Lattice-based Broadcast Authenticated Searchable Encryption for Cloud Storage

Yibo Cao, *Student Member, IEEE,* Shiyuan Xu, *Student Member, IEEE,* Xiu-Bo Chen, Gang Xu, and Siu-Ming Yiu, *Member, IEEE*

Abstract—The extensive use of cloud storage has created an urgent need to search and share data. Public key authenticated encryption with keyword search (PAEKS) allows for the retrieval from encrypted data, while resisting the insider keyword guessing attacks (IKGAs). Most PAEKS schemes only work with single-receiver model, exhibiting very limited applicability. To address this concern, there have been researches on broadcast authenticated encryption with keyword search (BAEKS) to achieve multi-receiver ciphertext search. But to our best knowledge, existing BAEKS schemes are susceptible to quantum computing attacks. In this paper, we propose lattice-based BAEKS, the first post-quantum broadcast authenticated encryption with keyword search, providing robust quantum-safety in multi-receiver model. Specifically, we leverage several lattice sampling algorithms and rejection sampling technique to construct our BAEKS scheme. Furthermore, we incorporate minimal cover set technique and lattice basis extension algorithm to construct an enhanced version, namely FS-BAEKS. Moreover, we give a rigorous security analysis of our scheme. Ultimately, the best computational overhead of BAEKS and Test algorithms in our BAEKS scheme delivers up to approximately 12-x and 402-x faster over prior arts when the number of receivers is six, respectively, which is practical for cloud storage systems.

Index Terms—Cloud storage, broadcast authenticated searchable encryption, lattice, forward security.

1 Introduction

Cloud storage provides users with searching and sharing their data between data senders and receivers without any geographical restrictions. It has numerous benefits, such as reducing local data maintenance, boosting data circulation, and improving service elasticity [1], [2]. Meanwhile, data privacy leakage problem in cloud storage is commonplace. In order to ensure data security as well as data availability, one possible method is to encrypt the data before sending it to the cloud server. Boneh et al. proposed a cryptographic primitive namely public key encryption with keyword search (PEKS) [3]. Through this technique, data receiver can search the keyword ciphertext uploaded by data sender, the specific process is depicted in Fig. 1.

Nevertheless, traditional PEKS schemes are vulnerable to the insider keyword guessing attacks (IKGAs) [3], [4], [5], [6], [7], [8], [9], [10], where an attacker (an external adversary or cloud server) can intercept a search trapdoor, select a keyword from a limited keyword space to compute the keyword ciphertext by encryption algorithm, and then obtain the keyword information by matching the obtained ciphertext and trapdoor to result in the privacy leakage. In order to resist these attacks, Huang et al. constructed public key authenticated encryption with keyword search (PAEKS) schemes [11] by involving the secret key of data sender in the encryption algorithm.

Most PAEKS primitives are primarily designed for single-receiver model [11], [12], [13], [14], [14], [15], [16], [17]. However, multi-receiver model are more common in real-world cloud storage systems [18]. Specifically, in the cloud-assisted healthcare scenarios, electronic medical records (EMR) are stored in a cloud server [19] [20], and when different departments' physicians need to access the same patient's EMR, a PAEKS scheme that supports multiple users will be more convenient. To address this concern, braodcast authenticated encryption with keyword search (BAEKS) was proposed [21], [22], [23], supporting multiple receivers to search keyword ciphertext uploaded by a data sender. Unfortunately, these schemes neither resist quantum computing attacks, nor avoid the secret key leakage problem. As far as we know, there exists no BAEKS schemes can explicitly enjoy quantum-safety and forward security.

In this paper, we propose lattice-based BAEKS, a novel broadcast authenticated encryption with keyword search over lattice. It supports multi-receiver ciphertext search and withstand the threat from cloud server (refers to IKGAs), protecting the data privacy in the cloud storage systems. Furthermore, we extend our BAEKS scheme to propose the FS-BAEKS scheme, which can solve the secret key leakage problems.

The constructions of BAEKS and FS-BAEKS primitives address two challenges in terms of data security for cloud storage. The first challenge is how to design a post-quantum BAEKS scheme, which is characterized by both multi-receiver support and IKGAs-resilience. Existing BAEKS schemes [21], [22], [23] are vulnerable to quantum computing attacks since their security relies on the traditional hardness assumptions, (i.e. discrete-logarithm (DL)

- *Corresponding author: Xiu-Bo Chen*
- *Y. Cao and X.-B. Chen are with the State Key Laboratory of Network and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China. (E-mail: caoyibo@bupt.edu.cn, flyover100@163.com).*
- *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).*
- *G. Xu is with the School of Information Science and Technology, North China University of Technology, Beijing, China. (E-mail: gx@ncut.edu.cn).*

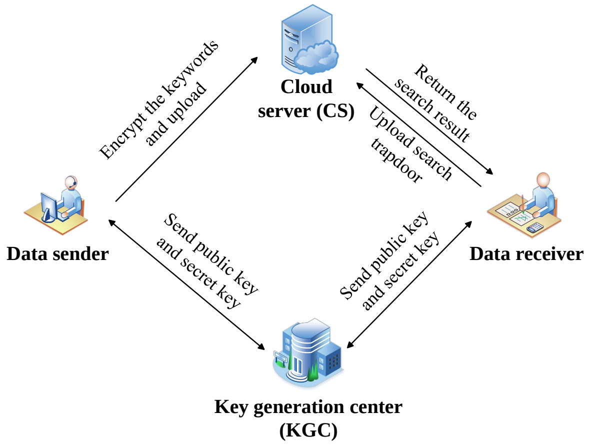


Fig. 1. The ciphertext search model for cloud storage.

hardness). Different from these previous BAEKS schemes, we introduce the lattice algebra structure and lattice basis sampling algorithm to construct the post-quantum BAEKS scheme. Specifically, we encrypt the keyword with a data sender's secret key and data receivers' public keys to support multi-receiver usage. To calculate a search trapdoor, a straightforward idea is to use SamplePre or SampleLeft algorithm, however, it is unable for each receiver in the data receivers set to search the keyword ciphertext, which is a difficult issue to be concerned. In our design, we come with a non-trivial way, by utilizing the SampleBasis and GenSamplePre algorithms with inputting a data receiver's secret key (lattice basis matrix) to generate an appropriate search trapdoor. More specially, our scheme has efficient computational overhead than current BAEKS schemes [21], [22] in terms of **BAEKS** and **Test** algorithms.

The second challenge is how to address the secret key leakage problem for lattice-based BAEKS. More specifically, a malicious adversary has the ability to calculate a trapdoor corresponding to a specific keyword if it obtained the data receivers' secret key. Then, the adversary can send it to cloud server in order to match the keyword ciphertext, thereby significantly compromising the security of keywords. To solve this problem, we achieve the forward security [24], it means that the secret key leakage in a future time period does not influenced the ciphertext security in past time periods. Technically, inspired by Yu et al. [25], we introduce the binary tree structure, minimal cover set technique and lattice basis extension algorithm to update the receivers' secret key. In a nutshell, our contributions are demonstrated as follows:

- We present a novel scheme namely lattice-based broadcast authenticated encryption with keyword search (BAEKS) in a quantum setting, as well as defining system models, formal definitions and two security models for it. Then, in order to ensure the security of data receivers' secret key, we propose lattice-based forward secure broadcast authenticated encryption with keyword search (FS-BAEKS) as the enhanced verison of BAEKS. As far as we know, numerous existing PEKS primitives cannot support multi-receiver model, and are vulnerable to several attacks, e.g. quantum computing attacks, IKGAs, secret key leakage attacks. Our schemes have the ability to resist all of aforementioned attacks simultaneously.
- We construct BAEKS scheme leveraged lattice algebra structure, several lattice sampling algorithms and rejection sampling technique, which supports multi-receiver ciphertext search to protect the data privacy in cloud storage systems. Concretely, through the SampleBasis and GenSamplePre algorithms, each receiver in the data receivers set can generate a proper search trapdoor. Moreover, based on our BAEKS, the binary tree structure, minimal cover set technique and lattice basis extension algorithm [25] is introduced, achieving time periods representation and data receivers' secret key update to construct FS-BAEKS scheme.
- Our BAEKS & FS-BAEKS schemes have been proven to be secure in IND-CKA and UF-IKGA models, which can be reduced to the LWE and SIS hardness in the random oracle model, respectively. Performance evaluation and comparison manifests that our BAEKS & FS-BAEKS schemes are more computationally efficient in terms of **BAEKS** and **Test** algorithms compared to the prior arts [21], [22]. In particular, for the computational overhead of our BAEKS scheme at the number of receivers $l = 6$, the **BAEKS** algorithm delivers up to $6\times$ and $12\times$, and the **Test** algorithm brings up to $120\times$ and $402\times$ faster over prior arts [21], [22], respectively. Moreover, the communication overhead has acceptable growth trend with the increment of receivers, time periods or security parameter.

The remainder of this paper is structured 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, formal definitions, and security models for BAEKS are then depicted in Section 4. A detailed explanation and its security analysis of BAEKS scheme is demonstrated in Section 5, while Section 6 focuses on the FS-BAEKS scheme, which is an enhanced version of BAEKS. In Section 7, we delve into the performance evaluation and comparison. Finally, we summarize this paper in Section 8.

2 Related Works

From the first PEKS scheme introduced by Boneh et al. [3], various variant-PEKS schemes have been presented. Byun et al. considered that the PEKS schemes does not resist to keyword guessing attacks (KGAs) [29]. Then, numerous researchers proposed numerous schemes to resist the KGAs under the external adversary. In detail, Baek et al. constructed an efficient SCFPEKS scheme, which designates a tester to remove a reliable channel between the data receiver and cloud server [7]. Rhee et al. enhanced the security models to proposed designated tester public key encryption with keyword search (dPEKS) [8], and constructed its generic construction [9]. Chen et al. introduced a semi-trusted participating entity namely keyword servers (KS) to construct a SA-PEKS scheme [10], and gave an instantiation utilizing full-domain hash RSA signatures. To resist to IKGAs,

TABLE 1
Comparison with the current state-of-art PEKS/PAEKS/BAEKS schemes

Schemes	Multi-receiver support	IKGA-resilience	Quantum-resistance	Forward security
Boneh et al. [3]	✗	✗	✗	✗
Baek et al. [7]	✗	✗	✗	✗
Rhee et al. [8]	✗	✗	✗	✗
Rhee et al. [9]	✗	✗	✗	✗
Chen et al. [10]	✗	✗	✗	✗
Huang et al. [11]	✗	✓	✗	✗
Liu et al. [12]	✗	✓	✓	✗
Liu et al. [13]	✗	✓	✓	✗
Cheng et al. [14]	✗	✓	✓	✗
Yao et al. [17]	✗	✓	✓	✗
Ali et al. [26]	✓	✗	✗	✗
Kiayias et al. [18]	✓	✗	✗	✗
Liu et al. [21]	✓	✓	✗	✗
Mukherjee [22]	✓	✓	✗	✗
Emura et al. [23]	✓	✓	✗	✗
Zhang et al. [27]	✗	✗	✓	✓
Yu et al. [25]	✗	✗	✓	✓
Yang et al. [28]	✗	✗	✓	✓
Xu et al. [15]	✗	✓	✓	✓
Our BAEKS	✓	✓	✓	✗
Our FS-BAEKS	✓	✓	✓	✓

Huang et al. introduced a public-key authenticated encryption with keyword search (PAEKS) scheme to implement authentication through a data owner's secret key, which can ensure that the keyword encryption procedure can only be performed by the data owner, and demonstrated a rigorous prove for proposed scheme in the random oracle model (ROM) [11]. Liu et al. put forward a generic construction for PAEKS and an instantiation over lattice to achieve the anti-quantum property [12], and enhanced its security [13]. Furthermore, Cheng et al. pointed out some security issues [13], [23], and constructed two PAEKS schemes over lattice [14]. Yao et al. then constructed a CCA-secure PAEKS scheme over ideal lattice, and demonstrated that the resistance of PAEKS scheme to IKGAs is equivalent to the unforgeability of keyword ciphertexts [17].

Since encrypted messages can be decrypted by a group of specified data users, broadcast encryption (BE), initialized by Fiat et al. [30], is more practical compared to one-to-one encryption and is exclusively used in numerous scenarios (e.g. content subscription and digital rights management). To mitigate the public key certificates storage overhead, Delerablèe et al. put forward an identity-based broadcast encryption scheme (IBBE), which keeps the ciphertext size constant and realized the CCA security in the ROM [31]. After that, Boneh et al. provided a broadcast hierarchical identity-based encryption (HIBE) scheme with short ciphertext [32]. Gentry et al. implemented the security of IBBE under standard model [33]. Ali et al. foresaw the combinability of BE and PEKS, and constructed a broadcast searchable keyword encryption scheme, which is a novel cryptographic primitive to search the keyword ciphertext encrypted by the public key of a group of specified data users [26]. Futhermore, an efficient broadcast encryption with keyword search (BEKS) is introduced by Kiayias et al., providing constant secret key and trapdoor size, and the server's storage overhead is independent of the number of data receivers, but is not resistant to IKGAs [18]. Enlightened by the concept of PAEKS, Liu et al. constructed a broadcast authenticated encryption with keyword search (BAEKS) cryptographic primitive to resist to IKGAs, and the ciphertext and trapdoor security was proved under the DBDH assumption [21]. In 2023, Mukherjee introduced a more stronger security model, and ensured the ciphertext and trapdoor security in the standard model [22]. Emura et al. put forward a generic construction of fully anonymous BAEKS, which provides the anonymity and consistency of keyword ciphertext and supports multi-receiver model [23]. However, none of aforementioned schemes can resist to quantum computing attacks, and there exists no post-quantum BAEKS scheme as so far.

In 2019, A lattice-based forward secure public key with keyword search (FS-PEKS) scheme is proposed by Zhang et al., which utilized lattice basis delegation to update the secret key [27]. After that, Yu et al. introduced the binary tree structure, minimal cover set technique and lattice basis extension to construct an efficient FS-PEKS scheme over lattice [25]. Then, Yang et al. presented a forward secure identity-based PEKS, namely FS-IBEKS, which instantiated two schemes over lattice to ensure security in the ROM and standard model, respectively [28]. For PAEKS primitive, Xu et al. constructed a forward secure PAEKS over lattice, namely FS-PAEKS, to achieve the IND-CKA and IND-IKGA secure [15]. However, there does not exist BAEKS scheme with forward security till now.

To sum up, there exists a valuable requirement to construct a BAEKS scheme and extend it to FS-BAEKS for achieving the multi-receiver support, IKGAs-resilience, quantum-resistance, and forward security. Table 1 compares aforementioned properties between our proposed BAEKS & FS-BAEKS schemes with the current state-of-art

TABLE 2
Glossary

Acronym	Definition
$[d]$	the number set $\{1, \cdots, d\}$
$i = [d]$	the iteration of each element in set $\{1, 2, \cdots, d\}$ with variable i
l	the number of data receivers
k	the length of a keyword
τ	the level number of binary tree
T	the number of time period, where $T = 2^\tau$
$\mathcal{W}$	the keyword set
$\mathbf{ck}$	the keyword owned by data sender
$\mathbf{tk}$	the keyword to be searched by specific data receiver
λ	the security parameter
pp	the public parameter
$(\mathbf{pk}_S, \mathbf{sk}_S)$	the public & secret keys of data sender
$(\mathbf{pk}_{R,i}, \mathbf{sk}_{R,i})$	the public & secret keys of data receiver i, where $i \in [l]$
$(\mathbf{pk}_{R,i}, \mathbf{sk}_{R,i,t})$	the public key & secret key of data receiver i with time period t
CT	the keyword ciphertext
CT_t	the keyword ciphertext with time period t
TD	the search trapdoor calculated by data receiver i, where $i \in [l]$
TD_t	the search trapdoor calculated by data receiver i with time period t, where $i \in [l]$

PEKS/PAEKS/BAEKS schemes.

3 Preliminaries

We provide a concise summary of the notations, including lattice, discrete Gaussian distribution, LWE & SIS hardness, lattice basis sampling and extension lemmas, and reject sampling lemma. Table 2 clarifies the acronyms and descriptions utilized in this paper.

Definition 1. [34] Suppose a matrix $\mathbf{M} = (\mathbf{m}_1, \mathbf{m}_2, \cdots, \mathbf{m}_m)$ is composed of m linearly independent vectors, the lattice Λ is defined as:

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

where $\mathbf{M}$ is a lattice basis of Λ.

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

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

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

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

Definition 3. Suppose a parameter $\sigma \in \mathbb{R}^+$, a center $\mathbf{c} \in \mathbb{Z}^m$, and any vector $\mathbf{v} \in \mathbb{Z}^m$, the discrete Gaussian distribution over Λ is defined as:

$$\mathcal{D}_{\Lambda,\sigma,\mathbf{c}}(\mathbf{v}) = \frac{\rho_{\sigma,\mathbf{c}}(\mathbf{v})}{\rho_{\sigma,\mathbf{c}}(\Lambda)},$$

for $\forall \mathbf{v} \in \Lambda$, where $\rho_{\sigma,\mathbf{c}}(\mathbf{v}) = \exp(-\pi \frac{\|\mathbf{v}-\mathbf{c}\|^2}{\sigma^2})$ and $\rho_{\sigma,\mathbf{c}}(\Lambda) = \sum_{\mathbf{v}\in\Lambda} \rho_{\sigma,\mathbf{c}}(\mathbf{v})$.

Definition 4. [36] Suppose several positive integer n,m,q, and an error distribution $\chi = \Psi_\alpha$, the $\text{LWE}_{n,m,q,\chi}$ hardness is defined as distinguishing two pairings $(\mathbf{M}, \mathbf{M}^\top\mathbf{s} + \mathbf{e})$ and $(\mathbf{M}, \mathbf{v})$, where $\mathbf{M} \leftarrow \mathbb{Z}_q^{n\times m}, \mathbf{s} \leftarrow \mathbb{Z}_q^n, \mathbf{e} \leftarrow \chi^m$, and $\mathbf{v} \leftarrow \mathbb{Z}_q^m$.

Definition 5. [35] Suppose several positive integer n,m,q, the $\text{SIS}_{n,m,q,\beta}$ hardness is defined as finding a non-zero vector $\mathbf{v} \in \mathbb{Z}^m \setminus \{\mathbf{0}\}$ s.t. $\mathbf{A}\mathbf{v} = \mathbf{0}$ and $\|\mathbf{v}\| \leq \beta$, where $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, and $\beta \geq \sqrt{m}q^{n/m}$.

Lemma 1. [37] Suppose a lattice Λ and its lattice basis $\mathbf{T_A}$, we obtain:

$$\Pr[\|\mathbf{v}\| > \sigma\sqrt{m} : \mathbf{v} \leftarrow \mathcal{D}_{\Lambda,\sigma}] \leq \mathsf{negl}(m),$$

where $\sigma \geq \|\widetilde{\mathbf{T_A}}\| \cdot \omega(\sqrt{\log m})$.

Lemma 2. [37] Suppose three positive integers n, m, q, where $q \geq 2$, and $m \geq 5n\log q$. After input several positive integers n, m, q, the probabilistic polynomial time (PPT) algorithm $\mathsf{TrapGen}(n, m, q)$ will calculate an uniform matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ together with a lattice basis $\mathbf{T_A} \in \mathbb{Z}_q^{m\times m}$ for $\Lambda_q^\perp(\mathbf{A})$, where $\mathbf{A}$ is statistically close to uniform distribution on $\mathbb{Z}^{n\times m}$ and $\|\widetilde{\mathbf{T_A}}\| \leq m\omega(\sqrt{\log m})$.

Lemma 3. [37] Suppose three positive integers n, m, q, where $q \geq 2$, and $m \geq 2n\log q$. After input a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, a lattice basis $\mathbf{T_A} \in \mathbb{Z}_q^{m\times m}$ for $\Lambda_q^\perp(\mathbf{A})$, and a Gaussian parameter $\sigma \leq \|\widetilde{\mathbf{T}}_\mathbf{A}\| \cdot \omega(\sqrt{\log m})$, the PPT algorithm $\mathsf{SamplePre}(\mathbf{A}, \mathbf{T_A}, \mathbf{u}, \sigma)$ will calculate a vector $\mathbf{e} \in \mathbb{Z}_q^m$ statistically close to $\mathcal{D}_{\Lambda_q^u(\mathbf{A}),\sigma}$, such that $\mathbf{A}\mathbf{e} = \mathbf{u} \bmod q$.

Suppose four positive integers n, m, q, k, a matrix $\mathbf{A} = (\mathbf{A}_1 \mid \cdots \mid \mathbf{A}_k) \in \mathbb{Z}_q^{n\times km}$, and a set $\mathcal{M} = \{i_1, i_2, \cdots, i_j\} \subset [k]$, we set $\mathbf{A}_\mathcal{M} := (\mathbf{A}_{i_1} \mid \mathbf{A}_{i_2} \mid \cdots \mid \mathbf{A}_{i_j}) \in \mathbb{Z}_q^{n\times jm}$. Then, we introduce the Lemma 4 and 5 as follows:

Lemma 4. [38] Suppose four positive integers n, m, q, k, where $q \geq 2$, and $m \geq 2n\log q$. After input a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times km}$, a lattice basis $\mathbf{T}_{\mathbf{A}_\mathcal{M}}$ for $\Lambda_q^\perp(\mathbf{A}_\mathcal{M})$, a set $\mathcal{M} \subset [k]$, and a Gaussian parameter $L \geq \|\widetilde{\mathbf{T}_{\mathbf{A}_\mathcal{M}}}\| \cdot \sqrt{km} \cdot \omega(\sqrt{\log km})$, the PPT algorithm $\mathsf{SampleBasis}(\mathbf{A}, \mathbf{T}_{\mathbf{A}_\mathcal{M}}, \mathcal{M}, L)$ will calculate a matrix $\mathbf{T}'_\mathbf{A}$, where $\mathbf{T}'_\mathbf{A}$ is a lattice basis of $\Lambda_q^\perp(\mathbf{A})$ and $\|\widetilde{\mathbf{T}'_\mathbf{A}}\| \leq L$ with overwhelming probability.

Lemma 5. [38] Suppose four positive integers n, m, q, k, where $q \geq 2$, and $m \geq 2n\log q$. After input a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times km}$, a lattice basis $\mathbf{T}_{\mathbf{A}_\mathcal{M}}$ for $\Lambda_q^\perp(\mathbf{A}_\mathcal{M})$, a set $\mathcal{M} \subset [k]$, a vector $\mathbf{u} \in \mathbb{Z}_q^n$, and a Gaussian parameter $\sigma \geq \|\widetilde{\mathbf{T}_{\mathbf{A}_\mathcal{M}}}\| \cdot \omega(\sqrt{\log km})$, the PPT algorithm $\mathsf{GenSamplePre}(\mathbf{A}, \mathbf{T}_{\mathbf{A}_\mathcal{M}}, \mathcal{M}, \mathbf{u}, \sigma)$ will output a vector $\mathbf{e} \in \mathbb{Z}^{km}$ statistically close in $\mathcal{D}_{\Lambda_q^u(\mathbf{A}),\sigma}$, such that $\mathbf{A}\mathbf{e} = \mathbf{u} \bmod q$.

Lemma 6. [39] Suppose four positive integers n, m, m', q, two matrices $\mathbf{A} \in \mathbb{Z}^{n\times m}$, $\mathbf{A}' \in \mathbb{Z}^{n\times m'}$. After input $\mathbf{A}'' = (\mathbf{A} \mid \mathbf{A}') \in \mathbb{Z}_q^{n\times(m+m')}$, and a basis $\mathbf{T_A} \in \mathbb{Z}_q^{m\times m}$ for $\Lambda_q^\perp(\mathbf{A})$, the deterministic polynomial time (DPT) algorithm $\mathsf{ExtBasis}(\mathbf{A}'', \mathbf{S})$ will calculate a lattice basis $\mathbf{T}_{\mathbf{A}''}$ for $\Lambda_q^\perp(\mathbf{A}'') \subseteq \mathbb{Z}_q^{m\times m''}$, where $\|\widetilde{\mathbf{T_A}}\| = \|\widetilde{\mathbf{T}_{\mathbf{A}''}}\|$, $m'' = m + m'$.

Lemma 7. [40] Suppose a vector space $W = \{\mathbf{w} \in \mathbb{Z}^m : \|\mathbf{w}\| \leq T\}$, a mapping $h : W \to \mathbb{R}$, a constant M, and a Gaussian parameter $\sigma = \omega(T\sqrt{\log m})$, where $\mathbf{w} \leftarrow h$, the following two distributions are defined as:

1) For $\mathbf{w} \leftarrow h$, $\mathbf{u} \leftarrow \mathcal{D}_{\sigma}^{m}$, obtain $(\mathbf{u}, \mathbf{w})$ with probability $\frac{1}{M}$.
2) For $\mathbf{w} \leftarrow h$, $\mathbf{u} \leftarrow \mathcal{D}_{\mathbf{w},\sigma}^{m}$, obtain $(\mathbf{u}, \mathbf{w})$ with probability $\min(\frac{\mathcal{D}_{\sigma}^{m}}{M \cdot \mathcal{D}_{\mathbf{w},\sigma}^{m}}, 1)$.

4 Framework Description

The system models, formal definitions, and security models of our BAEKS scheme are described in this sector.

4.1 System Models

The system models of our BAEKS scheme are illustrated in Fig. 2, which contains four participating entities: key generation center, data sender, data receivers, and cloud server.

1) **Key generation center (KGC)**: KGC is charged with executing the **Setup** algorithm to obtain the public parameters and calculate the public & secret keys for data sender and data receivers.
2) **Data sender**: Data sender owns massive data from different industries (e.g., medical data, logistics data, research data, etc.), extracts and encrypts the keywords from these data with its own secret key and a set of data receivers' public keys to calculate keyword ciphertext, and sends them to the cloud server.
3) **Data receivers**: Data receivers consist of users from different industries (e.g., doctor, manufacturer, researcher, etc.). To facilitate our BAEKS implementation, we assume that there are at most l data receivers. When a data receiver has a search requirement (e.g., the doctor in Fig. 2), it generates a search trapdoor by calling the **Trapdoor** algorithm, and uploads it to the cloud server. If there exists a matching ciphertext, it receives the search result from the cloud server.
4) **Cloud server (CS)**: After receiving the keyword ciphertext from a data sender and the search trapdoor from a specific data receiver, CS executes the **Test** algorithm to match the keyword ciphertext and the search trapdoor. If the match is successful, CS sends the search result to the data receiver. Otherwise, CS sends Null to it.

4.2 Formal Definitions

Our BAEKS scheme contains six algorithms $\Pi_{\text{BAEKS}} = (\textbf{Setup}, \textbf{KeyGen}_S, \textbf{KeyGen}_R, \textbf{BAEKS}, \textbf{Trapdoor}, \textbf{Test})$, the formal definitions of these algorithms is described as:

- $pp \leftarrow \textbf{Setup}(1^{\lambda})$: After inputting a security parameter λ, this algorithm publishes a public parameter pp.
- $(\mathbf{pk}_S, \mathbf{sk}_S) \leftarrow \textbf{KeyGen}_S(pp, \mathbf{T_A})$: After inputting the public parameter pp and a basis $\mathbf{T_A}$, this PPT algorithm publishes the public & secret keys $(\mathbf{pk}_S, \mathbf{sk}_S)$ of a data sender.
- $(\mathbf{pk}_{R,i}, \mathbf{sk}_{R,i}) \leftarrow \textbf{KeyGen}_R(pp)$: For $i = [l]$, after inputting the public parameter pp, this PPT algorithm publishes the public & secret keys $(\mathbf{pk}_{R,i}, \mathbf{sk}_{R,i})$ of the data receiver i.
- $\mathrm{CT} \leftarrow \textbf{BAEKS}(pp, \mathbf{ck}, \mathbf{sk}_S, \{\mathbf{pk}_{R,1}, \mathbf{pk}_{R,2}, \cdots, \mathbf{pk}_{R,l}\})$: After inputting the public parameter pp, a keyword $\mathbf{ck} \in \mathcal{W}$, a secret key $\mathbf{sk}_S$ of data sender, a set of data receivers' public keys $\{\mathbf{pk}_{R,1}, \mathbf{pk}_{R,2}, \cdots, \mathbf{pk}_{R,l}\}$, the data sender invokes this PPT algorithm to get the ciphertext CT corresponding to $\mathbf{ck}$.
- $\mathrm{TD} \leftarrow \textbf{Trapdoor}(pp, \mathbf{tk}, \mathbf{pk}_S, \{\mathbf{pk}_{R,1}, \mathbf{pk}_{R,2}, \cdots, \mathbf{pk}_{R,l}\}, \mathbf{sk}_{R,\gamma})$: After inputting the public parameter pp, a keyword $\mathbf{tk} \in \mathcal{W}$, a public key $\mathbf{pk}_S$ of data sender, a set of data receivers' public keys $\{\mathbf{pk}_{R,1}, \mathbf{pk}_{R,2}, \cdots, \mathbf{pk}_{R,l}\}$, and a secret key $\mathbf{sk}_{R,\gamma}$ of data receiver γ, the data receiver γ invokes this PPT algorithm to get the trapdoor TD corresponding to $\mathbf{tk}$.
- 1 or $0 \leftarrow \textbf{Test}(\mathrm{CT}, \mathrm{TD})$: The server processes this DPT algorithm to test if CT and TD correspond to the same keyword. If yes, it outputs 1. Otherwise, it outputs 0.

Definition 6. For any $pp \leftarrow \textbf{Setup}(1^{\lambda})$, $(\mathbf{pk}_S, \mathbf{sk}_S) \leftarrow \textbf{KeyGen}_S(pp, \mathbf{T_A})$, $(\mathbf{pk}_{R,i}, \mathbf{sk}_{R,i}) \leftarrow \textbf{KeyGen}_R(pp)$, $\mathrm{CT} \leftarrow \textbf{BAEKS}(pp, \mathbf{ck}, \mathbf{sk}_S, \{\mathbf{pk}_{R,1}, \mathbf{pk}_{R,2}, \cdots, \mathbf{pk}_{R,l}\})$, and $\mathrm{TD} \leftarrow \textbf{Trapdoor}(pp, \mathbf{tk}, \mathbf{pk}_S, \{\mathbf{pk}_{R,1}, \mathbf{pk}_{R,2}, \cdots, \mathbf{pk}_{R,l}\}, \mathbf{sk}_{R,\gamma})$, we say that our BAEKS primitive satisfies correctness, if $\Pr[\textbf{Test}(\mathrm{CT}, \mathrm{TD}) = 1)] = 1$ with a non-negligible probability when the keyword $\mathbf{ck} = \mathbf{tk}$.

4.3 Security Models

In this section, we define two security models of BAEKS scheme, namely ciphertext indistinguiability against chosen keyword attacks (IND-CKA), and trapdoor unforgability against insider keyword guessing attacks (UF-IKGA).

4.3.1 IND-CKA security

For the first part, we define the IND-CKA security model $\mathbf{Exp}_{\text{BAEKS},\mathcal{A}}^{\text{IND-CKA}}(\lambda)$ as follows:

1) **Setup**: Given a security parameter λ and many LWE instances, a challenger $\mathcal{C}$ invokes the $\textbf{Setup}(1^{\lambda})$ algorithm to calculate pp. Then, $\mathcal{C}$ processes the $\textbf{KeyGen}_S(pp, \mathbf{T_A})$ and $\textbf{KeyGen}_R(pp)$ algorithms to obtain a challenge sender's public & secret keys $(\mathbf{pk}_S^*, \mathbf{sk}_S^*)$ and the challenger receivers' public & secret keys $(\mathbf{pk}_{R,i}^*, \mathbf{sk}_{R,i}^*)$, where $i = [l]$, respectively. Then, $\mathcal{C}$ returns pp, $\mathbf{pk}_S^*$, $\{\mathbf{pk}_{R,1}^*, \mathbf{pk}_{R,2}^*, \cdots, \mathbf{pk}_{R,l}^*\}$ to the adversary $\mathcal{A}$.
2) **Phase 1**: $\mathcal{A}$ can adaptively perform three oracles in polynomial times.
   a) Hash Queries $\mathcal{O}_{H_1}$: Given a keyword $\mathbf{ck} \in \mathcal{W}$ from $\mathcal{A}$, $\mathcal{C}$ maintains a list L_{H_1} and searches $\mathbf{ck}$ in it, and then returns the answer to $\mathbf{A}$.
   b) Ciphertext Queries $\mathcal{O}_{\mathrm{CT}}$: Given a ciphertext keyword $\mathbf{ck} \in \mathcal{W}$ and a set of data receivers' public keys $\{\mathbf{pk}_{R,1}, \mathbf{pk}_{R,2}, \cdots, \mathbf{pk}_{R,l}\}$ from $\mathcal{A}$, $\mathcal{C}$ invokes the $\textbf{BAEKS}(pp, \mathbf{ck}, \mathbf{sk}_S^*, \{\mathbf{pk}_{R,1}, \mathbf{pk}_{R,2}, \cdots, \mathbf{pk}_{R,l}\})$ algorithm to calculate the ciphertext CT and sends it to $\mathcal{A}$.
   c) Trapdoor Queries $\mathcal{O}_{\mathrm{TD}}$: Given a keyword $\mathbf{tk} \in \mathcal{W}$, a public key $\mathbf{pk}_S$ of data sender, a set of data receivers' public keys $\{\mathbf{pk}_{R,1}^*, \mathbf{pk}_{R,2}^*, \cdots, \mathbf{pk}_{R,l}^*\}$ and $\gamma \in [l]$ from $\mathcal{A}$, $\mathcal{C}$ invokes the $\textbf{Trapdoor}(pp, \mathbf{tk}, \mathbf{pk}_S, \{\mathbf{pk}_{R,1}^*, \mathbf{pk}_{R,2}^*, \cdots, \mathbf{pk}_{R,l}^*\}, \mathbf{sk}_{R,\gamma}^*)$ to calculate the trapdoor TD and returns it to $\mathcal{A}$.
3) **Challenge**: $\mathcal{A}$ chooses $\mathbf{ck}_0, \mathbf{ck}_1 \in \mathcal{W}$ which have not been queried in **Phase 1** as two challenge ciphertext keywords, and sends them to $\mathcal{C}$. After that,

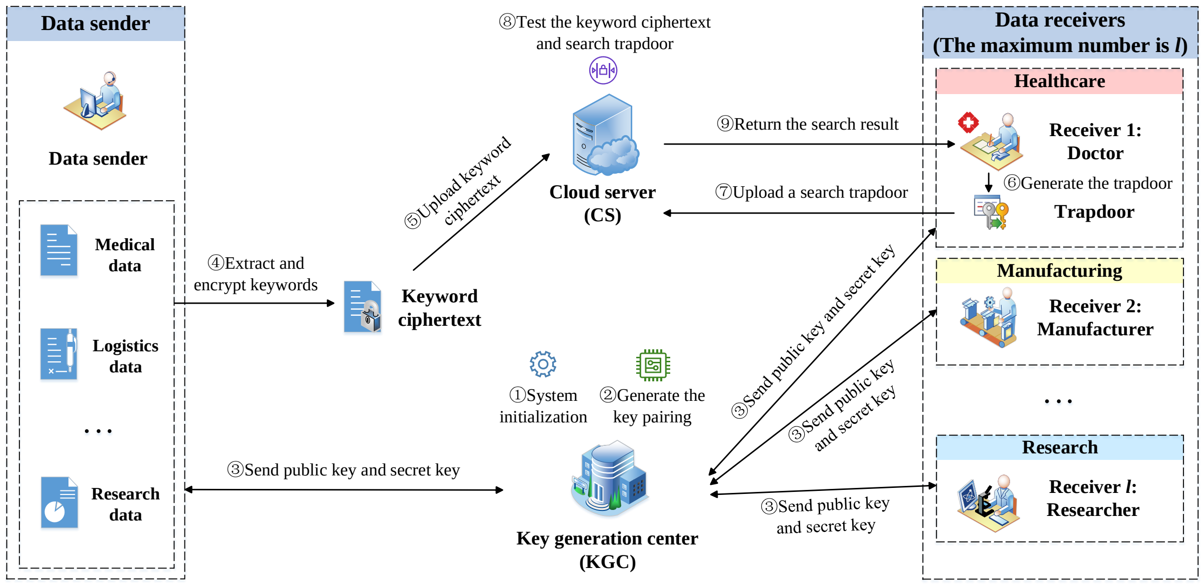


Fig. 2. System models of our BAEKS scheme for cloud storage.

$\mathcal{C}$ selects a random bit $\xi \in \{0,1\}$ and invokes the $\mathbf{BAEKS}(\mathbf{ck}_\xi, \mathbf{sk}_S^*, \{\mathbf{pk}_{R,1}^*, \mathbf{pk}_{R,2}^*, \cdots, \mathbf{pk}_{R,l}^*\})$ algorithm to obtain a challenge ciphertext CT_ξ. Finally, $\mathcal{C}$ returns CT_ξ to $\mathcal{A}$.

4) **Phase 2**: $\mathcal{A}$ executes these queries as above, neither $\mathbf{ck}_0$ nor $\mathbf{ck}_1$ can be queried.
5) **Guess**: A guess bit $\xi' \in \{0,1\}$ is outputted by $\mathcal{A}$. If $\xi' = \xi$, we say that $\mathcal{A}$ wins this game.

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

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

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

4.3.2 UF-IKGA security

For the second part, we define the UF-IKGA security model $\mathbf{Exp}_{\text{BAEKS},\mathcal{A}}^{\text{UF-IKGA}}(\lambda)$ as follows:

1) **Setup**: This part is same as the corresponding part in $\mathbf{Exp}_{\text{BAEKS},\mathcal{A}}^{\text{IND-CKA}}(\lambda)$.
2) **Phase 1**: $\mathcal{A}$ can adaptively perform three oracles in polynomial times.
   a) Hash Queries $\mathcal{O}_{H_2}$: Given a tuple $(\mathbf{c}_1, b)$ corresponding to the keyword ciphertext CT from $\mathcal{A}$, $\mathcal{C}$ maintains a list L_{H_2} and searches $(\mathbf{c}_1, b)$ in it, and then returns the answer to $\mathbf{A}$.
   b) Ciphertext Queries $\mathcal{O}_{\mathrm{CT}}$: This part is same as the corresponding part in $\mathbf{Exp}_{\text{BAEKS},\mathcal{A}}^{\text{IND-CKA}}(\lambda)$.
   c) Trapdoor Queries $\mathcal{O}_{\mathrm{TD}}$: This part is same as the corresponding part in $\mathbf{Exp}_{\text{BAEKS},\mathcal{A}}^{\text{IND-CKA}}(\lambda)$.
3) **Forgery**: $\mathcal{A}$ selects $\gamma \in [l]$ and sends it to $\mathcal{C}$. $\mathcal{C}$ invokes the $\mathbf{Trapdoor}(\mathbf{ck}^*, \mathbf{pk}_S^*, \{\mathbf{pk}_{R,1}^*, \mathbf{pk}_{R,2}^*, \cdots, \mathbf{pk}_{R,l}^*\}, \mathbf{sk}_{R,\gamma}^*)$ algorithm to obtain TD^*, and returns it to $\mathcal{A}$. Then, $\mathcal{A}$ forges a ciphertext CT^* corresponding to the challenge keyword $\mathbf{ck}^*$, and wins this game if the $\mathbf{Test}(\mathrm{CT}^*, \mathrm{TD}^*)$ algorithm publishes 1.

We define the advantage of $\mathcal{A}$ to win the above game $\mathbf{Exp}_{\text{BAEKS},\mathcal{A}}^{\text{UF-IKGA}}(\lambda)$ as:

$$\mathrm{Adv}_{\text{BAEKS},\mathcal{A}}^{\text{UF-IKGA}}(\lambda) = |\Pr[\mathbf{Test}(\mathrm{CT}^*, \mathrm{TD}^*) = 1]|.$$

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

5 Our Proposed BAEKS Scheme

We describe our proposed BAEKS scheme in this section, including the concrete construction, parameters setting and correctness analysis, and security analysis.

5.1 Concrete Construction

- $\mathbf{Setup}(1^\lambda)$: A security parameter 1^λ is inputted by the KGC, and then the public parameter pp is outputted according to the following procedures.
  1) Let the system parameters n, m, q, L, σ, k, and l.
  2) Invoke $(\mathbf{A}, \mathbf{T_A}) \leftarrow \mathsf{TrapGen}(n, m, q)$ to generate a uniformly matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and a basis $\mathbf{T_A} \in \mathbb{Z}^{m\times m}$ for $\Lambda_q^\perp(\mathbf{A})$.
  3) Choose a vector $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_q^n$ uniformly.
  4) Define two hash functions $H_1 : \{0,1\}^k \to \mathbb{Z}_q^{n\times m}$, and $H_2 : \mathbb{Z}_q^n \times \mathbb{Z}_q^{(l+1)m} \times \{0,1\} \to \{-1,0,1\}^m$.
  5) Output $pp := (n, m, q, L, \sigma, k, l, \mathbf{A}, \mathbf{u}, H_1, H_2)$ as the public parameter.
- $\mathbf{KeyGen}_S(pp, \mathbf{T_A})$: The KGC inputs a public parameter pp and a basis $\mathbf{T_A}$ and then returns the public & secret keys $(\mathbf{pk}_S, \mathbf{sk}_S)$ to a data sender according to the following procedures.
  1) Invoke $(\mathbf{A}_S, \mathbf{T}_{\mathbf{A}_S}) \leftarrow \mathsf{TrapGen}(n, m, q)$ to generate a 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) Parse the matrix $\mathbf{A}_S = (\mathbf{a}_{S,1}, \mathbf{a}_{S,2}, \cdots, \mathbf{a}_{S,m})$, which each vector $\mathbf{a}_{S,i} \in \mathbb{Z}^n$ for $i = [m]$.
3) For $i = [m]$, sample a vector $\mathbf{s}_i \in \mathbb{Z}_q^m$ as $\mathbf{s}_i \leftarrow \mathsf{SamplePre}(\mathbf{A}, \mathbf{T}_\mathbf{A}, \mathbf{a}_{S,i}, \sigma)$, where $\mathbf{s}_i$ s.t. $\mathbf{A}\mathbf{s}_i = \mathbf{a}_{S,i} \bmod q$ and $\mathbf{s}_i$ is statistically distributed in $\mathcal{D}^m_{\Lambda_q^{\mathbf{a}_{S,i}}(\mathbf{A}),\sigma}$.
4) Let a matrix $\mathbf{S} = (\mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s}_m) \in \mathbb{Z}^{m\times m}$, where $\mathbf{AS} = \mathbf{A}_s \bmod q$.
5) Output $\mathbf{pk}_S := \mathbf{A}_S$ and $\mathbf{sk}_S := (\mathbf{T}_{\mathbf{A}_S}, \mathbf{S})$ as the public & secret keys of data sender.

- **KeyGen**$_R(pp)$: For $i = [l]$, the KGC inputs a public parameter pp and then returns the public & secret keys $(\mathbf{pk}_{R,i}, \mathbf{sk}_{R,i})$ to the data receiver i according to the following procedures.

1) Invoke $(\mathbf{A}_{R,i}, \mathbf{T}_{\mathbf{A}_{R,i}}) \leftarrow \mathsf{TrapGen}(n, m, q)$ to generate a uniformly matrix $\mathbf{A}_{R,i} \in \mathbb{Z}_q^{n\times m}$ and a basis $\mathbf{T}_{\mathbf{A}_{R,i}} \in \mathbb{Z}^{m\times m}$ for $\Lambda_q^\perp(\mathbf{A}_{R,i})$.
2) Output $\mathbf{pk}_{R,i} := \mathbf{A}_{R,i}$ and $\mathbf{sk}_{R,i} := \mathbf{T}_{\mathbf{A}_{R,i}}$ as the public & secret keys of data receiver.

- **BAEKS**$(pp, \mathbf{ck}, \mathbf{sk}_S, \{\mathbf{pk}_{R,1}, \mathbf{pk}_{R,2}, \cdots, \mathbf{pk}_{R,l}\})$: A data sender inputs a public parameter pp, a keyword $\mathbf{ck} \in \mathcal{W}$, a sender's secret key $\mathbf{sk}_S$, a set of receivers' public keys $\{\mathbf{pk}_{R,1}, \mathbf{pk}_{R,2}, \cdots, \mathbf{pk}_{R,l}\}$, and then performs the following procedures.

1) Let a matrix $\mathbf{A}_R = (\mathbf{A}_{R,1} \mid \mathbf{A}_{R,2} \mid \cdots \mid \mathbf{A}_{R,l}) \in \mathbb{Z}_q^{n\times lm}$.
2) Calculate a matrix $\mathbf{A}_\mathbf{ck} = (\mathbf{A}_R \mid H_1(\mathbf{ck})) \in \mathbb{Z}^{n\times(l+1)m}$.
3) Select a random vector $\mathbf{v} \xleftarrow{\$} \mathbb{Z}_q^n$ uniformly, a random number $b \in \{0,1\}$, two noise vectors $\mathbf{x}_0 \xleftarrow{\$} \chi^{lm}$, $\mathbf{x}_1 \xleftarrow{\$} \chi^m$, and a noise number $x \xleftarrow{\$} \chi$.
4) Calculate a vector $\mathbf{c}_1 = \mathbf{A}_\mathbf{ck}^\top \mathbf{v} + (\mathbf{x}_0^\top \mid \mathbf{x}_1^\top)^\top \in \mathbb{Z}_q^{(l+1)m}$, and a number $c_2 = \mathbf{u}^\top\mathbf{v} + x + b\cdot\lfloor\frac{q}{2}\rfloor \in \mathbb{Z}_q$.
5) Select a vector $\mathbf{y} \xleftarrow{\$} \mathbb{Z}_q^m$ in $\mathcal{D}_\sigma^m$ uniformly.
6) Calculate a vector $\boldsymbol{\eta}_1 = H_2(\mathbf{Ay} \bmod q, \mathbf{c}_1, b) \in \{-1,0,1\}^m$ and another vector $\boldsymbol{\eta}_2 = \mathbf{S}\boldsymbol{\eta}_1 + \mathbf{y} \in \mathbb{Z}_q^m$ with the probability $\min(\frac{\mathcal{D}_\sigma^m}{M\cdot\mathcal{D}^m_{\mathbf{S}\boldsymbol{\eta}_1,\sigma}}, 1)$.
7) Output $\mathrm{CT} := (\mathbf{c}_1, c_2, \boldsymbol{\eta}_1, \boldsymbol{\eta}_2)$ as the ciphertext corresponding to the keyword $\mathbf{ck}$.

- **Trapdoor**$(pp, \mathbf{tk}, \mathbf{pk}_S, \{\mathbf{pk}_{R,1}, \mathbf{pk}_{R,2}, \cdots, \mathbf{pk}_{R,l}\}, \mathbf{sk}_{R,\gamma})$: A data receiver $\gamma \in [l]$ inputs a public parameter pp, a keyword $\mathbf{tk} \in \mathcal{W}$, a sender's public key $\mathbf{pk}_S$, a set of receivers' public keys $\{\mathbf{pk}_{R,1}, \mathbf{pk}_{R,2}, \cdots, \mathbf{pk}_{R,l}\}$, and secret keys $\mathbf{sk}_{R,\gamma}$ with receiver γ, and then performs the following procedures.

1) Calculate $\mathbf{A}_{\mathbf{tk},\gamma} = (\mathbf{A}_{R,\gamma} \mid H_1(\mathbf{tk})) \in \mathbb{Z}_q^{n\times 2m}$ and $\mathbf{A}_\mathbf{tk} = (\mathbf{A}_{R,1} \mid \cdots \mid \mathbf{A}_{R,l} \mid H_1(\mathbf{tk}))$.
2) Invoke $\mathbf{T}_{\mathbf{A}_{\mathbf{tk},\gamma}} \leftarrow \mathsf{SampleBasis}(\mathbf{A}_{\mathbf{tk},\gamma}, \mathbf{T}_{\mathbf{A}_{R,\gamma}}, \{1\}, L)$ to obtain a basis $\mathbf{T}_{\mathbf{A}_{\mathbf{tk},\gamma}} \in \mathbb{Z}^{2m\times 2m}$ for $\Lambda_q^\perp(\mathbf{A}_{\mathbf{tk},\gamma})$.
3) Sample a vector $\boldsymbol{\varepsilon}_i \in \mathbb{Z}^{(l+1)m}$ as $\boldsymbol{\varepsilon} \leftarrow \mathsf{GenSamplePre}(\mathbf{A}_\mathbf{tk}, \mathbf{T}_{\mathbf{A}_{\mathbf{tk},\gamma}}, \{\gamma, l+1\}, \mathbf{u}, \sigma)$, where $\boldsymbol{\varepsilon}$ s.t. $\mathbf{A}_\mathbf{tk}\boldsymbol{\varepsilon} = \mathbf{u} \bmod q$ and $\boldsymbol{\varepsilon}$ is statistically distributed in $\mathcal{D}^{(l+1)m}_{\Lambda_q^\mathbf{u}(\mathbf{A}_\mathbf{tk})}$.
4) Output $\mathrm{TD} := (\boldsymbol{\varepsilon}, \mathbf{pk}_S)$ as the trapdoor corresponding to the keyword $\mathbf{tk}$.

- **Test**$(\mathrm{CT}, \mathrm{TD})$: The cloud server inputs the ciphertext CT together with the trapdoor TD, and then processes the following procedures.

1) Parse $\mathrm{CT} = (\mathbf{c}_1, c_2, \boldsymbol{\eta}_1, \boldsymbol{\eta}_2)$ and $\mathrm{TD} = (\boldsymbol{\varepsilon}, \mathbf{pk}_S = \mathbf{A}_S)$.
2) Calculate a number $d = c_2 - \boldsymbol{\varepsilon}^\top\mathbf{c}_1 \in \mathbb{Z}_q$. If $\left|d - \lfloor\frac{q}{2}\rfloor\right| < \lfloor\frac{q}{4}\rfloor$, set $b' = 1$. Otherwise, set $b' = 0$.
3) Check $\|\boldsymbol{\eta}_2\| \overset{?}{\leq} 2\sigma\sqrt{m}$ and $\boldsymbol{\eta}_1 \overset{?}{=} H_2(\mathbf{A}\boldsymbol{\eta}_2 - \mathbf{A}_S\boldsymbol{\eta}_1, \mathbf{c}_1, b')$. If these two conditions are satisfied, output 1. Otherwise, output 0.

5.2 Parameters Setting and Correctness Analysis

The involved parameters of our BAEKS scheme is set as follows to fulfill the security requirements.

- $m \geq \lceil 5n\log q\rceil$ for the $\mathsf{TrapGen}$ lemma.
- $\sigma \geq km\cdot\omega(\log km)$ for $\mathsf{SamplePre}$ and $\mathsf{GenSamplePre}$ lemmas.
- $L \geq \mathcal{O}(m^{1.5})\cdot\omega(\log km)$ for SampleBasis lemma.
- $\alpha q > 2\sqrt{n}$ for LWE hardness.
- $q\alpha\sigma(l+1)m\omega(\sqrt{\log[(l+1)m]}) + \mathcal{O}(\sigma(l+1)m) < \frac{q}{5}$ for the correctness.

Based on the above parameter settings, we analyze the correctness of our BAEKS. We set that the data sender owns its the public & secret keys $(\mathbf{pk}_S := \mathbf{A}_S, \mathbf{sk}_S := (\mathbf{T}_{\mathbf{A}_S}, \mathbf{S}))$, a keyword $\mathbf{ck} \in \mathcal{W}$, and its ciphertext $\mathrm{CT} = (\mathbf{c}_1, c_2, \eta_\mathbf{1}, \eta_\mathbf{2})$. Moreover, the data receiver γ owns its public keys and secret keys $(\mathbf{pk}_{R,\gamma} := \mathbf{A}_{R,\gamma}, \mathbf{sk}_{R,\gamma} := \mathbf{T}_{A_{R,\gamma}})$, and the searched keyword $\mathbf{tk} \in \mathcal{W}$, and corresponding search trapdoor $\mathrm{TD} = (\boldsymbol{\varepsilon}, \mathbf{pk}_S)$.

On the one hand, for the condition $\left|d - \lfloor\frac{q}{2}\rfloor\right| < \lfloor\frac{q}{4}\rfloor$ in **Test** algorithm.

- If $\mathbf{ck} = \mathbf{tk}$, we have:

$$\begin{aligned}
d &= c_2 - \boldsymbol{\varepsilon}^\top\mathbf{c}_1\\
&= \mathbf{u}^\top\mathbf{v} + x + b\cdot\lfloor\frac{q}{2}\rfloor - \boldsymbol{\varepsilon}^\top(\mathbf{A}_\mathbf{ck}^\top\mathbf{v} + (\mathbf{x}_0^\top \mid \mathbf{x}_1^\top)^\top)\\
&= \mathbf{u}^\top\mathbf{v} + x + b\cdot\lfloor\frac{q}{2}\rfloor - \boldsymbol{\varepsilon}^\top(\mathbf{A}_\mathbf{tk}^\top\mathbf{v} + (\mathbf{x}_0^\top \mid \mathbf{x}_1^\top)^\top)\\
&= \mathbf{u}^\top\mathbf{v} + x + b\cdot\lfloor\frac{q}{2}\rfloor - \mathbf{u}^\top\mathbf{v} - ((\mathbf{x}_0^\top \mid \mathbf{x}_1^\top)\varepsilon_i)^\top\\
&= b\cdot\lfloor\frac{q}{2}\rfloor + x - \boldsymbol{\varepsilon}_i^\top(\mathbf{x}_0^\top \mid \mathbf{x}_1^\top)^\top,
\end{aligned}$$

where $x - \boldsymbol{\varepsilon}_i^\top(\mathbf{x}_0^\top \mid \mathbf{x}_1^\top)^\top$ is an error term, and it is bounded by:

$$\begin{aligned}
&|x - \boldsymbol{\varepsilon}_i^\top(\mathbf{x}_0^\top \mid \mathbf{x}_1^\top)^\top| \leq |x| + |(\mathbf{x}_0^\top \mid \mathbf{x}_1^\top)^\top|\\
&\leq q\alpha\sigma(l+1)m\omega(\sqrt{\log[(l+1)m]}) + \mathcal{O}(\sigma(l+1)m).
\end{aligned}$$

To recover b correctly, $|x - \boldsymbol{\varepsilon}_i^\top(\mathbf{x}_0^\top \mid \mathbf{x}_1^\top)^\top| < \frac{q}{5}$ needs to be fulfilled [41]. Then, we can obtain $b' = 1$.

- If $\mathbf{ck} \neq \mathbf{tk}$, we can obtain $b' = 1$ with negligible probability.

On the other hand, for the condition $\boldsymbol{\eta}_1 \overset{?}{=} H_2(\mathbf{A}\boldsymbol{\eta}_2 - \mathbf{A}_S\boldsymbol{\eta}_1, \mathbf{c}_1, b')$, we have:

$$\begin{aligned}
&\mathbf{A}\boldsymbol{\eta}_2 - \mathbf{A}_S\boldsymbol{\eta}_1 = \mathbf{A}(\mathbf{S}\boldsymbol{\eta}_1 + \mathbf{y}) - \mathbf{A}_S\boldsymbol{\eta}_1\\
&= \mathbf{A}_S\boldsymbol{\eta}_1 + \mathbf{Ay} - \mathbf{A}_S\boldsymbol{\eta}_1 = \mathbf{Ay} \bmod q.
\end{aligned}$$

Then, when $b' = 1$, we can obtain:

$$\boldsymbol{\eta}_1 = H_2(\mathbf{Ay} \bmod q, \mathbf{c}_1, b') = H_2(\mathbf{A}\boldsymbol{\eta}_2 - \mathbf{A}_S\boldsymbol{\eta}_1, \mathbf{c}_1, b').$$

To sum up, our BAEKS scheme satisfies correctness, where **Test** algorithm has the ability to match the keyword ciphertext CT with the search trapdoor TD successfully. Then, the cloud server sends the data ciphertext corresponding to CT to the data receiver γ as the search result. After receiving it, the data receiver decrypts it, and generates the data plaintext corresponding to the keyword $\mathbf{tk}$.

5.3 Security Analysis

We demonstrate that BAEKS scheme is security in the aforementioned security model, i.e. IND-CKA and UF-IKGA.

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

Proof If a PPT adversary $\mathcal{A}$ who has the ability to break the IND-CKA security with a non-negligible advantage, we can construct a challenger $\mathcal{C}$ who can solve the $\mathrm{LWE}_{n,m,q,\chi}$ hardness. The following procedures show the interaction between $\mathcal{A}$ and $\mathcal{C}$.

Setup: To begin with, the challenger $\mathcal{C}$ obtains several LWE instances $(b_j, \mathbf{a}_j) \in \mathbb{Z}_q \times \mathbb{Z}_q^n$ for $j = 0, 1, \cdots, (l+1)m$, such that all b_j are chosen randomly or equal to $\mathbf{a}_j^\top \mathbf{v} + x_j$, where $\mathbf{v} \in \mathbb{Z}^n$ and $x_j \xleftarrow{\$} \chi$. Then, $\mathcal{C}$ invokes the $\mathbf{Setup}(1^\lambda)$ algorithm to obtain a public parameter $pp = (\mathbf{A}, \mathbf{u}, H_1, H_2)$, where $\mathbf{A} \in \mathbb{Z}^{n\times m}$, $H_1 : \{0,1\}^k \to \mathbb{Z}_q^{n\times m}$, $H_2 : \mathbb{Z}_q^n \times \mathbb{Z}_q^{lm} \times \{0,1\} \to \{-1,0,1\}^m$ and $\mathbf{u} = \mathbf{a}_0$. In addition, $\mathcal{C}$ executes $(\mathbf{A}_S^*, \mathbf{T}_{\mathbf{A}_S}^*) \leftarrow \mathsf{TrapGen}(n, m, q)$ to obtain the challenge public key $\mathbf{pk}_S^* = \mathbf{A}_S^*$ of data sender. For $\mathbf{A}_S^* = (\mathbf{a}_{S,1}^*, \mathbf{a}_{S,2}^*, \cdots, \mathbf{a}_{S,m}^*)$ and $i = [m]$, $\mathcal{C}$ invokes $\mathbf{s}_i^* \leftarrow \mathsf{SamplePre}(\mathbf{A}, \mathbf{T}_\mathbf{A}, \mathbf{a}_{S,i}, \sigma)$ to obtain $\mathbf{s}_i^*$. After that, $\mathcal{C}$ obtains $\mathbf{S}^* = (\mathbf{s}_1^*, \mathbf{s}_2^*, \cdots, \mathbf{s}_m^*) \in \mathbb{Z}^{m\times m}$. Moreover, for $i = [l]$, $\mathcal{C}$ executes $(\mathbf{A}_{R,i}^*, \mathbf{T}_{\mathbf{A}_{R,i}}^*) \leftarrow \mathsf{TrapGen}(n, m, q)$, and sets the challenge receivers' public key $\mathbf{pk}_{R,i}^* = \mathbf{A}_{R,i}^* = (\mathbf{a}_{1+(i-1)m}, \mathbf{a}_{2+(i-1)m}, \cdots, \mathbf{a}_{m+(i-1)m})$. Finally, $\mathcal{C}$ returns pp, $\mathbf{pk}_S^*$, $\{\mathbf{pk}_{R,1}^*, \mathbf{pk}_{R,2}^*, \cdots, \mathbf{pk}_{R,l}^*\}$ to $\mathcal{A}$.

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

- **Hash Queries** $\mathcal{O}_{H_1}$: In this phase, $\mathcal{A}$ issues H_1 queries at most q_{H_1}. Firstly, the challenger $\mathcal{C}$ creates a empty list L_{H_1}, and selects $j^* \in [q_{H_1}]$ as a challenge query. For the j-th query, if $\mathbf{ck}_j$ has been queried, $\mathcal{C}$ returns $H_1(\mathbf{ck}_j)$ in L_{H_1} to $\mathcal{A}$. Otherwise, if $j^* \neq j$, $\mathcal{C}$ selects a random matrix in $\mathbb{Z}_q^{n\times m}$ as $H_1(\mathbf{ck}_j)$, and lets $L_{H_1} = L_{H_1} \cup \{\mathbf{ck}_j, H_1(\mathbf{ck}_j)\}$. Otherwise, $\mathcal{C}$ sets $H_1(\mathbf{ck}_j) = (\mathbf{a}_{lm+1}, \mathbf{a}_{lm+2}, \cdots, \mathbf{a}_{(l+1)m})$, and lets $L_{H_1} = L_{H_1} \cup \{\mathbf{ct}_j, H_1(\mathbf{ck}_j)\}$.
- **Ciphertext Queries** $\mathcal{O}_{\mathrm{CT}}$: $\mathcal{A}$ inputs the keyword $\mathbf{ck} \in \mathcal{W}$ and $\{\mathbf{pk}_{R,1}, \mathbf{pk}_{R,2}, \cdots, \mathbf{pk}_{R,l}\} = \{\mathbf{A}_{R,1}, \mathbf{A}_{R,2}, \cdots, \mathbf{A}_{R,l}\}$. The challenger $\mathcal{C}$ calculates $\mathbf{A}_\mathbf{ck} = (\mathbf{A}_R \mid H_1(\mathbf{ck}))$ where $\mathbf{A}_R = (\mathbf{A}_{R,1} \mid \mathbf{A}_{R,2} \mid \cdots \mid \mathbf{A}_{R,l})$. Then, $\mathcal{C}$ selects a random vector $\mathbf{v} \xleftarrow{\$} \mathbb{Z}_q^n$, a random number $b \in \{0,1\}$, two noise vectors $\mathbf{x}_0 \xleftarrow{\$} \chi^{lm}$, $\mathbf{x}_1 \xleftarrow{\$} \chi^m$, and a noise number $x \xleftarrow{\$} \chi$, computes $\mathbf{c}_1 = \mathbf{A}_\mathbf{ck}^\top \mathbf{v} + (\mathbf{x}_0^\top \mid \mathbf{x}_1^\top)^\top$ and $c_2 = \mathbf{u}^\top \mathbf{v} + x + b \cdot \lfloor \frac{q}{2} \rfloor$. Furthermore, $\mathcal{C}$ selects a vector $\mathbf{y} \xleftarrow{\$} \mathbb{Z}_q^m$ in $\mathcal{D}_\sigma^m$, and computes $\boldsymbol{\eta}_1 = H_2(\mathbf{Ay} \bmod q, \mathbf{c}_1, b)$ and $\boldsymbol{\eta}_2 = \mathbf{S}^* \boldsymbol{\eta}_1 + \mathbf{y}$ with the probability $\min(\frac{\mathcal{D}_\sigma^m}{M \cdot \mathcal{D}^m_{\mathbf{S}\boldsymbol{\eta}_1, \sigma}}, 1)$. Finally, $\mathcal{C}$ returns the ciphertext $\mathrm{CT} = (\mathbf{c}_1, c_2, \boldsymbol{\eta}_1, \boldsymbol{\eta}_2)$ to $\mathcal{A}$.
- **Trapdoor Queries** $\mathcal{O}_{\mathrm{TD}}$: For a chosen data receiver $\gamma \in [l]$, $\mathcal{A}$ inputs the keyword $\mathbf{tk} \in \mathcal{W}$, $\mathbf{pk}_{R,\gamma}^* = \mathbf{A}_{R,\gamma}^*$, $\mathbf{pk}_S = \mathbf{A}_S$. The challenger $\mathcal{C}$ calculates $\mathbf{A}_{\mathbf{tk},\gamma} = (\mathbf{A}_{R,\gamma^*} \mid H_1(\mathbf{tk}))$ and $\mathbf{A}_\mathbf{tk} = (\mathbf{A}_{R,1}^* \mid \cdots \mid \mathbf{A}_{R,l}^* \mid H_1(\mathbf{tk})) \in \mathbb{Z}^{n\times(l+1)m}$, and obtains $\mathbf{T}_{\mathbf{A}_{\mathbf{tk},\gamma}} \leftarrow \mathsf{SampleBasis}(\mathbf{A}_{\mathbf{tk},\gamma}, \mathbf{T}_{\mathbf{A}_{R,\gamma}}^*, \{1\}, L)$. Then, $\mathcal{C}$ samples $\boldsymbol{\varepsilon} \leftarrow \mathsf{GenSamplePre}(\mathbf{A}_\mathbf{tk}, \mathbf{T}_{\mathbf{A}_{\mathbf{tk},\gamma}}, \{\gamma, l+1\}, \mathbf{u}, \sigma)$, such that $\mathbf{A}_\mathbf{tk} \boldsymbol{\varepsilon} = \mathbf{u} \bmod q$. Finally, $\mathcal{C}$ returns the trapdoor $\mathrm{TD} = (\boldsymbol{\varepsilon}, \mathbf{pk}_S)$ to $\mathcal{A}$.

Challenge: $\mathcal{A}$ chooses $\mathbf{ck}_0, \mathbf{ck}_1 \in \mathcal{W}$ which have not been queried in **Phase 1**, and transmits it to the challenger $\mathcal{C}$. Then, $\mathcal{C}$ selects $\xi \in \{0,1\}$, and calculates a challenge ciphertext $(\mathbf{c}_{1,\xi}, c_{2,\xi}) \in \mathbb{Z}^{(l+1)m} \times \mathbb{Z}_q$, where: $\mathbf{c}_{1,\xi} = (b_1, \cdots, b_m, \cdots, b_{lm+1}, \cdots, b_{(l+1)m})^\top$, and $c_{2,\xi} = b_0 + b\lfloor \frac{q}{2} \rfloor, b \in \{0,1\}$. After that, $\mathcal{C}$ calculates $\boldsymbol{\eta}_1 = H_2(\mathbf{Ay} \bmod q, \mathbf{c}_1, b) \in \{-1,0,1\}^m$ and $\boldsymbol{\eta}_2 = \mathbf{S}^* \boldsymbol{\eta}_1 + \mathbf{y} \in \mathbb{Z}_q^m$ with the probability $\min(\frac{\mathcal{D}_\sigma^m}{M \cdot \mathcal{D}^m_{\mathbf{S}\boldsymbol{\eta}_1, \sigma}}, 1)$, and then returns $\mathrm{CT}_\xi = (\mathbf{c}_1, c_2, \boldsymbol{\eta}_1, \boldsymbol{\eta}_2)$ to $\mathcal{A}$.

Phase 2: $\mathcal{A}$ executes these queries as above, and promises neither $\mathbf{ck}_0$ nor $\mathbf{ck}_1$ can be queried.

Guess: $\mathcal{A}$ outputs a random bit $\xi' \in \{0,1\}$ after receiving CT_ξ. If $\xi' = \xi$, $\mathcal{A}$ wins this game, and the challenger $\mathcal{C}$ outputs 1 meaning $(b_j, \mathbf{a}_j)$ is sampled from the LWE oracle. Otherwise, $\mathcal{C}$ outputs 0 meaning $(b_j, \mathbf{a}_j)$ is sampled from the uniform distribution $\mathbb{Z}_q \times \mathbb{Z}_q^n$.

Analysis: If $\xi' = \xi$, for $j = [(l+1)m]$, the challenger $\mathcal{C}$ outputs 1 meaning $(b_j, \mathbf{a}_j)$ is sampled from the LWE oracle, then CT_ξ is valid. Let $\mathbf{x} = (x_1, \cdots, x_m, x_{m+1}, \cdots, x_{(l+1)m})^\top$, we have:

$$
\begin{aligned}
\mathbf{c}_{1,\xi} &= (b_1, \cdots, b_m, b_{m+1}, \cdots, b_{2m}, \cdots, b_{lm+1}, \cdots, b_{(l+1)m})^\top \\
&= (\mathbf{a}_1, \cdots, \mathbf{a}_m \mid \mathbf{a}_{m+1}, \cdots, \mathbf{a}_{2m} \mid \cdots \mid \mathbf{a}_{lm+1}, \\
&\quad \cdots, \mathbf{a}_{(l+1)m})^\top \mathbf{v} + (x_1, \cdots, x_m, \cdots, x_{(l+1)m})^\top \\
&= (\mathbf{A}_{R,1} \mid \mathbf{A}_{R,2} \mid \cdots \mid \mathbf{A}_{R,l} \mid H_1(\mathbf{ck}_\xi))^\top \mathbf{v} + \mathbf{x} \\
&= (\mathbf{A}_R \mid H_1(\mathbf{ck}_\xi))^\top \mathbf{v} + \mathbf{x} \\
&= \mathbf{A}_\mathbf{ck}^\top \mathbf{v} + \mathbf{x}.
\end{aligned}
$$

$$
c_{2,\xi} = b_0 + b\lfloor \frac{q}{2} \rfloor = \mathbf{a}_0^\top \mathbf{v} + x_0 + b\lfloor \frac{q}{2} \rfloor = \mathbf{u}^\top \mathbf{v} + x_0 + b\lfloor \frac{q}{2} \rfloor.
$$

In this case, $\mathcal{A}$ has the advantage ϵ_1 to solve LWE hardness, thus $\Pr[\xi' = \xi] = \frac{1}{2} + \epsilon_1$. Otherwise, $\mathcal{C}$ outputs 0 meaning $(b_j, \mathbf{a}_j)$ is obtained from the uniform distribution over $\mathbb{Z}_q \times \mathbb{Z}_q^n$, we can get $\Pr[\xi' = \xi] = \frac{1}{2}$. 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 $\mathrm{SIS}_{n,m,q,\beta}$ hardness holds, our proposed lattice-based BAEKS primitive satisfies UF-IKGA security in the random oracle model. For any PPT adversary $\mathcal{A}$, if $\mathcal{A}$ can compromise our scheme, then we can construct a PPT challenger $\mathcal{C}$ to solve the $\mathrm{SIS}_{n,m,q,\beta}$ hardness.

Proof If there exists an adversary $\mathcal{A}$ who can break the UF-IKGA security, then we has the ability to construct a challenger $\mathcal{C}$ who can find a solution of $\mathrm{SIS}_{n,m,q,\beta}$ hardness.

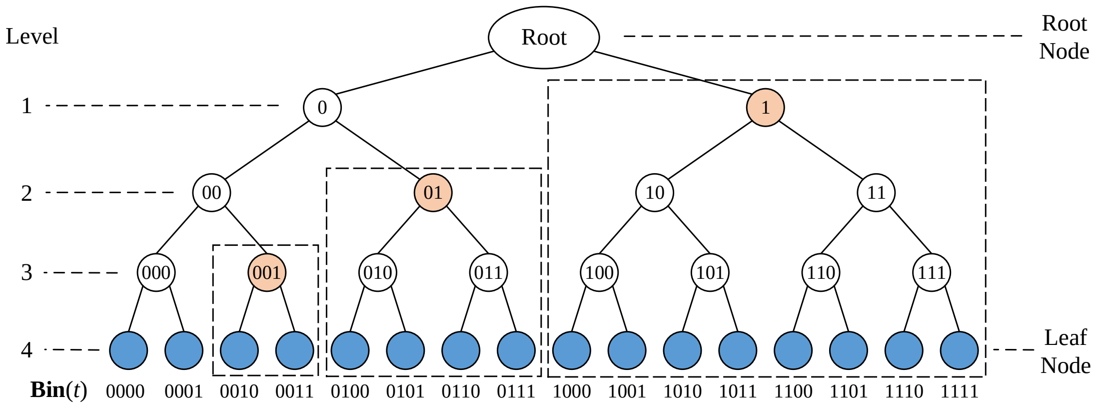


Fig. 3. The binary tree utilized to secret key update for data receivers, and the number of level $\tau = 4$.

The following procedures show the interaction between $\mathcal{A}$ and $\mathcal{C}$.

Setup: Firstly, a challenge keyword $\mathbf{ck}^*$ is selected by the adversary $\mathcal{A}$. Then, the challenger $\mathcal{C}$ invokes $\mathbf{Setup}(1^\lambda)$ to calculate the public parameter $pp = (H_1, H_2, \mathbf{u})$, where $H_1 : \{0,1\}^k \to \mathbb{Z}_q^{n\times m}$, $H_2 : \mathbb{Z}_q^{lm} \times \{0,1\} \to \mathbb{Z}_q^m$, and $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_q^n$. In addition, $\mathcal{C}$ executes $(\mathbf{A}_S^*, \mathbf{T}_{\mathbf{A}_S}^*) \leftarrow \mathsf{TrapGen}(n, m, q)$ to set the challenge public key $\mathbf{pk}_S^* = \mathbf{A}_S^*$ of data sender. For $\mathbf{A}_S^* = (\mathbf{a}_{S,1}^*, \mathbf{a}_{S,2}^*, \cdots, \mathbf{a}_{S,m}^*)$ and $i = [m]$, $\mathcal{C}$ invokes $\mathbf{s}_i^* \leftarrow \mathsf{SamplePre}(\mathbf{A}, \mathbf{T}_\mathbf{A}, \mathbf{a}_{S,i}, \sigma)$ to obtain $\mathbf{s}_i^*$. Moreover, for $i = [l]$, $\mathcal{C}$ executes $(\mathbf{A}_{R,i}^*, \mathbf{T}_{\mathbf{A}_{R,i}}^*) \leftarrow \mathsf{TrapGen}(n, m, q)$ to calculate the challenge receivers' public key $\mathbf{pk}_{R,i}^* = \mathbf{A}_{R,i}^*$. Finally, $\mathcal{C}$ returns pp, $\mathbf{pk}_S^*$, $\{\mathbf{pk}_{R,1}^*, \mathbf{pk}_{R,2}^*, \cdots, \mathbf{pk}_{R,l}^*\}$ to $\mathcal{A}$.

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

- **Hash Queries** $\mathcal{O}_{H_2}$: In this phase, $\mathcal{A}$ issues H_2 queries at most q_{H_2}. Firstly, the challenger $\mathcal{C}$ creates an empty list L_{H_2}, and selects $j^* \in [q_{H_2}]$ as challenge query. For the j-th query, if $(\mathbf{c}_{1,j}, b_j)$ has been queried, $\mathcal{C}$ returns $H_2(\mathbf{A}\mathbf{y}_j \bmod q, \mathbf{c}_{1,j}, b_j)$ in L_{H_2} to $\mathcal{A}$. Otherwise, if $j^* \neq j$, $\mathcal{C}$ selects $\mathbf{y}_j \in \mathbb{Z}^m$ from a uniform distribution on $\mathbb{Z}^m$, and sends $H_2(\mathbf{A}\mathbf{y}_j \bmod q, \mathbf{c}_{1,j}, b_j)$ to $\mathcal{A}$ and lets $L_{H_2} = L_{H_2} \cup \{\mathbf{c}_{1,j}, b_j, H_2(\mathbf{A}\mathbf{y}_j \bmod q, \mathbf{c}_{1,j}, b_j)\}$. Otherwise, $\mathcal{C}$ selects $\mathbf{y}^* \in \mathbb{Z}^m$, and sets $\mathbf{c}_1^* = \mathbf{c}_1$, $b^* = b$, which is a part of the forged ciphertext. Finally, $\mathcal{C}$ returns $H_2(\mathbf{A}\mathbf{y}^* \bmod q, \mathbf{c}_1^*, b^*)$ to $\mathcal{A}$, and lets $L_{H_2} = L_{H_2} \cup \{\mathbf{c}_1^*, b^*, H_2(\mathbf{A}\mathbf{y}^* \bmod q, \mathbf{c}_1^*, b^*)\}$.
- **Ciphertext Queries** $\mathcal{O}_{\mathrm{CT}}$: $\mathcal{A}$ inputs the keyword $\mathbf{ck} \in \mathcal{W}$ and $\{\mathbf{pk}_{R,1}, \mathbf{pk}_{R,2}, \cdots, \mathbf{pk}_{R,l}\} = \{\mathbf{A}_{R,1}, \mathbf{A}_{R,2}, \cdots, \mathbf{A}_{R,l}\}$. $\mathcal{C}$ calculates $\mathbf{A}_{\mathbf{ck}} = (\mathbf{A}_R \mid H_1(\mathbf{ck}))$, where $\mathbf{A}_R = (\mathbf{A}_{R,1} \mid \mathbf{A}_{R,2} \mid \cdots \mid \mathbf{A}_{R,l})$. Then, $\mathcal{C}$ selects a random vector $\mathbf{v} \xleftarrow{\$} \mathbb{Z}_q^n$, a random number $b \in \{0,1\}$, two noise vectors $\mathbf{x}_0 \xleftarrow{\$} \chi^{lm}$, $\mathbf{x}_1 \xleftarrow{\$} \chi^m$, and a noise number $x \xleftarrow{\$} \chi$, and computes $\mathbf{c}_1 = \mathbf{A}_{\mathbf{ck}}^\top \mathbf{v} + (\mathbf{x}_0^\top \mid \mathbf{x}_1^\top)^\top$ and $c_2 = \mathbf{u}^\top \mathbf{v} + x + b \cdot \lfloor \frac{q}{2} \rfloor$, and checks whether $(\mathbf{c}_1, b)$ has been queried in list L_{H_2}. If not, $\mathcal{C}$ selects $\mathbf{y} \in \mathbb{Z}^m$ randomly, and calculates $H_2(\mathbf{A}\mathbf{y} \bmod q, \mathbf{c}_1, b)$ and sets $L_{H_2} = L_{H_2} \cup \{\mathbf{c}_1, b, H_2(\mathbf{A}\mathbf{y} \bmod q, \mathbf{c}_1, b)\}$. After that, $\mathcal{C}$ sets $\boldsymbol{\eta}_1 = H_2(\mathbf{A}\mathbf{y} \bmod q, \mathbf{c}_1, b)$, and calculates $\boldsymbol{\eta}_2 = \mathbf{S}^* \boldsymbol{\eta}_1 + \mathbf{y}$ with the probability $\min(\frac{\mathcal{D}_\sigma^m}{M \cdot \mathcal{D}_{\mathbf{S}\boldsymbol{\eta}_1,\sigma}^m}, 1)$. Finally, $\mathcal{C}$ returns the ciphertext $\mathrm{CT} = (\mathbf{c}_1, c_2, \boldsymbol{\eta}_1, \boldsymbol{\eta}_2)$ to $\mathcal{A}$.
- **Trapdoor Queries** $\mathcal{O}_{\mathrm{TD}}$: For a chosen data receiver $\gamma \in [l]$, $\mathcal{A}$ inputs the keyword $\mathbf{tk} \in \mathcal{W}$, $\mathbf{pk}_{R,\gamma}^* = \mathbf{A}_{R,\gamma}^*$, $\mathbf{pk}_S = \mathbf{A}_S$. $\mathcal{C}$ calculates $\mathbf{A}_{\mathbf{tk},\gamma} = (\mathbf{A}_{R,\gamma^*} \mid H_1(\mathbf{tk}))$ and $\mathbf{A}_{\mathbf{tk}} = (\mathbf{A}_{R,1}^* \mid \cdots \mid \mathbf{A}_{R,l}^* \mid H_1(\mathbf{tk})) \in \mathbb{Z}^{n\times(l+1)m}$, and obtains $\mathbf{T}_{\mathbf{A}_{\mathbf{tk},\gamma}} \leftarrow \mathsf{SampleBasis}(\mathbf{A}_{\mathbf{tk},\gamma}, \mathbf{T}_{\mathbf{A}_{R,\gamma}}^*, \{1\}, L)$. Then, $\mathcal{C}$ samples $\boldsymbol{\varepsilon} \leftarrow \mathsf{GenSamplePre}(\mathbf{A}_{\mathbf{tk}}, \mathbf{T}_{\mathbf{A}_{\mathbf{tk},\gamma}}, \{\gamma, l+1\}, \mathbf{u}, \sigma)$, such that $\mathbf{A}_{\mathbf{tk}} \boldsymbol{\varepsilon} = \mathbf{u} \bmod q$. Finally, $\mathcal{C}$ returns the trapdoor $\mathrm{TD} = (\boldsymbol{\varepsilon}, \mathbf{pk}_S)$ to $\mathcal{A}$.

Forgery: $\mathcal{A}$ selects $\gamma \in [l]$ and transmits it to $\mathcal{C}$, $\mathcal{C}$ invokes the **Trapdoor**$(pp, \mathbf{ck}^*, \mathbf{pk}_S^*, \{\mathbf{pk}_{R,1}^*, \mathbf{pk}_{R,2}^*, \cdots, \mathbf{pk}_{R,l}^*\}, \mathbf{sk}_{R,\gamma}^*)$ algorithm to obtain TD^*, and sends it to $\mathcal{A}$. Then, $\mathcal{A}$ calculates $\mathrm{CT}^* = (\mathbf{c}_1^*, c_2^*, \boldsymbol{\eta}_1^*, \boldsymbol{\eta}_2')$ as a forged ciphertext corresponding to $\mathbf{ck}^*$, and wins this game if the **Test**$(\mathrm{CT}^*, \mathrm{TD}^*)$ algorithm inputs 1.

Analysis: Since $\mathrm{CT}^* = (\mathbf{c}_1^*, c_2^*, \boldsymbol{\eta}_1^*, \boldsymbol{\eta}_2')$ is a valid ciphertext, we can obtain $(\mathbf{c}_1^*, b^*, H_2(\mathbf{A}\mathbf{y}^* \bmod q, \mathbf{c}_1^*, b^*))$ in L_{H_2} such that $\boldsymbol{\eta}_1^* = H_2(\mathbf{A}\mathbf{y}^* \bmod q, \mathbf{c}_1^*, b^*)$, $\boldsymbol{\eta}_2^* = \mathbf{S}^* \boldsymbol{\eta}_1^* + \mathbf{y}$. In this way, we have $H_2(\mathbf{A}\boldsymbol{\eta}_2^* - \mathbf{A}_S \boldsymbol{\eta}_1^*, \mathbf{c}_1^*, b^*) = H_2(\mathbf{A}\boldsymbol{\eta}_2' - \mathbf{A}_S \boldsymbol{\eta}_1^*, \mathbf{c}_1^*, b^*)$. If $\mathbf{A}\boldsymbol{\eta}_2^* - \mathbf{A}_S \boldsymbol{\eta}_1^* \neq \mathbf{A}\boldsymbol{\eta}_2' - \mathbf{A}_S \boldsymbol{\eta}_1^*$, it reflects that $\mathcal{A}$ obtains a pre-image of hash function H_2. Otherwise, we get $\mathbf{A}\boldsymbol{\eta}_2^* - \mathbf{A}_S \boldsymbol{\eta}_1^* = \mathbf{A}\boldsymbol{\eta}_2' - \mathbf{A}_S \boldsymbol{\eta}_1^*$, thereby: $\mathbf{A}(\boldsymbol{\eta}_2' - \boldsymbol{\eta}_2^*) = \mathbf{0}$. In addition, we notice that $\boldsymbol{\eta}_2' - \boldsymbol{\eta}_2^* \neq 0$ and $\|\boldsymbol{\eta}_2'\| \leq 2\sigma\sqrt{m}$, $\|\boldsymbol{\eta}_2^*\| \leq 2\sigma\sqrt{m}$, we can calculate: $\|\boldsymbol{\eta}_2' - \boldsymbol{\eta}_2^*\| \leq 4\sigma\sqrt{m}$., and $\boldsymbol{\eta}_2' - \boldsymbol{\eta}_2^*$ is a solution of $\mathrm{SIS}_{q,n,m,\beta}$ hardness. □

6 Our Proposed FS-BAEKS Scheme

In this section, our FS-BAEKS scheme is proposed as an enhanced version of our BAEKS described in Section 5.

6.1 Concrete Construction

- **Setup**(1^λ): A security parameter 1^λ is inputted by the KGC, and then the public parameter pp is outputted according to the following procedures.
  1) Set the system parameters n, m, q, L, σ, k, and l.
  2) Initialize all nodes in the binary tree, set τ as the depth of binary tree, and $T = 2^\tau$, as in Fig. 3 (A example at $\tau = 4$).

3) For the root node, invoke $(\mathbf{A}, \mathbf{T}_{\mathbf{A}}) \leftarrow \mathsf{TrapGen}(n, m, q)$ to generate a uniformly matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and a basis $\mathbf{T}_{\mathbf{A}} \in \mathbb{Z}^{m \times m}$ for $\Lambda_q^{\perp}(\mathbf{A})$.
4) Choose a vector $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_q^n$ uniformly.
5) Define two hash functions $H_1 : \{0,1\}^k \rightarrow \mathbb{Z}_q^{n \times m}$ and $H_2 : \mathbb{Z}_q^n \times \mathbb{Z}_q^{(l+\tau+1)m} \times \{0,1\} \rightarrow \{-1,0,1\}^m$.
6) Output $pp := (n, m, q, L, \sigma, k, l, \tau, \mathbf{A}, \mathbf{u}, H_1, H_2)$ as the public parameter.

- $\mathbf{KeyGen}_S(pp, \mathbf{T}_{\mathbf{A}})$: The KGC inputs a public parameter pp and a basis $\mathbf{T}_{\mathbf{A}}$ and then returns the public & secret keys $(\mathbf{pk}_S, \mathbf{sk}_S)$ to a data sender according to the following procedures.

1) Invoke $(\mathbf{A}_S, \mathbf{T}_{\mathbf{A}_S}) \leftarrow \mathsf{TrapGen}(n, m, q)$ to obtain a 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) Parse the matrix $\mathbf{A}_S = (\mathbf{a}_{S,1}, \mathbf{a}_{S,2}, \cdots, \mathbf{a}_{S,m})$, which each vector $\mathbf{a}_{S,i} \in \mathbb{Z}^n$ for $i = [m]$.
3) For $i = [m]$, sample a vector $\mathbf{s}_i \in \mathbb{Z}_q^m$ as $\mathbf{s}_i \leftarrow \mathsf{SamplePre}(\mathbf{A}, \mathbf{T}_{\mathbf{A}}, \mathbf{a}_{S,i}, \sigma)$, where $\mathbf{s}_i$ s.t. $\mathbf{A}\mathbf{s}_i = \mathbf{a}_{S,i} \bmod q$ and $\mathbf{s}_i$ is statistically distributed in $\mathcal{D}^m_{\Lambda_q^{\mathbf{a}_{S,i}}(\mathbf{A}),\sigma}$.
4) Let a matrix $\mathbf{S} = (\mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s}_m) \in \mathbb{Z}^{m \times m}$, where $\mathbf{AS} = \mathbf{A}_s \bmod q$.
5) Output $\mathbf{pk}_S := \mathbf{A}_S$ and $\mathbf{sk}_S := (\mathbf{T}_{\mathbf{A}_S}, \mathbf{S})$ as the public & secret keys of the data sender.

- $\mathbf{KeyGen}_R(pp)$: For $i = [l]$, the KGC inputs a public parameter pp and then returns the public & initial secret key $(\mathbf{pk}_{R,i}, \mathbf{sk}_{R,i})$ to data receivers i according to the following procedures.

1) Invoke $(\mathbf{A}_{R,i,0}, \mathbf{T}_{\mathbf{A}_{R,i,0}}) \leftarrow \mathsf{TrapGen}(n, m, q)$ to generate a uniformly matrix $\mathbf{A}_{R,i,0} \in \mathbb{Z}_q^{n \times m}$ and a basis $\mathbf{T}_{\mathbf{A}_{R,i,0}} \in \mathbb{Z}^{m \times m}$ for $\Lambda_q^{\perp}(\mathbf{A}_{R,i,0})$.
2) Output $\mathbf{pk}_{R,i} := \mathbf{A}_{R,i,0}$ and $\mathbf{sk}_{R,i,0} := \mathbf{T}_{\mathbf{A}_{R,i,0}}$ as a public & initial secret key of the data receiver.

- $\mathbf{KeyUpdate}_R(pp, \mathbf{pk}_{R,i}, \mathbf{sk}_{R,i,t})$: For $i = [l]$, the KGC inputs a public parameter pp, a public key $\mathbf{pk}_{R,i}$ and secret key $\mathbf{sk}_{R,i,t}$ of data receiver with time period t, then returns its secret key $\mathbf{sk}_{R,i,t+1}$ with time period $t+1$ to this data receiver according to the following procedures, where $t \in \{0, 1, \cdots, T-1\}$. We set $\mathbf{bin}(t)$ as the τ bits binary representation of t, $\mathsf{Node}(\mathbf{bin}(t))$ as the minimal cover set of leaf node $\mathbf{bin}(t)$, which denotes the smallest set that includes an common ancestor node of each leaf node in $\{\mathbf{bin}(t), \mathbf{bin}(t+1), \cdots, \mathbf{bin}(T-1)\}$, and does not include any ancestor nodes of each leaf node in $\{\mathbf{bin}(0), \mathbf{bin}(1), \cdots, \mathbf{bin}(t-1)\}$. For example, in Fig. 3, $\mathsf{Node}(0010) = \{001, 01, 1\}$.

1) Parse $\mathbf{bin}(t) = (t_1, t_2, \cdots, t_\tau) \in \{0,1\}^\tau$.
2) Select several matrices $\mathbf{A}^{(0)}_{R,i,1}$, $\mathbf{A}^{(1)}_{R,i,1}$, $\cdots$, $\mathbf{A}^{(0)}_{R,i,\tau}$, $\mathbf{A}^{(1)}_{R,i,\tau} \in \mathbb{Z}_q^{n \times m}$.
3) Set the secret key $\mathbf{sk}_{R,i,0} = \mathbf{T}_{\mathbf{A}_{R,i,0}}$ with time period 0, and $\mathbf{sk}_{R,i,1} = \{\mathbf{T}_{\mathbf{A}_{R,i,0001}}, \mathbf{T}_{\mathbf{A}_{R,i,001}}, \mathbf{T}_{\mathbf{A}_{R,i,01}}, \mathbf{T}_{\mathbf{A}_{R,i,1}}\}$ with time period 1, due to $\mathsf{Node}(\mathbf{bin}(1)) = \mathsf{Node}(0001) = \{0001, 001, 01, 1\}$.
4) Update the secret key $\mathbf{sk}_{R,i,t}$ to $\mathbf{sk}_{R,i,t+1}$ according to the following procedures:
   a) Calculate the minimal cover set with time period $\mathsf{Node}(\mathbf{bin}(t))$ and $\mathsf{Node}(\mathbf{bin}(t+1))$,
   b) Obtain the basis of the node in set $\mathsf{Node}(\mathbf{bin}(t+1)) \setminus \mathsf{Node}(\mathbf{bin}(t))$, and remove the basis of the node in set $\mathsf{Node}(\mathbf{bin}(t)) \setminus \mathsf{Node}(\mathbf{bin}(t+1))$.
5) Invoke $\mathbf{T}_{\mathbf{A}_{R,i,\Theta_j}} \leftarrow \mathsf{ExtBasis}(\mathbf{A}_{R,i,\Theta_j}, \mathbf{T}_{A_{R,i,0}})$ or $\mathbf{T}_{\mathbf{A}_{R,i,\Theta_j}} \leftarrow \mathsf{ExtBasis}(\mathbf{A}_{R,i,\Theta_j}, \mathbf{T}_{A_{R,i,\Theta_\zeta}})$ to generate the aforementioned basis $\mathbf{T}_{\mathbf{A}_{R,i,\Theta_j}}$, where $\Theta_j = (\theta_1, \cdots, \theta_\zeta, \cdots, \theta_j) \in \{0,1\}^j$ as the nodes at j-th level, $j \in [\tau]$, $\zeta < j$, $\mathbf{A}_{R,i,\Theta_j} = (\mathbf{A}_{R,i,0} | \mathbf{A}^{(\theta_1)}_{R,i,1} | \cdots | \mathbf{A}^{(\theta_j)}_{R,i,j}) \in \mathbb{Z}_q^{n \times (j+1)m}$ and $\Theta_\zeta = (\theta_1, \cdots, \theta_\zeta) \in \{0,1\}^\zeta$.
6) Return $\mathbf{sk}_{R,i,t+1}$ as the secret key of data receiver i with time period $t+1$.

- $\mathbf{BAEKS}(pp, \mathbf{ck}, \mathbf{sk}_S, \{\mathbf{pk}_{R,1}, \mathbf{pk}_{R,2}, \cdots, \mathbf{pk}_{R,l}\}, t)$: A data sender inputs a public parameter pp, a keyword $\mathbf{ck} \in \mathcal{W}$, a sender's secret key $\mathbf{sk}_S$, a set of receivers' public keys $\{\mathbf{pk}_{R_1}, \mathbf{pk}_{R_2}, \cdots, \mathbf{pk}_{R_l}\}$, a time period t, and then performs the following procedures.

1) Let a matrix $\mathbf{A}_R = (\mathbf{A}_{R,1} \mid \mathbf{A}_{R,2} \mid \cdots \mid \mathbf{A}_{R,l}) \in \mathbb{Z}_q^{n \times lm}$, and $\mathbf{A}_t = (\mathbf{A}_1^{(t_1)} \mid \mathbf{A}_2^{(t_2)} \mid \cdots \mid \mathbf{A}_\tau^{(t_\tau)}) \in \mathbb{Z}_q^{n \times \tau m}$.
2) Calculate a matrix $\mathbf{A}_{\mathbf{ck},t} = (\mathbf{A}_R \mid \mathbf{A}_t \mid H_1(\mathbf{ck})) \in \mathbb{Z}^{n \times (l+\tau+1)m}$.
3) Select a random vector $\mathbf{v} \xleftarrow{\$} \mathbb{Z}_q^n$ uniformly, a random number $b \in \{0,1\}$, two noise vectors $\mathbf{x}_0 \xleftarrow{\$} \chi^{lm}$, $\mathbf{x}_1 \xleftarrow{\$} \chi^{\tau m}$, $\mathbf{x}_2 \xleftarrow{\$} \chi^m$, and a noise number $x \xleftarrow{\$} \chi$.
4) Calculate a vector $\mathbf{c}_1 = \mathbf{A}_{\mathbf{ck},t}^\top \mathbf{v} + (\mathbf{x}_0^\top \mid \mathbf{x}_1^\top \mid \mathbf{x}_2^\top)^\top \in \mathbb{Z}_q^{(l+\tau+1)m}$, and a number $c_2 = \mathbf{u}^\top \mathbf{v} + x + b \cdot \lfloor \frac{q}{2} \rfloor \in \mathbb{Z}_q$.
5) Select a vector $\mathbf{y} \xleftarrow{\$} \mathbb{Z}_q^m$ in $\mathcal{D}_\sigma^m$ uniformly.
6) Calculate a vector $\boldsymbol{\eta}_1 = H_2(\mathbf{Ay} \bmod q, \mathbf{c}_1, b) \in \{-1,0,1\}^m$ and another vector $\boldsymbol{\eta}_2 = \mathbf{S}\boldsymbol{\eta}_1 + \mathbf{y} \in \mathbb{Z}_q^m$ with the probability $\min(\frac{\mathcal{D}_\sigma^m}{M \cdot \mathcal{D}^m_{\mathbf{S}\boldsymbol{\eta}_1,\sigma}}, 1)$.
7) Output $\mathrm{CT}_t := (\mathbf{c}_1, c_2, \boldsymbol{\eta}_1, \boldsymbol{\eta}_2)$ as the ciphertext corresponding to the keyword $\mathbf{ck}$ with time period t.

- $\mathbf{Trapdoor}(pp, \mathbf{tk}, \mathbf{pk}_S, \{\mathbf{pk}_{R,1}, \mathbf{pk}_{R,2}, \cdots, \mathbf{pk}_{R,l}\}, \mathbf{sk}_{R,\gamma,t})$: A data receiver $\gamma \in [l]$ inputs a public parameter pp, a keyword $\mathbf{tk} \in \mathcal{W}$, a public key $\mathbf{pk}_S$ of data sender, a set of receivers' public keys $\{\mathbf{pk}_{R,1}, \mathbf{pk}_{R,2}, \cdots, \mathbf{pk}_{R,l}\}$, and secret keys $\mathbf{sk}_{R,\gamma,t}$ with receiver γ and time period t, and then performs the following procedures.

1) Let a matrix $\mathbf{A}_t = (\mathbf{A}_1^{(t_1)} \mid \mathbf{A}_2^{(t_2)} \mid \cdots \mid \mathbf{A}_\tau^{(t_\tau)}) \in \mathbb{Z}_q^{n \times \tau m}$.
2) Calculate two matrices $\mathbf{A}_{\mathbf{tk},\gamma,t} = (\mathbf{A}_{R,\gamma} \mid \mathbf{A}_t \mid H_1(\mathbf{tk})) \in \mathbb{Z}_q^{n \times (\tau+2)m}$ and $\mathbf{A}_{\mathbf{tk},t} = (\mathbf{A}_{R,1} \mid \cdots \mid \mathbf{A}_{R,l} \mid \mathbf{A}_t \mid H_1(\mathbf{tk})) \in \mathbb{Z}^{n \times (l+\tau+1)m}$.
3) If $\mathbf{sk}_{R,\gamma,t}$ does not contain $\mathbf{T}_{\mathbf{A}_{R,\gamma,t}}$, invoke $\mathbf{T}_{\mathbf{A}_{R,\gamma,t}} \leftarrow \mathsf{ExtBasis}(\mathbf{A}_{R,\gamma} \mid \mathbf{A}_t, \mathbf{T}_{\mathbf{A}_{R,\gamma,\Theta_j}})$ to obtain a basis $\mathbf{T}_{\mathbf{A}_{R,\gamma,t}}$ in $\mathbb{Z}^{(\tau+2)m \times (\tau+2)m}$ for $\Lambda_q^{\perp}(\mathbf{A}_{R,\gamma} \mid \mathbf{A}_t)$, where Θ_j is an ancestor node of $\mathbf{bin}(t)$ and $j < \tau$.
4) Invoke $\mathbf{T}_{\mathbf{A}_{\mathbf{tk},\gamma,t}} \leftarrow \mathsf{SampleBasis}(\mathbf{A}_{\mathbf{tk},\gamma,t}, \mathbf{T}_{\mathbf{A}_{R,\gamma,t}}, \{1\}, L)$ to obtain a basis $\mathbf{T}_{\mathbf{A}_{\mathbf{tk},\gamma,t}} \in \mathbb{Z}^{(\tau+2)m \times (\tau+2)m}$ for $\Lambda_q^{\perp}(\mathbf{A}_{\mathbf{tk},\gamma,t})$.
5) Sample a vector $\boldsymbol{\varepsilon}_t \in \mathbb{Z}^{(l+\tau+1)m}$ as $\boldsymbol{\varepsilon}_t \leftarrow \mathsf{GenSamplePre}(\mathbf{A}_{\mathbf{tk},t}, \mathbf{T}_{\mathbf{A}_{\mathbf{tk},\gamma,t}}, \{i, l+1, \cdots, l+\tau, l+\tau+1\}, \mathbf{u}, \sigma)$, where $\boldsymbol{\varepsilon}_t$ s.t. $\mathbf{A}_{\mathbf{tk},t}\boldsymbol{\varepsilon}_t = \mathbf{u} \bmod q$ and $\boldsymbol{\varepsilon}_t$ is statistically distributed in $\mathcal{D}^{(l+\tau+1)m}_{\Lambda_q^{\mathbf{u}}(\mathbf{A}_{\mathbf{tk},t})}$.

6) Output $\mathrm{TD}_t := (\boldsymbol{\varepsilon}_t, \mathbf{pk}_S)$ as the trapdoor corresponding to the keyword $\mathbf{tk}$.

- **Test**$(\mathrm{CT}_t, \mathrm{TD}_{i,t})$: The cloud server inputs the ciphertext CT_t together with the trapdoor TD_t, and then processes the following procedures.

1) Parse $\mathrm{CT}_t = (\mathbf{c}_1, c_2, \boldsymbol{\eta}_1, \boldsymbol{\eta}_2)$ and $\mathrm{TD}_t = (\boldsymbol{\varepsilon}_t, \mathbf{pk}_S = \mathbf{A}_S)$.
2) Calculate a number $d = c_2 - \boldsymbol{\varepsilon}_t^\top \mathbf{c}_1 \in \mathbb{Z}_q$. If $\left|d - \lfloor \frac{q}{2} \rfloor\right| < \lfloor \frac{q}{4} \rfloor$, set $b' = 1$. Otherwise, set $b' = 0$.
3) Check $\|\boldsymbol{\eta}_2\| \overset{?}{\leq} 2\sigma\sqrt{m}$ and $\boldsymbol{\eta}_1 \overset{?}{=} H_2(\mathbf{A}\boldsymbol{\eta}_2 - \mathbf{A}_S\boldsymbol{\eta}_1, \mathbf{c}_1, b')$. If two conditions are satisfied, output 1. Otherwise, output 0.

6.2 Correctness Analysis

Our FS-BAEKS scheme needs to additionally satisfy $q\alpha\sigma(l+\tau+1)m\omega(\sqrt{\log[(l+\tau+1)m]}) + \mathcal{O}(\sigma(l+\tau+1)m) < \frac{q}{5}$ for parameter settings. The correctness analysis between our BAEKS and FS-BAEKS are high symmetric, where the only difference is the condition $|d - \lfloor \frac{q}{2} \rfloor| < \lfloor \frac{q}{4} \rfloor$. We demonstrate the detail as follows:

- If $\mathbf{ck} = \mathbf{tk}$, we have:

$$
\begin{aligned}
d &= c_2 - \boldsymbol{\varepsilon}_t^\top \mathbf{c}_1 \\
&= \mathbf{u}^\top \mathbf{v} + x + b \cdot \lfloor \frac{q}{2} \rfloor - \boldsymbol{\varepsilon}_t^\top (\mathbf{A}_{\mathbf{ck},t}^\top \mathbf{v} + (\mathbf{x}_0^\top \mid \mathbf{x}_1^\top \mid \mathbf{x}_2^\top)^\top) \\
&= \mathbf{u}^\top \mathbf{v} + x + b \cdot \lfloor \frac{q}{2} \rfloor - \boldsymbol{\varepsilon}_t^\top (\mathbf{A}_{\mathbf{tk},t}^\top \mathbf{v} + (\mathbf{x}_0^\top \mid \mathbf{x}_1^\top \mid \mathbf{x}_2^\top)^\top) \\
&= \mathbf{u}^\top \mathbf{v} + x + b \cdot \lfloor \frac{q}{2} \rfloor - \mathbf{u}^\top \mathbf{v} - ((\mathbf{x}_0^\top \mid \mathbf{x}_1^\top \mid \mathbf{x}_2^\top)\boldsymbol{\varepsilon}_t)^\top \\
&= b \cdot \lfloor \frac{q}{2} \rfloor + x - \boldsymbol{\varepsilon}_t^\top (\mathbf{x}_0^\top \mid \mathbf{x}_1^\top \mid \mathbf{x}_2^\top)^\top,
\end{aligned}
$$

where $x - \boldsymbol{\varepsilon}_t^\top (\mathbf{x}_0^\top \mid \mathbf{x}_1^\top \mid \mathbf{x}_1^\top)^\top$ is an error term, and it is bounded by: $|x - \boldsymbol{\varepsilon}_t^\top (\mathbf{x}_0^\top \mid \mathbf{x}_1^\top \mid \mathbf{x}_2^\top)^\top| \leq |x| + |(x - \boldsymbol{\varepsilon}_t^\top (\mathbf{x}_0^\top \mid \mathbf{x}_1^\top \mid \mathbf{x}_2^\top)^\top| \leq q\alpha\sigma(l+\tau+1)m\omega(\sqrt{\log[(l+\tau+1)m]}) + \mathcal{O}(\sigma(l+\tau+1)m)$. To recover b correctly, $|x - \boldsymbol{\varepsilon}_t^\top (\mathbf{x}_0^\top \mid \mathbf{x}_1^\top \mid \mathbf{x}_2^\top)^\top| < \frac{q}{5}$ needs to be fulfilled [41]. Then, we can obtain $b' = 1$.

- If $\mathbf{ck} \neq \mathbf{tk}$, we can obtain $b' = 1$ with negligible probability.

The other proof is omitted by us since it is similar to Section 5.2.

6.3 Security Analysis

Theorem 3. Assume that the $\mathrm{LWE}_{n,m,q,\chi}$ hardness holds, our proposed lattice-based FS-BAEKS primitive satisfies IND-CKA security in the random oracle model. For any PPT adversary $\mathcal{A}$, if $\mathcal{A}$ can compromise our scheme with a non-negligible advantage ϵ_2, then we can construct a PPT challenger $\mathcal{C}$ to solve the $\mathrm{LWE}_{n,m,q,\chi}$ hardness with a non-negligible probability.

Proof The constructions between our BAEKS and FS-BAEKS are high symmetric, which only additionally introduced the time period t. Thus, this proof is omitted by us since it is similar to Theorem 1. □

Theorem 4. Assume that the $\mathrm{SIS}_{n,m,q,\beta}$ hardness holds, our proposed lattice-based FS-BAEKS primitive satisfies UF-IKGA security in the random oracle model. For any PPT adversary $\mathcal{A}$, if $\mathcal{A}$ can compromise our scheme, then we can construct a PPT challenger $\mathcal{C}$ to solve the $\mathrm{SIS}_{n,m,q,\beta}$ hardness.

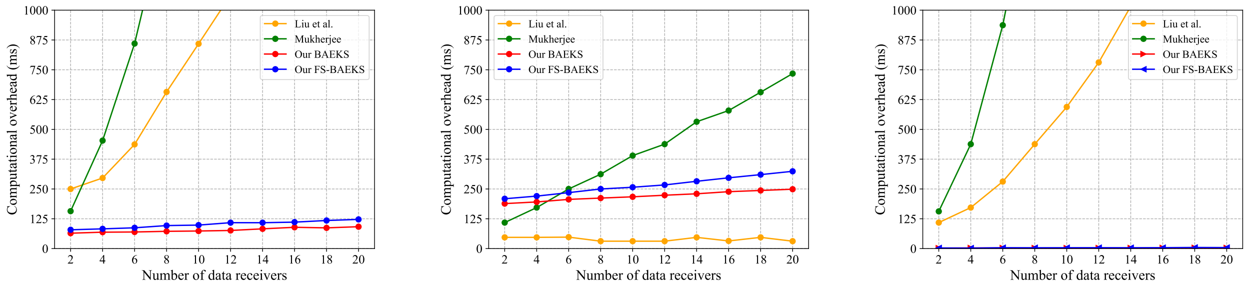
Proof The constructions between our BAEKS and FS-BAEKS are high symmetric which only additionally introduced the time period t. Thus, this proof is omitted by us since it is similar to Theorem 2. □

7 Performance Evaluation and Comparison with Prior Arts

We conduct a comparative analysis of the proposed BAEKS & FS-BAEKS schemes with other state-of-the-art BAEKS primitives in terms of computational and communication overhead. Our BAEKS and FS-BAEKS schemes were implemented in Python language with Numpy library, and all simulation experiments are accomplished on a laptop with 12-th Gen Intel(R) Core(TM) i7-12800HX CPU with 16 GB RAM under Windows 10. We set the parameters of our BAEKS and FS-BAEKS schemes as described in Section 5.2 and 6.2, respectively. For our BAEKS & FS-BAEKS schemes, we set $q = 4096$, $k = 1000$. When $n = 128$, we set $m = 7680$. When $n = 256$, we set $m = 15360$. Moreover, for schemes [21] and [22], the bilinear pairing is initialized by Type A elliptic curves: $y^2 = x^3 + x$, and the parameter $p = 512$.

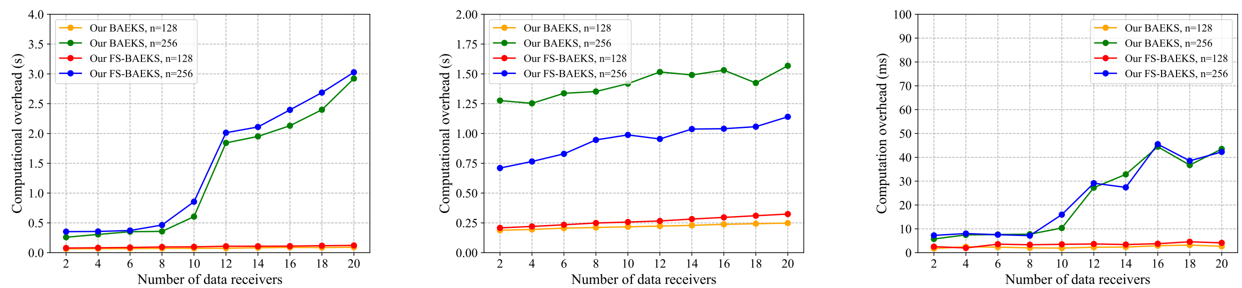
7.1 Computational overhead

As depicted in Fig. 4, we evaluate the computational overhead of our BAEKS & FS-BAEKS schemes compared to the current state-of-the-art BAEKS schemes [21], [22] at **BAEKS**, **Trapdoor** and **Test** algorithms. In Fig. 4(a), the computational overhead of our **BAEKS** algorithm is more efficient than prior arts [21], [22]. In detail, when $l = 6$, our BAEKS scheme requires only $69.55ms$ to encrypt the keywords, while the others require $437ms$ and $860ms$, respectively. Therefore, our BAEKS scheme is approximately $6\times$ and $12\times$ faster than [21] and [22]. Additionally, as the number of data receivers increases, our advantage will be further extended. Furthermore, the computational overhead of **BAEKS** algorithm in our BAEKS & FS-BAEKS schemes is directly proportional to the number of data receivers and has a very moderate growth rate. This growth rate is sufficient to support search operations with a large number of data receivers in cloud storage systems. As for Fig. 4(b), the computational overhead of **Trapdoor** algorithm in our BAEKS & FS-BAEKS schemes is slightly higher than that of [21] due to the sampling algorithm in lattice. However, our schemes offers a significant advantage over [22] as the number of data receivers increases. For instance, when $l = 20$, our BAEKS scheme only requires $248.99ms$ to generate a search trapdoor, which is approximately $2\times$ quicker than [22]. In Fig. 4(c), the computational overhead of **Test** algorithm in our BAEKS & FS-BAEKS schemes remains relatively constant as the number of data receivers l increases. To be more specific, when $l = 20$, the execution time in our BAEKS and FS-BAEKS schemes are only $2.68ms$ and $4.12ms$ respectively, which is approximately $120\times$ and $402\times$ quicker than prior arts [21], [22]. It is evident that our solutions significantly increases performance for the search operations with large amounts of cloud data.



(a) **BAEKS** algorithm (b) **Trapdoor** algorithm (c) **Test** algorithm

Fig. 4. Computational overhead comparison between our BAEKS & FS-BAEKS schemes and other BAEKS schemes [21], [22].



(a) **BAEKS** algorithm (b) **Trapdoor** algorithm (c) **Test** algorithm

Fig. 5. Computational overhead evaluation of our BAEKS & FS-BAEKS schemes with the number of data receivers l and security parameter n.

TABLE 3
Computational overhead evaluation

Schemes	**BAEKS** (s)		**Trapdoor** (s)		**Test** (ms)	
	$n = 128$	$n = 256$	$n = 128$	$n = 256$	$n = 128$	$n = 256$
Our BAEKS	0.091	2.921	0.248	1.568	2.68	43.51
Our FS-BAEKS	0.122	3.027	0.324	1.140	4.12	42.27

Subsequently, we evaluate the computational overhead of our BAEKS & FS-BAEKS schemes with different security parameters n in Fig. 5. It can be found that the computational overhead of **BAEKS**, **Trapdoor**, and **Test** algorithms reasonably increases as n changes from 128 to 256. Specifically, the computational overhead at $l = 20$ is presented in Table 3, which remains in the magnitude of milliseconds. Although the increase in the security parameter n may lead to a decrease in the efficiency, our BAEKS & FS-BAEKS schemes still maintain a significant advantage over [21] and [22] in terms of the **BAEKS** and **Test** algorithms. Moreover, the post-quantum security strength of our schemes is further enhanced, which is crucial for protecting the data privacy in cloud storage systems.

Notably, since the **Setup**, **KeyGen**$_S$, and **KeyGen**$_R$ algorithms are executed less frequently than the **BAEKS**, **Trapdoor**, and **Test** algorithms in real-world applications, which have little relevance to the search efficiency in cloud storage systems. Consequently, we only consider the **BAEKS**, **Trapdoor**, and **Test** algorithms for evaluation and comparison.

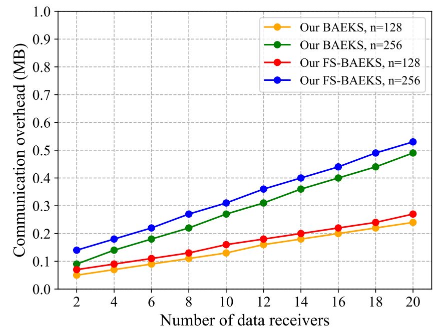
TABLE 4
Communication overhead comparison

Schemes	**BAEKS**	**Trapdoor**
Liu et al. [21]	$\lvert\mathbb{Z}_p\rvert + (t+2)\lvert\mathbb{G}_1\rvert$	$\lvert\mathbb{Z}_p\rvert$
Mukherjee [22]	$(l+1)(k+1)\lvert\mathbb{G}_1\rvert + l\lvert\mathbb{G}_T\rvert$	$2(k+1)\lvert\mathbb{G}_2\rvert$
Our BAEKS	$[(l+2)m+1]\lvert\mathbb{Z}_q\rvert + 2m$	$(l+n+1)m\lvert\mathbb{Z}_q\rvert$
Our FS-BAEKS	$[(l+\tau+2)m+1]\lvert\mathbb{Z}_q\rvert + 2m$	$(l+\tau+n+1)m\lvert\mathbb{Z}_q\rvert$

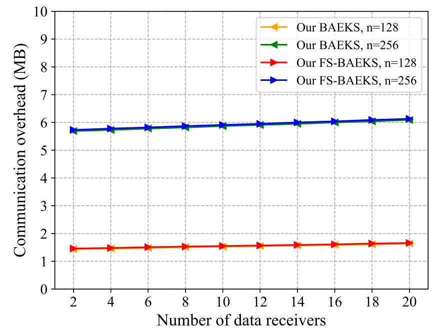
7.2 Communication overhead

For a BAEKS scheme, the transmission of keyword ciphertexts and search trapdoors among data senders, data receivers, and cloud server contributes to the communication overhead. This overhead relies on the size of the ciphertexts and trapdoors. In this way, we provide a theoretical comparison analysis of the communication overhead between our BAEKS & FS-BAEKS schemes and other state-of-the-art schemes [21] and [22] in Table 4, where $|\mathbb{G}_1|$, $|\mathbb{G}_2|$, $|\mathbb{G}_T|$, $|\mathbb{Z}_p|$, and $|\mathbb{Z}_q|$ represent the bit length of elements in $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$, $\mathbb{Z}_p$, and $\mathbb{Z}_q$, respectively.

Our scheme is based on lattice hardness, involving sampling operations on high-dimensional matrices, which is

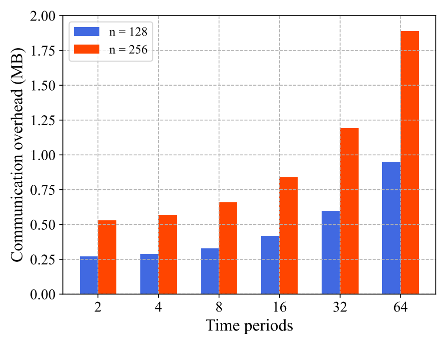


(a) **BAEKS** algorithm

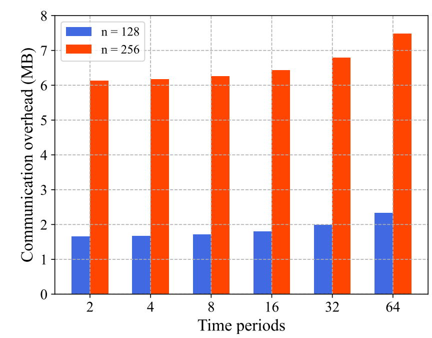


(b) **Trapdoor** algorithm

Fig. 6. Communication overhead evaluation of our BAEKS & FS-BAEKS schemes with the number of data receivers l and security parameter n.



(a) **BAEKS** algorithm



(b) **Trapdoor** algorithm

Fig. 7. Communication overhead evaluation of our FS-BAEKS with the time period t and security parameter n.

TABLE 5
Communication overhead evaluation

Schemes	**BAEKS** (MB)		**Trapdoor** (MB)	
	$n = 128$	$n = 256$	$n = 128$	$n = 256$
Our BAEKS	0.24	0.49	1.64	6.09
Our FS-BAEKS	0.27	0.53	1.66	6.13

different from the underlying constructions based on DL hardness. In this way, the size of our ciphertexts and trapdoors is larger than that of traditional DL-based schemes, which is a common issue. Therefore, as an acceptable trade-off for enhancing the security level to resist quantum computing attacks and secret key leakage attacks, the communication overhead of our BAEKS & FS-BAEKS schemes is higher compared to [21] and [22]. However, in cloud storage systems, BAEKS entities typically prioritize two aspects. Firstly, BAEKS scheme enjoys quantum-safety. Secondly, the computational operations involved in ciphertext generation, trapdoor generation, and search processes are efficient. Accordingly, it can be observed that our schemes introduce an acceptable communication overhead while ensuring post-quantum security and computational efficiency.

Fig. 6 illustrates the communication overhead of **BAEKS** and **Trapdoor** algorithms regarding to the security parameters $n = 128$ and $n = 256$, corresponding to the ciphertext and trapdoor size, respectively. The communication overhead of these two algorithms rises linearly as l is augmented. Moreover, the increment of the security parameter n does not produce an order-of-magnitude increase in the communication overhead, indicating that our BAEKS & FS-BAEKS schemes are scalable. As for Table 5, we give a specific communication overhead of our BAEKS & FS-BAEKS schemes at $l = 20$ and $\tau = 2$, e.g., when $n = 256$, the communication overhead of **BAEKS** algorithm in our FS-BAEKS scheme is $[(20+2+2) \times 15360+1] \times 12+2 \times 15360 \approx 0.53$MB.

For our FS-BAEKS scheme, the communication overhead of **BAEKS** and **Trapdoor** algorithms with τ is shown in Fig. 7 with setting the number of data receivers $l = 20$. Although the communication overhead is raised as t is increased, our FS-BAEKS scheme achieves forward security, has the ability to solve the secret key leakage attacks in cloud storage systems, and is more oriented to practicality. On the other hand, more larger security parameter n leads to a more pronounced trend in the communication overhead with time period t. It is acceptable for boosting the post-quantum security strength of our FS-BAEKS scheme.

8 Conclusion

In this paper, we propose a lattice-based broadcast authenticated encryption with keyword search (BAEKS) scheme. This scheme is designed to provide secure and efficient ciphertext search in multi-receiver model for cloud storage systems. To further enhance its security, we a forward-secure version of BAEKS called FS-BAEKS, which has the ability to mitigate secret key leakage problems. Our rigious security analysis demonstrates that both BAEKS and FS-BAEKS achieve IND-CKA and UF-IKGA security in the ROM. The comprehensive experimental evaluations also indicate that our proposed BAEKS & FS-BAEKS schemes offer significant advantages at computational efficiency of BAEKS and Test algorithms. In particular, when the number of data receivers is six, the best computational overhead of these two algorithms in our BAEKS scheme delivers up approximately 12-x and 402-x faster over current state-of-the-art BAEKS schemes, respectively. However, we acknowledge that further work is required to enhance the security level from the ROM to the standard model.

References

[1] J.-N. Liu, X. Luo, J. Weng, A. Yang, X. A. Wang, M. Li, and X. Lin, "Enabling efficient, secure and privacy-preserving mobile cloud storage," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 3, pp. 1518–1531, 2022.

[2] N. Andola, R. Gahlot, V. K. Yadav, S. Venkatesan, and S. Verma, "Searchable encryption on the cloud: a survey," *The Journal of Supercomputing*, vol. 78, no. 7, pp. 9952–9984, 2022.

[3] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Advances in Cryptology-EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004. Proceedings 23*. Springer, 2004, pp. 506–522.

[4] C. Bösch, P. Hartel, W. Jonker, and A. Peter, "A survey of provably secure searchable encryption," *ACM Computing Surveys (CSUR)*, vol. 47, no. 2, pp. 1–51, 2014.

[5] Q. Zheng, S. Xu, and G. Ateniese, "Vabks: Verifiable attribute-based keyword search over outsourced encrypted data," in *IEEE INFOCOM 2014-IEEE conference on computer communications*. IEEE, 2014, pp. 522–530.

[6] B. Chen, L. Wu, S. Zeadally, and D. He, "Dual-server public-key authenticated encryption with keyword search," *IEEE Transactions on Cloud Computing*, vol. 10, no. 1, pp. 322–333, 2019.

[7] J. Baek, R. Safavi-Naini, and W. Susilo, "Public key encryption with keyword search revisited," in *Computational Science and Its Applications–ICCSA 2008: International Conference, Perugia, Italy, June 30–July 3, 2008, Proceedings, Part I 8*. Springer, 2008, pp. 1249–1259.

[8] H. S. Rhee, J. H. Park, W. Susilo, and D. H. Lee, "Trapdoor security in a searchable public-key encryption scheme with a designated tester," *Journal of Systems and Software*, vol. 83, no. 5, pp. 763–771, 2010.

[9] H. S. Rhee, J. H. Park, and D. H. Lee, "Generic construction of designated tester public-key encryption with keyword search," *Information Sciences*, vol. 205, pp. 93–109, 2012.

[10] R. Chen, Y. Mu, G. Yang, F. Guo, X. Huang, X. Wang, and Y. Wang, "Server-aided public key encryption with keyword search," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 12, pp. 2833–2842, 2016.

[11] 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.

[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] ——, "Public-key authenticated encryption with keyword search: Cryptanalysis, enhanced security, and quantum-resistant instantiation," in *Proceedings of the 2022 ACM on Asia conference on computer and communications security*, 2022, pp. 423–436.

[14] 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.

[15] S. Xu, Y. Cao, X. Chen, S.-M. Yiu, and Y. Zhao, "Post-quantum public-key authenticated searchable encryption with forward security: General construction, implementation, and applications," *Cryptology ePrint Archive*, 2023.

[16] L. Cheng and F. Meng, "Server-aided public key authenticated searchable encryption with constant ciphertext and constant trapdoor," *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 1388–1400, 2023.

[17] L. Yao, J. Weng, A. Yang, X. Liang, Z. Wu, Z. Jiang, and L. Hou, "Scalable cca-secure public-key authenticated encryption with keyword search from ideal lattices in cloud computing," *Information Sciences*, vol. 624, pp. 777–795, 2023.

[18] A. Kiayias, O. Oksuz, A. Russell, Q. Tang, and B. Wang, "Efficient encrypted keyword search for multi-user data sharing," in *Computer Security–ESORICS 2016: 21st European Symposium on Research in Computer Security, Heraklion, Greece, September 26-30, 2016, Proceedings, Part I 21*. Springer, 2016, pp. 173–195.

[19] X. Chen, S. Xu, T. Qin, Y. Cui, S. Gao, and W. Kong, "Aq–abs: Anti-quantum attribute-based signature for emrs sharing with blockchain," in *2022 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2022, pp. 1176–1181.

[20] X. Chen, S. Xu, Y. He, Y. Cui, J. He, and S. Gao, "Lfs-as: lightweight forward secure aggregate signature for e-health scenarios," in *ICC 2022-IEEE International Conference on Communications*. IEEE, 2022, pp. 1239–1244.

[21] 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.

[22] S. Mukherjee, "Statistically consistent broadcast authenticated encryption with keyword search: Adaptive security from standard assumptions," in *Australasian Conference on Information Security and Privacy*. Springer, 2023, pp. 523–552.

[23] K. Emura *et al.*, "Generic construction of fully anonymous broadcast authenticated encryption with keyword search with adaptive corruptions," *IET Information Security*, vol. 2023, 2023.

[24] C. G. Günther, "An identity-based key-exchange protocol," in *Advances in Cryptology—EUROCRYPT'89: Workshop on the Theory and Application of Cryptographic Techniques Houthalen, Belgium, April 10–13, 1989 Proceedings 8*. Springer, 1990, pp. 29–37.

[25] X. Yu, L. Xu, X. Huang, and C. Xu, "An efficient lattice-based encrypted search scheme with forward security," in *International Conference on Network and System Security*. Springer, 2022, pp. 712–726.

[26] M. Ali, H. Ali, T. Zhong, F. Li, Z. Qin, and A. A. Abdelrahaman, "Broadcast searchable keyword encryption," in *2014 IEEE 17th International Conference on Computational Science and Engineering*. IEEE, 2014, pp. 1010–1016.

[27] X. Zhang, C. Xu, H. Wang, Y. Zhang, and S. Wang, "Fs-peks: Lattice-based forward secure public-key encryption with keyword search for cloud-assisted industrial internet of things," *IEEE Transactions on dependable and secure computing*, vol. 18, no. 3, pp. 1019–1032, 2019.

[28] X. Yang, X. Chen, J. Huang, H. Li, and Q. Huang, "Fs-ibeks: Forward secure identity-based encryption with keyword search from lattice," *Computer Standards & Interfaces*, vol. 86, p. 103732, 2023.

[29] J. W. Byun, H. S. Rhee, H.-A. Park, and D. H. Lee, "Off-line keyword guessing attacks on recent keyword search schemes over encrypted data," in *Workshop on secure data management*. Springer, 2006, pp. 75–83.

[30] A. Fiat and M. Naor, "Broadcast encryption," in *Advances in Cryptology—CRYPTO'93: 13th Annual International Cryptology Conference Santa Barbara, California, USA August 22–26, 1993 Proceedings 13*. Springer, 1994, pp. 480–491.

[31] C. Delerablée, "Identity-based broadcast encryption with constant size ciphertexts and private keys," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2007, pp. 200–215.

[32] D. Boneh and M. Hamburg, "Generalized identity based and broadcast encryption schemes," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2008, pp. 455–470.

[33] C. Gentry and B. Waters, "Adaptive security in broadcast encryption systems (with short ciphertexts)," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2009, pp. 171–188.

[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] 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.
[38] D. Cash, D. Hofheinz, and E. Kiltz, “How to delegate a lattice basis,” *Cryptology ePrint Archive*, 2009.
[39] 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.
[40] V. Lyubashevsky, “Lattice signatures without trapdoors,” in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2012, pp. 738–755.
[41] 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.