$
[48]	Single	✗	✓	$\mathcal{O}(l)$	$\mathcal{O}(2n)$
[49]	Conjunctive	✗	✗	$\mathcal{O}(l)$	$\mathcal{O}(n)$
[50]	Single	✗	✓	$\mathcal{O}(\ln)$	$\mathcal{O}(n)$
RKS	Conjunctive	✓	✓	$\mathcal{O}(m)$	$\mathcal{O}(2n + q)$
RKS+	Conjunctive	✓	✓	$\mathcal{O}(m)$	$\mathcal{O}(n + 2m + q)$

— **Notes.** “ n ” is the number of keywords in the dictionary; “ $|Atts|$ ” is the number of attributes in the system or access policy; “ $|S|$ ” is the number of user’s attributes; “ l ” is the number of query keywords; “ q ” is the number of users in the group ($q \ll n$); “ m ” is the number of revoked keywords ($m \ll n$).

- 1) **Bilinearity:** $e(g^a, g^b) = e(g, g)^{ab}$ holds for all $a, b \in \mathbb{Z}_p^*$, where $\mathbb{Z}_p$ is a field of prime order p ;
- 2) **Nondegeneracy:** $e(g, g) \neq 1$;
- 3) **Computability:** There is an efficient algorithm to compute $e(g, g)$.

B. Group Exchange Protocol

Specifically, the group first negotiates a common key g^α , and then publishes the group public key $e(g, g)^\alpha$, where α is a random element in $\mathbb{Z}_p$. For facilitating the keyword search, the valid tokens should be generated by authorized data users (DUs) having g^α , and match with the indexes generated by a certain DO using $e(g, g)^\alpha$. The specific group exchange protocol [19] Γ is shown as follows.

- 1) **Step 1:** Let $\mathbf{U} = \{U_1, \dots, U_q\}$ be a group of DUs, each DU $U_j (j \in [1, q])$ chooses a random element $\alpha_j \in \mathbb{Z}_p^*$ and broadcasts g^{α_j} to other group members.
- 2) **Step 2:** Each DU computes $X_j = (g^{\alpha_{j+1}} / g^{\alpha_{j-1}})^{\alpha_j}$ and then broadcasts it to other group members.
- 3) **Step 3:** Each DU computes the common key $CK_U = K_j$ by (1), and publishes the group public key $PK_U = e(K_j, g) = e(g, g)^\alpha$

$$\begin{aligned} K_j &= (g^{\alpha_{j-1}})^{q\alpha_j} \cdot X_j^{q-1} \cdot X_{j+1}^{q-2} \cdot \dots \cdot X_{j-2} \\ &= g^{\alpha_1\alpha_2 + \dots + \alpha_q\alpha_1} = g^\alpha. \end{aligned} \quad (1)$$

C. Conjunctive Keyword Search Scheme

The public-key encryption with conjunctive keyword search scheme [51] achieves multikeyword search in the multiuser setting, and consists of four polynomial-time algorithms.

- 1) **KeyGen:** Given the security parameter λ , trusted authority (TA) generates the public/secret key pair (g^{α_j}, α_j) for each group member $U_j \in \mathbf{U}$.
- 2) **Enc:** The DO generates the searchable indexes $\mathcal{I}$ for the keyword dictionary $\mathbf{W}$ by using public keys $\{g^{\alpha_j}\}$, and uploads $\mathcal{I}$ to CS.
- 3) **Token:** The user U_j produces a token $T_{W'}$ for query keywords $W' \subset \mathbf{W}$, and submits $T_{W'}$ to CS.

- 4) **Search**: CS matches $T_{W'}$ with $\mathcal{I}$, and sends the query results to U_j .

IV. PROBLEM FORMULATION

In this section, the problem formulation, including the system model, threat model, and security model, is defined in the following.

A. System Model

In this article, we consider a cloud-based EMR sharing application. As depicted in Fig. 2, the system model of RKS or RKS+ is composed of four entities, namely, TA, DOs, DUs, and CS. TA will be offline if it has distributed keys for DUs and published global parameters, and DOs belong or do not belong to the group. The role of each entity is shown as follows.

- 1) **TA**: TA is responsible for generating keys for a group of authorized DUs and publishing global parameters.
- 2) **DUs**: A group of DUs first negotiate a common key and publish the group public key. DUs in this group can generate valid tokens for query keywords and send them to CS.
- 3) **DOs**: DOs encrypt EMRs and build the corresponding indexes based on revoked keyword lists and the group public key, and send EMR ciphertexts and indexes to CS.
- 4) **CS**: CS first checks the legitimacy of query keywords, and then matches submitted tokens with indexes before returning query results to DUs.

In the above application, TA (e.g., trusted third-party) distributes keys for all DUs (Step ①). A group of DUs (e.g., multiple doctors) negotiate a common key and send the group public key to DOs, without the involvement of a group manager (Step ②). A certain DO (e.g., patient) first specifies a set of revoked keywords that are not allowed to query, then encrypts EMRs and builds indexes before sending ciphertexts to CS (Step ③). An authorized DU makes a search query by generating a token based on query keywords, common key and secret key, and then submits the token to CS (Step ④). Upon receiving the token, CS first checks whether these query keywords are revoked by DO. If these query keywords are allowed to query, CS matches the token with the entire indexes (Step ⑤). Finally, CS sends the relevant query results to DU (Step ⑥).

B. Threat Model

In our schemes, TA, DOs and nonrevoked DUs are fully trusted, but the revoked DUs may try to make queries based on their previous tokens. Consistent with previous works [52], [53], [54], CS is *honest-but-curious*, which follows the search protocols honestly but is very curious to infer some sensitive information about keywords in accessible indexes and tokens. CS with malicious behaviors is not within the scope of consideration in this article. If there are malicious behaviors on CS, we can solve it by adding query result verification. In addition, it requires that revoked DUs cannot collude with CS. Otherwise, the plaintext information will be

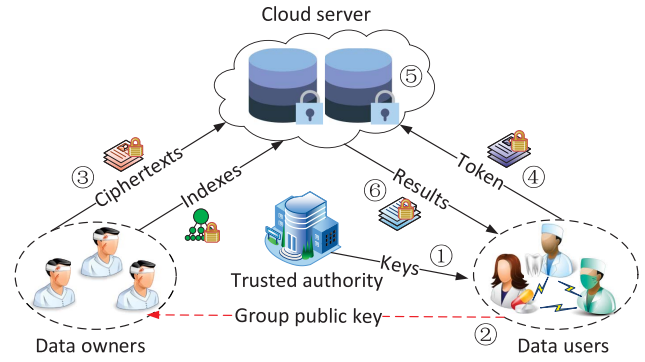


Fig. 2. System model of our schemes.

known to CS. The potential threats caused by revoked DUs or CS are shown as follows.

- 1) **Forward Security Threat**: The revoked DUs can successfully search previous indexes by using old or outdated tokens if the indexes are not updated.
- 2) **Token Linkability**: CS can infer whether two submitted tokens are generated from the same query keyword set if the token generation algorithm is deterministic.
- 3) **Chosen-Keyword Attack (CKA)**: CS or external attackers can distinguish indexes from ciphertexts of their chosen keywords by using public information.

C. Security Model

In the security analysis, we first consider a security game against CKA caused by CS or external attackers. In this game, the adversary $\mathcal{A}$ has access to DUs' public keys and is allowed to make token generation queries on behalf of authorized U_j . The security of our schemes is defined by the indistinguishability against CKA (IND-CKA), and the above security game conducted between $\mathcal{A}$ and challenger $\mathcal{C}$ is shown as follows:

Setup: Given the security parameter λ , $\mathcal{C}$ first runs **Setup**, **KeyGen**, **KeyExch** algorithms to output the global public parameters PP , public/secret key pairs $\{PK_j/SK_j\}$, group common/public key pair (CK_U, PK_U) , then sends $(PP, PK_U, \{PK_j\})$ to $\mathcal{A}$.

Phase 1: $\mathcal{A}$ adaptively makes a set of queries for various keyword sets $W_1, \dots, W_d$ to the token generation oracle.

- 1) $\mathcal{O}_{\text{Token}}(j, W_i)$: $\mathcal{C}$ first runs **Token** to generate the token T_{W_i} by using the secret key SK_j of U_j , then sends T_{W_i} to $\mathcal{A}$.

Challenge: $\mathcal{A}$ chooses a challenging keyword set W^* and sends it to $\mathcal{C}$. $\mathcal{C}$ first chooses a random bit $b \in \{0, 1\}$ and a random keyword set W . But the requirement is that W^* should not be distinguished from W for tokens queried in **Phase 1**. Then, $\mathcal{C}$ calls **Enc** to generate $\mathcal{I}_b$ for $W_0 = W^*$ or $W_1 = W$, and returns $\mathcal{I}_b$ to $\mathcal{A}$.

Phase 2: $\mathcal{A}$ additionally makes queries for keyword sets $W_{d+1}, W_{d+2}, \dots$ to the token generation oracle.

- 1) $\mathcal{O}_{\text{Token}}(j, W_i)$: For each keyword set $W_i \neq W_0$ or W_1 , $\mathcal{C}$ calls **Token** to generate the token T_{W_i} by using the secret key SK_j of U_j . If T_{W_i} is not distinguished for W_0, W_1 , $\mathcal{C}$ sends T_{W_i} to $\mathcal{A}$.

Guess: $\mathcal{A}$ outputs a guess bit $b' \in \{0, 1\}$, and successfully breaks this security game if $b' = b$.

Definition 1: We say that our schemes are secure against IND-CKA if for any adversary $\mathcal{A}$, it just has a negligible advantage ϵ in breaking the above security game, namely, $\text{Adv}_{\mathcal{A}}^{\text{IND-CKA}} = |\Pr[b' = b] - (1/2)| < \epsilon$.

As for the ciphertext associated with revoked keywords, we define its security by using the modified revocation game [20], [55], [56]. That is, our schemes achieve the Indistinguishability of selective revoked keyword lists against Chosen-Plaintext Attack (IND-CPA) as the revoked keyword ciphertexts are just associated with keyword identities. The above security game between $\mathcal{A}$ and challenger $\mathcal{C}$ is shown as follows:

Init: $\mathcal{A}$ first declares a revoked keyword list $L^* = \{w_1^*, \dots, w_m^*\}$.

Setup: Given the security parameter λ , $\mathcal{C}$ first runs *Setup*, *KeyGen*, *KeyExch* algorithms to output the global public parameters PP , public/secret key pairs $\{PK_j/SK_j\}$, group common/public key pair (CK_U, PK_U) , then sends $(PP, PK_U, \{PK_j\})$ to $\mathcal{A}$.

Phase 1: $\mathcal{A}$ adaptively makes a set of queries for various keyword sets $W_1, \dots, W_d$ to the token generation oracle, note that these issued keywords are not in L^* .

- 1) $\mathcal{O}_{\text{Token}}(j, W_i)$: $\mathcal{C}$ first runs *Token* to generate the token T_{W_i} by using the secret key SK_j of U_j , then sends T_{W_i} to $\mathcal{A}$.

Challenge: $\mathcal{A}$ chooses two challenging revoked keyword lists L_0, L_1 and sends them to $\mathcal{C}$. $\mathcal{C}$ chooses a random bit $b \in \{0, 1\}$ and calls *Enc* to output the revoked keyword ciphertexts, and sends them to $\mathcal{A}$, where $j \in [1, n]$.

Phase 2: This phase is similar to *Phase 1*, but the requirement is that $\mathcal{A}$ cannot make token generation queries for keywords in L_0, L_1 .

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

Definition 2: We say that our schemes are secure against IND-CPA if for any adversary, it does not have a nonnegligible advantage ϵ in breaking the above security game, namely, $\text{Adv}_{\mathcal{A}}^{\text{IND-CPA}} = |\Pr[b' = b] - (1/2)| < \epsilon$.

V. PROPOSED REVOCABLE KEYWORD SEARCH SCHEMES

In this section, we first demonstrate the technical overview of our schemes, then present their constructions and user revocation mechanism. To facilitate the understanding, we show some notation definitions in Table II.

A. Technical Overview

In this work, we model the access permissions of DUs in a group by using the group exchange protocol [19] and conjunctive keyword search scheme [51]. Unfortunately, simply combining the above two techniques [19], [51] does not achieve RKS. At a high level, our schemes are expected to allow each DO to specify a revoked keyword list L and then compute the index component $e(g, g)^{au}$ of revoked keywords by using the group public key $e(g, g)^a$, note that u is a random element in $\mathbb{Z}_p$ and is used to randomize the group public key

TABLE II
NOTATION DEFINITIONS IN OUR SCHEMES

Notations	Definitions
$W = \{w_1, \dots, w_n\}$	Keyword space
$\text{id} = (\text{id}_1, \dots, \text{id}_n)$	Keyword identities of W
$U = \{U_1, \dots, U_q\}$	A group of DUs
(PK_j, SK_j)	Public/secret key pair of U_j
(CK_U, PK_U)	Group common/public key pair of U
$L = \{w_1, \dots, w_m\}$	Revoked keyword list
$y = (y_1, \dots, y_n)$	Coefficient vector for L
$W' = \{w'_1, \dots, w'_l\}$	Queried keyword set
$\text{id}' = \{\text{id}'_1, \dots, \text{id}'_l\}$	Queried keyword identities of W'
$x_k = (x_{k,1}, \dots, x_{k,n})$	A vector for each w'_k
$(C, \mathcal{I})$	EMR ciphertexts/indexes
$T_{W'}$	Query token for W'
$(\text{Enc}(\cdot), \text{Dec}(\cdot))$	Encryption/decryption algorithms of AES

4

$e(g, g)^a$. Besides, each DO is required to generate other index components $\mathcal{I}'$ for all keywords W using the scheme [51]. Each group member U_j is allowed to generate a token for query keywords by using secret key g^{aj} and the common key CK_U . If each keyword queried by U_j belongs to L , CS should be prevented from recovering $e(g, g)^{au}$ and then abort this search process. Otherwise, CS should be able to recover $e(g, g)^{au}$ and then match the token with other index components $\mathcal{I}'$.

To achieve the keyword revocation, a naive solution is to build indexes for unrevoked keywords by using the public keys of authorized DUs in IBE or the access policy in ABE. Only DUs with valid secret keys or attributes that satisfy the access policy can make queries. However, this solution will incur high-encryption overheads when a certain DO allows DUs to query these revoked keywords, and the encryption overheads linearly increase with the number of authorized DUs or the number of attributes in the access policy. Another alternative way is to deploy temporary access control by embedding time intervals into keyword ciphertexts. But this solution incurs high-computation overheads when the revocation time is long, and brings high-communication overheads due to time synchronization.

To solve above challenging issues, we use a revocation list to achieve keyword revocation. In our basic RKS, we use the technique [57] to construct a coefficient vector $y = (y_1, \dots, y_n)$ for revoked keyword list $L = \{w_1, \dots, w_m\}$ ($m \ll n$) from the polynomial defined by (2), where $y_{m+2}, \dots, y_n$ are set as 0, and id_i is the identity of revoked keyword $w_i \in L$. It is worth noticing that the revoked keyword ciphertexts are components of keyword indexes $\mathcal{I}$

$$P_L(z) = \prod_{w_i \in L} (z - \text{id}_i) = \sum_{i=1}^{m+1} y_i z^{i-1}. \quad (2)$$

When making a search query for a keyword set $W' = \{w'_1, \dots, w'_l\}$, a certain DU first generates a vector $x_k = (x_{k,1}, \dots, x_{k,n})$ for each query keyword $w'_k \in W'$, where $x_{k,i} = (\text{id}'_k)^{i-1}$, $i \in [1, n]$, $k \in [1, l]$. Then, the DU uses x_k to generate the token components of $T_{W'}$. If $x_k \cdot y = 0$, it implies that id'_k is a root of $P_L(z)$. In other words, the query keyword belongs to L and CS cannot recover $e(g, g)^{au}$. If $x_k \cdot y \neq 0$, the query keyword is not revoked and CS can continue to match the token with indexes.

In RKS, the size of revoked keywords increases with the variable n instead of m . To overcome this defect of RKS, we use a novel secret sharing technique in the exponent [55] to generate the revoked keyword ciphertexts in RKS+. Given the revoked keyword list $L = \{w_1, \dots, w_m\}$ with identities $id = \{id_1, \dots, id_m\}$, RKS+ chooses a secret $u \in \mathbb{Z}_p^*$ and splits it into m shares such that $u = \sum_{k=1}^m u_k$, where $k \in [1, m]$. For each share u_k , the corresponding revoked keyword w_k has two components $I_{6,k} = g_1^{u_k}$ and $I_{6,k}^* = (g_2^{id_k} h_1)^{u_k}$, where $g_1 = g^b$, $g_2 = g^{b^2}$, $h_1 = h^b$, $b \in \mathbb{Z}_p^*$, note that g, h are generators of group $\mathbb{G}$. For each query keyword $w'_i \in W'$, we randomize its identity id'_i by a random element $v \in \mathbb{Z}_p^*$ and generate the token components $T_{4,i} = (g_1^{id'_i} h)^v$, $T_5 = g^{-v}$. If $id'_i \neq id_k$, it implies that the query keyword w'_i does not belong to L , and CS can get the random element $e(g, g_2)^{uv}$ by (3), note that $e(g_1, g_1) = e(g_2, g)$. With canceling $e(g, g_2)^{uv}$, CS can have $e(g, g)^{au}$. If $id'_i = id_k$, w'_i belongs to L and CS cannot cancel $e(g, g_2)^{uv}$

$$\frac{e\left(\prod_{k=1}^m I_{6,k}^{\frac{1}{(id'_i - id_k)}}, T_{4,i}\right)}{e\left(\prod_{k=1}^m (I_{6,k}^*)^{\frac{-1}{(id'_i - id_k)}}, T_5\right)} = e(g, g_2)^{uv}. \quad (3)$$

B. Construction of RKS

To support efficient keyword revocation in the group setting, we use the group key exchange protocol [19] and revoked keyword list constructed by predicate encryption [57] as the building blocks of RKS. Roughly speaking, the group members call the group key exchange protocol Γ to generate the common key and group public key. Note that the common key shared among authorized group members is used to generate valid tokens, and the group public key embedded in an index component is utilized to check whether query keywords belong to the revoked keyword list. The construction of RKS consists of six polynomial-time algorithms, namely, Setup, KeyGen, KeyExch, Enc, Token, and Search.

The specific construction of RKS is shown as follows:

Setup(1^λ): Given the security parameter λ , TA first generates the bilinear pairing parameters $(\mathbb{G}, \mathbb{G}_T, e, p, g)$, where g is a generator of group $\mathbb{G}$. Then, TA selects two hash functions $H_1 : \{0, 1\}^* \rightarrow \mathbb{G}$, $H_2 : \{0, 1\}^* \rightarrow \mathbb{G}$ and random elements $\beta, b_1, \dots, b_n \in \mathbb{Z}_p^*$. H_1, H_2 are chosen with different mapping rules for security concerns. Finally, TA computes $h = g^\beta$, $h_i = g^{b_i}$ ($i \in [1, n]$) and outputs the global public parameters PP by

$$PP = (\mathbb{G}, \mathbb{G}_T, e, p, g, H_1, H_2, h, \{h_i\}). \quad (4)$$

KeyGen(U, PP): Assume that there are a group $U = \{U_1, \dots, U_q\}$ of authorized DUs, TA first chooses a random element $\alpha_j \in \mathbb{Z}_p^*$ and computes g^{α_j} for each $U_j \in U$. Then, TA outputs the public/secret key pair (PK_j, SK_j) of U_j by (1), where $j \in [1, q]$

$$PK_j = g^{\alpha_j}, SK_j = \alpha_j. \quad (5)$$

Algorithm 1: Encryption Process of Enc

Input: Global public parameters PP, group U, group public key PK_U , EMR set M, keyword dictionary W and revoked keyword set L

Output: EMR ciphertexts $\mathcal{C}$, indexes $\mathcal{I}$ and coefficient vector y

```

1 foreach EMR  $rec \in M$  do
2    $\lfloor$  Encrypt  $rec$  as  $C_{rec} = \text{Enc}(rec)$ ;
3 Given the keyword dictionary  $W = \{w_1, \dots, w_n\}$  and revoked keyword set  $L = \{w_1, \dots, w_m\}$ ;
4 Specify a polynomial  $P_L(z)$  by calling (2);
5 Define a coefficient vector  $y = (y_1, \dots, y_n)$ ;
6 if  $m+2 \leq i \leq n$  then
7    $\lfloor$  Set  $y_i = 0$ ;
8 else
9    $\lfloor$   $y_i$  is the coefficient of  $P_L(z)$ ;
10 Choose random elements  $r, s, u \in \mathbb{Z}_p^*$ ;
11 for  $i = 1; i \leq n; i++ = 1$  do
12    $\lfloor$  Compute  $A_i = H_1(w_i)$ ,  $B_i = H_2(w_i)$ ;
13    $\lfloor$  Compute  $I_{i,1}, I_{i,2}$ ;
14 Compute  $I_1, I_2, I_3$ ;
15 for  $j = 1; j \leq q; j++ = 1$  do
16    $\lfloor$  Compute  $I_{4,j}$ ;
17 Set EMR ciphertexts  $\mathcal{C} = \{C_{rec}\}$ ;
18 Set EMR indexes  $\mathcal{I} = (I_1, I_2, I_3, \{I_{4,j}\}, \{I_{i,1}, I_{i,2}\})$ ;
19 return  $\mathcal{C}, \mathcal{I}, y$ .
```

KeyExch($U, \{PK_j, SK_j\}$): This group U calls the group key exchange protocol Γ to negotiate a common key CK_U , and publishes the group public key PK_U by

$$CK_U = g^\alpha = g^{\alpha_1 \alpha_2 + \dots + \alpha_q \alpha_1}$$

$$PK_U = e(g, CK_U) = e(g, g)^\alpha. \quad (6)$$

Enc(PK_U, PP, M, W, L, U): When a certain DO shares EMRs with the authorized group U, (s)he first specifies a revoked keyword list $L = \{w_1, \dots, w_m\} \subset W$ ($m \ll n$) with corresponding keyword identities $\{id_1, \dots, id_m\}$, and defines a coefficient vector $y = (y_1, \dots, y_n)$ of polynomial generated by (2). Then, DO encrypts EMRs and builds indexes as follows.

- 1) For each EMR $rec \in M$, DO utilizes the traditional symmetric encryption algorithm (e.g., AES) to encrypt it as $C_{rec} = \text{Enc}(rec)$, where $\text{Enc}(\cdot)$ is the encryption algorithm of AES.
- 2) For each keyword $w_i \in W = \{w_1, \dots, w_n\}$ with corresponding identities $id = (id_1, \dots, id_n)$, DO first chooses random elements $r, s, u \in \mathbb{Z}_p^*$, and computes $I_1, I_2, I_3, I_{i,1}$. Then, DO computes $I_{i,2}$ for each revoked keyword $w_i \in L$ and $I_{4,j}$ for each authorized $U_j \in U$. The terms $I_1, I_2, I_3, I_{4,j}, I_{i,1}, I_{i,2}$ are defined by (7), where $A_i = H_1(w_i)$, $B_i = H_2(w_i)$, $j \in [1, q]$, $i \in [1, n]$

$$I_1 = g^r, I_2 = e(g, g)^{u\alpha}, I_3 = g^s$$

$$I_{4,j} = PK_j^s, I_{i,1} = A_i^r B_i^s, I_{i,2} = h^{y_i u} h_i^u \quad (7)$$

Finally, DO sends the EMR ciphertexts $\mathcal{C}$, indexes $\mathcal{I}$ and the coefficient vector y to CS. The specific encryption process of Enc is shown in Algorithm 1.

Token(W', id', CK_U, PP, SK_j): When making a search query for a keyword set $W' = \{w'_1, \dots, w'_l\}$ that has corresponding identity set $id' = \{id'_1, \dots, id'_l\}$, a certain U_j first defines a vector $x_k = (x_{k,1}, \dots, x_{k,n})$ for each query

Algorithm 2: Token Generation Process of Token

Input: Global public parameters PP, query keyword set W' , the identity set id' of W' , common key CK_U and secret key SK_j of U_j

Output: Token $T_{W'}$ and vector set $\{x_k\}$ for W'

- 1 Given the query keyword set $W' = \{w'_1, \dots, w'_l\}$ and corresponding identity set $\text{id}' = \{\text{id}'_1, \dots, \text{id}'_l\}$;
- 2 Choose random elements $t, v \in \mathbb{Z}_p^*$;
- 3 **for** $k = 1; k \leq l; k++ = 1$ **do**
- 4 Define a vector $x_k = (x_{k,1}, \dots, x_{k,n})$;
- 5 **for** $i = 1; i \leq n; i++ = 1$ **do**
- 6 Compute $x_{k,i} = (\text{id}'_k)^{i-1}$;
- 7 Compute $T_{4,k}$;
- 8 Compute T_0, T_1, T_2, T_3, T_5 ;
- 9 Set the token $T_{W'} = \{T_0, T_1, T_2, T_3, \{T_{4,k}\}, T_5\}$;
- 10 **return** $T_{W'}, \{x_k\}$.

keyword w'_k , where $x_{k,i} = (\text{id}'_k)^{i-1}$, $k \in [1, l]$ and $i \in [1, n]$. Then, U_j chooses random elements $t, v \in \mathbb{Z}_p^*$ and computes $T_0, T_1, T_2, T_3, T_{4,k}, T_5$ by (8). Finally, U_j sends the token $T_{W'} = \{T_0, T_1, T_2, T_3, \{T_{4,k}\}, T_5\}$ and the vector set $\{x_k\}$ to CS. The specific token generation process is shown in Algorithm 2

$$T_0 = g^t, T_1 = \left(\prod_{k=1}^l H_1(w'_k) \right)^t, T_2 = \left(\prod_{k=1}^l H_2(w'_k) \right)^{t/\text{SK}_j}$$

$$T_3 = \text{CK}_U \cdot h^v, T_{4,k} = \left(\prod_{i=1}^n h_i^{x_{k,i}} \right)^v, T_5 = g^v. \quad (8)$$

Search(PP, $T_{W'}$, $\mathcal{C}$, $\mathcal{I}$, $\{x_k\}, y$): Upon receiving the token $T_{W'}$, CS first computes the intermediate variable $\eta_k = \prod_{i=1}^n I_{i,2}^{x_{k,i}}$. Then, CS checks whether each query keyword w'_k belongs to the revoked keyword list L by

$$I_2 \cdot e(\eta_k, T_5)^{1/x_k \cdot y} \stackrel{?}{=} e(I_3, T_3 \cdot T_{4,k}^{1/x_k \cdot y}). \quad (9)$$

If (9) does not hold, CS returns the empty set “ $\perp$,” which implies that the query keyword w'_k is revoked. Otherwise, CS continues to check whether the token $T_{W'}$ matches with indexes by (10) on condition that $W' \cap L = \emptyset$. If (10) also holds, CS returns the relevant results *Result* to U_j . Otherwise, CS returns the empty set $\perp$. The specific ciphertext search process is shown in Algorithm 3. It is worth noticing that U_j can decrypt these results by calling $\text{Dec}(\cdot)$ known only to DO and DUs

$$e\left(T_0, \prod_{k=1}^l I_{k,1}\right) \stackrel{?}{=} e(I_1, T_1) e(I_{4,j}, T_2). \quad (10)$$

Remark: RKS supports multikeyword search, which greatly reduces the ciphertext search costs. RKS allows the authorized group members $U = \{U_1, \dots, U_q\}$ to make valid search queries based on multiple keywords, but DUs outside of this group cannot as they do not have the common key CK_U . RKS also allows DOs to flexibly specify the revoked keyword list L . If each query keyword belongs to L , CS does not need to match the submitted token with each index, which further reduces the ciphertext search costs. However, in RKS, the sizes of public keys and revoked keyword ciphertexts are affected by the number of keywords in W . For example, the storage cost of index component $I_{i,2}$ linearly increases with the variable n rather than the number m of revoked keywords, note that

Algorithm 3: Ciphertext Search Process of Search

Input: Global public parameters PP, token $T_{W'}$, EMR ciphertexts $\mathcal{C}$, indexes $\mathcal{I}$ and vectors $\{x_k\}, y$

Output: Query results

- 1 Given the token $T_{W'}$, indexes $\mathcal{I}$ and vectors $\{x_k\}, y$;
- 2 **for** $k = 1; k \leq l; k++ = 1$ **do**
- 3 **for** $i = 1; i \leq n; i++ = 1$ **do**
- 4 Compute $I_{i,2}^{x_{k,i}}$;
- 5 Compute $\eta_k = \prod_{i=1}^n I_{i,2}^{x_{k,i}}$;
- 6 Check whether the query keyword $w'_k \in L$ by calling (9);
- 7 **if** (9) holds **then**
- 8 CS returns the empty set “ $\perp$ ”;
- 9 **else**
- 10 CS continues to call (9);
- 11 **if** $W' \cap L \neq \emptyset$ **then**
- 12 CS abort the search process;
- 13 **else**
- 14 CS calls (10) to check whether $T_{W'}$ matches with $\mathcal{I}$;
- 15 **if** (10) holds **then**
- 16 CS returns the query results *Result*;
- 17 **else**
- 18 CS returns the empty set “ $\perp$ ”;
- 19 **return** *Result* or “ $\perp$ ”.

$m \ll n$. Furthermore, RKS has to compute the vector x_k and intermediate variable η_k in **Token** and **Search**, respectively, which inevitably increases the computation overheads of token generation and ciphertext search.

It is worth noticing that our scheme RKS supports both static keyword revocation and dynamic keyword revocation. Suppose DO intends to change the revoked keyword list as $L' = \{w_1, \dots, w_{m'}\} \subset W(m' \ll n)$, then he just needs to redefine the coefficient vector by calling (2), without updating the terms $I_1, I_2, I_3, I_{4,j}, I_{i,1}, I_{i,2}$. Thus, our scheme RKS can be deployed in the dynamic setting in which the revoked keyword list frequently changes.

C. Construction of RKS+

Due to $m \ll n$, RKS incurs high computation and storage overheads when generating the ciphertexts of revoked keywords. The enhanced RKS (called RKS+) aims to make the ciphertext generation costs of revoked keywords independent of the number of keywords in the dictionary. In addition to the group key exchange technique, RKS+ also utilizes the novel secret sharing technique [55] to further reduce the extra computation overheads caused by vector constructions. The construction of RKS+ consists of six polynomial-time algorithms, namely, **Setup**, **KeyGen**, **KeyExch**, **Enc**, **Token**, and **Search**. As **KeyGen** and **KeyExch** are the same as those of RKS, we just demonstrate the modified algorithms as follows:

Setup(1^λ): Given the security parameter λ , TA generates the bilinear pairing parameters $(\mathbb{G}, \mathbb{G}_T, e, p)$. Then, TA chooses two hash functions $H_1 : \{0, 1\}^* \rightarrow \mathbb{G}$, $H_2 : \{0, 1\}^* \rightarrow \mathbb{G}$ and random elements $b \in \mathbb{Z}_p^*$, $g, h \in \mathbb{G}$, where g, h are two generators of group $\mathbb{G}$. H_1, H_2 are chosen with different mapping rules for security concerns. Finally, TA computes $g_1 = g^b$, $g_2 = g^{b^2}$, $h_1 = h^b$ and outputs the global public

parameters PP by

$$PP = (\mathbb{G}, \mathbb{G}_T, e, p, H_1, H_2, g, h, g_1, g_2, h_1). \quad (11)$$

Enc(PK_U, PP, M, W, L, U): When a certain DO shares EMRs with the authorized group U, (s)he first specifies a revoked keyword list $L = \{w_1, \dots, w_m\} \subset W (m < n)$ with corresponding keyword identities $id = \{id_1, \dots, id_m\}$, and chooses random elements $u, u_1, \dots, u_m \in \mathbb{Z}_p^*$ satisfying $u = \sum_{k=1}^m u_k$. Then, DO encrypts EMRs and builds indexes as follows.

- 1) For each EMR $rec \in M$, DO utilizes the traditional symmetric encryption algorithm $Enc(\cdot)$ (e.g., AES) to encrypt it as $C_{rec} = Enc(rec)$.
- 2) For each keyword $w_i \in W$, DO chooses random elements $r, s, u \in \mathbb{Z}_p^*$, and computes $I_1, I_2, I_3, I_{5,i}$. Then, DO computes $I_{6,k}, I_{6,k}^*$ for each revoked keyword $w_k \in L$ and $I_{4,j}$ for each $U_j \in U$. The terms $I_1, I_2, I_3, I_{4,j}, I_{5,i}, I_{6,k}, I_{6,k}^*$ are defined by (12), where $A_i = H_1(w_i)$, $B_i = H_2(w_i)$, $i \in [1, n]$, $k \in [1, m]$

$$\begin{aligned} I_1 &= g^r, I_2 = e(g, g)^{u\alpha}, I_3 = g^u, I_{4,j} = PK_j^s \\ I_{5,i} &= A_i^r B_i^s, I_{6,k} = g_1^{u_k}, I_{6,k}^* = \left(g_2^{id_k} h_1\right)^{u_k}. \end{aligned} \quad (12)$$

Finally, DO sends the EMR ciphertexts $\mathcal{C} = \{C_{rec}\}$, indexes $\mathcal{I} = (I_1, I_2, I_3, \{I_{4,j}\}, \{I_{5,i}\}, \{I_{6,k}, I_{6,k}^*\})$ and the revoked keyword identities $id = \{id_1, \dots, id_m\}$ to CS.

Token(W', id', CK_U, PP, SK_j): When making a search query for a query keyword set $W' = \{w'_1, \dots, w'_q\}$ that has corresponding identity set $id' = \{id'_1, \dots, id'_q\}$, a certain $U_j (j \in [1, q])$ first chooses random elements $t, v \in \mathbb{Z}_p^*$ and computes $T_0, T_1, T_2, T_3, T_{4,i}, T_5$ by (13), where $i \in [1, l]$. Finally, U_j sends the token $T_{W'} = \{T_0, T_1, T_2, T_3, \{T_{4,i}\}, T_5\}$ and the query keyword identities $id' = \{id'_i\}$ to CS

$$\begin{aligned} T_0 &= g^t, T_1 = \left(\prod_{i=1}^l H_1(w'_i)\right)^t, T_2 = \left(\prod_{i=1}^l H_2(w'_i)\right)^{t/SK_j} \\ T_3 &= CK_U \cdot g_2^v, T_{4,i} = \left(g_1^{id'_i} h_1\right)^v, T_5 = g^{-v} \end{aligned} \quad (13)$$

Remark: Keyword guessing attack refers to a situation where a eavesdropper or malicious server obtains a specific trapdoor and exhausts all possible keywords to generate ciphertext for testing, in order to determine the keywords contained in the trapdoor. The attacker attempts to use exhaustive methods to try various possible inputs to guess the keywords in the search trapdoor. However, this scheme supports conjunctive keyword queries, and the combination types of keywords are diverse, which makes the attacker's exhaustive method infeasible.

Search(PP, T_{W'}, C, I, id', id): Upon receiving the token $T_{W'}$, CS first calls (3) to compute $e(g, g_2)^{uv}$, then checks whether each query keyword w'_i belongs to the revoked keyword list L by

$$I_2 \cdot e(g, g_2)^{uv} \stackrel{?}{=} e(I_3, T_3). \quad (14)$$

If (14) does not hold, CS returns the empty set $\perp$, which means that the query keyword w'_i is revoked. Otherwise, CS continues to check whether the token $T_{W'}$ matches with

indexes by (10) on condition that $W' \cap L = \emptyset$. If (15) also holds, CS returns the relevant results $result$ to U_j . Otherwise, CS returns the empty set $\perp$

$$e\left(T_0, \prod_{i=1}^l I_{5,i}\right) \stackrel{?}{=} e(I_1, T_1) e(I_{4,j}, T_2). \quad (15)$$

Remark: RKS+ not only has the advantages (e.g., authorized group sharing, multikeyword search) of RKS but also makes the ciphertext generation overheads of revoked keywords linearly grow with the variable m rather than n . Furthermore, RKS+ avoids computing the variables x_k, η_k , which significantly reduces the token generation and ciphertext search costs. Although authorized DUs need to compute multiple token components $(T_{4,1}, \dots, T_{4,l})$, it does not incur a high-computation burden on DUs due to $l \ll n$. To protect keyword privacy, CS is just allowed to access keyword identities but not keyword dictionary.

Different from our basic scheme RKS, our enhanced scheme RKS+ needs to update a fraction of terms $\{I_{6,k}, I_{6,k}^*\}$ rather than the whole indexes $\mathcal{I}$. Thus, RKS+ can also support the dynamic keyword revocation efficiently. In the RKS+ scheme, we have optimized the keyword revocation process to minimize the update overhead. Instead of re-encrypting the entire keyword ciphertext, we only need to update the specific components related to the revoked keywords. This approach significantly reduces the computational cost and storage overhead associated with keyword revocation, making it more efficient and scalable compared to the basic RKS scheme.

D. User Revocations in Our Proposed Schemes

In the dynamic setting, some DUs may be revoked from $U = \{U_1, \dots, U_q\}$. For the newly formed group $U^* = \{U_1, \dots, U_{q^*}\}$, these group members call the group key exchange protocol Γ to generate the new common/public key pair (CK_U^*, PK_U^*) , where $CK_U^* = g^{\alpha^*}$, $PK_U^* = e(g, g)^{\alpha^*}$. To prevent revoked DUs having g^α from generating the valid token that satisfies the previous indexes, each DO recomputes $I_2^* = e(g, g)^{u\alpha^*}$ and then updates $I_2 = e(g, g)^{u\alpha}$ stored in CS. Thus, the revoked DUs cannot make valid search queries using the old or outdated tokens, thereby achieving the forward security. It is worth noticing that TA does not have to be online as it no longer needs to distribute public/secret key pairs for unrevoked users.

VI. SECURITY ANALYSIS

In this section, we formally prove that our schemes guarantee forward security and token unlinkability. When a certain DU is revoked from the group, this group will negotiate the new common key g^{α^*} and group public key $e(g, g)^{\alpha^*}$. Then, the DO updates the index component I_2 as $e(g, g)^{\alpha^*u}$. Thus, the revoked DUs having g^α cannot generate valid tokens, thereby guaranteeing the forward security. In Token, each DU uses the random element $t \in \mathbb{Z}_p^*$ to hide keywords, which further guarantees the token unlinkability.

Then, we formally prove that our proposed RKS or RKS+ is secure against IND-CKA by using the decision linear

Diffie–Hellman (DLDH) assumption. Let Z_1, Z_2, Z_3 be random elements in the group $\mathbb{G}$ and a, b, c be elements in the field $\mathbb{Z}_p$. Given a set of elements $(Z_1, Z_2, Z_3, Z_1^a, Z_2^b, Z_3^c)$, the goal of DLDH problem is to output $c = a + b$. It is worth noticing that an algorithm for solving the DLDH problem can be used to solve the decisional Diffie–Hellman (DDH) problem [51]. That is, the DLDH problem is still hard even though the DDH problem is easy in the bilinear groups. The advantage of an algorithm $\mathcal{A}$ is defined by

$$\text{Adv}_{\mathcal{A}}^{\text{DLDH}} = \left| \Pr \left[\mathcal{A}(Z_1, Z_2, Z_3, Z_1^a, Z_2^b, Z_3^{a+b}) = 1 \right] - \Pr \left[\mathcal{A}(Z_1, Z_2, Z_3, Z_1^a, Z_2^b, Z_3^c) = 1 \right] \right|. \quad (16)$$

Definition 3: We say that the DLDH assumption holds in the group $\mathbb{G}$ if there exists no adversary $\mathcal{A}$ that can solve the DLDH problem with a nonnegligible advantage ϵ .

Theorem 1: Our schemes (including RKS and RKS+) are secure against IND-CKA in the random oracle on condition that the DLDH assumption holds.

Proof: Assume that the adversary $\mathcal{A}$ can break our schemes with an advantage ϵ' , we model two hash functions $H_1(\cdot), H_2(\cdot)$ as random oracles, and construct another adversary $\mathcal{C}$ that can solve the DLDH problem by using $\mathcal{A}$.

Setup: Given DLDH parameters $(Z_1, Z_2, Z_3, V_1, V_2, V_3)$, $\mathcal{C}$ first chooses random elements $\pi_1, \dots, \pi_q \in \mathbb{Z}_p^*$ and sets $g = Z_1$, $\text{PK}_j = Z_2^{\pi_j}$, where $V_1 = Z_1^a$, $V_2 = Z_2^b$, $V_3 = Z_3^{a+b}$ or Z_3^c . The private key of $\mathcal{U}_j$ is defined as $\mu\pi_j$, where $Z_2 = Z_1^\mu$, $\mu \in \mathbb{Z}_p^*$. Then, $\mathcal{C}$ selects a random element $\theta \in \mathbb{Z}_p^*$ and keeps it secret. Finally, $\mathcal{C}$ sends the global public parameters PP and DUs' public keys $\{\text{PK}_j\}$ to $\mathcal{A}$, where $j \in [1, q]$.

Besides, $\mathcal{A}$ makes at most q_h keyword queries to $H_1(\cdot), H_2(\cdot)$ oracles. $\mathcal{C}$ responses to these queries by maintaining two lists, namely, $\mathcal{L}_{H_1} = (w_i, c_i, A_i, d_i)$ and $\mathcal{L}_{H_2} = (w_i, c_i, B_i, e_i)$. If the keyword w_i has been queried, $\mathcal{C}$ stores $A_i = H_1(w_i)$, $B_i = H_2(w_i)$ in $\mathcal{L}_{H_1}, \mathcal{L}_{H_2}$, respectively. Otherwise, $\mathcal{C}$ selects a random bit $c_i \in \{0, 1\}$ so that $\Pr[c_i = 1] = 1/nq_t$, where n is the number of keywords and q_t is the number of token generation queries that $\mathcal{A}$ makes at most. If $c_i = 0$, $\mathcal{C}$ chooses two random elements $d_i, e_i \in \mathbb{Z}_p^*$ and then sets $A_i = Z_1^{d_i}$, $B_i = Z_2^{e_i}$. Otherwise, $\mathcal{C}$ first chooses a random element $d_i \in \mathbb{Z}_p^*$ and computes $e_i = d_i/\mu$, then sets $A_i = Z_3^{d_i}$, $B_i = Z_3^{e_i}$. $\mathcal{C}$ adds (w_i, c_i, A_i, d_i) , (w_i, c_i, B_i, e_i) to hash lists $\mathcal{L}_{H_1}, \mathcal{L}_{H_2}$, respectively, and sends (A_i, B_i) to $\mathcal{A}$.

Phase 1: $\mathcal{A}$ makes a number of keyword sets to the following token generation oracle.

- 1) $\mathcal{O}_{\text{Token}}(j, \mathbf{W}_i)$: $\mathcal{A}$, on behalf of $\mathcal{U}_j$, issues a keyword set $\mathbf{W}'_i = \{w'_{i,1}, \dots, w'_{i,l}\}$ to $\mathcal{C}$ on behalf of $\mathcal{U}_j$. $\mathcal{C}$ picks two lists $(w'_{i,k}, c_{i,k}, A_{i,k}, d_{i,k})$, $(w'_{i,k}, c_{i,k}, B_{i,k}, e_{i,k})$ by issuing $H_1(\cdot), H_2(\cdot)$ oracles, respectively, where $k \in [1, l]$. If there is any $c_{i,k} = 1$ for $k \in [1, n]$, $\mathcal{C}$ aborts this process. Otherwise, $\mathcal{C}$ selects a random element $t_i \in \mathbb{Z}_p^*$ and returns the token $T_{\mathbf{W}'_i} = (T_{i,0}, T_{i,1}, T_{i,2})$, where $T_{i,0} = Z_1^{t_i}$, $T_{i,1} = Z_1^{t_i \sum_{k=1}^l d_{i,k}}$, $T_{i,2} = Z_1^{t_i \sum_{k=1}^l e_{i,k}/\pi_j}$.

Challenge: $\mathcal{A}$ chooses a targeted keyword set $\mathbf{W}^*$ and sends it to $\mathcal{C}$. $\mathcal{C}$ first chooses a random keyword set $\mathbf{W}$ and sets two challenging keyword sets $\mathbf{W}_0 = \mathbf{W}^* = \{w_{0,1}, \dots, w_{0,N}\}$, $\mathbf{W}_1 = \mathbf{W} = \{w_{1,1}, \dots, w_{1,n}\}$. The requirement is that $\mathbf{W}_0, \mathbf{W}_1$ were not queried in *Phase 1*. Then, $\mathcal{C}$ chooses a random

bit $\hat{b} \in \{0, 1\}$ and issues all keywords of $\mathbf{W}_{\hat{b}}$ to above hash oracles, thereby gaining two lists $(w_{\hat{b},i}, c_{\hat{b},i}, A_{\hat{b},i}, d_{\hat{b},i})$, $(w_{\hat{b},i}, c_{\hat{b},i}, B_{\hat{b},i}, e_{\hat{b},i})$. If there is no $c_{\hat{b},i} = 1$ for all $i \in [1, n]$, $\mathcal{C}$ aborts. Otherwise, $\mathcal{C}$ computes the challenging index components $(I_1, I_{4,j}, I_{\hat{b},5,i})$ and sends them to $\mathcal{A}$, where $I_1 = V_1$, $I_{4,j} = V_2^{\pi_j\theta}$, $I_{\hat{b},5,i} = V_1^{d_{\hat{b},i}} V_2^{e_{\hat{b},i}\theta}$ (in case $c_{\hat{b},i} = 0$) or $V_3^{d_{\hat{b},i}}$.

Phase 2: $\mathcal{A}$ continues to make token generation queries for keyword sets excepting $\mathbf{W}_0, \mathbf{W}_1$, but it cannot make token generation queries distinguishing $\mathbf{W}_0$ from $\mathbf{W}_1$. $\mathcal{C}$ responses to these queries as the same as *Phase 1*.

Guess: $\mathcal{A}$ outputs a guess bit $\hat{b}' \in \{0, 1\}$. If $\hat{b}' = \hat{b}$, $\mathcal{C}$ outputs 1 to show that $V_3 = Z_3^{a+b}$. Otherwise, $\mathcal{C}$ outputs 0 indicating that $V_3 = Z_3^c$.

From the above proof, we say that $\mathcal{C}$ should not abort in the token generation queries or *Challenge* phase for successfully solving the DLDH problem. The probability that $\mathcal{C}$ does not abort in the token generation queries is $(1 - 1/nq_t)^{nq_t} \geq (1/4)$, and that of *Challenge* phase is $1 - (1 - [1/nq_t])^n \geq (1/nq_t)$. Thus, the probability that $\mathcal{C}$ does not abort in the above simulation is greater than $(1/4nq_t)$. If $V_3 = Z_3^{a+b}$, the challenging index components $(I_1, I_{4,j}, I_{\hat{b},5,i})$ is a valid encryption of keyword set $\mathbf{W}_{\hat{b}}$. That is, $I_1 = V_1 = Z_1^a = g^a$, $I_{4,j} = Z_2^{\pi_j\theta} = Z_2^{b\pi_j\theta} = \text{PK}_j^{b\theta}$. If $c_{\hat{b},i} = 0$, $I_{\hat{b},5,i} = Z_1^{ad_{\hat{b},i}} Z_2^{be_{\hat{b},i}\theta} = A_{\hat{b},i}^a B_{\hat{b},i}^{b\theta}$. Otherwise, $I_{\hat{b},5,i} = V_3^{d_{\hat{b},i}} = Z_3^{(a+b)d_{\hat{b},i}} = Z_3^{ad_{\hat{b},i}} (Z_3^{d_{\hat{b},i}/\theta})^{b\theta} = A_{\hat{b},i}^a B_{\hat{b},i}^{b\theta}$.

$\mathcal{A}$'s view is the same as that of real attack game such that $|\Pr[\hat{b}' = \hat{b}] - (1/2)| \geq \epsilon'$. If $V_3 = Z_3^c$ and $\mathcal{C}$ does not abort in the simulation, then we have $|\Pr[\hat{b}' = \hat{b}]| = (1/2)$. Thus, $\mathcal{C}$'s probability that breaks the DLDH problem is defined as

$$\left| \Pr \left[\mathcal{C}(Z_1, Z_2, Z_3, Z_1^a, Z_2^b, Z_3^{a+b}) = 1 \right] - \Pr \left[\mathcal{C}(Z_1, Z_2, Z_3, Z_1^a, Z_2^b, Z_3^c) = 1 \right] \right| \geq \frac{\epsilon'}{4nq_t}$$

which contradicts with the DLDH assumption. In other words, our schemes are secure against IND-CKA in the random when the DLDH assumption holds. ■

Our RKS is selectively secure against IND-CPA under the decisional bilinear Diffie–Hellman (DBDH) assumption, which has a similar proof sketch in [20] and [56]. Given a set of elements $\Lambda = \{g, g^a, g^b, g^c\}$, the goal of DBDH problem is to distinguish $e(g, g)^{abc}$ from a random element R in the group $\mathbb{G}_T$.

Definition 4: We say that the DBDH assumption holds if there exists no algorithm $\mathcal{A}$ that can solve the DBDH problem with a nonnegligible advantage ϵ , namely, $|\Pr[\mathcal{A}(\Lambda, e(g, g)^{abc}) = 1] - \Pr[\mathcal{A}(\Lambda, R) = 1]| \geq \epsilon$.

Theorem 2: RKS is secure against IND-CPA on condition that the DBDH assumption holds.

Proof: Before proving this theorem, we first recall a lemma used in [56] and [58]. Given a DBDH instance (g, g^a, g^b, g^u, R) , then we construct a simulator $\mathcal{C}$ to simulate the security game as follows. ■

Lemma 1: Let $\mathbb{G}$ be a group of order p and g be its generator. Given the elements $(g^a, g^b, c_1 \neq 0, c_2) \in \mathbb{G}^2 \times \mathbb{Z}_p^* \times \mathbb{Z}_p$, there exists an algorithm $\mathcal{B}$ that can output a two-tuples $(g^{ab} \cdot (g^{ac_1} g^{c_2})^v, g^v)$, where $v \in \mathbb{Z}_p^*$.

Init: The adversary $\mathcal{A}$ first declares a revoked keyword list $L^* = \{w_1^*, \dots, w_{m^*}^*\}$ with corresponding vector $y^* = (y_1^*, \dots, y_n^*)$, and then gives them to $\mathcal{C}$.

Setup: $\mathcal{C}$ first calls **Setup**, **KeyGen**, **KeyExch** to publish parameters $PP, \{PK_j/SK_j\}, (CK_U, PK_U)$. Besides, $\mathcal{C}$ gives the token components associated with L^* . Then, $\mathcal{C}$ chooses a random vector $\tilde{v} = (\tilde{v}_1, \dots, \tilde{v}_n)$ and sets $h = g^a = g^b$, $e(g, g)^\alpha = e(g^a, g^b)$, $h_i = g^{v_i}$, where $v = -ay^* + \tilde{v} = (v_1, \dots, v_n)$.

Phase 1: $\mathcal{A}$ issues the keyword set $W = \{w_1, \dots, w_l\}$ to the token generation oracle. $\mathcal{C}$ constructs a vector x_k for each query keyword $w_k \in W$ and runs **Token** to generate the token components $(T_3, \{T_{4,k}\}, T_5)$ for W . $\mathcal{C}$ first outputs $(T_3, T_5) = (g^\alpha h^v, g^v)$ by calling the algorithm $\mathcal{B}$ with inputs $(g^a, g^b, 1, 0)$, then computes $T_{4,k} = T_5^{x_k \cdot v}$ by (17), where $v \in \mathbb{Z}_p^*$

$$\begin{aligned} T_{4,k} &= (g^v)^{x_k \cdot v} = \left((g^a)^{-x_k \cdot y^*} \cdot g^{x_k \cdot \tilde{v}} \right)^v \\ &= \left(\prod_{i=1}^n ((g^a)^{-y_i^*} \cdot g^{\tilde{v}_i})^{x_{k,i}} \right)^v = \left(\prod_{i=1}^n h_i^{x_{k,i}} \right)^v. \end{aligned} \quad (17)$$

Challenge: $\mathcal{A}$ submits two challenging revoked keyword lists L_0, L_1 . $\mathcal{C}$ selects a random bit $b \in \{0, 1\}$ and runs **Enc** to generate the index components $(I_{b,2}, I_{b,3}, I_{b,i,2})$ by (18), where $u \in \mathbb{Z}_p^*$. It is worth noticing that $(I_{b,2}, I_{b,3}, I_{b,i,2})$ have valid forms if $R = e(g, g)^{abu}$

$$\begin{aligned} I_{b,2} &= eg, g^{abu}; I_{b,3} = g^u; \\ I_{b,i,2} &= (g^u)^{\tilde{v}_i} = \left((g^a)^{y_i^*} \cdot (g^a)^{-y_i^*} \cdot g^{\tilde{v}_i} \right)^u = (h_i^{y_i^*} h_i)^u. \end{aligned} \quad (18)$$

Phase 2: This phase is similar to *phase 1*, but $\mathcal{A}$ is not allowed to make token generation queries for keywords in L_0, L_1 .

Guess: $\mathcal{A}$ outputs a guess bit $b' \in \{0, 1\}$. If $b' = b$, $\mathcal{C}$ returns 1 meaning $R = e(g, g)^{abu}$, which indicates that $\mathcal{C}$ has a nonnegligible advantage in the above simulation. Otherwise, $\mathcal{C}$ returns 0, which means that R is a random element in $\mathbb{G}_T$. If $\mathcal{A}$ breaks the above security game, $\mathcal{C}$ will have a nonnegligible advantage in solving the DBDH problem, which conflicts the DBDH assumption. Thus, $\mathcal{A}$ cannot break the above security game, which guarantees that RKS is secure against IND-CPA.

Our RKS+ is selectively secure against IND-CPA under the $\tilde{q}$ -decisional multiexponent bilinear Diffie-Hellman (MEBDH) assumption, which has a similar proof sketch in [20] and [55]. Given a set of elements Λ in the following, the goal of $\tilde{q}$ -MEBDH problem is to distinguish $e(g, g)^{\alpha u} \in \mathbb{G}_T$ from a random element R in $\mathbb{G}_T$, where $\alpha, u, a_1, \dots, a_{\tilde{q}} \in \mathbb{Z}_p^*$

$$\left\{ g, g^u, e(g, g)^\alpha, g^{a_i}, g^{a_i u}, g^{a_i a_j}, g^{\alpha/a_i^2} (i, j \in [1, \tilde{q}]) \right. \\ \left. g^{a_i a_j u}, g^{\alpha a_j/a_i^2}, g^{\alpha a_i/a_j^2} (i, j, k \in [1, \tilde{q}], i \neq j) \right\}.$$

Definition 5: We say that the $\tilde{q}$ -MEBDH assumption holds if there exists no algorithm $\mathcal{A}$ that can solve the $\tilde{q}$ -MEBDH problem with a nonnegligible advantage ϵ , namely, $|\Pr[\mathcal{A}(\Lambda, e(g, g)^{\alpha u}) = 1] - \Pr[\mathcal{A}(\Lambda, R) = 1]| \geq \epsilon$.

Theorem 3: RKS+ is secure against IND-CPA on condition that the $\tilde{q}$ -MEBDH assumption holds.

Proof: Assume that there exists an adversary $\mathcal{A}$ that can break the IND-CPA security game with a nonnegligible advantage ϵ and attack the index components of at most $\tilde{q}$ revoked keywords. Then, we build a simulator $\mathcal{C}$ that can solve the $\tilde{q}$ -MEBDH problem. Given the $\tilde{q}$ -MEBDH instances $(\Lambda, e(g, g)^{\alpha u})$, (Λ, R) , $\mathcal{C}$ simulates the security game as follows.

Init: $\mathcal{A}$ first declares a revoked keyword list $L^* = \{w_1^*, \dots, w_{m^*}^*\}$, and then gives L^* to $\mathcal{C}$, where $m^* \leq \tilde{q}$.

Setup: $\mathcal{C}$ first calls **Setup**, **KeyGen**, **KeyExch** to publish parameters $PP, \{PK_j/SK_j\}, (CK_U, PK_U)$. Besides, $\mathcal{C}$ gives the token components associated with L^* . Then, $\mathcal{C}$ sets $b = a_1 + \dots + a_{m^*}$ and publishes the following parameters $g, g_1, g_2, h, e(g, g)^\alpha$. Note that g_1, g_2, h are defined by (19) and the revocation keyword list L^* is reflected in the construction of h , where $y \in \mathbb{Z}_p^*$

$$\begin{aligned} g_1 &= g^b = \prod_{i \in [1, m^*]} g^{a_i}, g_2 = g^{b^2} = \prod_{i, j \in [1, m^*]} g^{a_i a_j} \\ h &= \prod_{i \in [1, m^*]} (g^{a_i})^{-\text{id}_i} g^{y_i}. \end{aligned} \quad (19)$$

Phase 1: $\mathcal{C}$ runs **Token** to generate the token components for the query keyword set $W = \{w_1, \dots, w_l\}$ submitted by $\mathcal{A}$. For each keyword identity id_i in W , $\mathcal{C}$ chooses $\sigma_i \in \mathbb{Z}_p^*$. With v_i , $\mathcal{C}$ first generates the following token components $T_3, T_{4,i}, T_5$ for each query keyword w_i , then sends them to $\mathcal{A}$. It is worth noticing that $\mathcal{C}$ can generate the valid token components $T_3, T_{4,i}, T_5$ by implicitly setting $v_i = -\alpha/a_i^2 + \sigma_i$

$$\begin{aligned} T_3 &= \prod_{\substack{j, k \in [1, l] \\ j \neq k}} g^{-\alpha a_j a_k / a_i^2} \prod_{j, k \in [1, l]} g^{a_j a_k \sigma_i} \\ T_{4,i} &= \left(\prod_{\substack{j \in [1, l] \\ j \neq i}} (g^{\frac{-\alpha a_j}{a_i^2}})^{\text{id}_i - \text{id}_j} g^{(\text{id}_i - \text{id}_j) a_j \sigma_i} \right) \left(g^{\frac{-\alpha}{a_i^2}} \right)^y g^{y \sigma_i} \\ T_5 &= g^{\alpha/a_i^2} g^{-\sigma_i}. \end{aligned}$$

Challenge: $\mathcal{A}$ submits two challenging revoked keyword lists L_0, L_1 . $\mathcal{C}$ first chooses a random bit $b \in \{0, 1\}$ and random elements $u'_1, \dots, u'_{m^*}$ such that $u' = \sum_{i=1}^{m^*} u'_i$, then generates the index components $I_{b,2}, I_{b,3}, I_{b,6,k}, I_{b,6,k}^*$ for revoked keywords as follows. It is worth noticing that $\mathcal{C}$ generates the valid index components for revoked keywords by implicitly setting $\tilde{u} = u + u'$ and $\tilde{u}_i = a_i u / b + u'_i$

$$\begin{aligned} I_{b,2} &= e(g, g)^{\alpha u'}, I_{b,3} = g^u g^{u'} \\ I_{b,6,k} &= g^{u a_i} \left(\prod_{j \in [1, m^*]} g^{a_j} \right)^{u'_i} \\ I_{b,6,k}^* &= \left(\prod_{\substack{j \in [1, m^*] \\ i \neq j}} (g^{u a_j a_i})^{\text{id}_i - \text{id}_j} \right) (g^{a_i u})^y (g^{b^2 \text{id}_i} h^b)^{u'_i}. \end{aligned}$$

Phase 2: This phase is similar to *phase 1*, but $\mathcal{A}$ is not allowed to make token generation queries for keywords in L_0, L_1 .

Guess: $\mathcal{A}$ outputs a guess bit $b' \in \{0, 1\}$. $\mathcal{C}$ outputs 0 to guess that $R = e(g, g)^{au}$ if $b' = b$. Otherwise, $\mathcal{C}$ outputs 1 meaning that R is a random element in $\mathbb{G}_T$. $\mathcal{C}$'s advantage in breaking the $\tilde{q}$ -MEBDH problem is defined as $\Pr[\mathcal{C}(\Lambda, R = e(g, g)^{au}) = 1] = (1/2) + \epsilon$ or $\Pr[\mathcal{C}(\Lambda, R) = 1] = (1/2)$. Thus, $\mathcal{C}$ can solve the $\tilde{q}$ -MEBDH problem with a nonnegligible advantage, which conflicts the $\tilde{q}$ -MEBDH assumption. That is, RKS+ is secure against IND-CPA. ■

VII. PERFORMANCE ANALYSIS

In this section, we evaluate the performance of our schemes in terms of theoretical complexities and experimental tests using a real-world data set.

A. Theoretical Complexities

To evaluate the feasibility and efficiency of our schemes, including RKS and RKS+, we compare them with the latest authorized keyword search (AKS) scheme [50].¹ In current research, the majority of keyword search schemes are oriented toward user permission revocation, rather than keyword revocation. This focus restricts the potential for more granular control mechanisms. Consequently, we have opted to conduct a comparative analysis with AKS, which incorporates time-controlled keyword search functionalities analogous to keyword revocation. Although there are limited solutions available for direct comparison, this approach nevertheless serves to underscore the distinctive advantages of our proposed solution. For the theoretical complexities, we mainly focus on several time-consuming operations, such as hash operation H , which maps the string of any length to the element of group $\mathbb{G}$, pairing operation P and modular exponentiation operation E (or E_T) in the 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 our proposed scheme and AKS in Table III.

In **KeyGen**, our schemes conduct one E operation to generate public/secret key pair for each DU, while AKS executes $P + 4E + E_T$ operations. AKS also needs to conduct one E operation to distribute the credential signature key pair for each DU, which incurs much more computation costs than our schemes. With the same reason, AKS also has higher storage cost than our schemes in terms of key generation. In **Enc**, AKS performs $(5E + H + E_T)n$ operations for all keywords in W . But our schemes need to generate ciphertexts for all keywords, including invoked keywords and DUs. Specifically, RKS and RKS+ execute $qE + n(4E + 2H)$, $qE + n(2E + 2H) + 3mE$, respectively. Because H is a time-consuming operation, these schemes have approximately equal computation costs. AKS takes about $n(5|\mathbb{G}| + |\mathbb{G}_T|)$ to store ciphertexts, but RKS and RKS+ spend about $q|\mathbb{G}| + 2n|\mathbb{G}|$, $q|\mathbb{G}| + n|\mathbb{G}| + 2m|\mathbb{G}|$. Thus, AKS has higher storage costs than our schemes due to $n \gg m$, $n \gg q$.

In **Token**, AKS conducts $l(7E + H)$ operations for l query keywords, and RKS and RKS+ perform about $2l(E + H) +$

¹This scheme achieves the time-controlled keyword search functionality, which is similar to the keyword revocation. Thus, we compare the performance of our schemes with the state-of-the-art scheme.

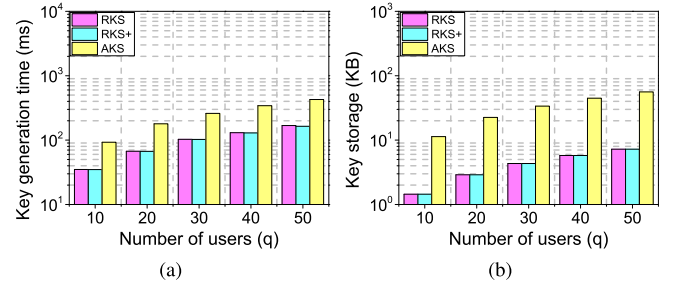


Fig. 3. Performance of KeyGen. (a) Computation cost. (b) Storage cost.

n/E , $2l(E + H) + lE$, respectively. As the token generation of RKS is affected by the vector dimension determined by n , RKS has higher computation cost than AKS due to $n \gg l$, but RKS+ is more efficient than AKS. As for the storage cost of token generation, AKS takes $5l|\mathbb{G}|$, and our schemes take $5|\mathbb{G}| + l|\mathbb{G}|$. Thus, AKS has higher storage cost when compared with our schemes. In **Search**, AKS that does not provide multikeyword search needs to conduct $5lnP$ operations for l query keywords. RKS that supports multikeyword search still performs $l(nE + 2P + E + E_T) + 3P$ operations for l query keywords, as it needs to execute $\ln E$ for intermediate variables $\{\eta_k\}$ and the inner product of vectors of length n . However, RKS+ conducts $l(3P + 2mE) + 3P$ operations that are independent of n . According to above analysis, RKS+ is much more efficient than AKS in terms of ciphertext search. As for the storage cost, AKS takes $5ln|\mathbb{G}|$, while RKS and RKS+ take $l(|\mathbb{G}| + 2|\mathbb{G}_T|) + 3|\mathbb{G}_T|$, $3l|\mathbb{G}_T| + 3|\mathbb{G}_T|$. Thus, AKS has higher storage cost than our schemes.

B. Experimental Tests

To assess the actual performance of our schemes and AKS, we conduct extensive experiments on macOS Catalina 10.15.4 with Intel Core i7 CPU 2.9 GHz by using Python3 language and pairing-based cryptography (PBC) Library. As we do not have access to public EMRs, the real-world data set used in these empirical experiments comes from the public Enron Email data set,² and has a size of 422 MB emails distributed in 3500 folders. The common Type A curve $E(F_q) : y^2 = x^3 + x$ selected has 80-bit security level, 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 an element in the file $\mathbb{Z}_p$ is 160 bits. For simplicity, we set the number of keywords as $n \in [1, 1000]$, the number of revoked keywords as $m \in [1, 25]$, the number of DUs in the group as $q \in [1, 50]$, and the number of query keywords as $l \in [1, 10]$.

From Fig. 3(a) and (b), we notice that the computation and storage costs of key generation linearly grow with the number of DUs in the group. Our schemes both take one modular exponentiation operation to distribute keys for each DU, while AKS not only generates the credential signature keys but also authorized keys. After key generation, DUs in our schemes run the key exchange protocol to negotiate the common key and group public key. The performance of key agreement is shown in Table IV. The computation and storage costs of

²<http://www.cs.cmu.edu/~enron/>

TABLE III
THEORETICAL COMPUTATION AND STORAGE COSTS: A COMPARATIVE SUMMARY

Algorithms		KeyGen	Enc	Token	Search
AKS [50]	CC	$q(P + 5E + E_T)$	$n(5E + H + E_T)$	$l(7E + H)$	$5lnP$
	SC	$q(G_T + 7 G + Z_p) + G + Z_p $	$n(5 G + G_T)$	$5l G $	$5ln G $
RKS	CC	qE	$2E + E_T + qE + n(4E + 2H)$	$3E + 2l(E + H) + nE$	$l(nE + 2P + E + E_T) + 3P$
	SC	$q(G + Z_p)$	$2 G + G_T + q G + 2n G $	$5 G + l G $	$l(G + 2 G_T) + 3 G_T $
RKS+	CC	qE	$2E + E_T + qE + n(2E + 2H) + 3mE$	$4E + 2l(E + H) + lE$	$l(3P + 2mE) + 3P$
	SC	$q(G + Z_p)$	$2 G + G_T + q G + n G + 2m G $	$5 G + l G $	$3l G_T + 3 G_T $

Notes. “CC”: Computation costs; “SC”: Storage costs; “n”: Number of keywords; “m”: Number of revoked keywords; “l”: Number of query keywords; “q”: Number of DUs in the group.

TABLE IV
PERFORMANCE OF KEY EXCHANGE PROTOCOL

values of q		10	20	30	40	50
CC (ms)	Step 1	16.57	33.60	52.54	66.25	84.93
	Step 2	15.57	30.35	46.50	59.42	78.12
	Step 3	28.25	65.69	118.18	179.69	264.79
	Total	60.39	129.60	217.22	305.36	427.83
SC (KB)	Step 1	5.63	23.75	54.38	97.50	153.13
	Step 2	5.63	23.75	54.38	97.50	153.13
	Step 3	2.50	5.00	7.50	10.00	12.50
	Total	13.76	52.50	116.26	205	318.76

Notes. “CC”: Computation costs; “SC”: Storage costs.

key agreement increase with the variable q in three phases. As for the computation cost, Step 1 or Step 3 conducts about qE operations, while Step 2 performs about $q(q-1)E$ operations. Regarding the storage cost, Step 1 or Step 2 takes about $[1/2]q(q-1)|G|$, and Step 3 takes about $q|G|$.

From Fig. 4, we notice that the computation and storage costs of AKS are affected by the variable n , while those of our schemes are affected by variables n, m, q . Thus, we evaluate the performance of Enc by varying different parameters. In Fig. 4(a) and (d), we set $m = 25, q = 50$ and $n \in [1, 1000]$. We notice that the computation and storage costs of three schemes linearly increase with the number of keywords in W . RKS has higher computation and storage costs than RKS+ as its ciphertext of revoked keywords is associated with the value of n rather than m . Without generating ciphertexts for revoked keywords, AKS has less computation cost than our schemes, but has higher storage cost. When setting $n = 1000, q = 50$ in Fig. 4(b) and (e), the computation and storage costs of AKS and RKS remain nearly constant, but are higher than those of RKS+. Although the ciphertext generation cost of RKS+ is affected by the variable m , it does not incur high overheads due to $m \ll n$. In Fig. 4(c) and (f), we fix $n = 1000, m = 25$ and notice that the computation and storage costs of AKS is independent of the variable $q \in [1, 50]$. But the ciphertext generation process of our schemes is affected by the value of q , but incurs less computation and storage burdens due to $q \ll n$. Thus, compared with AKS, our schemes have slightly higher computation burdens in Enc, but incur fewer storage overheads.

In Fig. 5, we show the computation and storage costs of token generation by varying different variables, including l, n , and m . In Fig. 5(a), and (d), we notice that the computation and storage costs of all schemes linearly increase with the value of $l \in [1, 10]$. RKS has a higher computation cost than the other two schemes as it needs to generate the vector x_k determined by the variable n , but it has less storage cost than

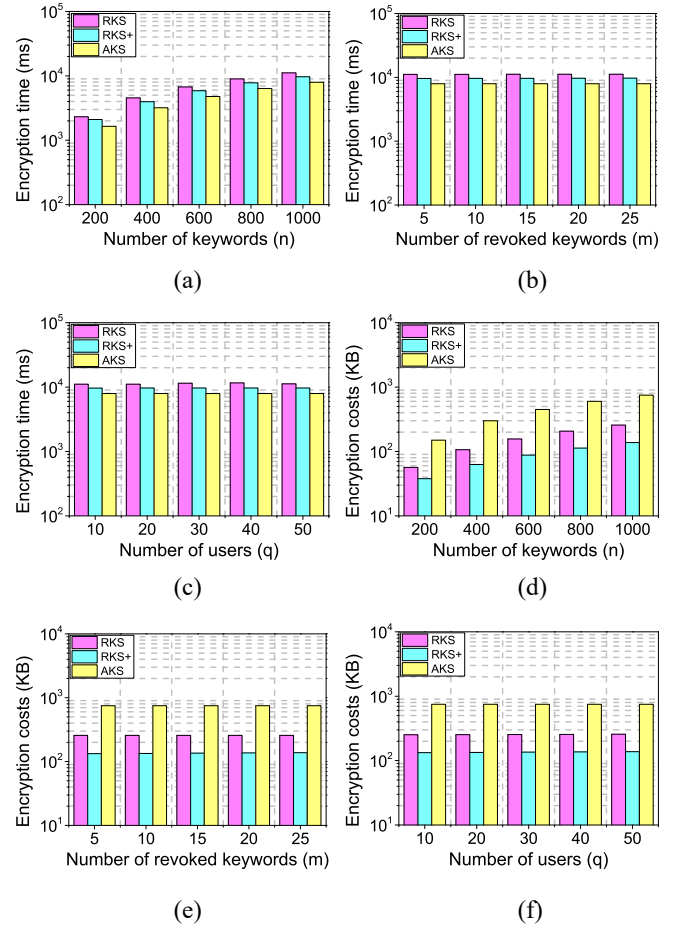


Fig. 4. Performance of Enc. (a) $m = 25, q = 50$. (b) $n = 1000, q = 50$. (c) $n = 1000, m = 25$. (d) $m = 25, q = 50$. (e) $n = 1000, q = 50$. (f) $n = 1000, m = 25$.

AKS. However, RKS+ is still much more efficient than AKS in terms of computation and storage costs. In Fig. 5(b) and (e), we set $l = 10, m = 25$ and vary n from 1 to 1000. We find that only the computation cost of RKS is affected by the variable $n \in [1, 1000]$, but the token generation costs of RKS+ and AKS keep almost unchanged. It is worth noticing that RKS+ is the most efficient among these schemes, and the storage cost of RKS is much less than that of AKS. In Fig. 5(c) and (f), two variables l, n are fixed as 10, 1000, respectively. We notice that the computation and storage costs of all schemes are not affected by the variable $m \in [1, 25]$. The token generation of RKS+ is much more efficient than AKS. But the computation cost of RKS is higher than that of AKS, this is because the token generation of RKS is affected by the variable $n \gg l, m$.

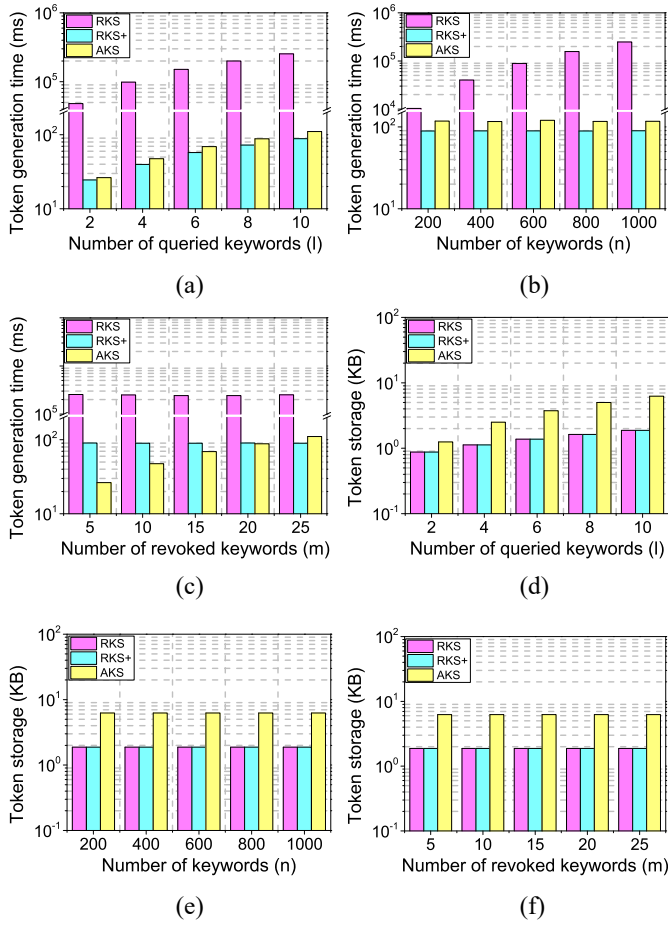


Fig. 5. Performance of Token. (a) $n = 1000, m = 25$. (b) $l = 10, m = 25$. (c) $l = 10, n = 1000$. (d) $n = 1000, m = 25$. (e) $l = 10, m = 25$. (f) $l = 10, n = 1000$.

The computation and storage costs of Search in the above three schemes are demonstrated in Fig. 6. In Fig. 6(a) and (d), we set $l = 10, m = 25$, we find that the search time of AKS and RKS linearly increases with the variable $n \in [1, 1000]$, and RKS+ that has constant search time is much more efficient than AKS and RKS. The storage costs of our schemes are independent of variable n , and are much less than that of AKS. In Fig. 6(b) and (e), with setting $n = 1000, m = 25$, the computation and storage costs of above three schemes linearly grow with the variable $l \in [1, 10]$. In this case, RKS+ has a minimal computation overhead, but RKS has a slightly higher computation cost than AKS. But RKS and RKS+ have approximately equal storage costs, and are superior to AKS in terms of storage cost. In Fig. 6(c) and (f), we set $l = 10, n = 1000$ and range m from 1 to 25. We notice that only the computation cost of RKS+ linearly increases with the variable m , and the search time and costs of AKS and RKS are not affected by m . Because RKS needs to check whether each query keyword belongs to the revoked keywords, RKS has a higher computation cost than AKS, but our proposed enhanced scheme RKS+ is much more efficient than AKS. Besides, our schemes have fewer storage costs than AKS.

From the above analysis, it is evident that RKS demonstrates high efficiency in keyword revocation operations, with an average processing time of only 2 s. This efficiency renders

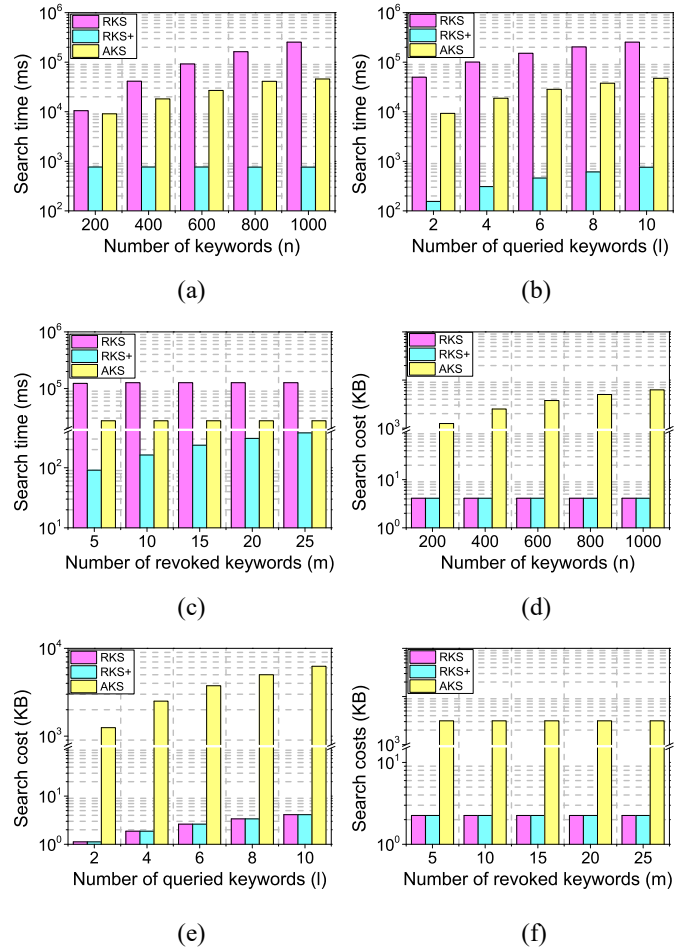


Fig. 6. Performance of Search. (a) $l = 10, m = 25$. (b) $n = 1000, m = 25$. (c) $l = 10, n = 1000$. (d) $l = 10, m = 25$. (e) $n = 1000, m = 25$. (f) $l = 10, n = 1000$.

it suitable for applications with moderate keyword update frequencies. However, as the number of keywords increases, both the public key size and computational overhead grow linearly, leading to higher computational costs relative to AKS and diminished performance in large data environments. Notably, RKS provides support for keyword revocation, a functionality absent in AKS. By contrast, RKS+ reduces the size of revoked keyword ciphertexts by approximately 40%, significantly lowering storage demands. It also cuts the time required for token generation and ciphertext retrieval by 50%, thereby substantially enhancing system performance and user experience. Compared to AKS, RKS+ proves to be significantly more efficient, and among all evaluated algorithms, RKS incurs lower storage costs than AKS. Despite these enhancements in operational performance, the initial setup time of RKS+ in multiuser environments remains relatively long, potentially impeding its deployment in real-time applications. Besides, the findings of experimental tests are consistent with theoretical analysis.

VIII. CONCLUSION

In this article, we proposed two RKS schemes in the group setting, namely, RKS and RKS+. RKS allows DOs to flexibly revoke keywords, and prevents the unauthorized

access permissions of revoked DUs by only updating an index component. But RKS has slightly higher computation costs in token generation and ciphertext search processes due to the construction of vectors. RKS+ achieves short ciphertext size, and provides efficient token generation and ciphertext search. Finally, we formally proved our schemes resist both IND-CKA and IND-CPA, and performed empirical experiments to demonstrate that its efficiency and feasibility in practical applications. Future extensions, including exploring more efficient and expressive keyword searches with fine-grained access control, protecting keyword identities, and implementing a prototype of the proposed approach (or its future extension) in a real-world setting for evaluation.

REFERENCES

- [1] Y. Yang and M. Ma, "Conjunctive keyword search with designated tester and timing enabled proxy re-encryption function for e-health clouds," *IEEE Trans. Inf. Forensics Security*, vol. 11, pp. 746–759, 2015.
- [2] H. Wu, Z. Peng, S. Guo, Y. Yang, and B. Xiao, "VQL: Efficient and verifiable cloud query services for blockchain systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 6, pp. 1393–1406, Jun. 2022, doi: [10.1109/TPDS.2021.3113873](https://doi.org/10.1109/TPDS.2021.3113873).
- [3] H. Cai, F. Ye, Y. Yang, Y. Zhu, J. Li, and F. Xiao, "Online pricing and trading of private data in correlated queries," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 3, pp. 569–585, Mar. 2021.
- [4] J. Li, Y. Zhang, J. Ning, X. Huang, G. S. Poh, and D. Wang, "Attribute based encryption with privacy protection and accountability for CloudIoT," *IEEE Trans. Cloud Comput.*, vol. 10, no. 2, pp. 762–773, Apr.–Jun. 2022.
- [5] Y. Liu, K. Xue, P. He, D. S. Wei, and M. Guizani, "An efficient, accountable, and privacy-preserving access control scheme for Internet of Things in a sharing economy environment," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6634–6646, Jul. 2020.
- [6] F. Guo, W. Susilo, and Y. Mu, "Identity-based encryption with random oracles," in *Introduction to Security Reduction*. Cham, Switzerland: Springer, 2018, pp. 215–227.
- [7] C. Ge, W. Susilo, J. Baek, Z. Liu, J. Xia, and L. Fang, "Revocable attribute-based encryption with data integrity in clouds," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 5, pp. 2864–2872, Sep./Oct. 2022, doi: [10.1109/TDSC.2021.3065999](https://doi.org/10.1109/TDSC.2021.3065999).
- [8] S. Chen, J. Li, Y. Zhang, and J. Han, "Efficient revocable attribute-based encryption with verifiable data integrity," *IEEE Internet Things J.*, vol. 11, no. 6, pp. 10441–10451, Mar. 2024.
- [9] 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. 2017.
- [10] 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.
- [11] F. Liu et al., "Volume-hiding range searchable symmetric encryption for large-scale datasets," *IEEE Trans. Dependable Secure Comput.*, early access, Nov. 21, 2023, doi: [10.1109/TDSC.2023.3335304](https://doi.org/10.1109/TDSC.2023.3335304).
- [12] 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.
- [13] 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, 2017.
- [14] Q. Zheng, S. Xu, and G. Ateniese, "VABKS: Verifiable attribute-based keyword search over outsourced encrypted data," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2014, pp. 522–530.
- [15] N. H. Sultan, N. Kaaniche, M. Laurent, and F. A. Barbhuiya, "Authorized keyword search over outsourced encrypted data in cloud environment," *IEEE Trans. Cloud Comput.*, vol. 10, no. 1, pp. 216–233, Jan.–Mar. 2022, doi: [10.1109/TCC.2019.2931896](https://doi.org/10.1109/TCC.2019.2931896).
- [16] R. Canetti, S. Halevi, and J. Katz, "A forward-secure public-key encryption scheme," *J. Cryptol.*, vol. 20, no. 3, pp. 265–294, 2007.
- [17] K. G. Paterson and E. A. Quaglia, "Time-specific encryption," in *Proc. Int. Conf. Security Cryptogr. Netw. (SCN)*, 2010, pp. 1–16.
- [18] 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, Jul./Sep. 2020.
- [19] M. Burmester and Y. Desmedt, "A secure and scalable group key exchange system," *Inf. Process. Lett.*, vol. 94, no. 3, pp. 137–143, 2005.
- [20] S. Sun, A. Sakzad, R. Steinfeld, J. K. Liu, and D. Gu, "Public-key puncturable encryption: Modular and compact constructions," in *Proc. IACR Int. Conf. Public-Key Cryptogr. (PKC)*, 2020, pp. 309–338.
- [21] J. Ning, J. Chen, K. Liang, J. K. Liu, C. Su, and Q. Wu, "Efficient encrypted data search with expressive queries and flexible update," *IEEE Trans. Services Comput.*, vol. 15, no. 3, pp. 1619–1633, May/Jun. 2022, doi: [10.1109/TSC.2020.3004988](https://doi.org/10.1109/TSC.2020.3004988).
- [22] Y. Wang and D. Papadopoulos, "Multi-user collusion-resistant searchable encryption for cloud storage," *IEEE Trans. Cloud Comput.*, vol. 11, no. 3, pp. 2993–3008, Jul.–Sep. 2023.
- [23] Z. Jiang, K. Zhang, L. Wang, and J. Ning, "Forward secure public-key authenticated encryption with conjunctive keyword search," *Comput. J.*, vol. 66, no. 9, pp. 2265–2278, 2023.
- [24] S. Niu, Y. Hu, Y. Su, S. Yan, and S. Zhou, "Attribute-based searchable encrypted scheme with edge computing for Industrial Internet of Things," *J. Syst. Archit.*, vol. 139, Jun. 2023, Art. no. 102889.
- [25] Q. Huang, Q. Wei, G. Yan, L. Zou, and Y. Yang, "Fast and privacy-preserving attribute-based keyword search in cloud document services," *IEEE Trans. Services Comput.*, vol. 16, no. 5, pp. 3348–3360, Sep./Oct. 2023.
- [26] 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. 2019.
- [27] Y. Miao, R. Deng, X. Liu, K.-K. R. Choo, H. Wu, and H. Li, "Multi-authority attribute-based keyword search over encrypted cloud data," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 4, pp. 1667–1680, Jul./Aug. 2021, doi: [10.1109/TDSC.2019.2935044](https://doi.org/10.1109/TDSC.2019.2935044).
- [28] F. Bao, R. H. Deng, X. Ding, and Y. Yang, "Private query on encrypted data in multi-user settings," in *Proc. Int. Conf. Inf. Security Pract. Exp. (ISPEC)*, 2008, pp. 71–85.
- [29] J. Shu, K. Yang, X. Jia, X. Liu, C. Wang, and R. Deng, "Proxy-free privacy-preserving task matching with efficient revocation in crowd-sourcing," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 1, pp. 117–130, Jan./Feb. 2021, doi: [10.1109/TDSC.2018.2875682](https://doi.org/10.1109/TDSC.2018.2875682).
- [30] D. Ghopur, J. Ma, X. Ma, J. Hao, T. Jiang, and X. Wang, "Puncturable key-policy attribute-based encryption scheme for efficient user revocation," *IEEE Trans. Services Comput.*, vol. 16, no. 6, pp. 3999–4011, Nov./Dec. 2023.
- [31] J. Zhang, S. Su, H. Zhong, J. Cui, and D. He, "Identity-based broadcast proxy re-encryption for flexible data sharing in VANETs," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 4830–4842, 2023.
- [32] X. Zhang, C. Xu, H. Wang, Y. Zhang, and S. Wang, "FS-PEKS: Lattice-based forward secure public-key encryption with keyword search for cloud-assisted Industrial Internet of Things," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 3, pp. 1019–1032, May/Jun. 2021, doi: [10.1109/TDSC.2019.2914117](https://doi.org/10.1109/TDSC.2019.2914117).
- [33] Y. Miao et al., "REKS: Role-based encrypted keyword search with enhanced access control for Outsourced cloud data," *IEEE Trans. Dependable Secure Comput.*, early access, Oct. 16, 2023, doi: [10.1109/TDSC.2023.3324640](https://doi.org/10.1109/TDSC.2023.3324640).
- [34] Y. Miao et al., "Verifiable Outsourced attribute-based encryption scheme for cloud-assisted mobile E-health system," *IEEE Trans. Dependable Secure Comput.*, early access, Jul. 4, 2023, doi: [10.1109/TDSC.2023.3292129](https://doi.org/10.1109/TDSC.2023.3292129).
- [35] B. Huang, P. Huang, H. Yuan, and S. Liang, "A verifiable ranked ciphertext retrieval scheme based on bilinear mapping," *Concurr. Comput., Pract. Exp.*, vol. 34, no. 12, Art. no. e5829, 2022.
- [36] 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](https://doi.org/10.1109/TDSC.2019.2940573).
- [37] J. Cui, H. Zhou, H. Zhong, and Y. Xu, "AKSER: Attribute-based keyword search with efficient revocation in cloud computing," *Inf. Sci.*, vol. 423, pp. 343–352, Jan. 2018.
- [38] S. Liu, J. Yu, Y. Xiao, Z. Wan, S. Wang, and B. Yan, "BC-SABE: Blockchain-aided searchable attribute-based encryption for cloud-IoT," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 7851–7867, Sep. 2020.
- [39] Y. Yang, X. Liu, X. Zheng, C. Rong, and W. Guo, "Efficient traceable authorization search system for secure cloud storage," *IEEE Trans. Cloud Comput.*, vol. 8, no. 3, pp. 819–832, Jul.–Sep. 2018.

- [40] S. Qiu, J. Liu, Y. Shi, and R. Zhang, "Hidden policy ciphertext-policy attribute-based encryption with keyword search against keyword guessing attack," *Sci. China Inf. Sci.*, vol. 60, no. 5, pp. 1–12, 2017.
- [41] 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. Clust., Cloud Internet Comput. (CCGRID)*, 2020, pp. 450–458.
- [42] C.-I. Fan and S.-Y. Huang, "Controllable privacy preserving search based on symmetric predicate encryption in cloud storage," *Future Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1716–1724, 2013.
- [43] Y. Miao et al., "Time-controllable keyword search scheme with efficient revocation in mobile E-health cloud," *IEEE Trans. Mobile Comput.*, vol. 23, no. 5, pp. 3650–3665, May 2024.
- [44] T. Yang, S. Ma, J. Du, C. Jiang, and Q. Huang, "Revocable public key encryption with equality test without pairing in cloud storage," *Comput. J.*, vol. 67, no. 2, pp. 642–657, 2024.
- [45] F. Meng and L. Cheng, "STR-ABKS: Server-aided traceable and revocable attribute-based encryption with keyword search," *IEEE Internet Things J.*, vol. 11, no. 7, pp. 12649–12659, Apr. 2024.
- [46] F. Luo, H. Wang, and X. Yan, "Re-PAEKS: Public-key authenticated re-encryption with keyword search," *IEEE Trans. Mobile Comput.*, early access, Mar. 5, 2024, doi: [10.1109/TMC.2024.3373626](https://doi.org/10.1109/TMC.2024.3373626).
- [47] J. Zhang and J. Mao, "Efficient public key encryption with revocable keyword search in cloud computing," *Clust. Comput.*, vol. 19, no. 3, pp. 1211–1217, 2016.
- [48] Y. Yu, J. Ni, H. Yang, Y. Mu, and W. Susilo, "Efficient public key encryption with revocable keyword search," *Secur. Commun. Netw.*, vol. 7, no. 2, pp. 466–472, 2014.
- [49] L. Xu, J. Li, X. Chen, W. Li, S. Tang, and H.-T. Wu, "Tc-PEDCKS: Towards time controlled public key encryption with delegatable conjunctive keyword search for Internet of Things," *J. Netw. Comput. Appl.*, vol. 128, pp. 11–20, Feb. 2019.
- [50] 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.
- [51] Y. H. Hwang and P. J. Lee, "Public key encryption with conjunctive keyword search and its extension to a multi-user system," in *Proc. Int. Conf. Pair. Based Cryptogr. (Pairing)*, 2007, pp. 2–22.
- [52] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in *Proc. IEEE INFOCOM*, 2010, pp. 1–9, doi: [10.1109/INFOCOM.2010.5462174](https://doi.org/10.1109/INFOCOM.2010.5462174).
- [53] X. Ding, P. Liu, and H. Jin, "Privacy-preserving multi-keyword top- k similarity search over encrypted data," *IEEE Trans. Dependable Secure Comput.*, vol. 16, no. 2, pp. 344–357, Mar./Apr. 2019.
- [54] L. Song, Y. Miao, J. Weng, K.-K. R. Choo, X. Liu, and R. H. Deng, "Privacy-preserving threshold-based image retrieval in cloud-assisted Internet of Things," *IEEE Internet Things J.*, vol. 9, no. 15, pp. 13598–13611, Aug. 2022.
- [55] A. Lewko, A. Sahai, and B. Waters, "Revocation systems with very small private keys," in *Proc. IEEE Symp. Security Privacy (SP)*, 2010, pp. 273–285.
- [56] S. Yamada, N. Attrapadung, G. Hanaoka, and N. Kunihiro, "A framework and compact constructions for non-monotonic attribute-based encryption," in *Proc. IACR Int. Conf. Public-Key Cryptogr. (PKC)*, 2014, pp. 275–292.
- [57] J. Katz, A. Sahai, and B. Waters, "Predicate encryption supporting disjunctions, polynomial equations, and inner products," in *Proc. Annu. Int. Conf. Theory Appl. Cryptogr. Techn.*, 2008, pp. 146–162.
- [58] D. Boneh and X. Boyen, "Secure identity based encryption without random oracles," in *Proc. Annu. Int. Cryptol. Conf. (CRYPTO)*, 2004, pp. 443–459.

Shuqin Liu (Member, IEEE) received the B.E. degree from the Department of Software Engineering, Xi'an Shiyong University, Xi'an, China, in 2007, the master's degree from the Department of computer science and technology, Ji'nan University, Guangzhou, China, in 2010, and the Ph.D. degree from the Department of Computer Science and Technology, Northwest University, Xi'an, in 2016.

She is also a Domestic Visitor with Xidian University, Xi'an, from September 2020 to September 2021. She is currently a Lecturer with the Department of Software Engineering, Xi'an University of Posts and Telecommunications, Xi'an. Her research interests include data security and privacy protection.

Yinbin Miao (Member, IEEE) received the B.E. degree from the Department of Telecommunication Engineering, Jilin University, Changchun, China, in 2011, and the Ph.D. degree from the Department of Telecommunication Engineering, Xidian University, Xi'an, China, in 2016.

He is also a Postdoctoral Fellow with Nanyang Technological University, Singapore, from September 2018 to September 2019, and a Postdoctoral Fellow with The City University of Hong Kong, Hong Kong, from December 2019 to December 2021. He is currently a Professor with the Department of Cyber Engineering, Xidian University. His research interests include information security and applied cryptography.

Feng Li received the B.E. degree from the School of Cyberspace Security, Hangzhou Dianzi University, Hangzhou, China, in 2018. He is currently pursuing the Ph.D. degree with the School of Cyber Engineering, Xidian University, Xi'an, China.

His current research interests include cloud computing security.

Xinghua Li received the M.E. and Ph.D. degrees in computer science from Xidian University, Xi'an, China, in 2004 and 2007, respectively.

He is currently a Professor with the School of Cyber Engineering, Xidian University. His research interests include wireless networks security, privacy protection, cloud computing, and security protocol formal methodology.

Kim-Kwang Raymond Choo (Senior Member, IEEE) received the Ph.D. degree in information security from Queensland University of Technology, Brisbane, QLD, Australia, in 2006.

He currently holds the Cloud Technology Endowed Professorship with The University of Texas at San Antonio, San Antonio, TX, USA.

Dr. Choo is the recipient of the 2019 IEEE Technical Committee on Scalable Computing Award for Excellence in Scalable Computing (Middle Career Researcher). He is an ACM Distinguished Speaker and an IEEE Computer Society Distinguished Visitor from 2021 to 2023, included in Web of Science's Highly Cited Researcher in the field of Cross-Field in 2020.

Robert H. Deng (Fellow, IEEE) received the B.E. degree from National University of Defence Technology, China, in 1981, the M.S. degree from Illinois Institute of Technology, United States of America, in 1983, and Ph.D. degree from Illinois Institute of Technology, United States of America, in 1985.

He is an AXA Chair Professor of Cybersecurity and the Professor of Information Systems with the School of Information Systems, Singapore Management University, Singapore, since 2004. His research interests include data security and privacy, multimedia security, network and system security.

Dr. Deng has received the Distinguished Paper Award (NDSS 2012), the Best Paper Award (CMS 2012), and the Best Journal Paper Award (IEEE Communications Society 2017). He served/is serving on the editorial boards for many international journals, including IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY and IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING.