Enabling Lattice-based Authentication Encrypted Search with Ciphertext Broadcast for Cloud Storage

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

***Abstract*—The development of cloud computing facilitates data outsourced sharing and storage, but also brings up several security issues. Public key authenticated encryption with keyword search (PAEKS) enables the encrypted search over cloud data while resisting the insider keyword guessing attacks (IKGAs). However, existing PAEKS schemes are limited to a single receiver, restricting application prospects in cloud storage. In addition, quantum computing attacks and key leakage issues further threaten the data security, which attracted extensive attention from researchers. Therefore, designing an encrypted search scheme to resist the above-mentioned attacks is still far-reaching. In this paper, we first propose BroSearch, a lattice-based authentication encrypted search with ciphertext broadcast. It utilizes lattice sampling algorithms to authenticate the keyword and offers searchability over broadcasting ciphertext while enjoying IKGAs-resistant in a quantum setting. To get around key leakage issues, we then incorporate the minimal cover set technique and lattice basis extension algorithm to construct FS-BroSearch, as an enhanced version. Furthermore, we give rigorous security analysis (IND-CKA and IND-IKGA) and comprehensive performance evaluation of both schemes. Specifically, the time cost of BroSearch is at least 0.61, 0.82, and 0.83 times compared to prior arts in terms of ciphertext calculation, trapdoor generation, and search procedures, which is practical and effective for cloud storage.**

***Index Terms*—Cloud storage, encrypted search, ciphertext broadcast, keyword authentication, lattice, forward security.**

I. Introduction

In recent years, more and more organizations have outsourced their data to cloud servers to reduce storage maintenance and enhance service elasticity [1], [2]. However, as semi-honest entities, cloud servers are vulnerable to a variety of malicious attacks, including internal keyword guessing attacks (IKGAs), which pose significant risks to data privacy. To alleviate this issue, public key authenticated encryption with keyword search (PAEKS) [3] has been widely studied. It involves encrypting keywords while authenticating them using a secret key of the data sender, enabling secure encrypted search and resisting IKGAs, thereby achieving the privacy-preserving of cloud data.

Most PAEKS primitives are primarily designed for a single-receiver model [3], [4], [5], [6], [7], [8], which involves four key entities: the data sender, the data receiver, the trusted authority, and the cloud server, as illustrated in Fig. 1. However, the expanding scale of cloud computing exposes significant limitations for this model. In most cases, the same data is distributed to multiple receivers simultaneously [9]. For instance, in a healthcare cloud service, the encrypted medical data owned by a specific patient is uploaded to the cloud, and multiple healthcare professionals (e.g. doctors, nurses, and pharmacists) need simultaneous access to collaborate effectively. A straightforward approach to mitigating this issue is to perform point-to-point encryption repeatedly. Nonetheless, the computational overhead grows significantly as the number of receivers increases. A more efficient alternative is broadcast authenticated encryption with keyword search (BAEKS), which has been formalized in previous studies [10], [11], [12]. BAEKS can broadcast a keyword ciphertext on a public channel, allowing a set of receivers to search it without incurring the additional computational overhead associated with multiple point-to-point encryptions.

Unfortunately, existing BAEKS schemes still face two challenges. On the one hand, these schemes rely on classical hardness assumptions (e.g. the discrete logarithm hardness), which are vulnerable to quantum computing attacks. Lattice-based cryptography, a quantum-resistant primitive widely adopted by researchers for data privacy-preserving in cloud storage systems [5], [6], [7], [8], [13], [14], [15], [16], offers a promising solution to this issue. On the other hand, in practical cloud applications, a malicious adversary could calculate a trapdoor corresponding to a specific keyword if it gains access to a data receiver's secret key. Then, the adversary sends it to the cloud server, allowing it to match the keyword ciphertext and thereby significantly compromise keyword security. To address this concern, numerous researchers have introduced the concept of forward security to several cryptographic systems [15], [17], [7], [18], but there does not exist a forward secure BAEKS scheme as so far.

Given this, IKGAs, quantum computing attacks, and key leakage issues are three serious threats that need to be addressed urgently in cloud storage systems. This situation leads us to the main questions addressed in this work:

Can we design an encrypted search scheme with ciphertext broadcast to resist above-mentioned threats?

In this paper, we address the aforementioned question in two milestones. Firstly, we propose BroSearch, a lattice-

Y. Cao and X.-B. Chen are with the Information Security Center, State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China. (E-mail: caoyibo@bupt.edu.cn, flyover100@163.com).

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, and also with the Information Security Center, State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China. (E-mail: gx@ncut.edu.cn).

Z. Li is with Institute for Network Sciences and Cyberspace, Tsinghua University, Beijing, 100084, China. (E-mail: zongpeng@tsinghua.edu.cn).

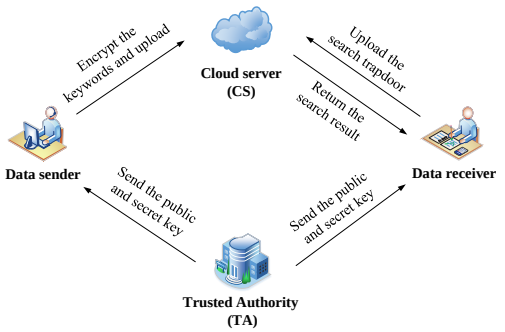


Fig. 1. A single-receiver encrypted search model for cloud storage.

based authentication encrypted search scheme with ciphertext broadcast for cloud storage systems. BroSearch can broadcast the keyword ciphertext to a set of receivers while preserving the functionality of encrypted search. Additionally, it offers enhanced data privacy-preserving in complex cloud storage environments, as it is resistant to both IKGAs and quantum computing attacks. To resist IKGAs, we embed a public matrix $\mathbf{U}$ into the ciphertext and trapdoor, while invoking SampleLeft algorithm to achieve keyword authentication. However, existing lattice-based PAEKS schemes [6], [19], [20] do not support ciphertext broadcast capability. Addressing these issues is not trivial. Unlike the above-mentioned schemes, we embed the public keys of all receivers in a broadcast list $\mathcal{L}$ into the ciphertext. Furthermore, during the trapdoor generation process, we innovatively leverage the GenSamplePre algorithm, allowing the calculation of a valid trapdoor by using the secret key of any receiver in $\mathcal{L}$. As a result, BroSearch successfully facilitates both ciphertext broadcast and encrypted search for cloud data.

Secondly, we extend BroSearch to develop the FS-BroSearch scheme in response to the key leakage issue. Inspired by Yu et al. [17], we introduce the binary tree structure, the minimal cover set technique, and a lattice basis extension algorithm to update the secret key. Thus, even if the secret key of a specific receiver is compromised during a given time period, an adversary is unable to generate a valid search trapdoor for any previous time periods, thus FS-BroSearch can mitigate the key leakage issue.

In a nutshell, our contributions are summarized as follows:

- We present BroSearch, a lattice-based authentication encrypted search scheme with ciphertext broadcast for cloud storage. BroSearch enables the simultaneous broadcasting of keyword ciphertext to a set of receivers, while providing resilience against IKGAs and quantum computing attacks. Additionally, we propose an enhanced version of BroSearch, named FS-BroSearch, which mitigates the key leakage issue.
- We construct BroSearch by leveraging lattice algebraic structures and lattice sampling algorithms. Specifically, by invoking the GenSamplePre algorithm initially, each receiver in the broadcast set can generate a valid search trapdoor. Furthermore, in FS-BroSearch, we introduce a binary tree structure, the minimal cover set technique, and a lattice basis extension algorithm to enable time-period representation and facilitate the secret key update for data receivers.
- We provide the IND-CKA and IND-IKGA security models for both schemes, and show a rigorous security analysis to demonstrate that their security can be reduced to the Learning With Errors (LWE) hardness, thereby ensuring their post-quantum security.
- We offer a detailed performance evaluation of BroSearch and FS-BroSearch in terms of both computational and communication overhead. In particular, the time cost of **Encrypt**, **trapdoor** and **Search** algorithms in BroSearch are at least $0.61\times$, $0.82\times$, and $0.83\times$ compared to other state-of-the-art schemes, which is more practical for cloud storage systems.

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

II. Related Works

Huang et al. introduced a public-key authenticated encryption with keyword search (PAEKS) scheme to implement keyword authentication through a data owner's secret key [3]. To achieve scalability for healthcare cloud storage, Cheng et al. presented a server-aided PAEKS scheme, ensuring the size of the ciphertext and trapdoor constant [4]. Chen et al. proposed a public-key authentication encryption with similar data search for pay-per-query, namely PAESS, which can prevent cloud servers and data users from colluding to deny chargebacks [21]. Liu et al. put forward a generic construction for PAEKS and an instantiation over lattice to achieve the anti-quantum property [5], and enhanced its security [22]. Furthermore, Cheng et al. pointed out some security issues [22], [12], and constructed two PAEKS schemes over lattice [6]. Following that, Liu et al. integrated attribute-based encryption [19] and proxy re-encryption [20] into lattice-based PAEKS to enhance the practicality for cloud storage systems.

Since encrypted messages can be decrypted by a specified group of authorized users, broadcast encryption (BE), first introduced by Fiat et al. [23], is often considered more practical in cloud storage systems. To enhance anonymity, Baee et al. combined message authentication scheme with beacon encryption, and then put forward an inter-vehicle broadcast authentication with encryption scheme for vehicle-to-vehicle (V2V) communication [24]. After that, Zhang et al. provided an identity-based broadcast proxy re-encryption scheme for fully anonymous data sharing [25]. Yin et al. constructed a new dual-mode identity-based BE scheme, named DM-IBBE, to protect sensitive data in smart contracts [26].

Ali et al. foresaw the combinability of BE and PEKS, and constructed a broadcast SE scheme, which is a novel crypto-

graphic primitive to search the keyword ciphertext encrypted by the public key of a group of specified data users [27]. Enlightened by the concept of PAEKS, Liu et al. constructed the BAEKS cryptographic primitive to resist IKGAs, and the ciphertext and trapdoor security were proved under the DBDH assumption [10]. Mukherjee introduced a stronger security model, and ensured the ciphertext and trapdoor security in the standard model [11]. 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 [12]. However, none of the aforementioned schemes can resist quantum computing attacks, and no post-quantum encrypted search scheme supporting ciphertext broadcast 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 [28]. 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 [17]. 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 [7].

To sum up, there exists a valuable requirement to construct an encrypted search scheme with ciphertext broadcast and extend it under the forward security property for resisting IKGAs, quantum computing attacks, and key leakage issues.

III. Preliminaries

Definition 1: 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: 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:

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

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

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

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: Suppose m independent pairs $(\mathbf{a}_i, b_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, and each sample is governed by the following either one to define the decisional $\mathrm{LWE}_{n,m,q,\chi}$ assumption:

1) Pseudo-random sample: $(\mathbf{a}_i, b_i) = (\mathbf{a}_i, \mathbf{a}_i^\top \mathbf{s} + e_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, where $\mathbf{s}$ is a randomly vector, e_i is an error, and $\mathbf{a}_i$ is an uniform vector.
2) Random sample: Randomly samples from $\mathbb{Z}_q^n \times \mathbb{Z}_q$.

Besides, the decisional $\mathrm{LWE}_{n,m,q,\chi}$ assumption is as hard as the worst-case SIVP and GapSVP problem [29].

Lemma 1: [30] Suppose two positive integers n, m, a prime q, where $m = \Theta(n \log q)$, the $\mathsf{TrapGen}$ algorithm returns a full-rank matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and its basis $\mathbf{T_A} \in \mathbb{Z}^{m\times m}$ over $\Lambda_q^\perp(\mathbf{A})$, such that $\mathbf{A}$ is $\mathsf{negl}(n)$-close to uniform and $\|\widetilde{\mathbf{T_A}}\| = O(\sqrt{n \log q})$ with all but negligible probability in n.

Lemma 2: [31] Suppose four integers n, m, m_1, q, and two matrices $\mathbf{A} \in \mathbb{Z}_q^{n\times m}, \mathbf{M}_1 \in \mathbb{Z}_q^{n\times m_1}$, $\mathbf{T_A} \in \mathbb{Z}_q^{m\times m}$ is a basis of $\Lambda_q^\perp(\mathbf{A})$, $\mathbf{u} \in \mathbb{Z}_q^n$ is a vector, and $\sigma > \|\widetilde{\mathbf{T_A}}\| \cdot \omega(\sqrt{\log(m+m_1)})$, the $\mathsf{SampleLeft}$ algorithm returns a vector $\mathbf{e} \in \mathbb{Z}_q^{m+m_1}$, such that $(\mathbf{A}|\mathbf{M}_1)\mathbf{e} = \mathbf{u} \bmod q$.

Lemma 3: [31] Suppose four integers n, k, m, q, and three matrices $\mathbf{A} \in \mathbb{Z}_q^{n\times k}, \mathbf{B} \in \mathbb{Z}_q^{n\times m}, \mathbf{R} \in \mathbb{Z}^{k\times m}$, $\mathbf{T_B} \in \mathbb{Z}_q^{m\times m}$ is a basis of $\Lambda^\perp(\mathbf{B})$, $\mathbf{u} \in \mathbb{Z}_q^n$ is a vector, and $\sigma > \|\widetilde{\mathbf{T_B}}\| \cdot \|\mathbf{R}\|\omega(\sqrt{\log(m)})$, the $\mathsf{SampleRight}$ algorithm returns a vector $\mathbf{e} \in \mathbb{Z}_q^{m+k}$, such that $(\mathbf{A}|\mathbf{A}\mathbf{R}+\mathbf{B})\mathbf{e} = \mathbf{u} \bmod q$.

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 as follows:

Lemma 4: [32] 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 5: [33] 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 6: [34] Suppose four positive integers n, m, m, k, q, σ, two matrices $\mathbf{A} \in \mathbb{Z}_q^{n\times m}, \mathbf{U} \in \mathbb{Z}_q^{n\times k}$, and a matrix $\mathbf{R} \in \mathbb{Z}^{m\times k}$ sampled from the distribution $\mathcal{D}_\sigma(\Lambda_q^{\mathbf{u}}(\mathbf{A})$ and a matrix $\mathbf{S}$ uniformly selected from $\{-1, 1\}^{m\times m}$, then the followings hold: $\|\mathbf{R}^\top\|_2 \leq \sigma\sqrt{mk}$, $\|\mathbf{R}\|_2 \leq \sigma\sqrt{mk}$, and $\|\mathbf{S}\|_2 \leq 20\sqrt{m}$.

TABLE I
Glossary

Acronym	Definition
l	the number of data receivers
τ	the level number of binary tree
T	the number of time period, where $T = 2^\tau$
$\mathcal{L}$	the broadcast list
$\mathbf{ck}$	the keyword owned by data sender
$\mathbf{tk}$	the keyword to be searched by data receiver
$(\mathbf{pk}_S, \mathbf{sk}_S)$	the public and secret keys of data sender
$(\mathbf{pk}_{R,i}, \mathbf{sk}_{R,i})$	the public and secret keys of data receiver i, where $i \in \mathcal{L}$
CT(resp. CT_t)	the ciphertext (resp. with time period t)
TD(resp. TD_t)	the trapdoor (resp. with time period t)

IV. Framework Description

We provide the system model of our encrypted search scheme for cloud storage, and describe the formal definitions and security models of our BroSearch and FS-BroSearch schemes. Table I clarifies the acronyms and descriptions.

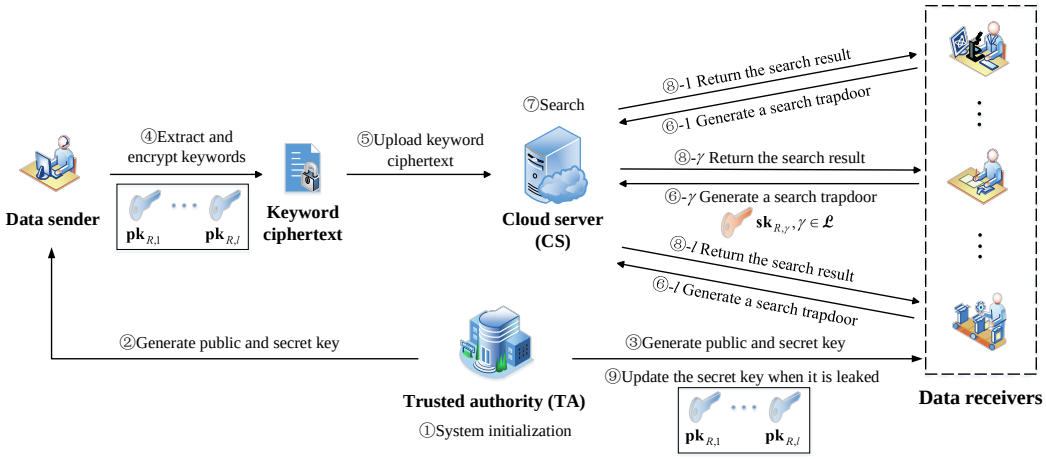


Fig. 2. An encrypted search model with ciphertext broadcast for cloud storage.

A. System Models

The system model in our paper is illustrated in Fig. 2, which contains four participating entities: trusted authority, data sender, data receivers, and cloud server.

1) **Trusted authority (TA)**: TA is charged with initializing the system to obtain the public parameters and calculate the public and secret keys for data sender and receivers. To facilitate trapdoor generation, the public keys of all receivers in the broadcast list are packaged and sent to each receiver. TA is a *fully trusted* entity in this system.
2) **Data sender**: As a *fully trusted* entity, data sender can extract the keywords from data and calculate a valid ciphertext to the cloud server with its secret key and several public keys of the receivers in a broadcast list.
3) **Data receivers**: Since the data receiver's secret key may be stolen by a malicious adversary, it is considered *semi-honest*. When a data receiver in the broadcast list has a search requirement, it generates a trapdoor to the cloud server by its secret key, and then receives a search result.
4) **Cloud server (CS)**: CS is *semi-honest*, upon receiving a keyword ciphertext from a data sender and search trapdoor from a data receiver, CS executes the search procedure to return a search result to the receiver.

B. Formal Definitions

Our BroSearch scheme is defined as $\Pi_{\text{BroSearch}} = (\textbf{Setup}, \textbf{KeyGen}_S, \textbf{KeyGen}_R, \textbf{Encrypt}, \textbf{Trapdoor}, \textbf{Search})$.

- $pp \leftarrow \textbf{Setup}(1^\lambda)$: After inputting a security parameter λ, this algorithm publishes a public parameter pp.
- $(\mathbf{pk}_S, \mathbf{sk}_S) \leftarrow \textbf{KeyGen}_S(pp)$: After inputting the public parameter pp, this algorithm publishes the public and secret keys $(\mathbf{pk}_S, \mathbf{sk}_S)$ of the data sender.
- $(\mathbf{pk}_{R,i}, \mathbf{sk}_{R,i}) \leftarrow \textbf{KeyGen}_R(pp)$: For $i \in \mathcal{L}$, after inputting the public parameter pp, this algorithm publishes the public and secret keys $(\mathbf{pk}_{R,i}, \mathbf{sk}_{R,i})$ of the data receiver i.
- $\text{CT} \leftarrow \textbf{Encrypt}(pp, \mathbf{ck}, \mathbf{pk}_S, \mathbf{sk}_S, \{\mathbf{pk}_{R,i}\}_{i\in\mathcal{L}})$: After inputting the public parameter pp, a keyword $\mathbf{ck} \in \mathbb{Z}_q^n$, the public and secret keys of data sender $(\mathbf{pk}_S, \mathbf{sk}_S)$, the public keys of data receiver in broadcast list $\{\mathbf{pk}_{R,i}\}_{i\in\mathcal{L}}$, the data sender invokes this algorithm to get a ciphertext CT.
- $\text{TD} \leftarrow \textbf{Trapdoor}(pp, \mathbf{tk}, \{\mathbf{pk}_{R,i}\}_{i\in\mathcal{L}}, \mathbf{sk}_{R,\gamma}, \mathbf{pk}_S)$: After inputting the public parameter pp, a keyword $\mathbf{tk} \in \mathbb{Z}_q^n$, the public keys of data receiver in broadcast list $\{\mathbf{pk}_{R,i}\}_{i\in\mathcal{L}}$, a secret key $\mathbf{sk}_{R,\gamma}$ of data receiver γ, and the public key of data sender $\mathbf{pk}_S$, the data receiver γ invokes this algorithm to get the trapdoor TD.
- $1 \text{ or } 0 \leftarrow \textbf{Search}(\text{CT}, \text{TD})$: The server processes this algorithm to test whether CT and TD correspond to the same keyword. If yes, it outputs 1. Otherwise, outputs 0.

Further, our FS-BroSearch scheme is regarded as an enhanced version of BroSearch, it contains seven following algorithms $\Pi_{\text{FS-BroSearch}} = (\textbf{Setup}, \textbf{KeyGen}_S, \textbf{KeyGen}_R, \textbf{KeyUpdate}_R, \textbf{Encrypt}, \textbf{Trapdoor}, \textbf{Search})$, which is similar as $\Pi_{\text{BroSearch}}$, except for the following algorithms.

- $(\mathbf{pk}_{R,i}, \mathbf{sk}_{R,i,0}) \leftarrow \textbf{KeyGen}_R(pp)$: For $i \in \mathcal{L}$, after inputting the public parameter pp, this algorithm publishes the public and initial secret keys $(\mathbf{pk}_{R,i}, \mathbf{sk}_{R,i,0})$ of the data receiver i.
- $\mathbf{sk}_{R,i,t+1} \leftarrow \textbf{KeyUpdate}_R(pp, \mathbf{pk}_{R,i}, \mathbf{sk}_{R,i,t})$: For $i \in \mathcal{L}$, after inputting a public parameter pp, the public and secret keys $(\mathbf{pk}_{R,i}, \mathbf{sk}_{R,i,t})$ of data receiver with time period t, this algorithm publishes its secret key $\mathbf{sk}_{R,i,t+1}$ with time period $t+1$.
- $\text{CT}_t \leftarrow \textbf{Encrypt}(pp, \mathbf{ck}, \mathbf{pk}_S, \mathbf{sk}_S, \{\mathbf{pk}_{R,i}\}_{i\in\mathcal{L}}, t)$: This definition is the same as the **Encrypt** algorithm in BroSearch, except for the introduction of time period t.
- $\text{TD}_t \leftarrow \textbf{Trapdoor}(pp, \mathbf{tk}, \{\mathbf{pk}_{R,i}\}_{i\in\mathcal{L}}, \mathbf{sk}_{R,\gamma,t}, \mathbf{pk}_S, t)$: This definition is the same as the **Trapdoor** algorithm in BroSearch, except for the introduction of time period t.

C. Security Models

We define two security models of our BroSearch scheme: ciphertext indistinguishability against chosen keyword attacks (IND-CKA) and ciphertext indistinguishability against insider keyword guessing attacks (IND-IKGA).

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

1) **Setup**: Given a challenge public key $\mathbf{pk}^*_S = \mathbf{A}_S$ of data sender and several challenge secret key in broadcast list $\{\mathbf{pk}^*_{R,i} = \mathbf{A}_{R,i}\}_{i\in\mathcal{L}^*}$, the challenger $\mathcal{C}$ calls the **Setup**(1^λ) algorithm to calculate a public parameter pp, and sends it to the adversary $\mathcal{A}$.
2) **Phase 1**: $\mathcal{A}$ can adaptively perform four oracles in PPT.
   a) $\mathcal{O}_{H_1}$: $\mathcal{A}$ inputs a keyword $\mathbf{ck}_j$ to issue H_1 queries at most q_{H_1} for $j \in [q_{H_1}]$, $\mathcal{C}$ calculates $H_1(\mathbf{ck}_j)$ and sends it to $\mathcal{A}$.
   b) $\mathcal{O}_{H_2}$: $\mathcal{A}$ inputs a keyword $\mathbf{ck}_j$ to issue H_2 queries at most q_{H_2} for $j \in [q_{H_2}]$, $\mathcal{C}$ calculates $H_2(\mathbf{ck}_j)$ and sends it to $\mathcal{A}$.
   c) $\mathcal{O}_{\text{CT}}$: Given a keyword $\mathbf{ck}$ and the receivers' public keys in broadcast list $\{\mathbf{pk}_{R,i}\}_{i\in\mathcal{L}}$ from $\mathcal{A}$ from $\mathcal{A}$, $\mathcal{C}$ calls **Encrypt**$(pp, \mathbf{ck}, \mathbf{pk}^*_S, \mathbf{sk}^*_S, \{\mathbf{pk}_{R,i}\}_{i\in\mathcal{L}})$ algorithm to generate a ciphertext CT, and sends it to $\mathcal{A}$.
   d) $\mathcal{O}_{\text{TD}}$: After obtained a keyword $\mathbf{tk}$, the sender's public key $\mathbf{pk}_S$ and a receiver $\gamma \in \mathcal{L}$ from $\mathcal{A}$, $\mathcal{C}$ calls **Trapdoor**$(pp, \mathbf{tk}, \{\mathbf{pk}^*_{R,i}\}_{i\in\mathcal{L}^*}, \mathbf{sk}^*_{R,\gamma}, \mathbf{pk}_S)$ algorithm to generate a trapdoor TD for $\mathcal{A}$.
3) **Challenge**: $\mathcal{A}$ chooses two challenge keywords $\mathbf{ck}^*_0, \mathbf{ck}^*_1$ which have not been queried in **Phase 1**, and sends them to $\mathcal{C}$. After that, $\mathcal{C}$ selects a random bit $\xi \in \{0,1\}$ and calls **Encrypt**$(pp, \mathbf{ck}^*_\xi, \mathbf{pk}^*_S, \mathbf{sk}^*_S, \{\mathbf{pk}^*_{R,i}\}_{i\in\mathcal{L}^*})$ 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**: $\mathcal{A}$ outputs a bit $\xi' \in \{0,1\}$. If $\xi' = \xi$, we say that $\mathcal{A}$ wins this game.

We define the advantage of $\mathcal{A}$ to win the above game $\mathbf{Exp}^{\text{IND-CKA}}_{\text{BroSearch},\mathcal{A}}(\lambda)$ as: $\text{Adv}^{\text{IND-CKA}}_{\text{BroSearch},\mathcal{A}}(\lambda) = |\Pr[\xi' = \xi] - \frac{1}{2}|$.

Definition 5: Our BroSearch primitive satisfies IND-CKA security, if any PPT malicious adversary wins the above game $\mathbf{Exp}^{\text{IND-CKA}}_{\text{BroSearch},\mathcal{A}}(\lambda)$ with a negligible advantage.

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

1) **Setup**: This procedure is the same as the corresponding part in $\mathbf{Exp}^{\text{IND-CKA}}_{\text{BroSearch},\mathcal{A}}(\lambda)$.
2) **Phase 1**: This procedure is the same as the corresponding part in $\mathbf{Exp}^{\text{IND-CKA}}_{\text{BroSearch},\mathcal{A}}(\lambda)$.
3) **Challenge**: $\mathcal{A}$ chooses two challenge keywords $\mathbf{tk}^*_0, \mathbf{tk}^*_1$ which have not been queried in **Phase 1**, and sends them to $\mathcal{C}$. After that, $\mathcal{C}$ selects a random bit $\xi \in \{0,1\}$ and calls **Trapdoor**$(pp, \mathbf{tk}^*_\xi, \{\mathbf{pk}^*_{R,i}\}_{i\in\mathcal{L}}, \mathbf{sk}^*_{R,\gamma}, \mathbf{pk}^*_S)$ algorithm to obtain a challenge trapdoor TD^*_ξ. Finally, $\mathcal{C}$ returns TD^*_ξ to $\mathcal{A}$.
4) **Phase 2**: $\mathcal{A}$ executes these queries as above, neither $\mathbf{tk}^*_0$ nor $\mathbf{tk}^*_1$ can be queried.
5) **Guess**: $\mathcal{A}$ outputs a bit $\xi' \in \{0,1\}$. If $\xi' = \xi$, we say that $\mathcal{A}$ wins this game.

We define the advantage of $\mathcal{A}$ to win the above game $\mathbf{Exp}^{\text{IND-IKGA}}_{\text{BroSearch},\mathcal{A}}(\lambda)$ as: $\text{Adv}^{\text{IND-IKGA}}_{\text{BroSearch},\mathcal{A}}(\lambda) = |\Pr[\xi' = \xi] - \frac{1}{2}|$.

Definition 6: Our BroSearch primitive satisfies IND-IKGA security, if any PPT adversary wins the above game $\mathbf{Exp}^{\text{IND-IKGA}}_{\text{BroSearch},\mathcal{A}}(\lambda)$ with a negligible advantage.

Correspondingly, our FS-BroSearch scheme also has two security models, $\mathbf{Exp}^{\text{IND-CKA}}_{\text{FS-BroSearch},\mathcal{A}}(\lambda)$ and $\mathbf{Exp}^{\text{IND-IKGA}}_{\text{FS-BroSearch},\mathcal{A}}(\lambda)$, which are highly symmetric with respect to $\mathbf{Exp}^{\text{IND-CKA}}_{\text{BroSearch},\mathcal{A}}(\lambda)$ and $\mathbf{Exp}^{\text{IND-IKGA}}_{\text{BroSearch},\mathcal{A}}(\lambda)$, except for the addition of the $\mathcal{O}_{\text{KU}}$ oracle and the introduction of time periods. $\mathcal{O}_{\text{KU}}$ is defined as follows:

- $\mathcal{O}_{\text{KU}}$: After obtaining a time period t for $i \in \mathcal{L}$ from $\mathcal{A}$, $\mathcal{C}$ calls **KeyUpdate**$_R(pp, \mathbf{pk}_{R,i}, \mathbf{sk}_{R,i,t-1})$ algorithm to calculate the secret key $\mathbf{sk}_{R,i,t}$ with time period t, and sends it to $\mathcal{A}$.

Based on this, we define the IND-CKA and IND-IKGA security of our FS-BroSearch scheme as follows:

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

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

V. The Design of BroSearch

A. Concrete Construction

The construction of our BroSearch scheme includes five procedures: System Initialization, Key Generation, Ciphertext Calculation, Trapdoor Generation, and Search.

1) System Initialization: TA initializes the entire system by calling the **Setup**(1^λ) algorithm through inputting a security parameter 1^λ. Firstly, it sets several system parameters n, m, q, σ, l, a gadget matrix $\mathbf{G} \in \mathbb{Z}_q^{n\times m}$, and a broadcast list $\mathcal{L} = \{1, \cdots, l\}$. Then, TA chooses two matrices $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and $\mathbf{U} \in \mathbb{Z}_q^{n\times n}$ uniformly. To ensure the security of our scheme in the ROM, two hash functions are defined as $H_1 : \mathbb{Z}_q^n \to \mathbb{Z}_q^{n\times m}$, $H_2 : \mathbb{Z}_q^n \to \mathbb{Z}_q^{n\times n}$. Finally, TA obtains the public parameter and sends it to other entities. The public parameter is defined as:

$$pp := (n, m, q, \sigma, l, \mathbf{G}, \mathcal{L}, \mathbf{A}, \mathbf{U}, H_1, H_2).$$

2) Key Generation: This procedure generates public and secret keys for the data sender and receiver through **KeyGen**$_S(pp)$ and **KeyGen**$_R(pp)$ algorithms, respectively.

For the data sender, TA invokes $(\mathbf{A}_S, \mathbf{T}_{\mathbf{A}_S}) \leftarrow \mathsf{TrapGen}(n, m, q)$ to generate an uniformly matrix $\mathbf{A}_S \in \mathbb{Z}_q^{n\times m}$ and a basis $\mathbf{T}_{\mathbf{A}_S} \in \mathbb{Z}^{m\times m}$ of $\Lambda_q^\perp(\mathbf{A}_S)$. After that, TA obtains the public and secret keys of the data sender:

$$\mathbf{pk}_S := \mathbf{A}_S, \mathbf{sk}_S := \mathbf{T}_{\mathbf{A}_S}.$$

Similarly, for the data receiver $i \in \mathcal{L}$, TA calls $(\mathbf{A}_{R,i}, \mathbf{T}_{\mathbf{A}_{R,i}}) \leftarrow \mathsf{TrapGen}(n, m, q)$ to generate an uniformly matrix $\mathbf{A}_{R,i} \in \mathbb{Z}_q^{n\times m}$ and a basis $\mathbf{T}_{\mathbf{A}_{R,i}} \in \mathbb{Z}^{m\times m}$ of $\Lambda_q^\perp(\mathbf{A}_{R,i})$, and defines the public and secret keys of the data receiver i:

$$\mathbf{pk}_{R,i} := \mathbf{A}_{R,i}, \mathbf{sk}_{R,i} := \mathbf{T}_{\mathbf{A}_{R,i}}.$$

At last, TA returns $(\mathbf{pk}_S, \mathbf{sk}_S)$ and $\{(\mathbf{pk}_{R,i}, \mathbf{sk}_{R,i})\}_{i\in\mathcal{L}}$ to a data sender and several data receivers in broadcast list $\mathcal{L}$ over a secure channel.

3) Ciphertext Calculation: To calculate the keyword ciphertext, the data sender calls the $\mathbf{Encrypt}(pp, \mathbf{ck}, \mathbf{pk}_S, \mathbf{sk}_S, \{\mathbf{pk}_{R,i}\}_{i\in\mathcal{L}})$ algorithm. In this procedure, the data sender inputs its secret key to authenticate the keyword $\mathbf{ck}$, which can resist IKGAs. On the other hand, the public keys of all the data receivers in the broadcast list $\mathcal{L}$ are embedded to enable the ciphertext broadcast.

Specifically, the data sender chooses a vector $\mathbf{s} \in \mathbb{Z}_q^n$ and several error vectors $\{\mathbf{e}_{R,i}\}_{i\in\mathcal{L}} \in \chi^m$, $\mathbf{e}_K \in \chi^m$, $\mathbf{e}_U \in \chi^n$. For $i \in \mathcal{L}$, it calculates a vector $\mathbf{c}_{R,i} = \mathbf{A}_{R,i}^\top \mathbf{s} + \mathbf{e}_{R,i} \in \mathbb{Z}_q^m$ to embed public keys of receivers in $\mathcal{L}$. Subsequently, the data sender calculate two vectors $\mathbf{c}_K = H_1(\mathbf{ck})^\top \mathbf{s} + \mathbf{e}_K \in \mathbb{Z}_q^m$ and $\mathbf{c}_U = \mathbf{U}^\top \mathbf{s} + \mathbf{e}_U \in \mathbb{Z}_q^n$. To authenticate the keyword $\mathbf{ck}$, it invokes the following algorithm to sample a vector $\boldsymbol{\varepsilon}_C \in \mathbb{Z}_q^{2m}$, where $\boldsymbol{\varepsilon}_C$ is statistically distributed in $\mathcal{D}^{2m}_{\Lambda_q^{\mathbf{c}_U}(\mathbf{A}_S|(\mathbf{A}+H_2(\mathbf{ck})\mathbf{G}))}$.

$$\boldsymbol{\varepsilon}_C \leftarrow \mathsf{SampleLeft}(\mathbf{A}_S, \mathbf{A} + H_2(\mathbf{ck})\mathbf{G}, \mathbf{T}_{\mathbf{A}_S}, \mathbf{c}_U, \sigma),$$

such that $(\mathbf{A}_S \mid (\mathbf{A} + H_2(\mathbf{ck})\mathbf{G}))\boldsymbol{\varepsilon}_C = \mathbf{c}_U \bmod q$.

To sum up, the data sender outputs the following ciphertext with the keyword $\mathbf{ck}$, and uploads it to CS.

$$\mathrm{CT} := (\{\mathbf{c}_{R,i}\}_{i\in\mathcal{L}}, \mathbf{c}_K, \boldsymbol{\varepsilon}_C).$$

4) Trapdoor Generation: A data receiver $\gamma \in \mathcal{L}$ performs the $\mathbf{Trapdoor}(pp, \mathbf{tk}, \{\mathbf{pk}_{R,i}\}_{i\in\mathcal{L}}, \mathbf{sk}_{R,\gamma}, \mathbf{pk}_S)$ algorithm to generate a trapdoor with an interested keyword $\mathbf{tk}$. This procedure is divided into two phases: the keyword embedding and the receiver's secret key processing.

For the keyword $\mathbf{tk}$, the data receiver chooses a vector $\mathbf{s}' \in \mathbb{Z}_q^n$, several error vectors $\mathbf{e}_S \in \chi^m$, $\mathbf{e}'_U \in \chi^n$, and a matrix $\mathbf{R}'_K \in \{-1, 1\}^{m\times m}$. Next, it calculates three vectors $\mathbf{t}_S = \mathbf{A}_S^\top \mathbf{s}' + \mathbf{e}_S \in \mathbb{Z}_q^m$, $\mathbf{t}_K = (\mathbf{A} + H_2(\mathbf{tk})\mathbf{G})^\top \mathbf{s}' + (\mathbf{R}'_K)^\top \mathbf{e}_S \in \mathbb{Z}_q^m$ and $\mathbf{t}_U = \mathbf{U}\mathbf{s}' + \mathbf{e}'_U \in \mathbb{Z}_q^n$ to embed this keyword.

Moreover, the data receiver generates a vector $\boldsymbol{\varepsilon}_T$ with its secret key $\mathbf{T}_{\mathbf{A}_{R,\gamma}}$ through the following algorithm:

$$\boldsymbol{\varepsilon}_T \leftarrow \mathsf{GenSamplePre}(\mathbf{A}_{R,1} \mid \cdots \mid \mathbf{A}_{R,l} \mid H_1(\mathbf{tk}), \mathbf{T}_{\mathbf{A}_{R,\gamma}}, \{\gamma\}, \mathbf{t}_u, \sigma),$$

such that $(\mathbf{A}_{R,1} \mid \cdots \mid \mathbf{A}_{R,l} \mid H_1(\mathbf{ck}))\boldsymbol{\varepsilon}_T = \mathbf{t}_U \bmod q$, where $\boldsymbol{\varepsilon}_T$ is statistically distributed in $\mathcal{D}^{(l+1)m}_{\Lambda_q^{\mathbf{t}_U}(\mathbf{A}_{R,1}|\cdots|\mathbf{A}_{R,l}|H_1(\mathbf{ck}))}$.

Eventually, the data receiver sends the following trapdoor to CS.

$$\mathrm{TD} := (\mathbf{t}_S, \mathbf{t}_K, \boldsymbol{\varepsilon}_T).$$

If the data receiver is in the broadcast list, it can generate the trapdoor used for the encrypted search by inputting its secret key. Otherwise, the data receiver cannot search on it, since the basis $\mathbf{T}_{\mathbf{A}_{R,\gamma}}$ does not match the matrix $(\mathbf{A}_{R,1} \mid \cdots \mid \mathbf{A}_{R,l} \mid H_1(\mathbf{tk}))$ in $\mathsf{GenSamplePre}$ algorithm.

5) Search: After receiving the trapdoor TD, CS iterates over the keywords CT uploaded on it, and calls the $\mathbf{Search}(\mathrm{CT}, \mathrm{TD})$ algorithm to obtain a search result.

In this procedure, CS parses $\mathrm{CT} = (\{\mathbf{c}_{R,i}\}_{i\in\mathcal{L}}, \mathbf{c}_K, \boldsymbol{\varepsilon}_C)$ and $\mathrm{TD} = (\mathbf{t}_S, \mathbf{t}_K, \boldsymbol{\varepsilon}_T)$, and calculates a number $d = \boldsymbol{\varepsilon}_C^\top(\mathbf{t}_S^\top \mid \mathbf{t}_K^\top)^\top - (\mathbf{c}_{R,1}^\top \mid \cdots \mid \mathbf{c}_{R,l}^\top \mid \mathbf{c}_K^\top)\boldsymbol{\varepsilon}_T \in \mathbb{Z}_q$.

If $|d| < \lfloor \frac{q}{4} \rfloor$, this procedure outputs 1 meaning that CT and TD correspond to the same keyword, and returns the search result to the data receiver. Otherwise, outputs 0.

B. Correctness Analysis

We analyze the correctness of our BroSearch. Given the data sender's public and secret keys $\mathbf{pk}_S = \mathbf{A}_S, \mathbf{sk}_S = \mathbf{T}_{\mathbf{A}_S}$, a keyword $\mathbf{ck} \in \mathbb{Z}_q^n$, and its ciphertext $\mathrm{CT} = (\{\mathbf{c}_{R,i}\}_{i\in\mathcal{L}}, \mathbf{c}_K, \boldsymbol{\varepsilon}_C)$. 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 $\mathbf{tk} \in \mathbb{Z}_q^n$, and corresponding search trapdoor $\mathrm{TD} = (\mathbf{t}_S, \mathbf{t}_K, \boldsymbol{\varepsilon}_T)$. If $\mathbf{ck} = \mathbf{tk}$, we have:

$$\begin{aligned}
d &= \boldsymbol{\varepsilon}_C^\top(\mathbf{t}_S^\top \mid \mathbf{t}_K^\top)^\top - (\mathbf{c}_{R,1}^\top \mid \cdots \mid \mathbf{c}_{R,l}^\top \mid \mathbf{c}_K^\top)\boldsymbol{\varepsilon}_T \\
&= \boldsymbol{\varepsilon}_C^\top((\mathbf{A}_S^\top \mathbf{s}' + \mathbf{e}_S)^\top \mid ((\mathbf{A} + H_2(\mathbf{tk})\mathbf{G})^\top \mathbf{s}' + \mathbf{R}'_K \mathbf{e}_S)^\top)^\top \\
&\quad - ((\mathbf{A}_{R,1}^\top \mathbf{s} + \mathbf{e}_{R,1})^\top \mid \cdots \mid (\mathbf{A}_{R,l}^\top \mathbf{s} + \mathbf{e}_{R,l})^\top \mid \\
&\quad (H_1(\mathbf{ck})^\top \mathbf{s} + \mathbf{e}_K)^\top)\boldsymbol{\varepsilon}_T \\
&= \boldsymbol{\varepsilon}_C^\top(\mathbf{A}_S \mid (\mathbf{A} + H_2(\mathbf{tk})\mathbf{G}))^\top \mathbf{s}' + \boldsymbol{\varepsilon}_C^\top(\mathbf{e}_S^\top \mid \mathbf{e}_S^\top \mathbf{R}'_K) \\
&\quad - \mathbf{s}^\top(\mathbf{A}_{R,1} \mid \cdots \mid \mathbf{A}_{R,l} \mid H_1(\mathbf{ck}))\boldsymbol{\varepsilon}_T - (\mathbf{e}_{R,1}^\top \mid \cdots \mid \mathbf{e}_{R,l}^\top \mid \mathbf{e}_K^\top)\boldsymbol{\varepsilon}_T \\
&= \mathbf{c}_U^\top \mathbf{s}' - \mathbf{s}^\top \mathbf{t}_U + \boldsymbol{\varepsilon}_C^\top(\mathbf{e}_S^\top \mid \mathbf{e}_S^\top \mathbf{R}'_K) - (\mathbf{e}_{R,1}^\top \mid \cdots \mid \mathbf{e}_{R,l}^\top \mid \mathbf{e}_K^\top)\boldsymbol{\varepsilon}_T \\
&= \mathbf{s}^\top \mathbf{U}\mathbf{s}' + \mathbf{e}_U^\top \mathbf{s}' - \mathbf{s}^\top \mathbf{U}\mathbf{s}' - \mathbf{s}^\top \mathbf{e}'_U + \boldsymbol{\varepsilon}_C^\top(\mathbf{e}_S^\top \mid \mathbf{e}_S^\top \mathbf{R}'_K) \\
&\quad - (\mathbf{e}_{R,1}^\top \mid \cdots \mid \mathbf{e}_{R,l}^\top \mid \mathbf{e}_K^\top)\boldsymbol{\varepsilon}_T \\
&= \mathbf{e}_U^\top \mathbf{s}' - \mathbf{s}^\top \mathbf{e}'_U + \boldsymbol{\varepsilon}_C^\top(\mathbf{e}_S^\top \mid \mathbf{e}_S^\top \mathbf{R}'_K) - (\mathbf{e}_{R,1}^\top \mid \cdots \mid \mathbf{e}_{R,l}^\top \mid \mathbf{e}_K^\top)\boldsymbol{\varepsilon}_T \\
&= \mathrm{error},
\end{aligned}$$

where $|\mathrm{error}| \le |\mathbf{e}_U^\top \mathbf{s}'| + |\mathbf{s}^\top \mathbf{e}'_U| + |\boldsymbol{\varepsilon}_C^\top(\mathbf{e}_S^\top \mid \mathbf{e}_S^\top \mathbf{R}'_K)| + |(\mathbf{e}_{R,1}^\top \mid \cdots \mid \mathbf{e}_{R,l}^\top \mid \mathbf{e}_K^\top)\boldsymbol{\varepsilon}_T| \le 2n\chi_m^2 + 40\sigma m^{\frac{3}{2}}\chi_m + (l+1)\sigma m \chi_m < \frac{q}{5}$.

C. Parameters Setting

In this section, we provide the parameter settings as:

- $m \ge \lceil 2n \log q \rceil$ for the $\mathsf{TrapGen}$ lemma.
- $\sigma \ge O(\sqrt{n \log q}) \cdot \omega(\sqrt{\log(l+1)m})$ for $\mathsf{SampleLeft}$ and $\mathsf{GenSamplePre}$ lemmas.
- $\chi_m \ge \sqrt{n} \cdot \omega(\log n)$ for LWE hardness.
- $2n\chi_m^2 + 40\sigma m^{\frac{3}{2}}\chi_m + (l+1)\sigma m \chi_m < \frac{q}{5}$.

D. Security Analysis

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

Theorem 1: Assume that the $\mathrm{LWE}_{n,m,q,\chi}$ hardness holds, our proposed lattice-based BroSearch 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 $\mathcal{A}$ can compromise 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 $\mathcal{C}$.

LWE Instances: $\mathcal{C}$ obtains several LWE instances $\{\mathbf{A}_{R,i}, \mathbf{a}_{R,i}\}_{i\in\mathcal{L}}$, $(\mathbf{A}_K, \mathbf{a}_K)$, which are sampled from either $\mathcal{O}_\mathrm{S}$ or $\mathcal{O}_\$$. If we sample them from $\mathcal{O}_\mathrm{S}$, they satisfy $\{\mathbf{a}_{R,i} = \mathbf{A}_{R,i}^\top \mathbf{s} + \mathbf{e}_{R,i}\}_{i\in\mathcal{L}} \in \mathbb{Z}_q^m$ and $\mathbf{a}_K = \mathbf{A}_K^\top \mathbf{s} + \mathbf{e}_K \in \mathbb{Z}_q^m$ for a secret vector $\mathbf{s} \in \mathbb{Z}_q^n$ and several error vectors $\{\mathbf{e}_{R,i}\}_{i\in\mathcal{L}} \in \mathbb{Z}_q^m$ and $\mathbf{e}_K \in \mathbb{Z}_q^m$. Otherwise, they are random over $\mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^m$.

Init: $\mathcal{C}$ selects a challenge public key $\mathbf{pk}_S^* = \mathbf{A}_S$ of data sender and several challenge secret key in broadcast list $\{\mathbf{pk}_{R,i}^* = \mathbf{A}_{R,i}\}_{i\in\mathcal{L}^*}$.

Setup: $\mathcal{C}$ obtains the public parameter pp through calling $\mathbf{Setup}(1^\lambda)$ algorithm, except that $\mathbf{A} = \mathbf{A}_S\mathbf{R}_K^* - h^*\mathbf{G} \in \mathbb{Z}_q^{n\times m}$ for a random value $h^* \in \mathbb{Z}_q$ and a matrix $\mathbf{R}_K^* \in \{-1,1\}^{m\times m}$, and then sends it to $\mathcal{A}$.

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

- $\mathcal{O}_{H_1}$: $\mathcal{A}$ inputs a keyword $\mathbf{ck}_j$ to issue H_1 queries at most q_{H_1} for $j \in [q_{H_1}]$. Initially, $\mathcal{C}$ creates a empty list L_{H_1}, and selects $\pi_1^* \in [q_{H_1}]$ as a challenge query. For the j-th query, if $j = \pi_1^*$, $\mathcal{C}$ sets $H_1(\mathbf{ck}_j) = \mathbf{A}_K$, selects $\mathbf{T}_{H_1(\mathbf{ck}_j)} \in \mathbb{Z}^{m\times m}$ randomly. Then, $\mathcal{C}$ adds $\{\mathbf{ck}_j, H_1(\mathbf{ck}_j), \mathbf{T}_{H_1(\mathbf{ck}_j)}\}$ to L_{H_1}, and then returns $H_1(\mathbf{ck}_j)$ to $\mathcal{A}$. Otherwise, if $\mathbf{ck}_j$ has been queried, $\mathcal{C}$ returns $H_1(\mathbf{ck}_j)$ in L_{H_1} to $\mathcal{A}$. Otherwise, $\mathcal{C}$ invokes $\mathsf{TrapGen}(n,m,q)$ to generate a matrix $\mathbf{A}_H \in \mathbb{Z}_q^{n\times m}$ and a basis $\mathbf{T}_{\mathbf{A}_H} \in \mathbb{Z}^{m\times m}$ of $\Lambda_q^\perp(\mathbf{A}_H)$. Subsequently, $\mathcal{C}$ sets $H_1(\mathbf{ck}_j) = \mathbf{A}_H$ and $\mathbf{T}_{H_1(\mathbf{ck}_j)} = \mathbf{T}_{\mathbf{A}_H}$, sends $H_1(\mathbf{ck}_j)$ to $\mathcal{A}$, and then adds $\{\mathbf{ck}_j, H_1(\mathbf{ck}_j), \mathbf{T}_{H_1(\mathbf{ck}_j)}\}$ to L_{H_1}.
- $\mathcal{O}_{H_2}$: In this phase, $\mathcal{A}$ issues H_2 queries at most q_{H_2} after inputting a keyword $\mathbf{ck}_j$ for $j \in [q_{H_2}]$. $\mathcal{C}$ first creates a empty list L_{H_2}, and selects $\pi_2^* \in [q_{H_2}]$ as a challenge query. For the j-th query, if $j = \pi_2^*$, $\mathcal{C}$ sets $H_2(\mathbf{ck}_j) = h^*$, adds $\{\mathbf{ck}_j, H_2(\mathbf{ck}_j)\}$ to L_{H_2}, and returns h^* to $\mathcal{A}$. Otherwise, if $\mathbf{ck}_j$ has been queried, $\mathcal{C}$ returns $H_2(\mathbf{ck}_j)$ in L_{H_2} to $\mathcal{A}$. Otherwise, $\mathcal{C}$ chooses a random value $h \in \mathbb{Z}_q$ as $H_2(\mathbf{ck}_j)$, sends it to $\mathcal{A}$, and adds $\{\mathbf{ck}_j, H_2(\mathbf{ck}_j)\}$ to L_{H_2}.
- $\mathcal{O}_{\mathrm{CT}}$: After obtaining a keyword $\mathbf{ck}$ and the receivers' public keys in broadcast list $\{\mathbf{pk}_{R,i}\}_{i\in\mathcal{L}}$ from $\mathcal{A}$, $\mathcal{C}$ calls $\mathbf{Encrypt}(pp, \mathbf{ck}, \mathbf{pk}_S^*, \mathbf{sk}_S^*, \{\mathbf{pk}_{R,i}\}_{i\in\mathcal{L}})$ algorithm to generate a ciphertext CT, and sends it to $\mathcal{A}$.
- $\mathcal{O}_{\mathrm{TD}}$: After obtaining a keyword $\mathbf{tk}$, the sender's public key $\mathbf{pk}_S$ and a receiver $\gamma \in \mathcal{L}$ from $\mathcal{A}$, $\mathcal{C}$ executes the $\mathcal{O}_{H_1}$ and $\mathcal{O}_{H_2}$ oracle to query the hash value of $\mathbf{tk}$. If $H_1(\mathbf{tk}) = \mathbf{A}_K$ or $H_2(\mathbf{tk}) = h^*$, this procedure is aborted by $\mathcal{C}$. Otherwise, $\mathcal{C}$ obtains $\{\mathbf{tk}, H_1(\mathbf{tk}), \mathbf{T}_{H_1(\mathbf{tk})}\}$ and $\{\mathbf{tk}, H_2(\mathbf{tk})\}$ in L_{H_1} and L_{H_2}. Then, $\mathcal{C}$ calculates $\mathbf{t}_S$, $\mathbf{t}_K$ and $\mathbf{t}_U$ as in **Trapdoor** algorithm, and invokes $\boldsymbol{\varepsilon}_T \leftarrow \mathsf{GenSamplePre}(\mathbf{A}_{R,1} \mid \cdots \mid \mathbf{A}_{R,l} \mid H_1(\mathbf{tk}), \mathbf{T}_{H_1(\mathbf{tk})}, \{l+1\}, \mathbf{t}_U, \sigma)$ such that $(\mathbf{A}_{R,1} \mid \cdots \mid \mathbf{A}_{R,l} \mid H_1(\mathbf{tk}))\boldsymbol{\varepsilon}_T = \mathbf{t}_U \bmod q$. Finally, $\mathcal{C}$ returns $\mathrm{TD} = (\mathbf{t}_S, \mathbf{t}_K, \boldsymbol{\varepsilon}_T)$ to $\mathcal{A}$.

Challenge: $\mathcal{A}$ transmits $\mathbf{ck}_0^*, \mathbf{ck}_1^* \in \mathbb{Z}_q^n$ to $\mathcal{C}$, which have not been queried in **Phase 1**. Then, $\mathcal{C}$ chooses $\xi \in \{0,1\}$, and calculates a challenge ciphertext $\mathrm{CT}_\xi^* = (\{\mathbf{c}_{R,i}^*\}_{i\in\mathcal{L}^*}, \mathbf{c}_K^*, \boldsymbol{\varepsilon}_C^*)$, where $\mathbf{c}_{R,i}^* = \mathbf{a}_{R,i}$ for $i \in \mathcal{L}^*$, $\mathbf{c}_K^* = \mathbf{a}_K$ and $\boldsymbol{\varepsilon}_C^* \leftarrow \mathbb{Z}_q^{2m}$. After that, $\mathcal{C}$ returns CT_ξ^* 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 $\mathcal{C}$ outputs 1. Otherwise, $\mathcal{C}$ outputs 0.

Analysis: We demonstrate two cases according to the sampling way of LWE instances. If these instances are sampled from the $\mathcal{O}_{\mathrm{S}}$, we have:

$$\mathbf{c}_{R,i}^* = \mathbf{a}_{R,i} = \mathbf{A}_{R,i}^\top\mathbf{s} + \mathbf{e}_{R,i}, \text{where } i \in \mathcal{L}^*,$$
$$\mathbf{c}_K^* = \mathbf{a}_K = \mathbf{A}_K^\top\mathbf{s} + \mathbf{e}_K = H_1(\mathbf{ck}_\xi^*)^\top\mathbf{s} + \mathbf{e}_K.$$

Thus, $\mathrm{CT}_\xi^* = (\{\mathbf{c}_{R,i}^*\}_{i\in\mathcal{L}^*}, \mathbf{c}_K^*, \boldsymbol{\varepsilon}_C^*)$ is a valid ciphertext. When $\mathcal{A}$ compromise our scheme with advantage ϵ_1, $\Pr[\xi' = \xi|\mathcal{O}_{\mathsf{LWE}} = \mathcal{O}_{\mathrm{S}}] = \frac{1}{2} + \epsilon_1$. Otherwise, CT_ξ^* is uniform over $\mathbb{Z}_q^{lm} \times \mathbb{Z}_q^m \times \mathbb{Z}_q^{2m}$. In this case, $\Pr[\xi' = \xi|\mathcal{O}_{\mathsf{LWE}} = \mathcal{O}_{\$}] = \frac{1}{2}$. Due to the probability of executing this procedure with probability $\frac{q_{H_1}+q_{H_2}-1}{q_{H_1}q_{H_2}}$, the advantage of $\mathcal{C}$ to solve the $\mathrm{LWE}_{n,m,q,\chi}$ hardness is $\frac{(q_{H_1}+q_{H_2}-1)\epsilon_1}{2q_{H_1}q_{H_2}}$, which is also non-negligible. □

Theorem 2: Assume that the $\mathrm{LWE}_{n,m,q,\chi}$ hardness holds, our proposed lattice-based BroSearch scheme satisfies IND-IKGA 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 If a PPT adversary $\mathcal{A}$ can compromise the IND-IKGA 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 $\mathcal{C}$.

LWE Instances: This procedure is the same as *Theorem 1*, except that $(\mathbf{A}_S, \mathbf{a}_S)$ and $(\mathbf{A}_K, \mathbf{a}_K)$ are sampled as LWE instances in this phase.

Init: $\mathcal{C}$ selects a challenge public key $\mathbf{pk}_S^* = \mathbf{A}_S$ of data sender and a challenge secret key $\mathbf{pk}_{R,\gamma}^* = \mathbf{A}_{R,\gamma}$ of data receiver where $\gamma \in \mathcal{L}^*$.

Setup: This procedure is the same as *Theorem 1*.

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

- $\mathcal{O}_{H_1}$: This procedure is the same as *Theorem 1*.
- $\mathcal{O}_{H_2}$: This procedure is the same as *Theorem 1*.
- $\mathcal{O}_{\mathrm{CT}}$: After obtaining a keyword $\mathbf{ck}$ and the receivers' public keys $\{\mathbf{pk}_{R,i}\}_{i\in\mathcal{L}}$ from $\mathcal{A}$, $\mathcal{C}$ executes the $\mathcal{O}_{H_1}$ and $\mathcal{O}_{H_2}$ oracle to query the hash value of $\mathbf{tk}$. If $H_1(\mathbf{tk}) = \mathbf{A}_K$ or $H_2(\mathbf{tk}) = h^*$, this procedure is aborted by $\mathcal{C}$. Otherwise, $\mathcal{C}$ obtains $\{\mathbf{tk}, H_1(\mathbf{tk}), \mathbf{T}_{H_1(\mathbf{tk})}\}$ and $\{\mathbf{tk}, H_2(\mathbf{tk})\}$ in L_{H_1} and L_{H_2}. Then, $\mathcal{C}$ calculates $\{\mathbf{c}_{R,i}\}_{i\in\mathcal{L}}$, $\mathbf{c}_K$ and $\mathbf{c}_U$ as in **Encrypt** algorithm, and invokes $\boldsymbol{\varepsilon}_C \leftarrow \mathsf{SampleRight}(\mathbf{A}_S, (H_2(\mathbf{ck}) - h^*)\mathbf{G}, \mathbf{R}_K^*, \mathbf{T}_\mathbf{G}, \mathbf{c}_U, \sigma)$ such that $(\mathbf{A}_S \mid (\mathbf{A} + H_2(\mathbf{ck})\mathbf{G}))\boldsymbol{\varepsilon}_C = \mathbf{c}_U \bmod q$. Finally, $\mathcal{C}$ returns $\mathrm{CT} = (\{\mathbf{c}_{R,i}\}_{i\in\mathcal{L}}, \mathbf{c}_K, \boldsymbol{\varepsilon}_C)$ to $\mathcal{A}$.
- $\mathcal{O}_{\mathrm{TD}}$: After obtaining a keyword $\mathbf{tk}$, the sender's public key $\mathbf{pk}_S$ and a receiver $\gamma \in \mathcal{L}$ from $\mathcal{A}$, $\mathcal{C}$ calls $\mathbf{Trapdoor}(pp, \mathbf{tk}, \{\mathbf{pk}_{R,i}^*\}_{i\in\mathcal{L}}, \mathbf{sk}_{R,\gamma}^*, \mathbf{pk}_S)$ algorithm to generate a trapdoor TD, and sends it to $\mathcal{A}$.

Challenge: $\mathcal{A}$ transmits $\mathbf{tk}_0^*, \mathbf{tk}_1^* \in \mathbb{Z}_q^n$ to $\mathcal{C}$, which have not been queried in **Phase 1**. Then, $\mathcal{C}$ chooses $\xi \in \{0,1\}$, and calculates a challenge ciphertext $\mathrm{TD}_\xi^* = (\mathbf{t}_S^*, \mathbf{t}_K^*, \boldsymbol{\varepsilon}_T^*)$, where $\mathbf{t}_S^* = \mathbf{a}_S$, $\mathbf{t}_K^* = \begin{pmatrix} \mathbf{a}_S \\ (\mathbf{R}_K^*)^\top\mathbf{a}_S \end{pmatrix}$ and $\boldsymbol{\varepsilon}_T^* \leftarrow \mathbb{Z}_q^{(l+1)m}$. After that, $\mathcal{C}$ returns TD_ξ^* to $\mathcal{A}$.

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

Guess: $\mathcal{A}$ outputs a random bit $\xi' \in \{0,1\}$ after receiving TD^*_ξ. If $\xi' = \xi$, $\mathcal{A}$ wins this game, and $\mathcal{C}$ outputs 1. Otherwise, $\mathcal{C}$ outputs 0.

Analysis: We demonstrate two cases according to the sampling way of LWE instances. If these instances are sampled from the $\mathcal{O}_{\mathrm{S}}$, we have:

$$\begin{aligned}\mathbf{t}^*_S &= \mathbf{a}_S = \mathbf{A}_S^\top \mathbf{s} + \mathbf{e}_S,\\ \mathbf{t}^*_K &= \begin{pmatrix}\mathbf{a}_S\\ (\mathbf{R}^*_K)^\top \mathbf{a}_S\end{pmatrix} = \begin{pmatrix}\mathbf{A}_S^\top \mathbf{s} + \mathbf{e}_S\\ (\mathbf{A}_S \mathbf{R}^*_K)^\top \mathbf{s} + (\mathbf{R}^*_K)^\top \mathbf{e}_S\end{pmatrix}\\ &= (\mathbf{A}_S \mid \mathbf{A} + H_2(\mathbf{tk}^*_\xi)\mathbf{G})^\top \mathbf{s} + (\mathbf{e}_S^\top \mid \mathbf{e}_S^\top \mathbf{R}_K)^\top.\end{aligned}$$

Thus, $\mathrm{TD}^*_\xi = (\mathbf{t}^*_S, \mathbf{t}^*_K, \boldsymbol{\varepsilon}^*_T)$ is a valid trapdoor. Similar to *Theorem 1*, the advantage of $\mathcal{C}$ to solve the $\mathrm{LWE}_{n,m,q,\chi}$ hardness is $\frac{(q_{H_1}+q_{H_2}-1)\epsilon_2}{2q_{H_1}q_{H_2}}$, which is also non-negligible. □

VI. The Design of FS-BroSearch: An Enhanced Version

A. Concrete Construction

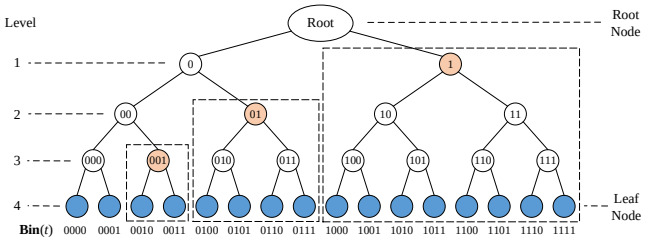


Fig. 3. The binary tree utilized to secret key update for data receivers, and the number of levels $\tau = 4$.

1) System Initialization: In this procedure, TA initializes the entire system by calling the $\mathbf{Setup}(1^\lambda)$ algorithm. It first sets several system parameters n, m, q, σ, l, a gadget matrix $\mathbf{G} \in \mathbb{Z}_q^{n\times m}$, and a broadcast list $\mathcal{L} = \{1, \cdots, l\}$. Secondly, TA initializes all nodes in the binary tree, and sets τ as the depth of the binary tree, where $T = 2^\tau$, as in Fig. 3 (A example at $\tau = 4$). Then, TA chooses several matrices $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, $\mathbf{U} \in \mathbb{Z}_q^{n\times n}$ and $\{\mathbf{A}_j^{(0)}, \mathbf{A}_j^{(1)}\}_{j\in[\tau]} \in \mathbb{Z}_q^{n\times m}$ uniformly. As same as in BroSearch, two hash functions are defined as $H_1 : \mathbb{Z}_q^n \to \mathbb{Z}_q^{n\times m}$ and $H_2 : \mathbb{Z}_q^n \to \mathbb{Z}_q^{n\times n}$. Finally, TA obtains the public parameter and sends it to other entities. The public parameter is defined as:

$$pp := (n, m, q, \sigma, l, \tau, T, \mathbf{G}, \mathcal{L}, \mathbf{A}, \mathbf{U}, \{\mathbf{A}_j^{(0)}, \mathbf{A}_j^{(1)}\}_{j\in[\tau]}, H_1, H_2)$$

2) Key Generation: This procedure generates public and secret keys for the data sender and receiver through $\mathbf{KeyGen}_S(pp)$ and $\mathbf{KeyGen}_R(pp)$ algorithms, respectively.

For the data sender, this procedure is identical to that of BroSearch. TA sets the public and secret keys of the data sender as:

$$\mathbf{pk}_S := \mathbf{A}_S, \mathbf{sk}_S := \mathbf{T}_{\mathbf{A}_S}.$$

For the data receiver $i \in \mathcal{L}$, TA Invoke $(\mathbf{A}_{R,i,0}, \mathbf{T}_{\mathbf{A}_{R,i,0}}) \leftarrow \mathsf{TrapGen}(n, m, q)$ to generate an 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 $\Lambda_q^\perp(\mathbf{A}_{R,i,0})$, and defines the public and initial secret keys of the data receiver i as:

$$\mathbf{pk}_{R,i} := \mathbf{A}_{R,i,0}, \mathbf{sk}_{R,i,0} := \mathbf{T}_{\mathbf{A}_{R,i,0}}$$

At last, TA sends $(\mathbf{pk}_S, \mathbf{sk}_S)$ and $\{(\mathbf{pk}_{R,I,0}, \mathbf{sk}_{R,I,0})\}_{i\in\mathcal{L}}$ to a data sender and several data receivers in broadcast list $\mathcal{L}$ over a secure channel.

3) Key Update: 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 a 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, $\mathsf{Node}(0010) = \{001, 01, 1\}$. The secret key of a data receiver $i \in \mathcal{L}$ with t is defined as the set of bases corresponding to several nodes in $\mathsf{Node}(\mathbf{bin}(t))$.

To achieve forward security, a data receiver $i \in \mathcal{L}$ updates its secret key through the $\mathbf{KeyUpdate}_R(pp, \mathbf{pk}_{R,i}, \mathbf{sk}_{R,i,t})$ algorithm when the secret key is compromised. In this procedure, the data receiver inputs the secret key $\mathbf{sk}_{R,i,t}$ with time period t and obtains $\mathbf{sk}_{R,i,t+1}$ with $t+1$, where $t \in \{0, 1, \cdots, T-2\}$. and the specific procedure can be divided into two cases.

If the time period $t = 0$, the data receiver owns the initial secret key $\mathbf{sk}_{R,i,0} = \mathbf{T}_{\mathbf{A}_{R,i,0}}$ with $t = 0$. Due to $\mathsf{Node}(\mathbf{bin}(1)) = \mathsf{Node}(0001) = \{0001, 001, 01, 1\}$, it obtains $\mathbf{sk}_{R,i,1} = \{\mathbf{T}_{\mathbf{A}_{R,i,0001}}, \mathbf{T}_{\mathbf{A}_{R,i,001}}, \mathbf{T}_{\mathbf{A}_{R,i,01}}, \mathbf{T}_{\mathbf{A}_{R,i,1}}\}$ with $t = 1$ through calling

$$\mathbf{T}_{\mathbf{A}_{R,i,\Theta}} \leftarrow \mathsf{ExtBasis}(\mathbf{A}_{R,i,\Theta}, \mathbf{T}_{\mathbf{A}_{R,i,0}}),$$

where $\Theta \in \{0001, 001, 01, 1\}$.

If the time period $t \geq 1$, the data receiver parses $\mathbf{bin}(t) = (t_1, t_2, \cdots, t_\tau) \in \{0,1\}^\tau$, and calculates the minimal cover set with time period $t+1$, denoted $\mathsf{Node}(\mathbf{bin}(t+1))$. For $\Theta_j = (\theta_1, \cdots, \theta_\zeta, \cdots, \theta_j) \in \mathsf{Node}(\mathbf{bin}(t+1))$, if there exists a basis $\mathbf{T}_{\mathbf{A}_{R,i,\Theta_j}}$ in $\mathbf{sk}_{R,i,t}$, the data receiver adds this basis to $\mathbf{sk}_{R,i,t+1}$. Otherwise, it sets a node $\Theta_\zeta = (\theta_1, \cdots, \theta_\zeta) \in \{0,1\}^\zeta$, which is an ancestor node of Θ_j in $\mathbf{sk}_{R,i,t}$, and then invoke the following algorithm to generate the aforementioned basis $\mathbf{T}_{\mathbf{A}_{R,i,\Theta_j}}$.

$$\mathbf{T}_{\mathbf{A}_{R,i,\Theta_j}} \leftarrow \mathsf{ExtBasis}(\mathbf{A}_{R,i,\Theta_j}, \mathbf{T}_{\mathbf{A}_{R,i,\Theta_\zeta}}),$$

where $\mathbf{A}_{R,i,\Theta_j} = (\mathbf{A}_{R,i,0} | \mathbf{A}_1^{(\theta_1)} | \cdots | \mathbf{A}_j^{(\theta_j)}) \in \mathbb{Z}_q^{n\times(j+1)m}$.

Ultimately, the data receiver i obtains $\mathbf{sk}_{R,i,t+1} := \{\mathbf{T}_{\mathbf{A}_{R,i,\Theta_j}}\}_{\Theta_j \in \mathsf{Node}(\mathbf{bin}(t+1))}$ as the secret key with $t+1$.

4) Ciphertext Calculation: To calculate the keyword ciphertext with time period t, the data sender calls the $\mathbf{Encrypt}(pp, \mathbf{ck}, \mathbf{pk}_S, \mathbf{sk}_S, \{\mathbf{pk}_{R,i}\}_{i\in\mathcal{L}}, t)$ algorithm.

The data sender parses $\mathbf{bin}(t) = (t_1, t_2, \cdots, t_\tau) \in \{0,1\}^\tau$, and chooses a vector $\mathbf{s} \in \mathbb{Z}_q^n$, several error vectors $\{\mathbf{e}_{R,i}\}_{i\in\mathcal{L}} \in \chi^m$, $\{\mathbf{e}_{T,j}\}_{j\in[\tau]} \in \chi^m$, $\mathbf{e}_K \in \chi^m$, $\mathbf{e}_U \in \chi^n$. For $j \in [\tau]$, it calculates several vectors $\mathbf{c}_{T,j} = (\mathbf{A}_j^{(t_j)})^\top \mathbf{s} + \mathbf{e}_{T,j} \in \mathbb{Z}_q^m$ to embed the time period into the ciphertext. As same as in BroSearch, the data sender calculates several vectors $\mathbf{c}_{R,i} = \mathbf{A}_{R,i,0}^\top \mathbf{s} + \mathbf{e}_{R,i} \in \mathbb{Z}_q^m$ for $i \in \mathcal{L}$, $\mathbf{c}_K = H_1(\mathbf{ck})^\top \mathbf{s} + \mathbf{e}_K \in \mathbb{Z}^m$, $\mathbf{c}_U = \mathbf{U}^\top \mathbf{s} + \mathbf{e}_U \in \mathbb{Z}_q^n$ and $\boldsymbol{\varepsilon}_C \leftarrow \mathsf{SampleLeft}(\mathbf{A}_S, \mathbf{A} + H_2(\mathbf{ck})\mathbf{G}, \mathbf{T}_{\mathbf{A}_S}, \mathbf{c}_U, \sigma)$ to achieve the ciphertext broadcast and keyword authentication.

To sum up, the data sender outputs the ciphertext corresponding to the keyword $\mathbf{ck}$ with time period t as:

$$\mathrm{CT}_t := (\{\mathbf{c}_{R,i}\}_{i\in\mathcal{L}}, \{\mathbf{c}_{T,j}\}_{j\in[\tau]}, \mathbf{c}_K, \boldsymbol{\varepsilon}_C).$$

5) Trapdoor Generation: With an interested keyword $\mathbf{tk} \in \mathbb{Z}_q^n$ as input, a data receiver $\gamma \in \mathcal{L}$ executes the **Trapdoor**$(pp, \mathbf{tk}, \{\mathbf{pk}_{R,i}\}_{i\in\mathcal{L}}, \mathbf{sk}_{R,\gamma,t}, \mathbf{pk}_S, t)$ algorithm to obtain a trapdoor with the keyword $\mathbf{tk}$ and time period t.

The date receiver first chooses a vector $\mathbf{s}' \in \mathbb{Z}_q^n$, several error vectors $\mathbf{e}_S \in \chi^m$, $\mathbf{e}'_U \in \chi^n$, and a matrix $\mathbf{R}'_K \in \{-1,1\}^{m\times m}$. Afterward, it calculates three vectors $\mathbf{t}_S = \mathbf{A}_S^\top \mathbf{s}' + \mathbf{e}_S \in \mathbb{Z}_q^m$, $\mathbf{t}_K = (\mathbf{A} + H_2(\mathbf{tk})\mathbf{G})^\top \mathbf{s}' + (\mathbf{R}'_K)^\top \mathbf{e}_S \in \mathbb{Z}_q^m$ and $\mathbf{t}_U = \mathbf{U}\mathbf{s}' + \mathbf{e}'_U \in \mathbb{Z}_q^n$ as the same as in BroSearch.

If $\mathbf{sk}_{R,\gamma,t}$ does not contain $\mathbf{T}_{\mathbf{A}_{R,\gamma,\mathbf{bin}(t)}}$, the data receiver invokes the following algorithms to obtain a basis $\mathbf{T}_{\mathbf{A}_{R,\gamma,\mathbf{bin}(t)}}$ in $\mathbb{Z}^{(\tau+1)m\times(\tau+1)m}$ for $\Lambda_q^\perp(\mathbf{A}_{R,\gamma,0} \mid \mathbf{A}_1^{(t_1)} \mid \cdots \mid \mathbf{A}_\tau^{(t_\tau)})$.

$$\mathbf{T}_{\mathbf{A}_{R,\gamma,\mathbf{bin}(t)}} \leftarrow \mathsf{ExtBasis}(\mathbf{A}_{R,\gamma,0} \mid \mathbf{A}_1^{(t_1)} \mid \cdots \mid \mathbf{A}_\tau^{(t_\tau)}, \mathbf{T}_{\mathbf{A}_{R,\gamma,\Theta_j}}),$$

where Θ_j is an ancestor node of $\mathbf{bin}(t)$ and $j < \tau$.

Following that, the data receiver generates a vector $\boldsymbol{\varepsilon}_T$ statistically distributed in $\mathcal{D}^{(l+\tau+1)m}_{\Lambda_q^{\mathbf{t}_U}(\mathbf{A}_{R,1,0}|\cdots|\mathbf{A}_{R,l,0}|\mathbf{A}_1^{(t_1)}|\cdots|\mathbf{A}_\tau^{(t_\tau)}|H_1(\mathbf{tk}))}$ through the following algorithm:

$$\boldsymbol{\varepsilon}_T \leftarrow \mathsf{GenSamplePre}(\mathbf{A}_{R,1,0} \mid \cdots \mid \mathbf{A}_{R,l,0} \mid \mathbf{A}_1^{(t_1)} \mid \cdots \mid \mathbf{A}_\tau^{(t_\tau)} \mid H_1(\mathbf{tk}), \mathbf{T}_{\mathbf{A}_{R,\gamma,\mathbf{bin}(t)}}, \{\gamma, l+1, \cdots, l+\tau\}, \mathbf{t}_U, \sigma),$$

such that $(\mathbf{A}_{R,1,0} \mid \cdots \mid \mathbf{A}_{R,l,0} \mid \mathbf{A}_1^{(t_1)} \mid \cdots \mid \mathbf{A}_\tau^{(t_\tau)} \mid H_1(\mathbf{tk}))\boldsymbol{\varepsilon}_T = \mathbf{t}_U \bmod q$. Eventually, the data receiver sends the trapdoor $\mathrm{TD}_t := (\mathbf{t}_S, \mathbf{t}_K, \boldsymbol{\varepsilon}_T)$ to cloud server.

6) Search: As similar as in BroSearch, CS calculates a number $d = \boldsymbol{\varepsilon}_C^\top(\mathbf{t}_S^\top \mid \mathbf{t}_K^\top)^\top - (\mathbf{c}_{R,1}^\top \mid \cdots \mid \mathbf{c}_{R,l}^\top \mid \mathbf{c}_{T,1}^\top \mid \cdots \mid \mathbf{c}_{T,\tau}^\top \mid \mathbf{c}_K^\top)\boldsymbol{\varepsilon}_T \in \mathbb{Z}_q$ If $|d| < \lfloor \frac{q}{4} \rfloor$, this procedure outputs 1 meaning that CT and TD correspond to the same keyword, and returns the search result to the data receiver. Otherwise, this procedure outputs 0.

B. Security Analysis

Theorem 3: Assume that the $\mathrm{LWE}_{n,m,q,\chi}$ hardness holds, our proposed lattice-based FS-BroSearch 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_3, 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 We define an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ as in the *Theorem 1*.

LWE Instances: This procedure is the same as *Theorem 1*, except that $\{\mathbf{A}_{R,i}, \mathbf{a}_{R,i}\}_{i\in\mathcal{L}}$, $(\mathbf{A}_K, \mathbf{a}_K)$ and $\{\mathbf{A}_{T,j}, \mathbf{a}_{T,j}\}_{j\in[\tau]}$ are sampled as LWE instances in this phase.

Init: $\mathcal{C}$ selects a challenge time period t^* where $\mathbf{bin}(t^*) = (t_1^*, \cdots, t_\tau^*) \in \{0,1\}^\tau$, a challenge public key $\mathbf{pk}_S^* = \mathbf{A}_S$ of data sender and several challenge secret key in broadcast list $\{\mathbf{pk}_{R,i}^* = \mathbf{A}_{R,i}\}_{i\in\mathcal{L}^*}$.

Setup: Base on *Theorem 1*, we add the generation procedure of several matrices $\{\mathbf{A}_i^{(0)}, \mathbf{A}_i^{(1)}\}_{i\in[\tau]}$. Specifically, $\mathcal{C}$ invokes $(\mathbf{A}_0, \mathbf{T}_{\mathbf{A}_0}) \rightarrow \mathsf{TrapGen}(n, m, q)$ to generate an uniformly matrix $\mathbf{A}_0 \in \mathbb{Z}_q^{n\times m}$ and a basis $\mathbf{T}_{\mathbf{A}_0} \in \mathbb{Z}^{m\times m}$ of $\Lambda_q^\perp(\mathbf{A}_0)$. For the node $\mathcal{T}_j = (t_1, \cdots, t_j)$ and $j = [\tau]$, if $\mathcal{T}_j = (t_1^*, \cdots, t_j^*)$, $\mathcal{C}$ sets $\mathbf{A}_j^{(t_j)} = \mathbf{A}_{T,j}$. Otherwise, $\mathcal{C}$ invokes $(\mathbf{A}_j^{(t_j)}, \mathbf{T}_{\mathbf{A}_j^{(t_j)}}) \rightarrow \mathsf{TrapGen}(n, m, q)$ to generate an uniformly matrix $\mathbf{A}_j^{(t_j)} \in \mathbb{Z}_q^{n\times m}$ and a basis $\mathbf{T}_{\mathbf{A}_j^{(t_j)}} \in \mathbb{Z}^{m\times m}$ of $\Lambda_q^\perp(\mathbf{A}_j^{(t_j)})$.

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

- $\mathcal{O}_{H_1}$: This procedure is the same as *Theorem 1*.
- $\mathcal{O}_{H_2}$: This procedure is the same as *Theorem 1*.
- $\mathcal{O}_{\mathrm{KU}}$: After obtaining a time period t for $i \in \mathcal{L}$ from $\mathcal{A}$, $\mathcal{C}$ calculates the secret key $\mathbf{sk}_{R,i,t}$ with time period t. Specifically, if $t \geq t^*$ where $\mathbf{bin}(t) = (t_1, \cdots, t_\tau)$, $\mathcal{C}$ invokes $\mathbf{T}_{\mathcal{T}_j} \leftarrow \mathsf{ExtBasis}(\mathbf{A}_{R,i,0} \mid \mathbf{A}_1^{(t_1)} \mid \cdots \mid \mathbf{A}_j^{(t_j)}, \mathbf{T}_{\mathbf{A}_j^{(t_j)}})$ to generate a basis $\mathbf{T}_{\mathcal{T}_j}$ for node $\mathcal{T}_j$. After that, $\mathcal{C}$ calculates $\mathbf{sk}_{R,i,t}$ which includes several bases in $\mathsf{Node}(\mathbf{bin}(t))$ as the secret key with time period t, and returns it to $\mathcal{A}$.
- $\mathcal{O}_{\mathrm{CT}}$: After obtaining a keyword $\mathbf{ck}$, the receivers' public keys in broadcast list $\{\mathbf{pk}_{R,i}\}_{i\in\mathcal{L}}$ and a time period t from $\mathcal{A}$, $\mathcal{C}$ calls **Encrypt**$(pp, \mathbf{ck}, \mathbf{pk}_S^*, \mathbf{sk}_S^*, \{\mathbf{pk}_{R,i}\}_{i\in\mathcal{L}}, t)$ algorithm to generate a ciphertext CT_t, and sends it to $\mathcal{A}$.
- $\mathcal{O}_{\mathrm{TD}}$: After obtaining a keyword $\mathbf{tk}$, the sender's public key $\mathbf{pk}_S$ and a time period t from $\mathcal{A}$, $\mathcal{C}$ selects a challenge receiver $\gamma \in \mathcal{L}^*$, and executes the $\mathcal{O}_{H_1}$ and $\mathcal{O}_{H_2}$ oracle to query the hash value of $\mathbf{tk}$. If $H_1(\mathbf{tk}) = \mathbf{A}_K$ or $H_2(\mathbf{tk}) = h^*$, this procedure is aborted by $\mathcal{C}$. Otherwise, $\mathcal{C}$ calculates $\mathbf{t}_S$, $\mathbf{t}_K$, $\mathbf{t}_U$ and $\mathbf{T}_{\mathbf{A}_{R,\gamma,\mathbf{bin}(t)}}$ as in **Trapdoor** algorithm, and invokes $\boldsymbol{\varepsilon}_T \leftarrow \mathsf{GenSamplePre}(\mathbf{A}_{R,1,0} \mid \cdots \mid \mathbf{A}_{R,l,0} \mid \mathbf{A}_1^{(t_1)} \mid \cdots \mid \mathbf{A}_\tau^{(t_\tau)} \mid H_1(\mathbf{tk}), \mathbf{T}_{H_1(\mathbf{tk})}, \{\gamma, l+1, \cdots, l+\tau+1\}, \mathbf{t}_U, \sigma)$ such that $(\mathbf{A}_{R,1,0} \mid \cdots \mid \mathbf{A}_{R,l,0} \mid \mathbf{A}_1^{(t_1)} \mid \cdots \mid \mathbf{A}_\tau^{(t_\tau)} \mid H_1(\mathbf{tk}))\boldsymbol{\varepsilon}_T = \mathbf{t}_U \bmod q$. Finally, $\mathcal{C}$ returns $\mathrm{TD}_t = (\mathbf{t}_S, \mathbf{t}_K, \boldsymbol{\varepsilon}_T)$ to $\mathcal{A}$.

Challenge: Base on *Theorem 1*, we add $\mathbf{c}_{T,j}^* = \mathbf{a}_{T,j}$ for $j \in [\tau]$.

Phase 2: This procedure is the same as *Theorem 1*.

Guess: This procedure is the same as *Theorem 1*.

Analysis: Due to *Theorem 1* and $\mathbf{c}_{T,j}^* = \mathbf{a}_{T,j} = \mathbf{A}_{T,j}^\top \mathbf{s} + \mathbf{e}_{T,j}$ for $j \in [\tau]$, $\mathrm{CT}_{t,\xi}^* = (\{\mathbf{c}_{R,i}^*\}_{i\in\mathcal{L}^*}, \{\mathbf{c}_{T,j}^*\}_{j\in[\tau]}, \mathbf{c}_K^*, \boldsymbol{\varepsilon}_C^*)$ is a valid ciphertext. Similar to *Theorem 1*, the advantage of $\mathcal{C}$ to solve the $\mathrm{LWE}_{n,m,q,\chi}$ hardness is $\frac{(q_{H_1}+q_{H_2}-1)\epsilon_3}{2q_{H_1}q_{H_2}}$, which is also non-negligible. □

Theorem 4: Assume that the $\mathrm{LWE}_{n,m,q,\chi}$ hardness holds, our proposed lattice-based FS-BroSearch scheme satisfies IND-IKGA 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_4, 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 We define an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ as in the *Theorem 2*.

LWE Instances: This procedure is the same as *Theorem 2*.

Init: This procedure is the same as *Theorem 2*.

Setup: This procedure is the same as *Theorem 2*.

TABLE II
THEORETICAL COMPUTATIONAL OVERHEAD COMPARISON

Schemes	Encrypt	Trapdoor	Search
FS-PEKS [28]	$(nm^2+nml_t+nl_t)T_M+T_I+T_H$	$T_{NBD}+T_{SP}+nm^2T_M+T_I+T_H$	ml_tT_M
PAEKS [6]	$l_sT_{SL}+(n^2m+n^2l_s+3m^2l_s+4nml_s)T_M+T_H$	$l_rT_{SL}+(n^2m+n^2l_r+3m^2l_r+4nml_r)T_M+T_H$	$8ml_sl_rT_M$
ABAEKS [19]	$T_{SL}+[(2\|att\|+3)nm+n^2+(\|att\|+1)m^2]T_M+T_H$	$T_{SL}+(n^2+m^2+2nm)T_M+T_{Epk}+T_H$	$5mT_M+T_{Ect}$
Re-PAEKS [20]	$T_{SL}+(n^2+3m^2+5nm)T_M+T_H$	$T_{SL}+(n^2+3m^2+5nm)T_M+T_H$	$8mT_M$
Our BroSearch	$T_{SL}+[n^2m+n^2+(l+1)nm]T_M+2T_H$	$T_{GSP}+(n^2m+n^2+m^2+2nm)T_M+2T_H$	$(l+3)mT_M$
Our FS-BroSearch	$T_{SL}+[n^2m+n^2+(l+\tau+1)nm]T_M+2T_H$	$T_{EB}+T_{GSP}+(n^2m+n^2+m^2+2nm)T_M+2T_H$	$(l+\tau+3)mT_M$

Note: l_t: The security-level of testing; l_s, l_r: The execution number of SampleLeft for sender and receiver defined in [6]; $|att|$: The length of attributes; l: The number of data receivers.

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

- $\mathcal{O}_{H_1}$: This procedure is the same as *Theorem 2*.
- $\mathcal{O}_{H_2}$: This procedure is the same as *Theorem 2*.
- $\mathcal{O}_{\mathrm{KU}}$: This procedure is the same as *Theorem 2*.
- $\mathcal{O}_{\mathrm{CT}}$: This procedure is the same as *Theorem 2*, except that a time period t is obtained and $\mathcal{C}$ calculates $\{\mathbf{c}_{R,i}\}_{i\in\mathcal{L}}$, $\{\mathbf{c}_{T,j}\}_{j\in[\tau]}$, $\mathbf{c}_K$ and $\mathbf{c}_U$ as in **Encrypt** algorithm.
- $\mathcal{O}_{\mathrm{TD}}$: This procedure is the same as *Theorem 2*, except that a time period t is obtained and $\mathcal{C}$ calls $\mathbf{Trapdoor}(pp, \mathbf{tk}, \{\mathbf{pk}^*_{R,i}\}_{i\in\mathcal{L}}, \mathbf{sk}^*_{R,\gamma}, \mathbf{pk}_S, t)$ algorithm to generate a trapdoor TD_t.

Challenge: This procedure is the same as *Theorem 2*, except that $\mathcal{C}$ samples $\boldsymbol{\varepsilon}^*_T \leftarrow \mathbb{Z}_q^{(l+\tau+1)m}$, and returns $\mathrm{TD}^*_{t,\xi}$ to $\mathcal{A}$.

Phase 2: This procedure is the same as *Theorem 2*.

Guess: This procedure is the same as *Theorem 2*.

Analysis: Similar to *Theorem 2*, $\mathrm{TD}^*_{t,\xi} = (\mathbf{t}^*_S, \mathbf{t}^*_K, \boldsymbol{\varepsilon}^*_T)$ is a valid ciphertext. The advantage of $\mathcal{C}$ to solve the $\mathrm{LWE}_{n,m,q,\chi}$ hardness is $\frac{(q_{H_1}+q_{H_2}-1)\epsilon_4}{2q_{H_1}q_{H_2}}$, which is also non-negligible. □

VII. PERFORMANCE EVALUATION AND COMPARISON

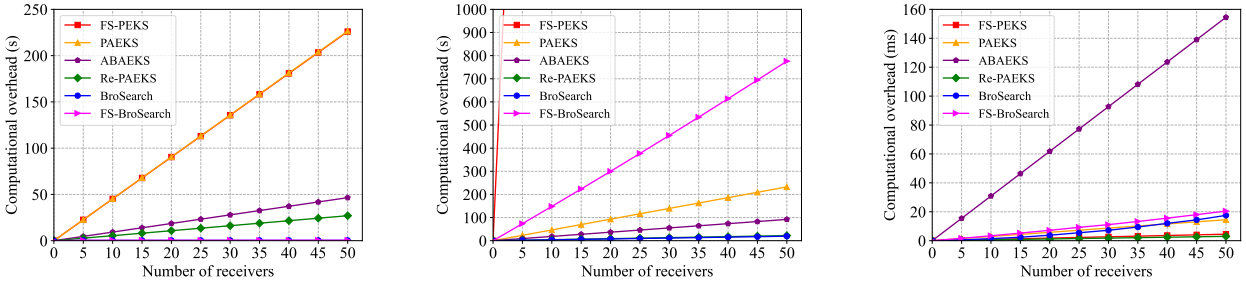
We evaluate the computational and communication overheads of our BroSearch and FS-BroSearch schemes. To ensure fairness, we set the parameters $q = 4097$ and $n = 64$ and compare with other lattice-based PEKS schemes [28], [6], [19], [20]. All experiments were implemented in Python using the NumPy library, which was conducted on a laptop with a 12th Gen Intel(R) Core(TM) i7-12800HX CPU, 16 GB of RAM, and running in Windows 10. Each experiment is conducted independently in each round.

A. Computational Overhead

To conduct a theoretical comparison of the computational overhead between our two designs and other state-of-the-art approaches [28], [6], [19], [20], we focus solely on the following operations: T_H denotes the time cost of a hash function; T_I represents the time cost of matrix inversion; T_M indicates the time cost of multiplication; T_{NBD}, T_{EB}, T_{SP}, T_{SL} and T_{GSP} correspond to the time costs of the NewBasisDel, ExtBasis, SamplePre, SampleLeft and GenSamplePre algorithms, respectively; T_{Epk} and T_{Ect} represent the time costs of the Eval_{pk} and Eval_{ct} algorithms, as defined in [19].

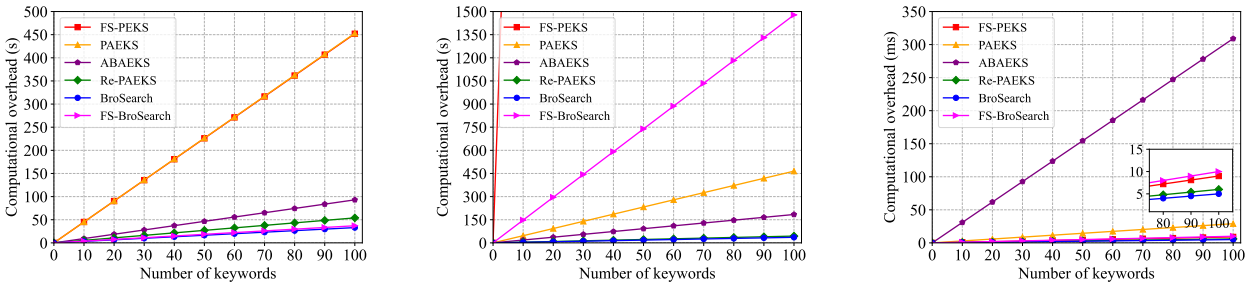
As shown in Table II, when encrypting a keyword, our design avoids the time-consuming matrix inversion operations and multiple invocations of the SampleLeft algorithm, realizing much more efficient than FS-PEKS and PAEKS, and comparable to ABAEKS and Re-PAEKS. Furthermore, in a multi-receiver scenario, our scheme enables direct broadcasting of the ciphertext to the receivers, eliminating the need for additional operations such as embedding receiver attribute [19] or re-encryption [20], which is well-suited for cloud storage applications. For the **Trapdoor** algorithm, similar to the encryption process, our BroSearch scheme demonstrates superior efficiency compared to FS-PEKS and PAEKS, while remaining comparable to ABAEKS and Re-PAEKS. Further, our FS-BroSearch scheme invokes the ExtBasis algorithm to achieve forward security, whereas FS-PEKS employs the NewBasisDel algorithm. As $T_{NBD} >> T_{EB}$, the efficiency of FS-BroSearch represents a significant advantage over FS-PEKS. While this additional operation introduces computational overhead compared to PAEKS, ABAEKS, and Re-PAEKS, our FS-BroSearch scheme offers resilience against key leakage issues. In the **Search** algorithm, as in FS-PEKS, PAEKS, and Re-PAEKS, the time cost of our schemes is primarily determined by a constant-level multiplication operation, which is more efficient than ABAEKS.

Fig. 4 demonstrates the computational overhead of our BroSearch and FS-BroSearch schemes compared to FS-PEKS [28], PAEKS [6], ABAEKS [19] and RE-PAEKS [20] with the number of receivers l. In Fig. 4(a), since our BroSearch and FS-BroSearch schemes allow a ciphertext to be broadcast to multiple receivers simultaneously, the time costs of **Encrypt** algorithm in BroSearch and FS-BroSearch schemes are more efficient than other four schemes. Fig. 4(b) illustrates the computational overhead of the trapdoor generation procedure. Due to the additional cost of invoking the ExtBasis algorithm, the time cost of FS-BroSearch is higher compared to BroSearch, PAEKS, ABAEKS, and Re-PAEKS, which is considered a trade-off between security and efficiency. Moreover, BroSearch demonstrates superior efficiency relative to PAEKS, ABAEKS, and Re-PAEKS. For example, when setting $l = 1$, PAEKS takes $4.65s$, ABAEKS takes $1.84s$, and Re-PAEKS takes $0.44s$, while our BroSearch only takes $0.36s$. In Fig. 4(c), the search cost of BroSearch and FS-BroSearch is much



(a) **Encrypt** algorithm (b) **Trapdoor** algorithm (c) **Search** algorithm

Fig. 4. Computational overhead comparison with the number of data receivers l.



(a) **Encrypt** algorithm (b) **Trapdoor** algorithm (c) **Search** algorithm

Fig. 5. Computational overhead comparison with the number of keywords k.

TABLE III
THE COMPUTATIONAL OVERHEAD COMPARISON WHEN $k = 1$

Schemes	Encrypt	Trapdoor	Search
FS-PEKS [28]	$4.52s$	$553.51s$	$0.09ms$
PAEKS [6]	$4.53s$	$4.65s$	$0.29ms$
ABAEKS [19]	$0.93s$	$1.84s$	$3.09ms$
Re-PAEKS [20]	$0.54s$	$0.44s$	$0.06ms$
BroSearch	$0.33s$	$0.36s$	$0.05ms$
FS-BroSearch	$0.37s$	$14.78s$	$0.10ms$

TABLE IV
THEORETICAL COMMUNICATION OVERHEAD COMPARISON

Schemes	Secret key	Ciphertext	Trapdoor
FS-PEKS [28]	$m^2\|\mathbb{Z}_q\|$	$l_t(m+1)\|\mathbb{Z}_q\|$	$m\|\mathbb{Z}_q\|$
PAEKS [6]	$m^2\|\mathbb{Z}_q\|$	$8ml_s\|\mathbb{Z}_q\|$	$8ml_r\|\mathbb{Z}_q\|$
ABAEKS [19]	$4m^2\|\mathbb{Z}_q\|$	$(\|att\|+4)m\|\mathbb{Z}_q\|$	$5m\|\mathbb{Z}_q\|$
Re-PAEKS [20]	$m^2\|\mathbb{Z}_q\|$	$8m\|\mathbb{Z}_q\|$	$8m\|\mathbb{Z}_q\|$
BroSearch	$m^2\|\mathbb{Z}_q\|$	$(l+3)m\|\mathbb{Z}_q\|$	$(l+3)m\|\mathbb{Z}_q\|$
FS-BroSearch	$m^2\|\mathbb{Z}_q\|$	$(l+\tau+3)m\|\mathbb{Z}_q\|$	$(l+\tau+3)m\|\mathbb{Z}_q\|$

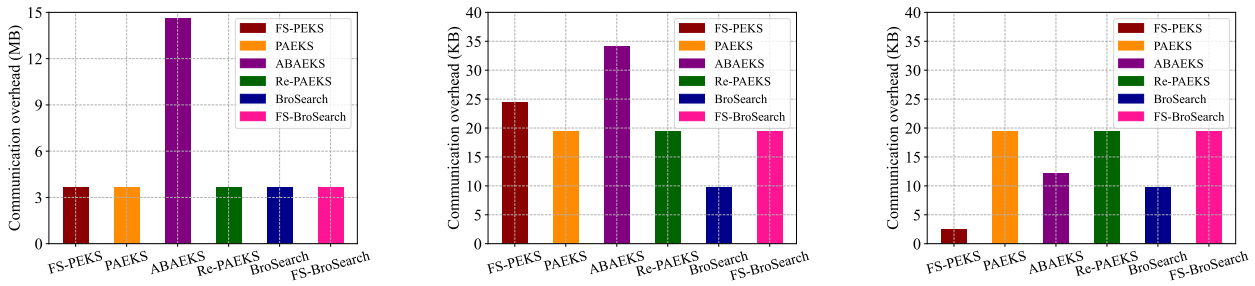
more efficient than that of ABAEKS and is largely consistent with FS-PEKS, PAEKS, and Re-PAEKS schemes, which is practical in cloud storage systems.

As depicted in Fig. 5, we illustrate the computational overhead with a different number of keywords k during ciphertext generation, trapdoor generation, and search procedures for these six schemes. In Fig. 5(a), the time cost of our BroSearch and FS-BroSearch schemes is significantly lower than that of FS-PEKS and PAEKS, which aligns with the theoretical values presented in Tab. II. Moreover, since a lower-dimensional matrix is used to invoke the SampleLeft algorithm, the actual encryption overhead in our schemes is more efficient than that of ABAEKS and Re-PAEKS, with this advantage increasing as k grows. Fig. 5(b) and Fig. 5(c) show the computational overhead of the **Trapdoor** and **Search** algorithms in these six schemes, respectively. We observe that the time costs of our schemes are proportional to the number of keywords k, and these results are consistent with those shown in Fig. 4(b) and Fig. 4(c). For example, when $k = 1$, the time costs of these six schemes are shown in Table III. Specifically, the time costs of the **Encrypt**, **Trapdoor**, and **Search** algorithms in our BroSearch scheme are $0.33s$, $0.36s$, and $0.05ms$, respectively, which are at least $0.61\times$, $0.82\times$, and $0.83\times$ compared to the others [28], [6], [19], [20].

Notably, since the **Setup**, **KeyGen**$_S$, and **KeyGen**$_R$ algorithms are executed less frequently than the **Encrypt**, **Trapdoor**, and **Search** algorithms in real-world applications, which have minimal impact on execution efficiency in cloud storage systems. Consequently, we focus only on the evaluation and comparison of ciphertext generation, trapdoor generation, and search procedures.

B. Communication Overhead

The transmission of the secret key, ciphertext, and trapdoor among participating entities contributes to the overall communication overhead. In this context, we provide a theoretical comparison of the communication overhead between our BroSearch and FS-BroSearch schemes and other state-of-the-art schemes [28], [6], [19], [20] in Table IV, where $|\mathbb{Z}_q|$ denotes the bit length of elements in $\mathbb{Z}_q$. For the data sender's secret key, the theoretical length across these six schemes



(a) Secret key size of receivers (b) Ciphertext size (c) Trapdoor size

Fig. 6. Communication overhead comparison.

is $m^2|\mathbb{Z}_q|$, and thus it will not be repeated. As indicated in Table IV, the secret key size for a data receiver in our schemes is equivalent to that in FS-PEKS, PAEKS, and Re-PAEKS, and outperforms ABAEKS due to the integration of the access policy. During ciphertext and trapdoor generation, the ciphertext size in our BroSearch (and FS-BroSearch) is identical to the trapdoor size, which is influenced by the number of receivers l (and for FS-BroSearch, also by the time period τ). While the communication overhead of our schemes does not present a substantial advantage, our approach offers significant benefits, including ciphertext broadcasting and forward security, distinguishing it from other schemes.

Subsequently, we set the parameters as follows: $q = 4097$, $|\mathbb{Z}_q| = 13$, $n = 64$, $m = \lceil 2n \log q \rceil = 1537$, $l_t = 10$, $l_s = l_r = 4$, $|att| = 10$, $l = 1$, and $\tau = 4$. We then evaluate the communication overhead of our BroSearch and FS-BroSearch schemes. For example, the ciphertext size of BroSearch and FS-BroSearch is given by $(l+3)m|\mathbb{Z}_q| = (1+3) \times 1537 \times 13 \approx 9.76$KB and $(l+\tau+3)m|\mathbb{Z}_q| = (1+4+3) \times 1537 \times 13 \approx 19.51$KB, respectively. In Fig. 6, we compare these evaluation results with those of prior works [28], [6], [19], [20]. As shown, our schemes demonstrate advantages over existing methods in terms of ciphertext size.

VIII. Conclusion

In this paper, we propose BroSearch, providing secure and efficient encrypted search with ciphertext broadcast for cloud storage systems. Furthermore, we propose a forward-secure version, called FS-BroSearch, which successfully mitigates secret key leakage problems. Rigorous security analysis demonstrates that both schemes achieve IND-CKA and IND-IKGA security in the ROM. Comprehensive experimental evaluations also indicate that our BroSearch offers significant advantages in the computational efficiency. We acknowledge that further work is required to enhance the security level from the ROM to the standard model.

References

[1] Y. Yang, Y. Chen, F. Chen, and J. Chen, "An efficient identity-based provable data possession protocol with compressed cloud storage," *IEEE TIFS*, vol. 17, pp. 1359–1371, 2022.

[2] K. Zhang, Z. Jiang, J. Ning, and X. Huang, "Subversion-resistant and consistent attribute-based keyword search for secure cloud storage," *IEEE TIFS*, vol. 17, pp. 1771–1784, 2022.

[3] 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.

[4] L. Cheng and F. Meng, "Server-aided public key authenticated searchable encryption with constant ciphertext and constant trapdoor," *IEEE TIFS*, vol. 19, pp. 1388–1400, 2023.

[5] 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.

[6] L. Cheng and F. Meng, "Public key authenticated encryption with keyword search from lwe," in *ESORICS*. Springer, 2022, pp. 303–324.

[7] S. Xu, Y. Cao, X. Chen, Y. Zhao, and S.-M. Yiu, "Post-quantum public-key authenticated searchable encryption with forward security: General construction, and applications," in *Inscrypt*. Springer, 2023, pp. 274–298.

[8] 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.

[9] A. Kiayias, O. Oksuz, A. Russell, Q. Tang, and B. Wang, "Efficient encrypted keyword search for multi-user data sharing," in *ESORICS*. Springer, 2016, pp. 173–195.

[10] X. Liu, K. He, G. Yang, W. Susilo, J. Tonien, and Q. Huang, "Broadcast authenticated encryption with keyword search," in *ACISP*. Springer, 2021, pp. 193–213.

[11] S. Mukherjee, "Statistically consistent broadcast authenticated encryption with keyword search: Adaptive security from standard assumptions," in *ACISP*. Springer, 2023, pp. 523–552.

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

[13] J. Jiang and D. Wang, "Qpase: Quantum-resistant password-authenticated searchable encryption for cloud storage," *IEEE TIFS*, 2024.

[14] S. Xu, Y. Cao, X. Chen, Y. Guo, Y. Yang, F. Guo, and S.-M. Yiu, "Post-quantum searchable encryption supporting user-authorization for outsourced data management," in *ACM CIKM*, 2024, pp. 2702–2711.

[15] Y. Cao, S. Xu, X. Chen, Y. He, and S. Jiang, "A forward-secure and efficient authentication protocol through lattice-based group signature in vanets scenarios," *Computer Networks*, vol. 214, p. 109149, 2022.

[16] X. Chen, S. Xu, Y. Cao, Y. He, and K. Xiao, "Aqrs: Anti-quantum ring signature scheme for secure epidemic control with blockchain," *Computer Networks*, vol. 224, p. 109595, 2023.

[17] X. Yu, L. Xu, X. Huang, and C. Xu, "An efficient lattice-based encrypted search scheme with forward security," in *NSS*. Springer, 2022, pp. 712–726.

[18] X. Chen, S. Xu, S. Gao, Y. Guo, S.-M. Yiu, and B. Xiao, "Fs-llrs: Lattice-based linkable ring signature with forward security for cloud-assisted electronic medical records," *IEEE TIFS*, 2024.

[19] F. Luo, H. Wang, C. Lin, and X. Yan, "Abaeks: Attribute-based authenticated encryption with keyword search over outsourced encrypted data," *IEEE TIFS*, 2023.

[20] F. Luo, H. Wang, and X. Yan, "Re-paeks: Public-key authenticated re-encryption with keyword search," *IEEE TMC*, 2024.

[21] L. Chen, J. Li, J. Li, and J. Weng, "Paess: Public-key authentication encryption with similar data search for pay-per-query," *IEEE TIFS*, 2024.

[22] Z.-Y. Liu, Y.-F. Tseng, R. Tso, M. Mambo, and Y.-C. Chen, "Public-key authenticated encryption with keyword search: Cryptanalysis, enhanced security, and quantum-resistant instantiation," in *ACM AsiaCCS*, 2022, pp. 423–436.

[23] A. Fiat and M. Naor, "Broadcast encryption," in *CRYPTO*. Springer, 1994, pp. 480–491.

[24] M. A. R. Baee, L. Simpson, X. Boyen, E. Foo, and J. Pieprzyk, "Ali: Anonymous lightweight inter-vehicle broadcast authentication with encryption," *IEEE TDSC*, vol. 20, no. 3, pp. 1799–1817, 2022.

[25] J. Zhang, S. Su, H. Zhong, J. Cui, and D. He, "Identity-based broadcast proxy re-encryption for flexible data sharing in vanets," *IEEE TIFS*, vol. 18, pp. 4830–4842, 2023.

[26] H. Yin, Y. Zhu, G. Guo, and W. C.-C. Chu, "Privacy-preserving smart contracts for confidential transactions using dual-mode broadcast encryption," *IEEE Transactions on Reliability*, vol. 73, no. 2, pp. 1090–1103, 2023.

[27] M. Ali, H. Ali, T. Zhong, F. Li, Z. Qin, and A. A. Abdelrahaman, "Broadcast searchable keyword encryption," in *IEEE CSE*. IEEE, 2014, pp. 1010–1016.

[28] 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 TDSC*, vol. 18, no. 3, pp. 1019–1032, 2019.

[29] 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.

[30] C. Gentry, C. Peikert, and V. Vaikuntanathan, "Trapdoors for hard lattices and new cryptographic constructions," in *ACM STOC*, 2008, pp. 197–206.

[31] S. Agrawal, D. Boneh, and X. Boyen, "Efficient lattice (h) ibe in the standard model," in *EUROCRYPT*. Springer, 2010, pp. 553–572.

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

[33] 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.

[34] D. Boneh, C. Gentry, S. Gorbunov, S. Halevi, V. Nikolaenko, G. Segev, V. Vaikuntanathan, and D. Vinayagamurthy, "Fully key-homomorphic encryption, arithmetic circuit abe and compact garbled circuits," in *EUROCRYPT*. Springer, 2014, pp. 533–556.