RABAEKS: Revocable Attribute-based Authenticated Encrypted Search over Lattice for Multi-receiver Cloud Storage

Yibo Cao, Shiyuan Xu, Xiu-Bo Chen, and Siu-Ming Yiu

Abstract—With the widespread development of cloud storage, searching over the encrypted data (without decryption) has become a crucial issue. Public key authenticated encryption with keyword search (PAEKS) retrieves encrypted data, and resists inside keyword guessing attacks (IKGAs). Most PAEKS schemes cannot support access control in multi-receiver models. To address this concern, attribute-based authenticated encryption with keyword search (ABAEKS) has been studied. However, the access privilege for the ciphertext may change, and the conventional cryptographic primitives are not resistant to quantum computing attacks, which exhibits a limited applicability and poor security for cloud storage. In this paper, we propose RABAEKS, the first post-quantum revocable attribute-based authenticated encrypted search scheme for multi-receiver cloud storage. Our design enables cloud server enforces the access control of data receivers in the search process. For practical consideration, we further introduce a revocation mechanism of data receivers, which makes the access control more dynamic. We then define and rigorously analyze the security our scheme. Through the performance evaluations and comparisons, our computational overhead of ciphertext generation, trapdoor generation and search algorithm are at least $20\times$, $1.67\times$ and $1897\times$ faster than prior arts, respectively, which is practical for cloud storage.

Index Terms—Cloud storage, authenticated searchable encryption, revocation, access control.

1 Introduction

Cloud storage has facilitated the retrieval and sharing of big data to a greater extent. Data owners can upload large amounts of data to cloud servers (e.g., Amazon Web Services, IBM Cloud), effectively reducing local storage overheads and enhancing service elasticity [1], [2]. Meanwhile, cloud storage also causes serious privacy and security concerns, i.e., the disclosure of sensitive data [3], [4], [5], [6], [7], [8]. To protect data privacy, many researchers adopt encryption-before-outsourcing method, but this creates a barrier to query data. In order to ensure data privacy without losing data availability, public key encryption with authenticated keyword search (PAEKS) scheme has been formalized. It takes the secret key of data owner as the input to the encryption algorithm to resist keyword guessing attacks (KGAs) and inside keyword guessing attacks (IKGAs) [9]. Utilizing PAEKS, data receivers can encrypt their data before uploading to cloud servers, allowing the cloud server to search directly over the encrypted data, thereby ensuring data privacy.

Unfortunately, most PAEKS schemes only support single data receiver [9], [10], [11], [12], [13], which is not practical for cloud storage, as the data outsourcing process typically involves multiple data receivers. A straightforward multi-receiver PAEKS scheme can be implemented by taking each data receiver to submit a search trapdoor one by one, but this would introduce additional computational and communication overheads. To address this limitation, some researchers have presented multi-receiver PAEKS schemes [14], [15], [16], [17], where the specific process of ciphertext search is presented in Fig. 1. Among these schemes, the most representative one is a key policy attribute-based authenticated encryption with keyword search (ABAEKS) scheme through introducing the attribute-based encryption (ABE) into PAEKS [17]. It assigns an attribute to keyword ciphertext, allowing a receiver can access the specific ciphertext that meets the access policy. Comparing to key policy ABAEKS, a ciphertext policy ABAEKS scheme can provide that the data (or keyword) ciphertext can be accessed by the specific data receivers to realize the access control in cloud storage practically.

However, designing a ciphertext policy ABAEKS scheme still faces two significant problems. On the one hand, in real-world cloud scenarios, the access privilege for the data (or keyword) ciphertext may change, causing a specific data receiver to lose access privilege [18]. To bridge the gap, revocation mechanism serves as a cornerstone to achieve dynamic access control, which provides the security for the data in systems [19], [20], [21], [22], [23], [24]. It usually requires the cloud server maintaining a revocation list, rendering the data (or keyword) ciphertext inaccessible to the data receivers on that list. When the revocation list changes, the data (or keyword) ciphertext will also be updated. On the other hand, quantum computers pose a severe security threat to the conventional cryptography [25], [26]. For example, Shor's algorithm can solve the discrete logarithmic (DL) hardness in probabilistic polynomial time (PPT) [27]. To address this, lattice-based cryptography is considered as a promising approach to resist quantum computing attacks.

- *Corresponding author: Xiu-Bo Chen*
- *Y. Cao and X.-B. Chen are with the Information Security Center, State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China. (E-mail: caoyibo@bupt.edu.cn, flyover100@163.com).*
- *Y. Cao, S. Xu, and S.-M. Yiu are with the Department of Computer Science, The University of Hong Kong, Pok Fu Lam, Hong Kong. (E-mail: syxu2@cs.hku.hk, smyiu@cs.hku.hk).*

Specifically, the learning with errors (LWE) hardness is a common assumption in lattice-based cryptography, which as hard as worst-case standard lattice assumptions, and has been widely used in the lattice cryptographic constructions [28], [29]. Therefore, it is crucial to construct a lattice-based revocable ABAEKS scheme for cloud storage.

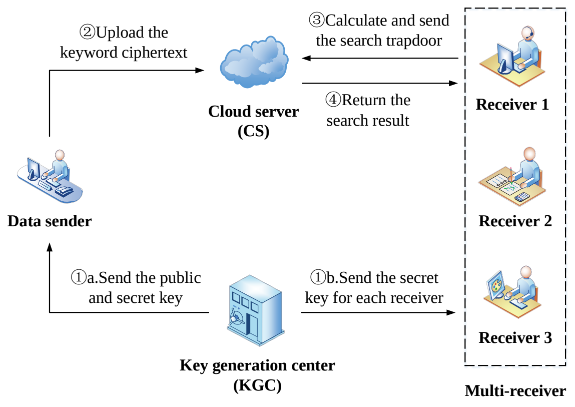

②Upload the keyword ciphertext
③Calculate and send the search trapdoor
④Return the search result
Cloud server (CS)
Receiver 1
Receiver 2
Receiver 3
Data sender
①a.Send the public and secret key
①b.Send the secret key for each receiver
Key generation center (KGC)
Multi-receiver

Fig. 1. The ciphertext search in multi-receiver models.

To get around the above-mentioned issues, in this paper, we propose RABAEKS, a post-quantum revocable attribute-based authenticated encrypted search scheme for multi-receiver cloud storage. We leverage the philosophy of lattice hardness to against quantum computing attacks. We also construct the attribute set $\mathcal{S}$ and access policy $(\mathcal{W}, t)$, and use the Lagrange interpolation algorithm to achieve access control for data receivers. Accordingly, when a data receiver submits a search trapdoor, the cloud server initially determines whether its attribute satisfies the access policy. With regard to the revocation mechanism, we implement the construction of revocation list $\mathcal{R}$ utilized the binary tree structure [24] and KUNodes algorithm [30]. While a trivial method to construct a RABAEKS scheme is to combine a ABAEKS scheme [17] with the revocation method [24] directly, this approach emerges two challenges. On the one hand, in [17], the attribute is embedded in the keyword ciphertext (aka. key-policy), which cannot provide attributes to the data receivers. On the other hand, finding a correct lattice basis for the SampleLeft algorithm is impossible (as detailed in Sect. 5.1). To address these issues, we introduce GenSamplePre algorithm to construct a ciphertext-policy solution by using the secret key (lattice basis) of a data receiver to sample a vector as part of the search trapdoor. We hereby summarize our four-fold contributions as follows.

- We propose a post-quantum revocable attribute-based authenticated encrypted search scheme for multi-receiver cloud storage, namely RABAEKS, which not only achieves multi-receiver and dynamic access control, but also provides resistance to quantum computing attacks and IKGAs. To the best of our knowledge, our design is the first lattice-based ciphertext policy RABAEKS scheme.
- Our RABAEKS design is based on the lattice algebraic structure, which adopts several lattice sampling algorithms. From a higher perspective, the secret key for the data owner and data receivers are generated by TrapGen algorithm and SampleBasis algorithm, respectively, and a search trapdoor is calculated by GenSamplePre algorithm to realize the keyword ciphertext search.
- We implement the access control structure through the threshold secret sharing technique based on Lagrange interpolation. Additionally, we introduce a revocation list, the binary tree structure and KUNodes algorithm to achieve the receiver revocation and ciphertext update, thereby offering the dynamic access control.
- We provide the security model and security analysis in the random oracle model (ROM), which demonstrates that our RABAEKS scheme enjoys IND-CKA and IND-KGA in a quantum setting. Furthermore, comprehensive performance evaluations and comparisons indicate that our RABAEKS is more efficient in terms of ciphertext generation, trapdoor generation and search algorithms compared to previous prior arts [31], [32], [33]. In particular, these three algorithms offer significant improvements over prior arts, with up to $20\times$, $1.67\times$, and $1897\times$ faster performance.

The rest of this paper is organized as follows. Section 2 presents numerous related works to showcase recent advancements. Following that, Section 3 provides an introduction to the preliminary concepts. The system models, threat models, formal definitions, and security models for RABAEKS are then depicted in Section 4. A detailed explanation and its security analysis of RABAEKS scheme is demonstrated in Section 5, while Section 6 focuses on the security analysis of our RABAEKS scheme. In Section 7, we delve into the performance evaluation and comparison. Finally, we summarize this paper in Section 8.

2 Related Works

We review the literature in the context of PAEKS, attribute-based encrypted search and revocation scheme, and lattice-based solution.

2.1 Public Key Authenticated Encryption with Keyword Search

Huang et al. [9] formalized the first PAEKS scheme, to resist IKGAs during ciphertext search, which has been proven secure in the ROM. In their scheme, a data sender not only encrypts the keywords but also authenticates them with its own secret key so that the keyword ciphertext can only be calculated by itself. Following this, a great deal of PAEKS schemes have been conducted. For example, He et al. [10] proposed a certificateless PAEKS scheme, ensuring the security of the ciphertext search for Industrial Internet of Things (IIoT). To fight against adaptively-chosen-targets adversaries, Lu et al. [11] then improved the adversary model and security definitions of PAEKS by designing a lightweight PAEKS scheme to avoid the bilinear pairing operations. However, those schemes still suffer from the poor utility for cloud storage systems.

In order to improve the practicality, numerous research contributed to designing the multi-receiver PAEKS schemes [14], [15], [16]. In 2021, a broadcast authenticated encryption

TABLE 1
Comparison with the current state-of-art schemes

Schemes	Keyword Search	IKGAs-resilience	Multi-receiver	Data Receiver Revocation	Quantum Resistance
Huang et al. [9]	✓	✓	✗	✗	✗
He et al. [10]	✓	✓	✗	✗	✗
Lu et al. [11]	✓	✓	✗	✗	✗
Liu et al. [14]	✓	✓	✓	✗	✗
Chenam et al. [15]	✓	✓	✓	✗	✗
Sun et al. [16]	✓	✓	✓	✗	✗
Miao et al. [31]	✓	✗	✓	✗	✗
Yang et al. [32]	✓	✗	✓	✗	✗
Yang et al. [33]	✓	✗	✓	✗	✗
Han et al. [19]	✗	✗	✓	✓	✗
Deng et al. [20]	✗	✗	✓	✓	✗
Ge et al. [21]	✗	✗	✓	✓	✗
Liu et al. [12]	✓	✓	✗	✗	✓
Cheng et al. [13]	✓	✓	✗	✗	✓
Luo et al. [17]	✓	✓	✓	✗	✓
Zhao et al. [22]	✗	✗	✓	✓	✓
Luo et al. [23]	✗	✗	✓	✓	✓
Guo et al. [24]	✗	✗	✓	✓	✓
Our RABAEKS	✓	✓	✓	✓	✓

with keyword search (BAEKS) was proposed by introducing the broadcast encryption to PAKES [14]. Subsequently, Chenam et al. [15] constructed a designated cloud server-based multi-receiver certificateless authenticated searchable encryption scheme with conjunctive keyword. After that, Sun et al. [16] presented a multi-receiver certificateless authenticated searchable encryption scheme for concealing search patterns, without the need of expensive bilinear pairing operations. Nevertheless, none of these aforementioned schemes can support access control.

2.2 Attribute-based Encrypted Search and Revocation Schemes

Attribute-based encrypted search schemes can provide a ciphertext search under multi-receiver scenario to achieve the privacy-preserving for cloud storage. For instance, Miao et al. [31] utilized a hidden access policy to propose a ciphertext policy attribute-based keyword search scheme, named ABKS-SM. After that, Yang et al. [32] introduced the attribute-based keyword search to present an efficient and provably secure data selective sharing and acquisition (DSA) scheme, which not only ensures the security for the cloud data, but also achieves the access control in a fine-grained manner. Moreover, a dual traceable distributed attribute-based encryption with subset keyword search scheme (abbr. DT) was constructed by Yang et al. [33] to realize data source and user trace. Most recently, Luo et al. [17] presented an attribute-based authenticated encryption with keyword search (ABAEKS) primitive to realize access control by leverage the philosophy of ABE. However, when the data receiver's status was changed, they cannot offer the dynamic access control.

Revocation mechanism can dynamically adjust the access privilege of data receivers when their status have been changed, which serves as a cornerstone towards to the access control systems [21]. To achieve the revocation, Han et al. [19] proposed a traceable and revocable ciphertext-policy attribute-based encryption (CP-ABE) scheme, through introducing a binary tree to generate revocation information and embedding it in the ciphertext to support the ciphertext update when revoking. In addition, Deng et al. [20] presented a revocable attribute-based data storage (RADS) scheme. In their scheme, revoked users cannot access either the newly uploaded files or the old ones, thereby protect the data privacy. Along with this direction, Ge et al. [21] formulated a formal definition and security model for the revocable attribute-based encryption with data integrity protection (RABE-DI) scheme, which further considers the data integrity. However, most existing schemes are constructed based on the DL hardness, and are not resilient to quantum computing attacks.

2.3 Lattice-based PAEKS and Revocation Schemes

With the development of quantum computers, conventional cryptographic primitives have been seriously threatened. Lattice-based cryptography is considered as the most promising post-quantum solution, which have been widely utilized in both PAEKS and revocation schemes.

In 2022, Liu et al. [12] proposed a generic construction of PAEKS and a lattice-based instantiation. Then, Cheng et al. [13] enhanced the security level of lattice-based PAEKS scheme to the fully ciphertext indistinguishablity (fully CI) and fully trapdoor indistinguishability (fully TI). To support the multi-receiver model, Luo et al. [17] formalized an attribute-based authenticated encryption with keyword search scheme over lattice, named ABAEKS, to realize the IND-CKA and IND-KGA security under quantum computing attacks.

With regard to the revocation mechanism, a revocable ABE scheme over lattice was presented by Zhao et al. [22] for privacy-preserving in the cloud storage system, and its security was reduced to the Ring-LWE hardness. After that, Luo et al. [23] constructed a revocable key policy ABE scheme reduced on LWE hardness, which utilizes the KUNodes algorithm to realize revocation. Nevertheless, during the revocation process of their scheme, the secret key of data receiver is necessary, which causes the secret key leakage concerns. To mitigate this, Guo et al. [24] proposed a novel lattice-based revocable ciphertext policy ABE scheme that also introduces the KUNodes algorithm.

Consequently, to our best knowledge, none of the existing ABAEKS schemes can not only support receiver revocation, but also resist quantum computing attacks. The property comparison with the existing state-of-the-art solutions is demonstrated in Table 1.

3 Preliminaries

In this section, we introduce the preliminaries toward to our RABAEKS scheme, i.e., lattice definition, discrete Gaussian distribution, LWE hardness, and lattice basis sampling algorithms.

Definition 1. [34] Assume that a matrix $\mathbf{E} = (\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_m)$ is composed of m linearly independent vectors, we define the lattice Λ as:

$$\Lambda = \Lambda(\mathbf{E}) = \{x_1\mathbf{e}_1 + x_2\mathbf{e}_2 + \cdots + x_m\mathbf{e}_m | x_i \in \mathbb{Z}, i \in [m]\}.$$

We say that $\mathbf{E}$ is a lattice basis of Λ.

Definition 2. [35] Given three integers n, m, q, and a matrix $\mathbf{E} \in \mathbb{Z}_q^{n\times m}$, we define a q-ary integer lattice as:

$$\Lambda_q(\mathbf{E}) := \{\mathbf{h} \in \mathbb{Z}^m | \exists \mathbf{s} \in \mathbb{Z}_q^n, \mathbf{E}^\top \mathbf{s} = \mathbf{h} \bmod q\}.$$

$$\Lambda_q^\perp(\mathbf{E}) := \{\mathbf{h} \in \mathbb{Z}^m | \mathbf{E}\mathbf{h} = \mathbf{0} \bmod q\}.$$

$$\Lambda_q^{\mathbf{u}}(\mathbf{E}) := \{\mathbf{h} \in \mathbb{Z}^m | \mathbf{E}\mathbf{h} = \mathbf{u} \bmod q\}.$$

Definition 3. Given a parameter $\sigma \in \mathbb{R}^+$, a center $\mathbf{c} \in \mathbb{Z}^m$, and any vector $\mathbf{x} \in \mathbb{Z}^m$, we define the discrete Gaussian distribution over Λ as: $\mathcal{D}_{\Lambda,\sigma,\mathbf{c}}(\mathbf{x}) = \frac{\rho_{\sigma,\mathbf{c}}(\mathbf{x})}{\rho_{\sigma,\mathbf{c}}(\Lambda)}$. For $\forall \mathbf{x} \in \Lambda$, we say that $\rho_{\sigma,\mathbf{c}}(\mathbf{x}) = \exp(-\pi \frac{\|\mathbf{x}-\mathbf{c}\|^2}{\sigma^2})$, and $\rho_{\sigma,\mathbf{c}}(\Lambda) = \sum_{\mathbf{x}\in\Lambda} \rho_{\sigma,\mathbf{c}}(\mathbf{x})$.

Definition 4. [36] Given three integers n,m,q, and an error distribution $\chi = \Psi_\alpha$, we define the $\mathrm{LWE}_{n,m,q,\chi}$ hardness as distinguishing $(\mathbf{A}, \mathbf{A}^\top\mathbf{s} + \mathbf{e})$ and $(\mathbf{A}, \mathbf{x})$, where $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}, \mathbf{s} \leftarrow \mathbb{Z}_q^n, \mathbf{e} \leftarrow \chi^m$, and $\mathbf{x} \leftarrow \mathbb{Z}_q^m$. Moreover, according to [37], the secret vector $\mathbf{s}$ can be selected in χ^n.

Lemma 1. [38] Given a lattice Λ and its lattice basis $\mathbf{T_A}$, we can say: $\Pr[\|\mathbf{x}\| > \sigma\sqrt{m} : \mathbf{x} \leftarrow \mathcal{D}_{\Lambda,\sigma}] \leq \mathsf{negl}(m)$, and $\sigma \geq \|\widetilde{\mathbf{T_A}}\| \cdot \omega(\sqrt{\log m})$.

Lemma 2. [39] Given three integers n, $q \geq 2$, and $m \geq 2n\log q$, the probabilistic polynomial time (PPT) algorithm $\mathsf{TrapGen}(n, m, q)$ computes a uniform matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and its basis $\mathbf{T_A} \in \mathbb{Z}_q^{m\times m}$ over $\Lambda_q^\perp(\mathbf{A})$. We can say that $\mathbf{A}$ is statistically close to uniform distribution over $\mathbb{Z}^{n\times m}$ and $\|\widetilde{\mathbf{T_A}}\| \leq \mathcal{O}(\sqrt{n\log m})$.

Assume that four positive integers n, $q \geq 2$, τ, $m \geq 2n\log q$, a matrix $\mathbf{E} = (\mathbf{E}_1 \mid \mathbf{E}_2 \mid \cdots \mid \mathbf{E}_\tau) \in \mathbb{Z}_q^{n\times\tau m}$, and a set $\mathcal{N} = \{i_1, i_2, \cdots, i_j\} \subset [\tau]$, we let $\mathbf{E}_\mathcal{N} := (\mathbf{E}_{i_1} \mid \mathbf{E}_{i_2} \mid \cdots \mid \mathbf{E}_{i_j}) \in \mathbb{Z}_q^{n\times jm}$. Now, we give the following two Lemmas 3 and 4.

Lemma 3. [40] Taking a matrix $\mathbf{E} \in \mathbb{Z}_q^{n\times\tau m}$, a basis $\mathbf{T}_{\mathbf{E}_\mathcal{N}}$ over $\Lambda_q^\perp(\mathbf{E}_\mathcal{N})$, a set $\mathcal{N} \subset [\tau]$, and a Gaussian parameter $L \geq \|\widetilde{\mathbf{T}_{\mathbf{E}_\mathcal{N}}}\| \cdot \sqrt{\tau m} \cdot \omega(\sqrt{\log \tau m})$ as input, the PPT algorithm $\mathsf{SampleBasis}(\mathbf{E}, \mathbf{T}_{\mathbf{E}_\mathcal{N}}, \mathcal{N}, L)$ computes a matrix $\mathbf{T}'_\mathbf{E}$. We say that $\mathbf{T}'_\mathbf{E}$ is a lattice basis of $\Lambda_q^\perp(\mathbf{E})$, and $\|\widetilde{\mathbf{T}'_\mathbf{E}}\| \leq \sigma$ with overwhelming probability.

Lemma 4. [40] Taking a matrix $\mathbf{E} \in \mathbb{Z}_q^{n\times\tau m}$, a lattice basis $\mathbf{T}_{\mathbf{E}_\mathcal{N}}$ over $\Lambda_q^\perp(\mathbf{E}_\mathcal{N})$, a set $\mathcal{N} \subset [\tau]$, a vector $\mathbf{u} \in \mathbb{Z}_q^n$, and a Gaussian parameter $\sigma \geq \|\widetilde{\mathbf{T}_{\mathbf{E}_\mathcal{N}}}\| \cdot \omega(\sqrt{\log \tau m})$ as input, the PPT algorithm $\mathsf{GenSamplePre}(\mathbf{E}, \mathbf{T}_{\mathbf{E}_\mathcal{N}}, \mathcal{N}, \mathbf{u}, \sigma)$ returns a vector $\mathbf{e} \in \mathbb{Z}^{\tau m}$, which is statistically close to the distribution over $\mathcal{D}_{\Lambda_q^u(\mathbf{E}),\sigma}$, satisfying $\mathbf{E}\mathbf{e} = \mathbf{u} \bmod q$.

Lemma 5. [41] Taking a matrix $\mathbf{E} \in \mathbb{Z}_q^{n\times m}$ and its basis $\mathbf{T_E} \in \mathbb{Z}_q^{m\times m}$, a vector $\mathbf{t} \in \mathbb{Z}_q^n$, a matrix $\mathbf{H} \in \mathbb{Z}_q^{n\times k}$, and $\sigma \leq \|\widetilde{\mathbf{T}}_\mathbf{E}\| \cdot \omega(\sqrt{\log(m+k)})$ as input, the $\mathsf{SampleLeft}(\mathbf{E}, \mathbf{H}, \mathbf{T_E}, \mathbf{t}, \sigma)$ algorithm returns a vector $\mathbf{s} \in \mathbb{Z}^{m+k}$, which is statistically close over the distribution $\mathcal{D}_{\Lambda_q^{\mathbf{t}}([\mathbf{E}|\mathbf{H}]),\sigma}$, satisfying $[\mathbf{E}|\mathbf{H}] \cdot \mathbf{s} = \mathbf{t} \mod q$.

Lemma 6. [41] Taking two matrices $\mathbf{E}, \mathbf{M} \in \mathbb{Z}_q^{n\times m}$ and a basis $\mathbf{T_M}$ over $\Lambda_q^\perp(\mathbf{M})$, a matrix $\mathbf{H} \in \mathbb{Z}_q^{m\times m}$, a vector $\mathbf{v} \in \mathbb{Z}_q^n$, and $\sigma > \|\widetilde{\mathbf{T}}_\mathbf{E}\| \cdot s_1(\mathbf{H}) \cdot \omega(\sqrt{\log m})$ as input, the $\mathsf{SampleRight}(\mathbf{E}, \mathbf{M}, \mathbf{H}, \mathbf{T_M}, \mathbf{v}, \sigma)$ algorithm returns a vector $\mathbf{t} \in \mathbb{Z}^{2m}$, which is statistically close to the distribution over $\mathcal{M}_{\Lambda_q^{\mathbf{v}}([\mathbf{E}\|\mathbf{EH}+\mathbf{M}]),\sigma}$.

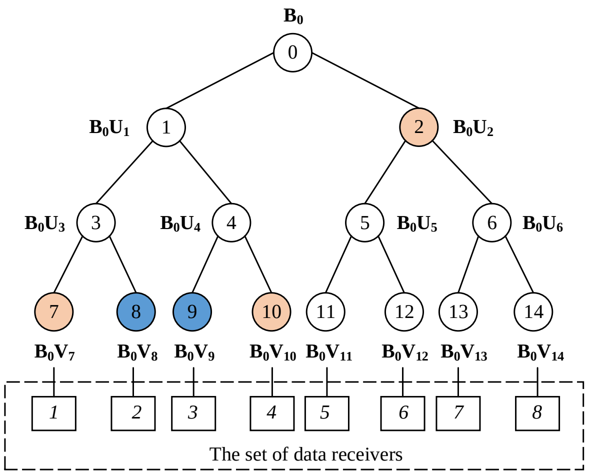


Fig. 2. The binary tree $\mathcal{T}$ for revocation.

Definition 5. [24], [30] Assuming that a set of data receivers is $\mathcal{U} := \{1, \cdots, |\mathcal{U}|\}$, a revocation list is $\mathcal{R}$, and a root matrix is $\mathbf{B}_0 \in \mathbb{Z}_q^{n\times m}$, we construct a binary tree $\mathcal{T}$ structure as showed in Fig. 2. For the non-leaf nodes $i \in [1, |\mathcal{U}| - 2]$ and the leaf nodes $j \in [|\mathcal{U}| - 1, 2|\mathcal{U}| - 2]$, we calculate $\mathbf{B}_i = \mathbf{B}_0\mathbf{U}_i$, and $\mathbf{B}_j = \mathbf{B}_0\mathbf{V}_j$, where $\mathbf{U}_i \leftarrow \{0, 1\}^{m\times m}$ is a reversible matrix such that each row and column has only one unit element and the rest of elements are zero, $\mathbf{V}_j \leftarrow \{-1, 1\}^{m\times m}$ is a reversible matrix. In $\mathcal{T}$, each leaf node corresponds to a data receiver in $\mathcal{U}$, and $\omega(\xi)$ represents the iteration of each node

Algorithm 1 KUNodes

```
Input: A binary tree $\mathcal{T}$, and a revocation list $\mathcal{R}$
Output: A set $\mathcal{Y}$
$\mathcal{X} \leftarrow \emptyset, \mathcal{Y} \leftarrow \emptyset$.
for $i \in \mathcal{R}$ do
    $\mathcal{X} = \mathcal{X} \cup \omega(i)$.
end
for $j \in \mathcal{X}$ do
    if $j_{\text{LeftNode}} \notin \mathcal{X}$ then
        $\mathcal{Y} = \mathcal{Y} \cup \{j_{\text{LeftNode}}\}$.
    end
    if $j_{\text{RightNode}} \notin \mathcal{X}$ then
        $\mathcal{Y} = \mathcal{Y} \cup \{j_{\text{RightNode}}\}$.
    end
end
if $\mathcal{Y} = \emptyset$ then
    $\mathcal{Y} = \mathcal{Y} \cup \{0\}$.
end
return $\mathcal{Y}$.
```

number in the path from root to this leaf. For example, $\omega(3) = \{0, 1, 4, 9\}$, and $\omega(5) = \{0, 2, 5, 11\}$. Then, we define the KUNodes$(\mathcal{T}, \mathcal{R})$ as Algorithm 1. For instance, if $\mathcal{R} = \{2, 3\}$, the KUNodes$(\mathcal{T}, \mathcal{R}) = \{2, 7, 10\}$ according to **Algorithm 1**. If a data receiver (i.e. leaf node) ξ in the revocation list $\mathcal{R}$, KUNodes$(\mathcal{T}, \mathcal{R}) \cap \omega(\xi) = \emptyset$. Otherwise, KUNodes$(\mathcal{T}, \mathcal{R}) \cap \omega(\xi)$ has one and only one element.

Definition 6. Given a hash function family $\mathcal{H} := \{H : X \to Y\}$ where $0 \in Y$ and a $(Q+1)$-dimension vector $\mathbf{x} = (x_0, \cdots, x_Q) \in X^{Q+1}$ as input, we define that $\mathcal{H}$ is (Q, p_{min}, p_{max}) abort-resistant if $p(\mathbf{x}) := \Pr[H(x_0) = 0 \cap H(x_1) \neq 0 \cap \cdots \cap H(x_Q) \neq 0] \in [p_{min}, p_{max}]$ for all $\mathbf{x} \in X^{Q+1}$ with $x_0 \neq \{x_1, \cdots, x_Q\}$ and $H \in \mathcal{H}$. Specifically, we define a abort-resistant hash function $H : \mathbb{Z}_q^k \setminus \{0\}^k \to \mathbb{Z}_q$ as $H(\mathbf{m}) := 1 + \sum_{i=1}^{k} \theta_i m_i$, where q is a prime, $\boldsymbol{\theta} := (\theta_1, \cdots, \theta_k) \in \mathbb{Z}_q^k$, and $\mathbf{m} := (m_1, \cdots, m_k) \in \mathbb{Z}_q^k \setminus \{0\}^k$.

Lemma 7. [42] Assume that q is a prime and $0 \leq Q \leq q$, the hash function $H : \mathbb{Z}_q^k \setminus \{0\}^k \to \mathbb{Z}_q$ defined in ***Definition 6*** is $(Q, \frac{1}{q}(1 - \frac{Q}{q}), \frac{1}{q})$ abort-resistant.

4 Framework Formulation

In this section, we show the system models, threat models, formal definitions and security models. The notations utilized in our scheme are defined in Table 2.

TABLE 2
Nomenclature

Symbol	Description
λ	The security parameter
$\|kw\|$	Then length of keyword
$\|att\|$	The number of system attribute
l	The number of receiver attribute, where $l \leq \|att\|$
$\mathcal{ATT}$	The system attribute, where $\mathcal{ATT} := \{1, \cdots, \|att\|\}$
$\mathcal{U}$	The set of data receivers
$\mathcal{R}$	The revocation list
pp	The public parameter
msk	The system master secret key
(pk_S, sk_S)	The public and secret key of data sender
$\mathcal{S}$	The receiver attribute, where $\mathcal{S} \subseteq \mathcal{ATT}$
sk_R	The secret key of data receiver
$(\mathcal{W}, t)$	The access policy, where $\mathcal{W} \subset \mathcal{ATT}$, and $t \in [1, \|att\|)$
$\mathbf{ck}$	The ciphertext keyword, where $\mathbf{ck} \in \{0,1\}^{\|kw\|}$
CT	The keyword ciphertext
$\mathbf{tk}$	The trapdoor keyword, where $\mathbf{tk} \in \{0,1\}^{\|kw\|}$
TD	The search trapdoor

4.1 System Models

In our RABAEKS scheme, there are four entities involved, Key generation center (KGC), Data sender, Data receivers, and Cloud server(CS), as depicted in Fig. 3.

The KGC is in charge of initializing the system and generating the key for data sender and receivers. For a data sender, the KGC calculates its public and secret key according to the public parameter. For a data receiver, the KGC calculates its secret key according to the public parameter, its attribute, and master secret key.

The data sender has a collection of data files, and extracts the keyword from it. After receiving the public and secret key from KGC, the data sender encrypts the keyword with its own secret key and an access policy through invoking **RABAEKS** algorithm, to obtain the keyword ciphertext that can only be accessed by some specific data receivers. Finally, the data sender uploads the ciphertext to CS.

Each data receiver owns an attribute in our scheme, which can be utilized to generate its secret key in KGC. When the data receiver has a search requirement, it calculates a search trapdoor using **Trapdoor** algorithm, and sends it to CS. If the attribute meets the access condition (i.e. meets the attribute policy $(\mathcal{W}, t)$, and is not in the revocation list) and the trapdoor matches the corresponding ciphertext, the data receiver obtains the search result from the CS.

The CS provides the data storage and access control functions. It stores the keyword ciphertext with an access policy and a revocation list (RL). After received a search trapdoor from a data receiver, CS checks whether its attribute meets access condition, including its attribute meets access policy defined by the data sender and it is not in the RL. If the access condition is matched, CS performs the **Test** algorithm, and sends corresponding search result to the data receiver. Moreover, when the RL is updated, CS invokes the **CTUpdate** algorithm to refresh the keyword ciphertext.

4.2 Threat Models

Assuming that the communication channel is secure, we present the threat assumptions of four entities as follows.

- **KGC** is designed to be *fully trusted*. It generates valid public and secret keys for data sender and data receivers, and is not interested in the data files and keywords stored in the cloud server.
- **Data Sender** is considered to be *fully trusted*. It protects the secret key and submits the valid keyword ciphertext to the cloud server honestly.
- **Data Receivers** are assumed to be *malicious*. They generate valid trapdoors according to the keyword to be searched, while they are interested in speculating on

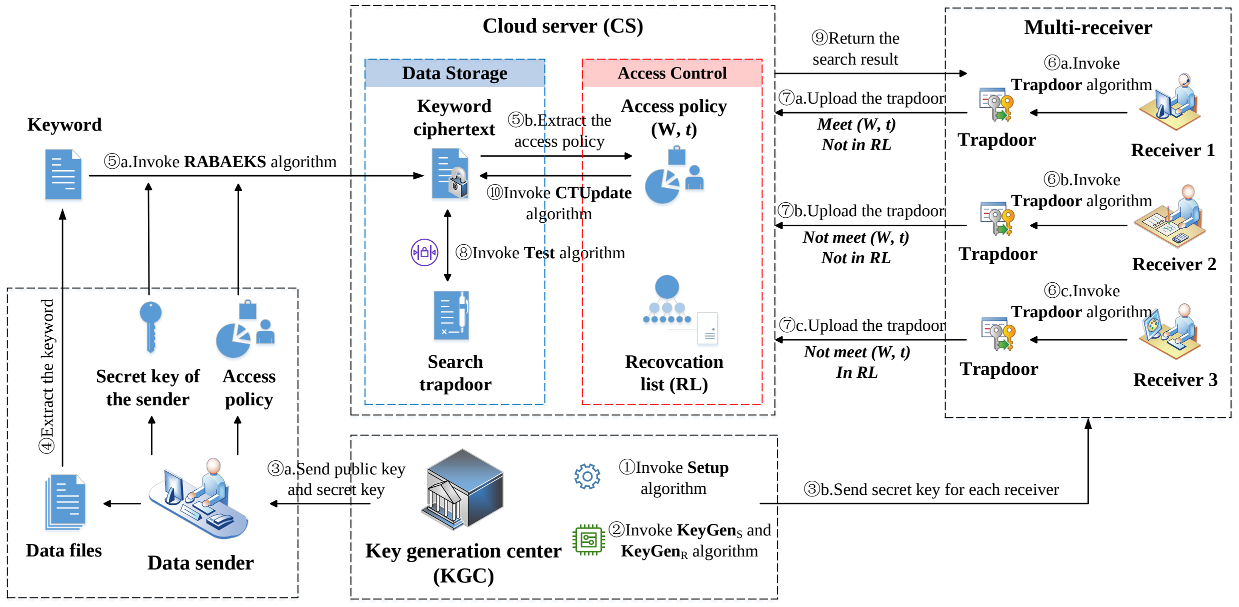


Fig. 3. System models of our RABAEKS for cloud storage.

the search context of other receivers by performing an unauthorised search.

- **CS** is considered to be *honest-but-curious*. It strictly executes the search algorithm and returns the correct search results to data receivers that satisfy the access conditions, while it is it is eager to obtain the data files and keywords.

4.3 Formal Definitions

Our RABAEKS scheme consists of seven algorithms Π_{RABAEKS} = (**Setup**, **KeyGen**$_S$, **KeyGen**$_R$, **RABAEKS**, **Trapdoor**, **Test**, **CTUpdate**). We describe their formal definitions as follows.

- $(pp, msk) \leftarrow$ **Setup**(λ): Given a security parameter λ, this algorithm returns a public parameter pp and a master secret key msk.
- $(pk_S, sk_S) \leftarrow$ **KeyGen**$_S(pp)$: Given a public parameter pp, this algorithm returns the public and secret keys (pk_S, sk_S) to the data sender.
- $sk_R \leftarrow$ **KeyGen**$_R(pp, \mathcal{S}, msk)$: Given a public parameter pp, a receiver attribute set $\mathcal{S} \subseteq \mathcal{ATT}$, and a master secret key msk, this algorithm returns the secret keys sk_R to the data receiver.
- CT $\leftarrow$ **RABAEKS**$(pp, pk_S, sk_S, (\mathcal{W}, t), \mathbf{ck})$: Given a public parameter pp, a data sender's public and secret keys pk_S and sk_S, an access policy $(\mathcal{W}, t)$, and a ciphertext keyword **ck**, this algorithm returns the keyword ciphertext CT to the cloud server.
- TD $\leftarrow$ **Trapdoor**$(pp, pk_S, sk_R, (\mathcal{S}, \xi), \mathbf{tk})$: Given a public parameter pp, a data sender's public key pk_S, a data receiver's secret key sk_R, an attribute set $\mathcal{S}$ corresponding to the data receiver ξ, and a trapdoor keyword **tk**, this algorithm returns the trapdoor TD to the cloud server.
- 1 or 0 $\leftarrow$ **Test**$(pp, (\mathcal{S}, \xi), (\mathcal{W}, t)$, CT, TD$)$: Given a public parameter pp, an attribute set $\mathcal{S}$ corresponding to the data receiver ξ, an access policy $(\mathcal{W}, t)$, a keyword ciphertext CT, and a trapdoor TD, this algorithm returns 1 or 0 to represent the ciphertext and trapdoor whether correspond to the same keyword.
- CT$'$ $\leftarrow$ **CTUpdate**$(pp, \mathcal{R}', \text{CT})$: Given a public parameter pp, an updated revocation list $\mathcal{R}'$, and a keyword ciphertext CT, this algorithms returns an updated keyword ciphertext CT$'$ to the cloud server.

4.4 Security Models

4.4.1 IND-CKA security

We define the IND-CKA security model $\mathbf{Exp}^{\text{IND-CKA}}_{\text{RABAEKS},\mathcal{A}}(\lambda)$ as below.

1) **Setup**: Giving a security parameter λ as input, the challenger $\mathcal{C}$ invokes the **Setup**(λ) algorithm to calculate a public parameter pp and a master secret key msk.
2) **Phase 1**: In polynomial times, $\mathcal{A}$ perform the following queries in an adaptive manner.
   a) **Secret Key Queries** $\mathcal{O}_{RK}$: $\mathcal{A}$ submits (pp, S, msk) to $\mathcal{C}$, and then $\mathcal{C}$ invokes **KeyGen**$_R$ algorithm to compute a secret key sk_R. Ultimately, $\mathcal{C}$ returns sk_R to $\mathcal{A}$.
   b) **Ciphertext Queries** $\mathcal{O}_{CT}$: $\mathcal{A}$ delivers $((\mathcal{W}, t), \mathbf{ck})$ to $\mathcal{C}$, and then $\mathcal{C}$ executes **RABAEKS**$(pp, pk_S, sk_S, (\mathcal{W}, t), \mathbf{ck})$ algorithm to compute a ciphertext CT associated to **ck**. Eventually, $\mathcal{C}$ returns CT to $\mathcal{A}$.
   c) **Trapdoor Queries** $\mathcal{O}_{TD}$: $\mathcal{A}$ sends $((\mathcal{S}, \xi), \mathbf{tk})$ to $\mathcal{C}$, and then $\mathcal{C}$ executes **Trapdoor**$(pp, pk_S, sk_R, (\mathcal{S}, \xi), \mathbf{tk})$ algorithm to compute a trapdoor TD associated to **tk**. Finally, $\mathcal{C}$ returns TD to $\mathcal{A}$.
3) **Challenge**: $\mathcal{A}$ chooses $\mathbf{ck}_0^*, \mathbf{ck}_1^*$ which have not been queried in **Phase 1** as two challenge ciphertext keywords, and sends them to $\mathcal{C}$. After that, $\mathcal{C}$ selects a random bit $\mu \in \{0, 1\}$ and invokes the

$\mathbf{RABAEKS}(pp, pk_S, sk_S, (\mathcal{W}, t), \mathbf{ck}^*_\mu)$ algorithm to obtain a challenge ciphertext CT^*_μ. Ultimately, $\mathcal{C}$ returns CT^*_μ to $\mathcal{A}$.

4) **Phase 2**: $\mathcal{A}$ continues to access $\mathcal{O}_{RK}$, $\mathcal{O}_{CT}$, and $\mathcal{O}_{TD}$ oracles, but subject to the restriction with neither $\mathbf{ck}^*_0$ nor $\mathbf{ck}^*_1$ can be queried.
5) **Guess**: $\mathcal{A}$ guesses a bit $\mu' \in \{0,1\}$. If $\mu' = \mu$, we say that $\mathcal{A}$ wins this game.

We hereby define the advantage of $\mathcal{A}$ to win the above game $\mathbf{Exp}^{\text{IND-CKA}}_{\text{RABAEKS},\mathcal{A}}(\lambda)$ as:

$$\mathrm{Adv}^{\text{IND-CKA}}_{\text{RABAEKS},\mathcal{A}}(\lambda) = |\Pr[\mu' = \mu] - \frac{1}{2}|.$$

Definition 7. Our RABAEKS primitive satisfies IND-CKA security, if any PPT adversary wins the above game $\mathbf{Exp}^{\text{IND-CKA}}_{\text{RABAEKS},\mathcal{A}}(\lambda)$ with a negligible advantage.

4.4.2 IND-KGA security

We define the IND-KGA security model $\mathbf{Exp}^{\text{IND-KGA}}_{\text{RABAEKS},\mathcal{A}}(\lambda)$ as below.

1) **Setup**: Giving a security parameter λ as input, the challenger $\mathcal{C}$ invokes the $\mathbf{Setup}(\lambda)$ algorithm to calculate a public parameter pp and a master secret key msk.
2) **Phase 1**: In polynomial times, $\mathcal{A}$ perform the following queries in an adaptive manner.
   a) **Secret Key Queries** $\mathcal{O}_{RK}$: The query is same as the corresponding query in $\mathbf{Exp}^{\text{IND-CKA}}_{\text{RABAEKS},\mathcal{A}}(\lambda)$.
   b) **Ciphertext Queries** $\mathcal{O}_{CT}$: The query is same as the corresponding query in $\mathbf{Exp}^{\text{IND-CKA}}_{\text{RABAEKS},\mathcal{A}}(\lambda)$.
   c) **Trapdoor Queries** $\mathcal{O}_{TD}$: The query is same as the corresponding query in $\mathbf{Exp}^{\text{IND-CKA}}_{\text{RABAEKS},\mathcal{A}}(\lambda)$.
3) **Challenge**: $\mathcal{A}$ chooses $\mathbf{tk}^*_0, \mathbf{tk}^*_1$ which have not been queried in **Phase 1** as two challenge trapdoor keywords, and sends them to $\mathcal{C}$. After that, $\mathcal{C}$ selects a random bit $\mu \in \{0,1\}$ and invokes the $\mathbf{Trapdoor}(pp, pk_S, sk_R, (\mathcal{S}, \xi), \mathbf{tk}^*_\mu)$ algorithm to obtain a challenge trapdoor TD^*_μ. Ultimately, $\mathcal{C}$ returns TD^*_μ to $\mathcal{A}$.
4) **Phase 2**: $\mathcal{A}$ continues to access $\mathcal{O}_{RK}$, $\mathcal{O}_{CT}$, and $\mathcal{O}_{TD}$ oracles, but subject to the restriction with neither $\mathbf{tk}^*_0$ nor $\mathbf{tk}^*_1$ can be queried.
5) **Guess**: $\mathcal{A}$ guesses a bit $\mu' \in \{0,1\}$. If $\mu' = \mu$, we say that $\mathcal{A}$ wins this game.

We hereby define the advantage of $\mathcal{A}$ to win the above game $\mathbf{Exp}^{\text{IND-KGA}}_{\text{RABAEKS},\mathcal{A}}(\lambda)$ as:

$$\mathrm{Adv}^{\text{IND-KGA}}_{\text{RABAEKS},\mathcal{A}}(\lambda) = |\Pr[\mu' = \mu] - \frac{1}{2}|.$$

Definition 8. Our RABAEKS primitive satisfies IND-KGA security, if any PPT adversary wins the above game $\mathbf{Exp}^{\text{IND-KGA}}_{\text{RABAEKS},\mathcal{A}}(\lambda)$ with a negligible advantage.

5 Our Proposed RABAEKS Scheme

In this section, we propose the overview of our design. After that, we present the concrete construction of RABAEKS scheme, and analyze its correctness.

5.1 Design Rationale

In traditional PAEKS schemes, keyword ciphertexts are often stored in cloud servers, and data receivers submit a trapdoor to search for their interested keywords. Unfortunately, they can only support single-receiver model, which loses scalability for cloud storage.

In 2023, Luo et al. [17] incorporated the idea of ABE into PAEKS to construct a lattice-based ABAEKS scheme, thereby support the multi-receiver model. Nevertheless, they did not take the revocation function into consideration, which makes it impractical for the real-world applications.

To bridge the gap, a straightforward method is to combine ABAEKS with a data receiver revocation method [24], but this approach presents two challenges to be considered. On the one hand, in [17], the attribute is embedded in the keyword ciphertext (aka. key-policy), which cannot provide attributes to the data receivers. On the other hand, it is necessary to find a correct basis for lattice $\Lambda^\perp_q(\mathbf{A}_0 \mid (\mathbf{A} + \sum_{i=1}^{|kw|} tk_i \mathbf{W}_i) \mid \mathbf{M}' \mid \mathbf{A}_i \mid \mathbf{B}_{w_\tau})$ or $\Lambda^\perp_q((\mathbf{A} + \sum_{i=1}^{|kw|} tk_i \mathbf{W}_i) \mid \mathbf{M}' \mid \mathbf{A}_i \mid \mathbf{B}_{w_\tau})$, but we can only obtain a basis $\mathbf{T}_{(\mathbf{A}_0|\mathbf{M}')}$ for lattice $\Lambda^\perp_q(\mathbf{A}_0 \mid \mathbf{M}')$ during key generation. As a result, there is no suitable lattice basis available for the lattice basis sampling algorithm.

To solve the aforementioned problems, we initially construct a ciphertext-policy solution by embedding the attribute into the secret key of data receivers and the access policy into the keyword ciphertext, thereby addressed the first challenge. In addition, given a short basis for $\Lambda^\perp_q(\mathbf{A}')$, we innovatively invoke $\mathsf{GenSamplePre}$ algorithm to sample a vector, where $\mathbf{A}'$ is the horizontal concatenation of any number of matrices in $(\mathbf{A}_0 \mid (\mathbf{A} + \sum_{i=1}^{|kw|} tk_i \mathbf{W}_i) \mid \mathbf{M}' \mid \mathbf{A}_i \mid \mathbf{B}_{w_\tau})$, e.g. $(\mathbf{A}_0 \mid \mathbf{M}')$. In this way, we can utilize the secret key $\mathbf{T}_{(\mathbf{A}_0|\mathbf{M}')}$ to be a suitable basis, thereby addressed the second and third challenges. Consequently, we can construct a lattice-based RABAEKS scheme that supports dynamic access control in a multi-receiver model, offering a more practical searchable encryption scheme for cloud storage.

5.2 Our Concrete Construction

The design of our RABAEKS includes six phases: *System Initialization*, *Key Generation*, *Ciphertext Generation*, *Trapdoor Generation*, *Search Phase*, and *Ciphertext Update*.

5.2.1 System Initialization

The KGC sets up this system by calling the $\mathbf{Setup}(\lambda)$ algorithm. A security parameter λ is inputted, and this algorithm outputs the public parameter pp and master key msk according to these following procedures.

1) Set the system parameters $n, m, q, \sigma, L, |att|, l, |kw|, \mathcal{ATT}, \mathcal{U}, \mathcal{R}$.
2) Calculate an attribute set $\mathcal{ATT} = \{1, \cdots, |att|\}$.
3) Invoke $(\mathbf{A}_0, \mathbf{T}_{\mathbf{A}_0}) \leftarrow \mathsf{TrapGen}(n, m, q)$ to obtain an uniformly matrix $\mathbf{A}_0 \in \mathbb{Z}_q^{n \times m}$ and a basis $\mathbf{T}_{\mathbf{A}_0} \in \mathbb{Z}^{m \times m}$ for $\Lambda^\perp_q(\mathbf{A}_0)$.
4) Select two random matrices $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}, \overline{\mathbf{A}} \xleftarrow{\$} \mathbb{Z}_q^{n \times n}$.
5) For $i \in \mathcal{ATT}$, select a random matrix $\mathbf{A}_i \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$.
6) For $i \in [|kw|]$, select a random matrix $\mathbf{W}_i \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$.

7) Select a random matrix $\mathbf{M} := \{\mathbf{M}_1 | \cdots | \mathbf{M}_{|att|}\} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$, where $\{\mathbf{M}_i\}_{i\in[|att|]} \in \mathbb{Z}_q^{n\times\eta}$, and $m = \eta|att|$.
8) Construct a binary tree $\mathcal{T}$ and many matrices $\mathbf{B}_0 \in \mathbb{Z}_q^{n\times m}$, $\{\mathbf{U}_i\}_{i\in[1,|\mathcal{U}|-2]}$, $\{\mathbf{V}_i\}_{i\in[|\mathcal{U}|-1,2|\mathcal{U}|-2]}$ defined in ***Definition*** 5.
9) Return the public parameter $pp := (n, m, q, \sigma, L, |att|, l, |kw|, \mathbf{A}_0, \mathbf{A}, \{\mathbf{A}_i\}_{i\in\mathcal{ATT}}, \{\mathbf{W}_i\}_{i\in[|kw|]}, \mathbf{M}, \mathcal{ATT}, \mathcal{U}, \mathcal{R}, \mathbf{B}_0, \{\mathbf{U}_i\}_{i\in[1,|\mathcal{U}|-2]}, \{\mathbf{V}_i\}_{i\in[|\mathcal{U}|-1,2|\mathcal{U}|-2]}, \mathcal{T})$, and the master key $msk := \mathbf{T}_{\mathbf{A}_0}$.

5.2.2 Key Generation

The KGC calculates the public and secret key of the data sender and the secret key of data receiver by calling the $\mathbf{KeyGen}_S(pp)$ and $\mathbf{KeyGen}_R(pp, msk)$ algorithms, respectively.

For the $\mathbf{KeyGen}_S(pp)$ algorithm, a public parameter pp is inputted, and this algorithm outputs the public key pk_S and secret key sk_S with the data sender according to these following procedures.

1) Invoke $(\mathbf{A}_S, \mathbf{T}_{\mathbf{A}_S}) \leftarrow \mathsf{TrapGen}(n, m, q)$ to obtain an uniformly matrix $\mathbf{A}_S \in \mathbb{Z}_q^{n\times m}$ and a basis $\mathbf{T}_{\mathbf{A}_S} \in \mathbb{Z}^{m\times m}$ for $\Lambda_q^{\perp}(\mathbf{A}_S)$.
2) Return the public key $pk_S := \mathbf{A}_S$ and the secret key $sk_S := \mathbf{T}_{\mathbf{A}_S}$ for the data sender.

For the $\mathbf{KeyGen}_R(pp, S, msk)$ algorithm, a public parameter pp, a receiver attribute set $\mathcal{S} \subseteq \mathcal{ATT}$ and a master key msk are inputted, and this algorithm outputs the secret key sk_R with the data receiver according to these following procedures.

1) Parse the matrix $\mathbf{M} = (\mathbf{M}_1 \mid \cdots \mid \mathbf{M}_{|att|})$. For $i \in [|att|]$, if $i \in \mathcal{S}$, set $\mathbf{M}'_i = \mathbf{M}_i$. Otherwise, set $\mathbf{M}'_i = \mathbf{0}$. Let a matrix $\mathbf{M}' = (\mathbf{M}'_1 \mid \cdots \mid \mathbf{M}'_{|att|}) \in \mathbb{Z}_q^{n\times m}$.
2) Invoke $\mathbf{T}_{(\mathbf{A}_0|\mathbf{M}')} \leftarrow \mathsf{SampleBasis}(\mathbf{A}_0 \mid \mathbf{M}', \mathbf{T}_{\mathbf{A}_0}, \{1\}, L)$ to obtain a basis $\mathbf{T}_{(\mathbf{A}_0|\mathbf{M}')} \in \mathbb{Z}^{2m\times 2m}$ for $\Lambda_q^{\perp}(\mathbf{A}_0 \mid \mathbf{M}')$.
3) Return the secret key $sk_R := \mathbf{T}_{(\mathbf{A}_0|\mathbf{M}')}$ to the data receiver.

5.2.3 Ciphertext Generation

The data sender calculates a keyword ciphertext by calling the $\mathbf{RABAEKS}(pp, pk_S, sk_S, (\mathcal{W}, t), \mathbf{ck})$ algorithm. A public parameter pp, the public and secret keys (pk_S, sk_S) with the data sender, an access policy $(\mathcal{W}, t)$ where $\mathcal{W} \subset \mathcal{ATT}$, and a keyword $\mathbf{ck} \in \{0,1\}^{|kw|}$ are inputted, and this algorithm outputs a keyword ciphertext CT according to these following procedures.

1) Parse the keyword $\mathbf{ck} = (ck_1, \cdots, ck_{|kw|}) \in \{0,1\}^{|kw|}$.
2) Select a random vector $\mathbf{s} \xleftarrow{\$} \chi^n$, and many noise vectors $\mathbf{x}_0 \xleftarrow{\$} \chi^m$, $\mathbf{x}_1 \xleftarrow{\$} \chi^m$, $\overline{\mathbf{x}} \xleftarrow{\$} \chi^n$, and $\mathbf{x}_\mathbf{M} \xleftarrow{\$} \chi^m$.
3) Calculate three vectors $\mathbf{c}_0 = \mathbf{A}_0^\top\mathbf{s}+\mathbf{x}_0 \in \mathbb{Z}_q^m$, $\mathbf{c}_1 = (\mathbf{A}+\sum_{i=1}^{|kw|} ck_i\mathbf{W}_i)^\top\mathbf{s} + \mathbf{x}_1 \in \mathbb{Z}_q^m$, and $\overline{\mathbf{c}} = \overline{\mathbf{A}}^\top\mathbf{s} + \overline{\mathbf{x}} \in \mathbb{Z}_q^n$.
4) Invoke $\mathbf{c}_2 \leftarrow \mathsf{SampleLeft}(\mathbf{A}_S, \mathbf{A} + \sum_{i=1}^{|kw|} ck_i\mathbf{W}_i, \mathbf{T}_{\mathbf{A}_S}, \overline{\mathbf{c}}, \sigma)$ to obtain a vector $\mathbf{c}_2 \in \mathbb{Z}_q^{2m}$ statistically distributed in $\mathcal{D}^{2m}_{\Lambda_q^{\overline{\mathbf{c}}}(\mathbf{A}_S|\mathbf{A}+\sum_{i=1}^{|kw|} ck_i\mathbf{W}_i)}$, s.t. $(\mathbf{A}_S \mid \mathbf{A} + \sum_{i=1}^{|kw|} ck_i\mathbf{W}_i)\mathbf{c}_2 = \overline{\mathbf{c}} \bmod q$.
5) For $i \in \mathcal{W}$, select a noise vector $\mathbf{x}_i \xleftarrow{\$} \chi^m$, and calculate a vector $\mathbf{c}_i = \mathbf{A}_i^\top\mathbf{s} + \mathbf{x}_i \in \mathbb{Z}_q^m$.
6) Calculate a vector $\mathbf{c}_\mathbf{M} = \mathbf{M}^\top\mathbf{s} + \mathbf{x}_\mathbf{M} \in \mathbb{Z}_q^m$.
7) For $k \in \mathsf{KUNodes}(\mathcal{T}, \mathcal{R})$, select a noise vector $\mathbf{x}_k \xleftarrow{\$} \chi^m$, and calculate a vector $\mathbf{c}_k = \mathbf{B}_k^\top\mathbf{s} + \mathbf{x}_k \in \mathbb{Z}_q^m$.
8) Return the ciphertext $\mathrm{CT} := (\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2, \{\mathbf{c}_i\}_{i\in\mathcal{W}}, \mathbf{c}_\mathbf{M}, \{\mathbf{c}_k\}_{k\in\mathsf{KUNodes}(\mathcal{T},\mathcal{R})})$ corresponding to the keyword $\mathbf{ck}$.

5.2.4 Trapdoor Generation

The data receiver $\xi \in \mathcal{U}$ calculates a search trapdoor by calling the $\mathbf{Trapdoor}(pp, pk_S, sk_R, (\mathcal{S}, \xi), \mathbf{tk})$ algorithm. A public parameter pp, the public key pk_S with the data sender, the secret keys sk_R and the attribute set $\mathcal{S}$ with the data receiver ξ, and a keyword $\mathbf{tk} \in \{0,1\}^{|kw|}$ are inputted, and this algorithm outputs a search trapdoor TD according to these following procedures.

1) Parse the keyword $\mathbf{tk} = (tk_1, \cdots, tk_{|kw|}) \in \{0,1\}^{|kw|}$.
2) Select some random matrices $\mathbf{R}' \xleftarrow{\$} \{-1,1\}^{m\times m}$, a random vector $\mathbf{s}' \xleftarrow{\$} \chi^n$, and two noise vectors $\mathbf{x}'_0 \xleftarrow{\$} \chi^m$, and $\overline{\mathbf{x}}' \xleftarrow{\$} \chi^n$.
3) Calculate two vectors $\mathbf{t}_0 = (\mathbf{A}_S \mid \mathbf{A} + \sum_{i=1}^{|kw|} tk_i\mathbf{W}_i)^\top\mathbf{s}' + \begin{pmatrix}\mathbf{x}'_0\\ \mathbf{R}'^\top\mathbf{x}'_0\end{pmatrix} \in \mathbb{Z}_q^{2m}$, and $\overline{\mathbf{t}} = \overline{\mathbf{A}}\mathbf{s}' + \overline{\mathbf{x}}' := (\overline{t}_1, \cdots, \overline{t}_n) \in \mathbb{Z}_q^n$, where $\mathbf{R}' = \sum_{i=1}^{|kw|} ck_i\mathbf{R}'_i \in \mathbb{Z}^{m\times m}$.
4) Calculate a set $\omega(\xi) = \{w_0, \cdots, w_\tau\}$.
5) For $j \in [n]$, select many polynomials $p_j(x) \in \mathbb{Z}_q[x]$, such that $p_j(0) = \overline{t}_j \in \mathbb{Z}_q$.
6) For $j \in \mathcal{S}$, set the vector $\hat{\mathbf{t}}_j = (p_1(j), \cdots, p_n(j))^\top \in \mathbb{Z}_q^n$, and invoke $\mathbf{t}_j \leftarrow \mathsf{GenSamplePre}(\mathbf{A}_0 \mid (\mathbf{A} + \sum_{i=1}^{|kw|} tk_i\mathbf{W}_i) \mid \mathbf{M}' \mid \mathbf{A}_j \mid \mathbf{B}_{w_\tau}, \mathbf{T}_{(\mathbf{A}_0|\mathbf{M}')}, \{1,3\}, \hat{\mathbf{t}}_j, \sigma)$ to obtain a vector $\mathbf{t}_j \in \mathbb{Z}_q^{5m}$ statistically distributed in $\mathcal{D}^{5m}_{\Lambda_q^{\hat{\mathbf{t}}_j}(\mathbf{A}_0|(\mathbf{A}+\sum_{i=1}^{|kw|} tk_i\mathbf{W}_i)|\mathbf{M}'|\mathbf{A}_j|\mathbf{B}_{w_\tau})}$, s.t. $(\mathbf{A}_0 \mid (\mathbf{A} + \sum_{i=1}^{|kw|} tk_i\mathbf{W}_i) \mid \mathbf{M}' \mid \mathbf{A}_j \mid \mathbf{B}_{w_\tau})\mathbf{t}_j = \hat{\mathbf{t}}_j \bmod q$.
7) Return the trapdoor $\mathrm{TD} := (\mathbf{t}_0, \{\mathbf{t}_j\}_{j\in\mathcal{S}})$ corresponding to the keyword $\mathbf{tk}$.

5.2.5 Search Phase

The cloud server executes the $\mathbf{Test}(pp, (\mathcal{S}, \xi), (\mathcal{W}, t), \mathrm{CT}, \mathrm{TD})$ algorithm to search the matched keyword ciphertext corresponding to the trapdoor. A public parameter pp, the attribute set $\mathcal{S}$ with data receiver ξ, the access policy $(\mathcal{W}, t)$, the keyword ciphertext CT, and the trapdoor TD are inputted, and this algorithm outputs 1 or 0 according to the following judgements.

1) Parse the matrix $\mathbf{c}_\mathbf{M} = \begin{pmatrix}\mathbf{c}_{\mathbf{M},1}\\ \vdots\\ \mathbf{c}_{\mathbf{M},l}\end{pmatrix}$. For $i \in [|att|]$, if $i \in \mathcal{S}$, set $\mathbf{c}_{\mathbf{M}',i} = \mathbf{c}_{\mathbf{M},i}$. Otherwise, set $\mathbf{c}_{\mathbf{M}',i} = 0$. Let a matrix $\mathbf{c}_{\mathbf{M}'} = \begin{pmatrix}\mathbf{c}_{\mathbf{M}',1}\\ \vdots\\ \mathbf{c}_{\mathbf{M}',l}\end{pmatrix} \in \mathbb{Z}_q^m$.
2) If $|\mathcal{S} \cap \mathcal{W}| < t$ or $\mathsf{KUNodes}(\mathcal{T}, \mathcal{R}) \cap \omega(\xi) = \emptyset$, return 0.
3) Otherwise, the set $|\mathcal{S} \cap \mathcal{W}| \geq t$.
   a) Select a set $\mathcal{J} \subset \mathcal{S} \cap \mathcal{W}$, where $|\mathcal{J}| = t$.
   b) Set $\gamma = \mathsf{KUNodes}(\mathcal{T}, \mathcal{R}) \cap \omega(\xi)$:
      - If γ is a non-leaf node, $\mathbf{c}'_\gamma = \mathbf{V}_{\omega_\tau}^\top(\mathbf{U}_\gamma^\top)^{-1}\mathbf{c}_\gamma$.
      - If γ is a leaf node, $\mathbf{c}'_\gamma = \mathbf{c}_{\omega_\tau}$.
   c) Calculate a number $r = \mathbf{c}_2^\top\mathbf{t}_0 - \sum_{j\in\mathcal{J}} L_j(\mathbf{c}_0^\top \mid \mathbf{c}_1^\top \mid \mathbf{c}_{\mathbf{M}'}^\top \mid \mathbf{c}_j^\top \mid \mathbf{c}'^\top_\gamma)\mathbf{t}_j \in \mathbb{Z}_q$, where $L_j = \frac{\prod_{i\in\mathcal{J},i\neq j} -i}{\prod_{i\in\mathcal{J},i\neq j}(j-i)}$.

The correctness analysis of our RABAEKS scheme

To prove the correctness of our RABAEKS scheme, we suppose the public and secret key of data sender $(pk_S, sk_S) = (\mathbf{A}_S, \mathbf{T}_{\mathbf{A}_S})$, the secret key of data receiver $sk_R = \mathbf{T}_{(\mathbf{A}_0|\mathbf{M}')}$, a ciphertext keyword $\mathbf{ck}$ and its ciphertext CT $:= (\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2, \{\mathbf{c}_i\}_{i\in\mathcal{W}}, \mathbf{c}_\mathbf{M},$
$\{\mathbf{c}_k\}_{k\in\mathsf{KUNodes}(\mathcal{T},\mathcal{R})})$, the trapdoor keyword $\mathbf{tk}$ and corresponding search trapdoor TD $:= (\mathbf{t}_0, \{\mathbf{t}_i\}_{i\in\mathcal{S}})$. When $|\mathcal{S}\cap\mathcal{W}| \geq t$ and $\mathbf{ck} = \mathbf{tk}$, we set $\mathcal{J} \subset \mathcal{S}\cap\mathcal{W}$ and $\gamma = \mathsf{KUNodes}(\mathcal{T},\mathcal{R}) \cap \omega(\xi)$, where $|\mathcal{J}| = t$.

If γ is a non-leaf node, $\mathbf{c}'_\gamma = \mathbf{V}_{\omega_\tau}^\top(\mathbf{U}_\gamma^\top)^{-1}\mathbf{c}_\gamma = \mathbf{V}_{\omega_\tau}^\top(\mathbf{U}_\gamma^\top)^{-1}(\mathbf{B}_\gamma^\top\mathbf{s} + \mathbf{x}_\gamma) = \mathbf{V}_{\omega_\tau}^\top(\mathbf{U}_\gamma^\top)^{-1}(\mathbf{U}_\gamma^\top\mathbf{B}_0^\top\mathbf{s} + \mathbf{x}_\gamma) = \mathbf{V}_{\omega_\tau}^\top\mathbf{B}_0^\top\mathbf{s} + \mathbf{V}_{\omega_\tau}^\top(\mathbf{U}_\gamma^\top)^{-1}\mathbf{x}_\gamma = \mathbf{B}_{\omega_\tau}^\top\mathbf{s} + \mathbf{x}'_{\omega_\tau}$, where $\mathbf{x}'_{\omega_\tau} = \mathbf{V}_{\omega_\tau}^\top(\mathbf{U}_\gamma^\top)^{-1}\mathbf{x}_\gamma$. If γ is a leaf node, $\mathbf{c}'_\gamma = \mathbf{c}_{\omega_\tau} = \mathbf{B}_{\omega_\tau}^\top\mathbf{s} + \mathbf{x}'_{\omega_\tau}$, where $\mathbf{x}'_{\omega_\tau} = \mathbf{x}_{\omega_\tau}$.

Based on the aforementioned conditions, we have:

$$\begin{aligned}
r &= \mathbf{c}_2^\top\mathbf{t}_0 - \sum_{j\in\mathcal{J}} L_j(\mathbf{c}_0^\top|\mathbf{c}_1^\top|\mathbf{c}_{\mathbf{M}'}^\top|\mathbf{c}_j^\top|\mathbf{c}'^\top_\gamma)\mathbf{t}_j \\
&= \mathbf{c}_2^\top(\mathbf{A}_S|(\mathbf{A} + \sum_{i=1}^{|kw|} tk_i\mathbf{W}_i))^\top\mathbf{s}' + \mathbf{c}_2^\top\begin{pmatrix}\mathbf{x}'_0\\ \mathbf{R}'^\top\mathbf{x}'_0\end{pmatrix} \\
&\quad - \sum_{j\in\mathcal{J}} L_j((\mathbf{s}^\top\mathbf{A}_0 + \mathbf{x}_0^\top)|(\mathbf{s}^\top(\mathbf{A} + \sum_{i=1}^{|kw|} tk_i\mathbf{W}_i) + \mathbf{x}_0^\top\mathbf{R})|(\mathbf{s}^\top\mathbf{M}' + \mathbf{x}_{\mathbf{M}'}^\top)|(\mathbf{s}^\top\mathbf{A}_j + \mathbf{x}_j^\top)|(\mathbf{s}^\top\mathbf{B}_{\omega_\tau} + \mathbf{x}'^\top_{\omega_\tau}))\mathbf{t}_j \\
&= \overline{\mathbf{c}}^\top\mathbf{s}' - \sum_{j\in\mathcal{J}} L_j\mathbf{s}^\top(\mathbf{A}_0|(\mathbf{A} + \sum_{i=1}^{|kw|} tk_i\mathbf{W}_i)|\mathbf{M}'|\mathbf{A}_j|\mathbf{B}_{\omega_\tau})\mathbf{t}_j + \mathbf{c}_2^\top\begin{pmatrix}\mathbf{x}'_0\\ \mathbf{R}'^\top\mathbf{x}'_0\end{pmatrix} - \sum_{j\in\mathcal{J}} L_j(\mathbf{x}_0^\top|\mathbf{x}_0^\top\mathbf{R}|\mathbf{x}_{\mathbf{M}'}^\top|\mathbf{x}_j^\top|\mathbf{x}'^\top_{\omega_\tau})\mathbf{t}_j \\
&= \overline{\mathbf{c}}^\top\mathbf{s}' - \sum_{j\in\mathcal{J}} L_j\mathbf{s}^\top\hat{\mathbf{t}_j} + \mathbf{x}_L = \overline{\mathbf{c}}^\top\mathbf{s}' - \mathbf{s}^\top\overline{\mathbf{t}} + \mathbf{x}_L = \mathbf{s}^\top\overline{\mathbf{A}}\mathbf{s}' + \overline{\mathbf{x}}^\top\mathbf{s}' - \mathbf{s}^\top\overline{\mathbf{A}}\mathbf{s}' - \mathbf{s}^\top\overline{\mathbf{x}'} + \mathbf{x}_L \\
&= \mathbf{x}_L + \overline{\mathbf{x}}^\top\mathbf{s}' - \mathbf{s}^\top\overline{\mathbf{x}'}, \text{ where } \mathbf{x}_L = \mathbf{c}_2^\top\begin{pmatrix}\mathbf{x}'_0\\ \mathbf{R}'^\top\mathbf{x}'_0\end{pmatrix} - \sum_{j\in\mathcal{J}} L_j(\mathbf{x}_0^\top|\mathbf{x}_0^\top\mathbf{R}|\mathbf{x}_{\mathbf{M}'}^\top|\mathbf{x}_j^\top|\mathbf{x}'^\top_{\omega_\tau})\mathbf{t}_j.
\end{aligned}$$

If $|r| = |\mathbf{x}_L + \overline{\mathbf{x}}^\top\mathbf{s}' - \mathbf{s}^\top\overline{\mathbf{x}'}| < \frac{q}{5}$ [42], the cloud server will return 1, which represents that the ciphertext and trapdoor correspond to the same keyword.

Fig. 4. The correctness analysis of our RABAEKS scheme.

d) If $|r - \lfloor\frac{q}{2}\rfloor| \leq \lfloor\frac{q}{4}\rfloor$, return 1; otherwise, return 0.

5.2.6 Ciphertext Update

The cloud server updates the ciphertext with the change of revocation list by calling the **CTUpdate**$(pp, \mathcal{R}', \mathrm{CT})$ algorithm. A public parameter pp, the updated revocation list $\mathcal{R}'$, and the keyword ciphertext CT are inputted, and this algorithm outputs an updated keyword ciphertext CT$'$ according to these following procedures.

1) For $k' \in \mathsf{KUNodes}(\mathcal{T},\mathcal{R}')$, set ζ is an arbitrary non-leaf node in $\mathcal{R}$.
   a) If k' is a non-leaf node, calculate a vector $\mathbf{c}'_{k'} = \mathbf{U}_{k'}^\top(\mathbf{U}_\zeta^\top)^{-1}\mathbf{c}_\zeta \in \mathbb{Z}_q^n$.
   b) If k' is a leaf node, calculate a vector $\mathbf{c}'_{k'} = \mathbf{V}_{k'}^\top(\mathbf{U}_\zeta^\top)^{-1}\mathbf{c}_\zeta \in \mathbb{Z}_q^n$.
2) Return an updated keyword ciphertext CT$'$ $:= (\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2, \{\mathbf{c}_i\}_{i\in\mathcal{W}}, \mathbf{c}_\mathbf{M}, \{\mathbf{c}'_{k'}\}_{k'\in\mathsf{KUNodes}(\mathcal{T},\mathcal{R}')})$.

5.3 Correctness Analysis and Parameter Setting

In Fig. 4, we provide a correctness analysis of our RABAEKS scheme. Then, to ensure our RABAEKS scheme can execute successfully, the involved parameters are set as follows.

- $|r| = |\mathbf{x}_L + \overline{\mathbf{x}}^\top\mathbf{s}' - \mathbf{s}^\top\overline{\mathbf{x}'}| \leq (5t+1)(q\sigma m\alpha\omega(\sqrt{\log m}) + \mathcal{O}(\sigma m^{\frac{3}{2}})) + 2n(2q^2\alpha^2\omega(\log m) + \mathcal{O}(m)) < \frac{q}{5}$ for a correct RABAEKS.
- $m \geq \lceil 2n\log q\rceil$ for TrapGen algorithm.
- $L \geq \mathcal{O}(m^{\frac{3}{2}}) \cdot \omega(\log 2m)$ for SampleBasis algorithm.
- $\sigma \geq 5m \cdot \omega(\log 5m)$ for GenSamplePre algorithm.
- $\alpha q > 2\sqrt{n}$ for LWE hardness.

6 Security Analysis

In this section, we rigorously prove the security of our scheme, which covers IND-CKA and IND-KGA security.

Theorem 1. Assume that the $\mathrm{LWE}_{n,m,q,\chi}$ hardness holds, our proposed lattice-based RABAEKS scheme satisfies IND-CKA security in the ROM. For any PPT adversary $\mathcal{A}$, if $\mathcal{A}$ can compromise our scheme with a non-negligible advantage ϵ_1, then a PPT challenger $\mathcal{C}$ can be constructed to solve the $\mathrm{LWE}_{n,m,q,\chi}$ hardness with a non-negligible probability.

Proof Assume that there exists a PPT adversary $\mathcal{A}$ that has ability to break the IND-CKA secure with a non-negliable advantage ϵ_1, we can construct a challenger $\mathcal{C}$ that solves the decisional $\mathrm{LWE}_{n,m,q,\chi}$ hardness with the same advantage. In the Initialization phase, $\mathcal{A}$ sends a challenge access structure $(\mathcal{W}^*, t^*)$ and a revocation list $\mathcal{R}^*$ to $\mathcal{C}$, where $t^* \in [1, |att|)$. Let $\mathcal{ATT} = \{1, \cdots, |att|\}$, such that $\mathcal{W}^* \subset \mathcal{ATT}$, the interaction between $\mathcal{A}$ and $\mathcal{C}$ is showed as follows.

Setup: The challenger $\mathcal{C}$ samples $\{(\mathbf{A}_j, \mathbf{b}_j)\}_{j\in[0,|att|+1]}$, $\{(\mathbf{W}_j, \mathbf{u}_j)\}_{j\in[|kw|]}$, $\{(\mathbf{B}_j, \mathbf{v}_j)\}_{j\in[1,2|\mathcal{U}|-2]}$ and $\{(\mathbf{a}_i, b_i)\}_{i\in[m]}$ from $\mathcal{O}$. Then, $\mathcal{C}$ selects a random matrix $\overline{\mathbf{A}} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$, sets the matrix $\mathbf{A} = \mathbf{A}_{|att|+1}$, $\{\mathbf{A}_i\}_{i\in\mathcal{ATT}} = \{\mathbf{A}_j\}_{j\in[1,|att|+1]}$, $\mathbf{M} = (\mathbf{a}_1 \mid \cdots \mid$

$\mathbf{a}_m) \in \mathbb{Z}_q^{n\times m}$, and sends the public parameter $pp = (n, m, q, \sigma, L, |att|, l, |kw|, \mathbf{A}_0, \mathbf{A}, \{\mathbf{A}_i\}_{i\in\mathcal{ATT}}, \{\mathbf{W}_i\}_{i\in[|kw|]}, \mathbf{M}, \mathcal{ATT}, \mathcal{U}, \mathcal{R}, \mathbf{B}_0, \{\mathbf{U}_i\}_{i\in[1,|\mathcal{U}|-2]}, \{\mathbf{V}_i\}_{i\in[|\mathcal{U}|-1,2|\mathcal{U}|-2]}, \mathcal{T})$ to $\mathcal{A}$.

Phase 1: The adversary $\mathcal{A}$ executes these following queries adaptively.

- **Secret Key Queries** $\mathcal{O}_{RK}$: If $\mathcal{S}$ meets the access policy $(\mathcal{W}^*, t^*)$, $\mathcal{C}$ returns $\perp$ to $\mathcal{A}$. Otherwise, $\mathcal{C}$ calculates $\mathbf{M}' = (\mathbf{M}_1 \mid \cdots \mid \mathbf{M}_{|att|})$, where $\mathbf{M}'_i = \mathbf{M}_i$ if $i \in \mathcal{S}$ and $\mathbf{M}'_i = \mathbf{0}$ if $i \notin \mathcal{S}$, for $i \in [|att|]$. Then, $\mathcal{C}$ invokes $\mathbf{T}_{(\mathbf{A}_0|\mathbf{M}')} \leftarrow \mathsf{SampleBasis}(\mathbf{A}_0 \mid \mathbf{M}', \mathbf{T}_{\mathbf{A}_0}, \{1\}, L)$ to obtain a basis $\mathbf{T}_{(\mathbf{A}_0|\mathbf{M}')} \in \mathbb{Z}^{2m\times 2m}$ for $\Lambda_q^{\perp}(\mathbf{A}_0 \mid \mathbf{M}')$. Finally, $\mathcal{C}$ returns the secret key with data receiver $sk_R = \mathbf{T}_{(\mathbf{A}|\mathbf{M}')}$ to $\mathcal{A}$.
- **Ciphertext Queries** $\mathcal{O}_{CT}$: After $\mathcal{A}$ inputs $((\mathcal{W}, t), \mathbf{ck})$, $\mathcal{C}$ executes **RABAEKS**$(pp, pk_S, sk_S, (\mathcal{W}, t), \mathbf{ck})$ to generate the ciphertext CT with the keyword $\mathbf{ck}$, and then returns it to $\mathcal{A}$.
- **Trapdoor Queries** $\mathcal{O}_{TD}$: After $\mathcal{A}$ inputs $((\mathcal{S}, \xi), \mathbf{tk})$, $\mathcal{C}$ executes the following procedures according to different conditions.
  1) If $\mathcal{S}$ does not meet $(\mathcal{W}^*, t^*)$ and $\xi \in \mathcal{R}^*$, where $|\mathcal{S}\cap\mathcal{W}^*| < t^*$. $\mathcal{C}$ calculates a secret key sk_R with data receiver by executing $\mathcal{O}_{RK}$, and calculates $\mathbf{t}_0$ as same as the process in **Trapdoor**$(pp, pk_S, sk_R, (\mathcal{S}, \xi), \mathbf{tk})$. Then, $\mathcal{C}$ selects a subset $\mathcal{J} \subseteq \mathcal{S}$ such that $|\mathcal{J}| = t$. For $i \in \mathcal{J}$, $\mathcal{C}$ samples a vector $\mathbf{t}_i \xleftarrow{\$} \mathcal{D}_{\mathbb{Z}_q^{5m},\sigma}$, and calculates $\hat{\mathbf{t}}_i = (\mathbf{A}_0 \mid (\mathbf{A} + \sum_{i=1}^{|kw|} tk_i\mathbf{W}_i) \mid \mathbf{M}' \mid \mathbf{A}_i \mid \mathbf{B}_{w_\tau})\mathbf{t}_i \bmod q$. Consequently, the pairing $\{\bar{\mathbf{t}}, \{\hat{\mathbf{t}}_i\}\}$ can be obtained. Through the Lagrange interpolation, the polynomials $p_1(x), \cdots, p_n(x)$ can be recovered. For $j \in \mathcal{S}$, $\hat{\mathbf{t}}_j = (p_1(j), \cdots, p_n(j))^\top$, $\mathcal{C}$ invokes $\mathbf{t}_j \leftarrow \mathsf{GenSamplePre}(\mathbf{A}_0 \mid (\mathbf{A} + \sum_{i=1}^{|kw|} tk_i\mathbf{W}_i) \mid \mathbf{M}' \mid \mathbf{A}_j \mid \mathbf{B}_{w_\tau}, \mathbf{T}_{(\mathbf{A}_0|\mathbf{M}')}, \{1,3\}, \hat{\mathbf{t}}_i, \sigma)$ to obtain a vector $\mathbf{t}_j \in \mathbb{Z}_q^{5m}$. Finally, $\mathcal{C}$ returns the trapdoor $\mathrm{TD} = (\mathbf{t}_0, \{\mathbf{t}_i\}_{i\in\mathcal{S}})$ to $\mathcal{A}$.
  2) If $\mathcal{S}$ meets $(\mathcal{W}^*, t^*)$ and $\xi \in \mathcal{R}^*$, $\mathcal{C}$ calculates a vector $\mathbf{t}_0$ firstly. Then, $\mathcal{C}$ calculates $\{\mathbf{t}_i\}_{i\in\mathcal{S}}$ as same as 1), and returns $\mathrm{TD} = (\mathbf{t}_0, \{\mathbf{t}_i\}_{i\in\mathcal{S}})$ to $\mathcal{A}$.
  3) If $\mathcal{S}$ does not meet $(\mathcal{W}^*, t^*)$ and $\xi \notin \mathcal{R}^*$, $\mathcal{C}$ calculates a vector $\mathbf{t}_0$ firstly, and then invokes $\mathbf{t}_i \leftarrow \mathsf{GenSamplePre}(\mathbf{A}_0 \mid (\mathbf{A} + \sum_{i=1}^{|kw|} tk_i\mathbf{W}_i) \mid \mathbf{M}' \mid \mathbf{A}_i \mid \mathbf{B}_{w_\tau}, \mathbf{T}_{(\mathbf{A}_0|\mathbf{M}')}, \{1,3\}, \hat{\mathbf{t}}_i, \sigma)$ to obtain a vector $\mathbf{t}_i \in \mathbb{Z}_q^{5m}$, where $\hat{\mathbf{t}}_i = (p_1(i), \cdots, p_n(i))^\top \in \mathbb{Z}_q^n$ and $p_1(x), \cdots, p_n(x) \in \mathbb{Z}_q[x]$ such that $(p_1(0), \cdots, p_n(0))^\top = \bar{\mathbf{t}}$ for $i \in \mathcal{S}$. Then, $\mathcal{C}$ returns $\mathrm{TD} = (\mathbf{t}_0, \{\mathbf{t}_i\}_{i\in\mathcal{S}})$ to $\mathcal{A}$.
  4) If $\mathcal{S}$ meets $(\mathcal{W}^*, t^*)$ and $\xi \notin \mathcal{R}^*$, $\mathcal{C}$ returns $\perp$ to $\mathcal{A}$.

Challenge: $\mathcal{A}$ selects two keywords $\mathbf{ck}_0^*$, $\mathbf{ck}_1^*$ which have not been queried in **Phase 1**, and sends them to $\mathcal{C}$. Then, $\mathcal{C}$ selects a bit $\mu \in \{0,1\}$ randomly, and encrypts $\mathbf{ck}_\mu^*$ with the access policy $(\mathcal{W}^*, t^*)$. Specially, $\mathcal{C}$ calculates $\mathbf{c}_0^* = \mathbf{b}_0$, $\mathbf{c}_1^* = \mathbf{b}_{|att|+1} + \sum_{i=1}^{|kw|} ck_i^*\mathbf{u}_i$, and samples a random vector $\mathbf{c}_2^* \xleftarrow{\$} \mathcal{D}_{\mathbb{Z}_q^{2m},\sigma}$. For $i \in \mathcal{W}^*$, $\mathcal{C}$ calculates $\mathbf{c}_i^* = \mathbf{b}_i$. For $k \in \mathsf{KUNodes}(\mathcal{T}, \mathcal{R}^*)$, $\mathbf{c}_k^* = \mathbf{v}_j$, where $j \in [1, 2|\mathcal{U}|-2]$. After that, $\mathcal{C}$ calculates $\mathbf{c}_{\mathbf{M}}^* = (b_1, \cdots, b_m)^\top$. Finally, $\mathcal{C}$ returns $\mathrm{CT}_\mu^* = (\mathbf{c}_0^*, \mathbf{c}_1^*, \mathbf{c}_2^*, \{\mathbf{c}_i^*\}_{i\in\mathcal{W}^*}, \mathbf{c}_{\mathbf{M}}^*, \{\mathbf{c}_k^*\}_{k\in\mathsf{KUNodes}(\mathcal{T},\mathcal{R}^*)})$ to $\mathcal{A}$.

Phase 2: The adversary $\mathcal{A}$ continues to query from $\mathcal{O}_{RK}$, $\mathcal{O}_{CT}$, and $\mathcal{O}_{TD}$ as same as in **Phase 1** except that the $((\mathcal{S}, \xi), \mathbf{tk})$ could be queried in $\mathcal{O}_{TD}$ when $\mathcal{S}$ meets $(\mathcal{W}^*, t^*)$ and $\mathbf{tk} \notin (\mathbf{ck}_0^*, \mathbf{ck}_1^*)$.

Guess: The adversary $\mathcal{A}$ outputs $\mu' \in \{0,1\}$. If $\mu' = \mu$, $\mathcal{A}$ wins this game and $\mathcal{C}$ outputs $\mathcal{O} = \mathcal{O}_{LWE}$. Otherwise, $\mathcal{C}$ outputs $\mathcal{O} = \mathcal{O}'_{LWE}$.

Analysis: If $\mu' = \mu$, $\mathcal{C}$ outputs $\mathcal{O} = \mathcal{O}_{LWE}$ meaning that $\{(\mathbf{A}_j, \mathbf{b}_j)\}_{j\in[0,|att|+1]}$, $\{(\mathbf{W}_j, \mathbf{u}_j)\}_{j\in[|kw|]}$, $\{(\mathbf{B}_j, \mathbf{v}_j)\}_{j\in[1,2|\mathcal{U}|-2]}$ and $\{(\mathbf{a}_i, b_i)\}_{i\in[m]}$ are sampled from $\mathcal{O}_{LWE}$ such that $\mathbf{b}_j = \mathbf{A}_j^\top\mathbf{s} + \mathbf{x}_j$, $\mathbf{u}_j = \mathbf{W}_j^\top\mathbf{s} + \mathbf{x}_j$, $\mathbf{v}_j = \mathbf{B}_j^\top\mathbf{s} + \mathbf{x}_j$, and $b_i = \mathbf{a}_i^\top\mathbf{s} + x_i$, where $\mathbf{s} \in \chi^n$ is a secret vector and $\mathbf{x}_j \leftarrow \chi^m$, $x_i \leftarrow \chi$ are noise vectors. Then, we can obtain the following equations:

$$\begin{aligned}
\mathbf{c}_0^* &= \mathbf{b}_0 = \mathbf{A}_0^\top\mathbf{s} + \mathbf{x}_0\\
\mathbf{c}_1^* &= \mathbf{b}_{|att|+1} + \sum_{i=1}^{|kw|} ck_i^*\mathbf{u}_i\\
&= (\mathbf{A}_{|att|+1} + \sum_{i=1}^{|kw|} ck_i^*\mathbf{W}_i)^\top\mathbf{s} + (\mathbf{x}_{|att|+1} + \sum_{i=1}^{|kw|} ck_i^*\mathbf{x}_i)\\
\mathbf{c}_2^* &\in \mathcal{D}_{\mathbb{Z}^{2m},\sigma}\\
\mathbf{c}_i^* &= \mathbf{b}_i = \mathbf{A}_i^\top\mathbf{s} + \mathbf{x}_i, i \in \mathcal{W}^*\\
\mathbf{c}_k^* &= \mathbf{v}_j = \mathbf{B}_j^\top\mathbf{s} + \mathbf{x}_j, k \in \mathsf{KUNodes}(\mathcal{T}, \mathcal{R}^*), j \in [1, 2|\mathcal{U}| - 2]\\
\mathbf{c}_{\mathbf{M}}^* &= (b_1, \cdots, b_m)^\top = \mathbf{M}^\top\mathbf{s} + (\mathbf{x}_1^\top \mid \cdots \mid \mathbf{x}_m^\top)^\top
\end{aligned}$$

Consequently, $\mathrm{CT}_\mu^* = (\mathbf{c}_0^*, \mathbf{c}_1^*, \mathbf{c}_2^*, \{\mathbf{c}_k^*\}_{k\in\mathsf{KUNodes}(\mathcal{T},\mathcal{R}^*)}, \{\mathbf{c}_i^*\}_{i\in\mathcal{W}^*}, \mathbf{c}_{\mathbf{M}}^*)$ is a valid ciphertext, and assume that $\mathcal{A}$ can compromise our RABAEKS scheme with advantage ϵ_1, thus $\Pr[\mu' = \mu \mid \mathcal{O} = \mathcal{O}_{LWE}] = \frac{1}{2} + \epsilon_1$. On the other hand, $\mathcal{C}$ outputs $\mathcal{O} = \mathcal{O}'_{LWE}$ meaning that $\{(\mathbf{A}_j, \mathbf{b}_j)\}_{j\in[0,|att|+1]}$, $\{(\mathbf{W}_j, \mathbf{u}_j)\}_{j\in[|kw|]}$, $\{(\mathbf{B}_j, \mathbf{v}_j)\}_{j\in[1,2|\mathcal{U}|-2]}$, and $\{(\mathbf{a}_i, b_i)\}_{i\in[m]}$ are sampled over $\mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^m$, $\mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^m$, and $\mathbb{Z}_q^n \times \mathbb{Z}_q$, respectively, thus $\Pr[\mu' = \mu \mid \mathcal{O} = \mathcal{O}'_{LWE}] = \frac{1}{2}$. Considering the successful execution of this IND-CKA game, the challenger $\mathcal{C}$ has advantage $\frac{\epsilon_1}{2}$ to solve the $\mathrm{LWE}_{n,m,q,\chi}$ hardness. □

Theorem 2. *Assume that the $\mathrm{LWE}_{n,m,q,\chi}$ hardness holds, our proposed lattice-based RABAEKS scheme satisfies IND-KGA security in the ROM. For any PPT adversary $\mathcal{A}$, if $\mathcal{A}$ can compromise our scheme with a non-negligible advantage ϵ_2, then a PPT challenger $\mathcal{C}$ can be constructed to solve the $\mathrm{LWE}_{n,m,q,\chi}$ hardness with a non-negligible probability.*

Proof Assume that there exists a PPT adversary $\mathcal{A}$ that has ability to break the IND-KGA secure with a non-negliable advantage ϵ_2, we can construct a challenger $\mathcal{C}$ that solves the decisional $\mathrm{LWE}_{n,m,q,\chi}$ hardness with the same advantage. In the Initialization phase, $\mathcal{A}$ sends a challenge attribute $(\mathcal{S}^*, \xi^*)$ and a revocation list $\mathcal{R}^*$ to $\mathcal{C}$, where $t^* \in [1, |att|)$. Let $\mathcal{ATT} = \{1, \cdots, |att|\}$, where $\mathcal{W} \subseteq \mathcal{ATT}$, the interaction between $\mathcal{A}$ and $\mathcal{C}$ is showed as follows.

Setup: The challenger $\mathcal{C}$ samples $(\mathbf{A}_S, \mathbf{b})$ from $\mathcal{O}$. Then, $\mathcal{C}$ selects many random matrices $\mathbf{A}_0 \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$, $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$, $\{\mathbf{A}_i\}_{i\in\mathcal{ATT}} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$, $\overline{\mathbf{A}} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$, $\mathbf{M} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$, $\{\mathbf{B}_i\}_{i\in[2|\mathcal{U}|-2]} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$, sets a abort-resistant hash function $H : \mathbb{Z}_q^{|kw|} \setminus \{0\}^{|kw|} \to \mathbb{Z}_q$ defined in ***Definition 6***, many matrices $\{\mathbf{W}_i =$

$\mathbf{A}_S\mathbf{R}'_i + \theta_i\mathbf{A}\}_{i\in[|kw|]}$ where $\boldsymbol{\theta} = (\theta_1, \cdots, \theta_{|kw|}) \in \mathbb{Z}_q^{|kw|}$ and $\mathbf{R}'_i \in \{-1,1\}^{m\times m}$, and sends the public parameter $pp = (n, m, q, \sigma, L, |att|, l, |kw|, \mathbf{A}_0, \mathbf{A}, \{\mathbf{A}_i\}_{i\in\mathcal{ATT}}, \{\mathbf{W}_i\}_{i\in[|kw|]}, \mathbf{M}, \mathcal{ATT}, \mathcal{U}, \mathcal{R}, \mathbf{B}_0, \{\mathbf{U}_i\}_{i\in[1,|\mathcal{U}|-2]}, \{\mathbf{V}_i\}_{i\in[|\mathcal{U}|-1,2|\mathcal{U}|-2]}, \mathcal{T})$ to $\mathcal{A}$.

Phase 1: The adversary $\mathcal{A}$ executes these following queries adaptively.

- **Secret Key Queries** $\mathcal{O}_{RK}$: The query process is same as $\mathcal{O}_{RK}$ described in ***Theorem 1***.
- **Ciphertext Queries** $\mathcal{O}_{CT}$: The adversary $\mathcal{A}$ will query the $\mathcal{O}_{CT}$ q_H times. For the each query, $\mathcal{A}$ inputs $((\mathcal{W}, t), \mathbf{ck})$, if $H(\mathbf{ck}) = 0$, this game is aborted, and $\mathcal{A}$ outputs $\mu' \in \{0,1\}$ randomly. Otherwise, $\mathcal{C}$ calculates $\mathbf{c}_0$, $\mathbf{c}_1$, $\bar{\mathbf{c}}$, $\{\mathbf{c}_i\}_{i\in\mathcal{W}}$, $\mathbf{c}_{\mathbf{M}}$, and $\{\mathbf{c}_k\}_{k\in\mathsf{KUNodes}(\mathcal{T},\mathcal{R}^*)}$ as same as in $\textbf{RABAEKS}(pp, pk_S, sk_S, (\mathcal{W}, t), \mathbf{ck})$ algorithm. Then, $\mathcal{C}$ invokes $\mathbf{c}_2 \leftarrow \mathsf{SampleRight}(\mathbf{A}_S, (1+\sum_{i=1}^{|kw|}\theta_i ck_i)\mathbf{A}, \mathbf{R}', \mathbf{T}_{\mathbf{A}}, \bar{\mathbf{c}}, \sigma)$ to generate a vector $\mathbf{c}_2 \in \mathbb{Z}_q^{2m}$ statistically distributed in $\mathcal{D}^{2m}_{\Lambda_q^{\bar{c}}(\mathbf{A}_S|(\mathbf{A}+\sum_{i=1}^{|kw|}ck_i\mathbf{W}_i))}$, s.t. $(\mathbf{A}_S \mid (\mathbf{A} + \sum_{i=1}^{|kw|}ck_i\mathbf{W}_i))\mathbf{c}_2 = \bar{\mathbf{c}} \bmod q$, where $\mathbf{R}' = \sum_{i=1}^{|kw|}ck_i\mathbf{R}'_i$. Finally, $\mathcal{C}$ returns the ciphertext $\mathrm{CT} := (\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2, \{\mathbf{c}_i\}_{i\in\mathcal{W}}, \mathbf{c}_{\mathbf{M}}, \{\mathbf{c}_k\}_{k\in\mathsf{KUNodes}(\mathcal{T},\mathcal{R})})$ to $\mathcal{A}$.
- **Trapdoor Queries** $\mathcal{O}_{TD}$: After $\mathcal{A}$ inputs $((\mathcal{S}, \xi), \mathbf{tk})$, $\mathcal{C}$ executes $\textbf{Trapdoor}(pp, pk_S, sk_R, (\mathcal{S}, \xi), \mathbf{tk})$ to generate the ciphertext TD with the keyword $\mathbf{tk}$, and then returns it to $\mathcal{A}$.

Challenge: $\mathcal{A}$ selects two keywords $\mathbf{tk}_0^*$, $\mathbf{tk}_1^*$ which have not been queried in **Phase 1**, and sends them to $\mathcal{C}$. Then, $\mathcal{C}$ selects a bit $\mu \in \{0,1\}$ randomly. If $H(\mathbf{tk}_\mu^*) = 0$, this game is aborted, and $\mathcal{A}$ outputs $\mu' \in \{0,1\}$ randomly. Otherwise, $\mathcal{C}$ generates the trapdoor TD_μ^* with the keyword $\mathbf{tk}_\mu^*$ and the attribute $(\mathcal{S}^*, \xi^*)$. Specially, $\mathcal{C}$ calculates $\mathbf{t}_0^* = \begin{pmatrix}\mathbf{b}\\(\mathbf{R}')^\top\mathbf{b}\end{pmatrix}$, and samples a random vector $\mathbf{t}_i^* \xleftarrow{\$} \mathcal{D}_{\mathbb{Z}_q^{5m},\sigma}$ for $i \in \mathcal{S}^*$. Finally, $\mathcal{C}$ returns $\mathrm{TD}_\mu^* = (\mathbf{t}_0^*, \{\mathbf{t}_i^*\}_{i\in\mathcal{S}})$ to $\mathcal{A}$.

Phase 2: The adversary $\mathcal{A}$ continues to query from $\mathcal{O}_{RK}$, $\mathcal{O}_{CT}$, and $\mathcal{O}_{TD}$ as same as in **Phase 1** except that the $((\mathcal{W}, t), \mathbf{ck})$ could be queried in $\mathcal{O}_{CT}$ when $\mathbf{ck} \notin (\mathbf{tk}_0^*, \mathbf{tk}_1^*)$.

Guess: The adversary $\mathcal{A}$ outputs $\mu' \in \{0,1\}$ as the answer to $\mathrm{LWE}_{n,m,q,\chi}$ hardness. Otherwise, $\mathcal{A}$ outputs $\mu' \in \{0,1\}$ randomly. Then, the challenger $\mathcal{C}$ checks whether $H(\mathbf{tk}_\mu^*) = 0$ and $H(\mathbf{ck}_i) \neq 0$ for $i \in [q_H]$. If not, $\mathcal{C}$ overwrites $\mu' \in \{0,1\}$ and aborts this game. After that, $\mathcal{C}$ selects a bit $\rho \in \{0,1\}$, where $\Pr[\rho = 1] = \gamma(\mathbf{tk}_\mu^*, \mathbf{ck}_1, \cdots, \mathbf{ck}_{q_H})$ and the function $\gamma()$ is defined in [43]. If $\rho = 1$, $\mathcal{C}$ overwrites $\mu' \in \{0,1\}$ and aborts this game as an artificial abort. If $\mu' = \mu$, $\mathcal{A}$ wins this game and $\mathcal{C}$ outputs $\mathcal{O} = \mathcal{O}_{LWE}$. Otherwise, $\mathcal{C}$ outputs $\mathcal{O} = \mathcal{O}'_{LWE}$.

Analysis: If $\mu' = \mu$, $\mathcal{C}$ outputs $\mathcal{O} = \mathcal{O}_{LWE}$ meaning that $(\mathbf{A}_S, \mathbf{b})$ is sampled from $\mathcal{O}_{LWE}$ such that $\mathbf{b} = \mathbf{A}_S^\top\mathbf{s} + \mathbf{x}'_0$, where $\mathbf{s} \in \chi^n$ is a secret vector and $\mathbf{x}'_0 \leftarrow \chi^m$ is a noise vector. Then, we can obtain the following equations:

$$\mathbf{t}_0^* = \begin{pmatrix}\mathbf{b}\\(\mathbf{R}')^\top\mathbf{b}\end{pmatrix} = \begin{pmatrix}\mathbf{A}_S^\top\mathbf{s} + \mathbf{x}'_0\\(\mathbf{A}_S\mathbf{R}')^\top\mathbf{s} + (\mathbf{R}')^\top\mathbf{x}'_0\end{pmatrix}$$
$$= (\mathbf{A}_S \mid (\mathbf{A} + \sum_{i=1}^{|kw|} tk_i\mathbf{W}_i))^\top\mathbf{s} + \begin{pmatrix}\mathbf{x}'_0\\(\mathbf{R}')^\top\mathbf{x}'_0\end{pmatrix}$$
$$\mathbf{t}_i^* \in \mathcal{D}_{\mathbb{Z}^{2m},\sigma}, i \in \mathcal{S}$$

Consequently, $\mathrm{TD}_\mu^* = (\mathbf{t}_0^*, \{\mathbf{t}_i\}_{i\in\mathcal{W}})$ is a valid trapdoor. From the Lemma 7 and $q_H \leq \frac{q}{2}$, we can obtain $|p_{max} - p_{min}| = \frac{1}{q} - \frac{1}{q}(1 - \frac{q_H}{q}) = \frac{q_H}{q^2}$, and $p_{min} = \frac{1}{q}(1 - \frac{q_H}{q}) \geq \frac{1}{2q}$. In order to realize $|p_{max} - p_{min}|$ is negligible, we need to make q large enough. After that, according to the Lemma 28 in [42], the advantage to solve the $\mathrm{LWE}_{n,m,q,\chi}$ hardness is $\epsilon' \geq \frac{p_{min}}{2}\epsilon_2 \geq \frac{\epsilon_2}{4q}$, considering the artificial abort of this IND-KGA game. □

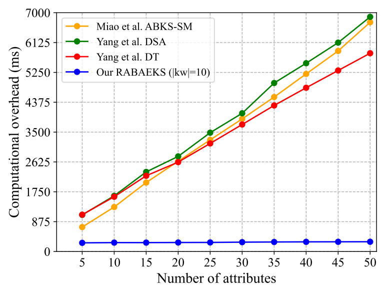
7 Performance Evaluation and Comparison with Prior Arts

In this section, we implement our RABAEKS scheme using Python language with Numpy library, and the hardware environment is accomplished on a laptop with 12-th Gen Intel(R) Core(TM) i7-12800HX CPU with 16 GB RAM under Windows 10 operating system. According to the parameter setting described in Section 5.3, we select $q = 4096$, $n = 256$ and $m = \lceil 2n\log q\rceil = 6144$ to simulate our **RABAEKS**, **Trapdoor**, and **Test** algorithms, and assume that the number of $\mathcal{S}$ is equal to the number of $\mathcal{W}$. For other state-of-the-art schemes, we set $|\mathbb{G}| = |\mathbb{G}_T| = 1024bits$, $|\mathbb{Z}_p| = 1024bits$, and the bilinear pairing is initialized by Type A elliptic curves $y^2 = x^3 + x$.

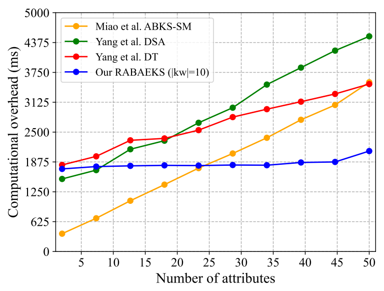
7.1 Computational overhead

Fig. 5 depicts the computational overhead comparison between our RABAEKS and prior arts in the context of ciphertext generation, trapdoor generation and search phase, as the the number of attributes l increases. As for Fig. 5(a), we set the length of keyword $|kw| = 10$ in our RABAEKS scheme, which does not impact the computational overhead of other schemes. It is evident that the computational overhead for all schemes is proportional to l when calculating keyword ciphertexts, with our RABAEKS scheme proving to be significantly more efficient than all other solutions. Regarding Fig. 5(b), the computational overhead of our trapdoor generation algorithm is comparable to the other three schemes. When the attributes number $l \geq 25$, our RABAEKS scheme exhibits lower computational overhead than the other three schemes, with a more gradual increase as l grows. For the search phase, as shown in Fig. 5(c), the computational overhead remains relatively stable as the number of attributes l increases. Consequently, when there are numerous keyword ciphertexts and search requests in the cloud server, our solution proves to be much more practical due to its enhanced search efficiency. In Table 3, we illustrate the specific computational overhead when the attributes number is $l = 50$. Specifically, the computational overhead of ciphertext generation, trapdoor generation and search phases is $286ms$, $2103ms$, and $0.30ms$, which is at least $20\times$, $1.67\times$ and $1897\times$ faster than prior arts, respectively.

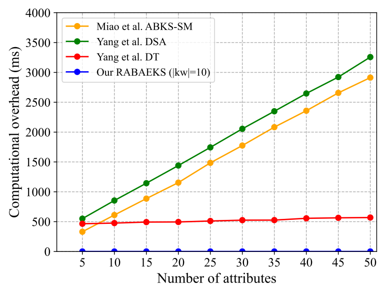
As depicted in Fig. 6, we assess the effects of varying keyword length (i.e., $|kw| = 10, 30, 50$) on the ciphertext generation, trapdoor generation, and search phase in our RABAEKS scheme, as the number of attributes l increases. As shown in Fig. 6(a), the computational overhead for the ciphertext generation algorithm at $|kw| = 10, 30, 50$ are closely aligned, indicating that increasing keyword length



(a) Ciphertext generation



(b) Trapdoor generation

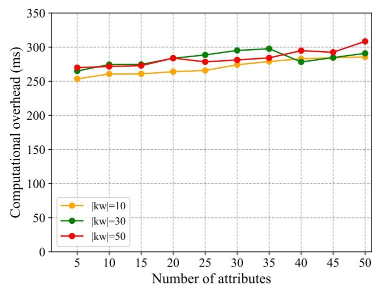


(c) Search phase

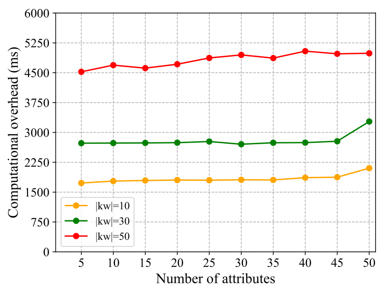
Fig. 5. Computational overhead comparison between our RABAEKS and other state-of-the-art schemes.

TABLE 3
Computational overhead evaluation

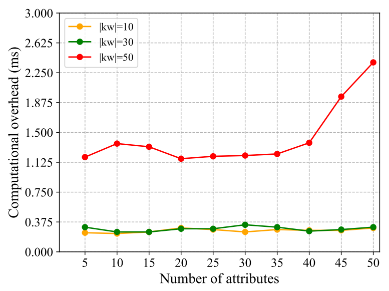
Schemes	**Ciphertext generation** (ms)	**Trapdoor generation** (ms)	**Search phase** (ms)
Miao et al. ABKS-SM	6720	3544	2913
Yang et al. DSA	6878	4505	3257
Yang et al. DT	5812	3506	569
Our RABAEKS	286	2103	0.30



(a) Ciphertext generation



(b) Trapdoor generation



(c) Search phase

Fig. 6. Computational overhead with the attributes number l between the different length of keyword $|kw|$.

has minimal impact on the efficiency of keyword ciphertext computation. Furthermore, we compare the computational overhead of trapdoor generation with the results presented in Fig. 6(b). It is noted that the increment of keyword length significantly impacts the efficiency compared to the attributes number. In detail, when $l = 50$, the computational overhead for three keyword lengths are $2103ms$, $3275ms$, and $4989ms$, respectively. Moreover, the search efficiency are presented in Fig. 6(c) when the keyword length $|kw| = 10, 30, 50$. Concretely, the cost of our RABAEKS scheme is similar for $|kw| = 10$ and $|kw| = 30$ (i.e. $0.30ms$ and $0.31ms$), but it becomes higher at $|kw| = 50$, reaching $2.38ms$. Nevertheless, our solution remains much more efficient than the other three schemes, which take $2913ms$, $3257ms$, and $569ms$, respectively. This is because our approach relies solely on number and matrix operations, such as matrix modulo multiplication, and avoids expensive pairing operations. As a result, our RABAEKS scheme offers a superior search experience for entities in cloud storage.

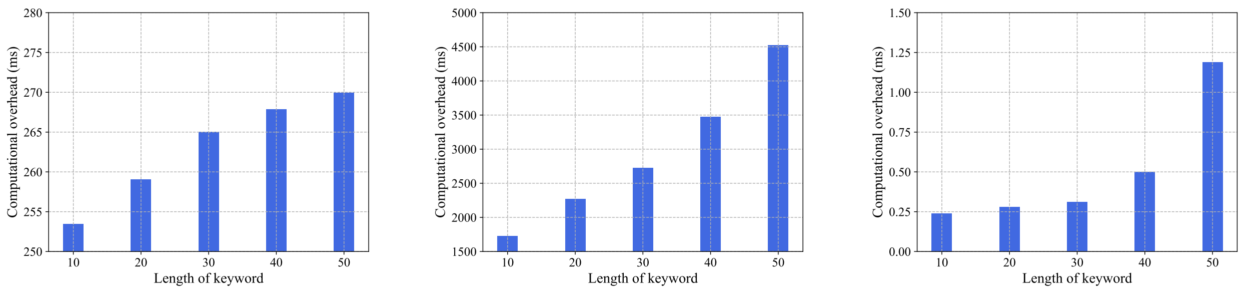
To further demonstrate the efficiency of our scheme in relation to the increasing keyword length, we choose the number of attributes $l = 10$ and vary the keyword length values (i.e., $|kw| = 10, 20, 30, 40, 50$). Based on aforementioned parameters setting, in Fig. 7, we show the computational overhead of our ciphertext generation, trapdoor generation and search phase, which provides further corroboration of the result in Fig. 6.

TABLE 4
Glossary

Acronym	**Definition**
l	the number of attributes
l_i	the number of possible values for each attribute
$\lvert\mathbf{M}_{row}\rvert$	the row number of access matrix $\mathbf{M}$
ϕ	the order of keyword polynomial in DT
$\lvert\mathsf{KUNodes}(\mathcal{T}, \mathcal{R})\rvert$	the number of $\mathsf{KUNodes}(\mathcal{T}, \mathcal{R})$
$\lvert\mathbb{Z}_p\rvert$ (resp. $\lvert\mathbb{Z}_q\rvert$)	the element length in $\mathbb{Z}_p$ (resp. $\mathbb{Z}_q$)
$\lvert\mathbb{G}\rvert$ (resp. $\lvert\mathbb{G}_T\rvert$)	the element length in $\mathbb{G}$ (resp. $\mathbb{G}_T$)

7.2 Communication overhead

For searchable encryption schemes, the communication overhead relies on the keyword ciphertext and search trapdoor size transmitted between the data sender, data receiver, and cloud server. Firstly, we analyze the communication

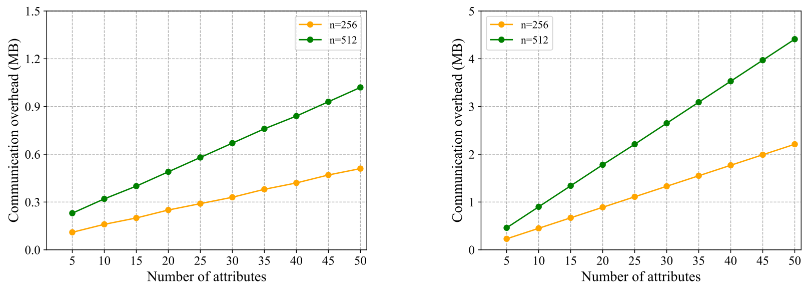


(a) Ciphertext generation (b) Trapdoor generation (c) Search phase

Fig. 7. Computational overhead with the keyword length $|kw|$.

TABLE 5
Communication overhead comparison

Schemes	Ciphertext generation	Trapdoor generation
Miao et al. ABKS-SM.	$(l+2)\|\mathbb{Z}_p\| + (\sum_{i=1}^{l}(l_i + \|\mathbf{M}_{row}\| + l + 2)\|\mathbb{G}\| + 3\|\mathbb{G}_T\|$	$2\|\mathbb{Z}_p\| + \|\mathbb{G}\|$
Yang et al. DSA	$(2l+3)\|\mathbb{G}\|$	$(4l+2)\|\mathbb{G}\|$
Yang et al. DT	$(2l+\phi+4)\|\mathbb{G}\| + (\|\mathbf{M}_{row}\|+2)\|\mathbb{G}_T\|$	$(l+\phi+3)\|\mathbb{G}\| + l\|\mathbb{G}_T\|$
Our RABAEKS	$(l + \|\mathsf{KUNodes}(\mathcal{T},\mathcal{R})\| + 5)m\|\mathbb{Z}_q\|$	$(5l+1)m\|\mathbb{Z}_q\|$



(a) Ciphertext generation (b) Trapdoor generation

Fig. 8. Communication overhead with the number of attributes l between different security parameter n.

overhead of the keyword ciphertext and trapdoor in our RABAEKS and other prior arts. In Table. 4, we list the length parameters for the theoretical evaluation, and the result is showed in Table. 5. Specifically, the ciphertext and trapdoor size of other three schemes are simply the linear combinations of element length on group, and grow linearly with the number of attributes l. For our RABAEKS scheme, the ciphertext and trapdoor size increases with the security parameters n, m, the number of attributes l, and the size of $\mathsf{KUNodes}(\mathcal{T},\mathcal{R})$. Since the ciphertext and trapdoor are matrices, our communication overhead is higher than the other three pairing-based schemes, which is a common problem for lattice-based cryptographic schemes. Although our RABAEKS scheme introduces additional communication overhead, it can resist quantum attacks, providing post-quantum security for keywords in cloud storage scenarios, which is not available in the other three schemes.

In Fig. 8, we analyze the communication overhead in calculating the ciphertext and trapdoor with the number of attributes l between $n = 256$ and $n = 512$. As l growing, the ciphertext and trapdoor size increase linearly. Since there are more components related to l in the trapdoor than the ciphertext, the trapdoor size increases more significantly. Moreover, the enhancement of security parameter (m doubles from 256 to 512) leads to the growing in the communication overhead. For example, we set $l = 50$, $n = 512$, $m = 12288$, the ciphertext size is $(50+3+5)\times 12288 \times 12 = 1.02MB$, and the trapdoor size is $(5\times 50+1)\times 12288\times 12 = 4.41MB$. When n doubles from 256 to 512, the ciphertext and trapdoor size are change from $0.51MB$ to $1.02MB$, and $2.21MB$ to $4.41MB$, respectively.

8 Conclusion and Future Work

In this paper, we propose RABAEKS, a post-quantum revocable attribute-based authenticated encrypted search scheme for multi-receiver cloud storage. Our design embeds attributes into the data receiver's secret key and access policies into the keyword ciphertext, to realize the keyword ciphertext search and access control simultaneously. In addition, we introduce revocation of data receivers to construct a more dynamic access control mechanism. The rigorous security analysis demonstrates that our RABAEKS achieves IND-CKA and IND-KGA security in the ROM. The comprehensive experimental evaluations also indicate that the computational overhead of our ciphertext generation, trapdoor generation, and search algorithms are at least $20\times$, $1.67\times$, and $1897\times$ faster than state-of-the-art schemes. Although our design reduces the IND-CKA and IND-KGA security to LWE hardness, it has only been proven in the ROM. As such, improving security in the standard model is an interesting direction for future work. Besides, our scheme is not resistant to the secret key leakage attacks, i.e., the security of keywords cannot be guaranteed when the data owner/receiver's secret key has been compromised. Addressing this vulnerability is also a valuable area for exploration.

References

[1] C. Cai, J. Weng, X. Yuan, and C. Wang, "Enabling reliable keyword search in encrypted decentralized storage with fairness," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 1, pp. 131–144, 2018.

[2] K. He, J. Guo, J. Weng, J. Weng, J. K. Liu, and X. Yi, "Attribute-based hybrid boolean keyword search over outsourced encrypted data," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 6, pp. 1207–1217, 2018.

[3] Y. Chen, W. Li, F. Gao, Q. Wen, H. Zhang, and H. Wang, "Practical attribute-based multi-keyword ranked search scheme in cloud computing," *IEEE Transactions on Services Computing*, vol. 15, no. 2, pp. 724–735, 2019.

[4] Y. Miao, R. H. Deng, K.-K. R. Choo, X. Liu, J. Ning, and H. Li, "Optimized verifiable fine-grained keyword search in dynamic multi-owner settings," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 4, pp. 1804–1820, 2019.

[5] C. Ge, W. Susilo, Z. Liu, J. Xia, P. Szalachowski, and L. Fang, "Secure keyword search and data sharing mechanism for cloud computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 6, pp. 2787–2800, 2020.

[6] K. Zhang, M. Wen, R. Lu, and K. Chen, "Multi-client sub-linear boolean keyword searching for encrypted cloud storage with owner-enforced authorization," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 6, pp. 2875–2887, 2020.

[7] C. Huang, D. Liu, A. Yang, R. Lu, and X. Shen, "Multi-client secure and efficient dpf-based keyword search for cloud storage," *IEEE Transactions on Dependable and Secure Computing*, vol. 21, no. 1, pp. 353–371, 2023.

[8] S. Xu, X. Chen, and Y. He, "Evchain: An anonymous blockchain-based system for charging-connected electric vehicles," *Tsinghua Science and Technology*, vol. 26, no. 6, pp. 845–856, 2021.

[9] Q. Huang and H. Li, "An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks," *Information Sciences*, vol. 403, pp. 1–14, 2017.

[10] D. He, M. Ma, S. Zeadally, N. Kumar, and K. Liang, "Certificateless public key authenticated encryption with keyword search for industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 8, pp. 3618–3627, 2017.

[11] Y. Lu and J. Li, "Lightweight public key authenticated encryption with keyword search against adaptively-chosen-targets adversaries for mobile devices," *IEEE Transactions on Mobile Computing*, vol. 21, no. 12, pp. 4397–4409, 2021.

[12] Z.-Y. Liu, Y.-F. Tseng, R. Tso, M. Mambo, and Y.-C. Chen, "Public-key authenticated encryption with keyword search: A generic construction and its quantum-resistant instantiation," *The Computer Journal*, vol. 65, no. 10, pp. 2828–2844, 2022.

[13] L. Cheng and F. Meng, "Public key authenticated encryption with keyword search from lwe," in *European Symposium on Research in Computer Security*. Springer, 2022, pp. 303–324.

[14] X. Liu, K. He, G. Yang, W. Susilo, J. Tonien, and Q. Huang, "Broadcast authenticated encryption with keyword search," in *Australasian Conference on Information Security and Privacy*. Springer, 2021, pp. 193–213.

[15] V. B. Chenam and S. T. Ali, "A designated cloud server-based multi-user certificateless public key authenticated encryption with conjunctive keyword search against ikga," *Computer Standards & Interfaces*, vol. 81, p. 103603, 2022.

[16] L. Sun, Z. Cao, X. Dong, J. Shen, M. Wang, and J. Chen, "dm-claeks: Pairing-free designated-tester multi-recipient certificateless authenticated encryption with keyword search for concealing search patterns," *Journal of Systems Architecture*, vol. 144, p. 103007, 2023.

[17] F. Luo, H. Wang, C. Lin, and X. Yan, "Abaeks: Attribute-based authenticated encryption with keyword search over outsourced encrypted data," *IEEE Transactions on Information Forensics and Security*, 2023.

[18] 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 Transactions on Services Computing*, 2023.

[19] D. Han, N. Pan, and K.-C. Li, "A traceable and revocable ciphertext-policy attribute-based encryption scheme based on privacy protection," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 1, pp. 316–327, 2020.

[20] H. Deng, Z. Qin, Q. Wu, Z. Guan, and H. Yin, "Revocable attribute-based data storage in mobile clouds," *IEEE Transactions on Services Computing*, vol. 15, no. 2, pp. 1130–1142, 2020.

[21] C. Ge, W. Susilo, J. Baek, Z. Liu, J. Xia, and L. Fang, "Revocable attribute-based encryption with data integrity in clouds," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 5, pp. 2864–2872, 2021.

[22] S. Zhao, R. Jiang, and B. Bhargava, "Rl-abe: A revocable lattice attribute based encryption scheme based on r-lwe problem in cloud storage," *IEEE Transactions on Services Computing*, vol. 15, no. 2, pp. 1026–1035, 2020.

[23] F. Luo, S. Al-Kuwari, H. Wang, F. Wang, and K. Chen, "Revocable attribute-based encryption from standard lattices," *Computer Standards & Interfaces*, vol. 84, p. 103698, 2023.

[24] L. Guo, L. Wang, X. Ma, and Q. Ma, "A new revocable attribute based encryption on lattice," in *International Conference on Provable Security*. Springer, 2023, pp. 309–326.

[25] S. Xu, X. Chen, Y. Guo, S.-M. Yiu, S. Gao, and B. Xiao, "Efficient and secure post-quantum certificateless signcryption for internet of medical things," *Cryptology ePrint Archive*, 2024.

[26] M. Yang, H. Wang, and D. He, "Pm-abe: Puncturable bilateral fine-grained access control from lattices for secret sharing," *IEEE Transactions on Dependable and Secure Computing*, 2024.

[27] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM review*, vol. 41, no. 2, pp. 303–332, 1999.

[28] S. Xu, Y. Cao, X. Chen, Y. Guo, Y. Yang, F. Guo, and S.-M. Yiu, "Lattice-based public key encryption with authorized keyword search: Construction, implementation, and applications," *Cryptology ePrint Archive*, 2023.

[29] G. Xu, S. Xu, Y. Cao, F. Yun, Y. Cui, Y. Yu, and K. Xiao, "Ppseb: A postquantum public-key searchable encryption scheme on blockchain for e-healthcare scenarios," *Security and Communication Networks*, vol. 2022, no. 1, p. 3368819, 2022.

[30] A. Boldyreva, V. Goyal, and V. Kumar, "Identity-based encryption with efficient revocation," in *Proceedings of the 15th ACM conference on Computer and communications security*, 2008, pp. 417–426.

[31] Y. Miao, X. Liu, K.-K. R. Choo, R. H. Deng, J. Li, H. Li, and J. Ma, "Privacy-preserving attribute-based keyword search in shared multi-owner setting," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 3, pp. 1080–1094, 2019.

[32] K. Yang, J. Shu, and R. Xie, "Efficient and provably secure data selective sharing and acquisition in cloud-based systems," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 71–84, 2022.

[33] Y. Yang, R. H. Deng, W. Guo, H. Cheng, X. Luo, X. Zheng, and C. Rong, "Dual traceable distributed attribute-based searchable encryption and ownership transfer," *IEEE Transactions on Cloud Computing*, vol. 11, no. 1, pp. 247–262, 2021.

[34] M. Ajtai, "Generating hard instances of lattice problems," in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, 1996, pp. 99–108.

[35] C. Peikert, "An efficient and parallel gaussian sampler for lattices," in *Annual Cryptology Conference*. Springer, 2010, pp. 80–97.

[36] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *Journal of the ACM (JACM)*, vol. 56, no. 6, pp. 1–40, 2009.

[37] B. Applebaum, D. Cash, C. Peikert, and A. Sahai, "Fast cryptographic primitives and circular-secure encryption based on hard learning problems," in *Advances in Cryptology-CRYPTO 2009: 29th*

Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings. Springer, 2009, pp. 595–618.

[38] C. Gentry, C. Peikert, and V. Vaikuntanathan, "Trapdoors for hard lattices and new cryptographic constructions," in *Proceedings of the fortieth annual ACM symposium on Theory of computing*, 2008, pp. 197–206.

[39] D. Micciancio and C. Peikert, "Trapdoors for lattices: Simpler, tighter, faster, smaller." in *Eurocrypt*, vol. 7237. Springer, 2012, pp. 700–718.

[40] D. Cash, D. Hofheinz, and E. Kiltz, "How to delegate a lattice basis," *Cryptology ePrint Archive*, 2009.

[41] D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert, "Bonsai trees, or how to delegate a lattice basis," *Journal of cryptology*, vol. 25, pp. 601–639, 2012.

[42] S. Agrawal, D. Boneh, and X. Boyen, "Efficient lattice (h) ibe in the standard model," in *Advances in Cryptology–EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30–June 3, 2010. Proceedings 29.* Springer, 2010, pp. 553–572.

[43] B. Waters, "Efficient identity-based encryption without random oracles," in *Advances in Cryptology–EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005. Proceedings 24.* Springer, 2005, pp. 114–127.