Quantum-resistant Public-key Authenticated Encryption with Keyword Search for Industrial Internet of Things

Zi-Yuan Liu[1], Yi-Fan Tseng[1], Raylin Tso[1], and Masahiro Mambo[2]

[1] Department of Computer Science, National Chengchi University, Taipei, Taiwan
{zyliu, yftseng, raylin}@cs.nccu.edu.tw
[2] Institute of Science and Engineering, Kanazawa University, Kakuma-machi, Kanazawa 920-1192, Japan
mambo@ec.t.kanazawa-u.ac.jp

Abstract. The industrial Internet of Things (IIoT) integrates sensors, instruments, equipment, and industrial applications, enabling traditional industries to automate and intelligently process data. To reduce the cost and demand of required service equipment, IIoT relies on cloud computing to further process and store data. However, the means for ensuring the privacy and confidentiality of the outsourced data and the maintenance of flexibility in the use of these data remain unclear. Public-key authenticated encryption with keyword search (PAEKS) is a variant of public-key encryption with keyword search that not only allows users to search encrypted data by specifying keywords but also prevents insider keyword guessing attacks (IKGAs). However, all current PAEKS schemes are based on the discrete logarithm assumption and are therefore vulnerable to quantum attacks. Additionally, the security of these schemes are only proven under random oracle and are considered insufficiently secure. In this study, we first introduce a generic PAEKS construction that enjoys the security under IKGAs in the standard model. Based on the framework, we propose a novel instantiation of quantum-resistant PAEKS that is based on ring learning with errors assumption. Compared with its state-of-the-art counterparts, our instantiation is more efficient and secure.

Keywords: Public-key authenticated encryption with keyword search · Insider keyword guessing attacks · Industrial IoT · Quantum-resistant

1 Introduction

The Internet of Things (IoT) is a system that connects a large set of devices to a network, where these devices can communicate with each other over the network. Industrial IoT (IIoT) is a particular type of IoT that fully utilizes the advantages of IoT for remote detection, monitoring, and management in industry. Because the volume of data and computation in industry is very large, and long-term storage is required, IIoT is highly reliant on cloud computing technology to reduce the cost of storage and computing environments (Fig 1). Despite the numerous benefits of processing IIoT data through cloud computing, industrial data typically have commercial value and thus necessitate privacy protection when such sensitive data are offloaded to the cloud. Therefore, to ensure data confidentiality, sensitive data should be encrypted before being uploaded to the cloud.

In addition to data confidentiality, data sharing is indispensable in IIoT. For instance, in an industrial organization, the administrator in the information department (*i.e.,* the data sender) must share the data collected from IoT devices with an administrator from another department (*i.e.,* the data receiver). To ensure data confidentiality, the data sender encrypts the data by using the public key of the data receivers. However, in such a method, if the data receiver wants to retrieve the data from the ciphertext stored in the cloud, the data receiver must download all the ciphertext and further decrypt it, which consumes considerable time and resources.

Public-key encryption with keyword search (PEKS), first introduced by Boneh [6], is highly suited to the aforementioned application environment because PEKS makes the ciphertext searchable. Furthermore, in PEKS, a data sender not only uploads encrypted data but also and uploads the encrypted keywords related to the data using the data receiver's public key. To download the data related to a specified keyword, the data receiver can use their private key to generate a corresponding trapdoor and submit the trapdoor to the cloud server. The cloud server can then identify encrypted keywords corresponding to the trapdoor and then returns the corresponding encrypted data to the data receiver. A secure PEKS scheme is required to ensure that the ciphertext and trapdoor leak no keyword information to the malicious outsiders. However, Byun [7] noted that having only the two aforementioned security requirements is insufficient because the cloud server may be malicious, where the malicious cloud server guesses the keyword hiding in the trapdoor—a type of attack called insider keyword guessing attacks (IKGAs). In particular, because the cloud server can adaptively generate a ciphertext for any keyword by using the data receiver's public key, through trial and error, test for that self-made ciphertext that is matched with the trapdoor received from the data receiver. As mentioned in [7], because the keyword space is not large enough, there is a high probability that keyword-related information searched for by the data receiver is leaked to the malicious cloud server.

To prevent IKGA, some early PEKS schemes have used additional servers to perform tests, in place of the original server. This method is called designated-tester PEKS [37] or dual-server PEKS [12,13,11,32,10]. When servers do not collude, IKGAs do not occur. However, using additional servers can significantly increase the cost of communication. Furthermore, the means for ensuring that servers do not collude remain unclear. Recently, Huang and Li [20] introduced a new cryptography primitive called public-key authenticated encryption with keyword search (PAEKS). In this primitive, the data sender not only generates but also authenticates ciphertext, whereas a trapdoor generated from the data receiver is only valid to the ciphertext authenticated by the specific data sender. Therefore, the cloud server cannot perform IKGAs. Because of the higher efficiency and greater convenience compared with designated-tester PEKS schemes, many PAEKS schemes [18,25,34,35,36,24,40] have been formulated for further application in IoT and IIoT as well as in cloud computing environments.

Unfortunately, these PAEKS schemes are only proven under random oracle model (ROM). As described in [23,4,8], ROM can be said to be unnatural and markedly different from the construction of the real world; thus, there is both a theoretical drawback and also a practical concern of the constructions proven under ROM. How to obtain a secure PAEKS scheme avoiding such heuristics is still an important question.

Shor [39,38] reported on quantum algorithms that can violate the traditional number-theoretic assumptions, such as the integer factoring assumption and discrete logarithm assumption. In particular, the advent of the 53-qubit quantum computer, proposed by Arute *et al.* [3], may improve quantum computing technology and affect the existing cryptographic systems. Because the security of existing PAEKS schemes is based on the discrete logarithm assumption, quantum computers can come to pose a potential threat to existing schemes. Hence, the means of constructing a quantum-resistant PEAKS scheme is an emerging issue among scholars and practitioners.

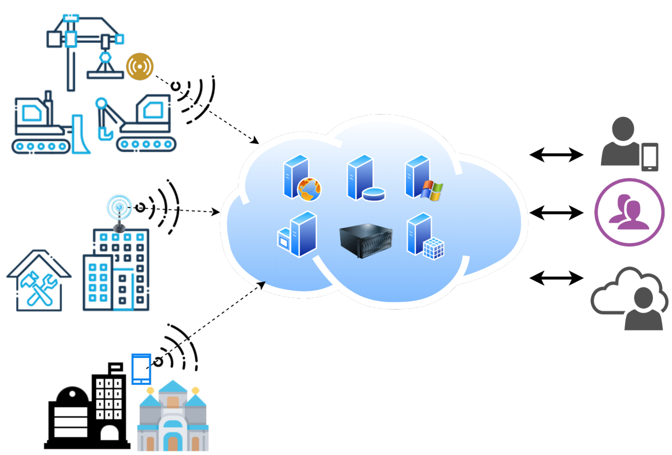

Fig. 1. Typical network architecture for IIoT.

1.1 Our Contribution

In this paper, we introduce a novel solution for constructing a quantum-resistant PAEKS scheme for use in IIoT. At a high level, the original keyword space is commonly found and easy to test. Our strategy is to allow a data sender and data receiver to generate an "extended keyword" from an original keyword without interacting with each other. In this method, the ciphertext and trapdoor are generated using the extended keyword instead of the original keyword. Because the keyword space increases after the keyword is extended, the malicious cloud server cannot generate a valid ciphertext to perform IKGAs.

Accordingly, we provide a generic PAEKS construction by leveraging a two-tier identity-based key encapsulation mechanism (IBKEM), a pseudorandom generator (PRG), and anonymous identity-based encryption (IBE). We also present two rigorous proofs to show that our construction satisfies the security requirements of PAEKS. These requirements are indistinguishability against chosen keyword attacks (IND-CKA) and indistinguishability against IKGAs (IND-IKGA) under a multi-user setting in a standard model, without ROM. Furthermore, we first employ

Katsumata and Yamada's adaptively anonymous IBE [21] to obtain a two-tier IBKEM under the ring learning with errors (RLWE) assumption. We then combine the scheme in [21] with the two-tier IBKEM scheme to obtain an instantiation of PAKES. Because the security of [21] is inherited, we obtain the first quantum-resistant instantiation of PAEKS. The comparison results of our scheme with other state-of-the-art PAKES schemes are presented in Table 2 and Figure 3; our instantiation was demonstrated to be not only more secure but also more efficient with respect to ciphertext generation, trapdoor generation, and testing.

1.2 Related Work

The PEKS schemes against IKGAs can be separated into three categories: designated-tester (or called dual-server) PEKS, PAEKS, and witness-based searchable encryption.

The concept of designated-tester PEKS was first introduced by Rhee *et al.* [37], who proposed a PEKS scheme that supports trapdoor indistinguishability. Chen *et al.* [12,13,11] followed this concept and proposed a variant scheme, called dual-server PEKS, which can be used against IKGAs if the servers do not collude with each other. However, Huang [19] indicates that [12,13,11] are susceptible to IKGAs. Recently, Chen *et al.* [10] introduced an efficient dual-server scheme that is resistant to IKGAs without needing any pairing computations. In addition, Mao *et al.* [32] suggested a quantum-resistant designated-tester PEKS scheme, which is also the first lattice-based PEKS that is protected from IKGAs. However, the above schemes requires that servers do not collude with each other, which is difficult to guarantee in many scenarios. Moreover, construction costs and communications costs are increased in this method.

Considering these limitations, scholars thus began to study methods for constructing trapdoors that are only valid for certain ciphertexts. Fang *et al.* [15,16] first considered using a one-time signature to authenticate the ciphertext, while having the trapdoor be valid only for the authenticated ciphertext, a method that improved resistance to IKGA. Huang and Li [20] formally defined the system model and security model for PAEKS. Noroozi and Eslami [34] first considered Huang and Li's scheme [20] is not secure against IKGAs and further improved [20] without incurring additional cost complexity. To resist quantum attacks, Zhang *et al.* [41] proposed a lattice-based PAEKS scheme; however, Liu *et al.* [26] recently demonstrated that the security model of that work is flawed and therefore cannot withstand IKGAs. Pakniat *et al.* [35] introduced the first certificateless PAEKS scheme for an IoT environment. Moreover, Li *et al.* [24] and Qin *et al.* [36] further prevented malicious adversary eavesdrops on the transmission channel of ciphertext and trapdoor, and executes the test algorithm to determine whether the two ciphertexts shared the same keyword. Although the aforementioned PAEKS schemes resist IKGAs, these schemes are based on the discrete logarithm assumption, which make them vulnerable to attacks from quantum computers.

Ma *et al.* [31] introduced a cryptographic primitive called "witness-based searchable encryption," in which the trapdoor is valid only when the ciphertext has a witness relation to the trapdoor. Chen *et al.* [14] formulated an improvement to reduce the complexity of the trapdoor size. Inspired by [31], Liu *et al.* [27] introduced a new concept called "designated-ciphertext searchable encryption," where the trapdoor is designated to a ciphertext; this concept affords users with a quantum-resistant instantiation. Despite their advantages, however, these schemes require the data sender to interact with the data receiver; moreover, they incur additional communication costs and are inapplicable to many scenarios.

1.3 Organization of the Paper

The rest of the paper is organized as follows. Section 2 introduces the preliminaries, and Section 3 recalls the definition of the building blocks used in our generic construction. Moreover, Section 4 provides the definition and security requirement of the PAEKS. Next, Sections 5 and 6 introduce our generic constriction before providing the security proofs. Section 7 elaborates on the first quantum-resistant PAEKS instantiation, and Section 8 details the analysis of the communication cost and computation cost incurred in the related PAEKS schemes. Finally, Section 9 concludes this study.

2 Preliminary

For simplicity and readability, we use the notations in Table 1 throughout the manuscript.

2.1 Lattices

We now introduce the basic concepts underlying lattices that are used in our instantiation. An m-dimension lattice Λ is an additive discrete subgroup of $\mathbb{R}^m$, which can be defined as follows.

Table 1. Notations

Notation	Description
1^λ	Security parameter
$\mathcal{A}$	Adversary
$\mathcal{B}$	Challenger
$\mathcal{O}$	Oracle
$\mathit{\Pi}$	PAEKS
$\mathit{\Psi}$	IBE
$\mathit{\Omega}$	2-tier IBKEM
F	Pseudorandom generator
IDS	Identity space
CS	Ciphertext space
KS	Shared key space
PS	Plaintext space
W	Keyword space
$\mathbb{N}, \mathbb{Z}, \mathbb{R}$	Natural number, integer number, real number
$\mathbb{G}_1, \mathbb{G}_T$	Cyclic group
$\mathbf{v}, \mathbf{V}$	Vector, matrix
$a\|b$	Concatenation of element a and b
$s \xleftarrow{\$} S$	Sampling an element s from S uniformly at random
$\widetilde{\mathbf{T}}$	Gram-Schmidt orthogonalization of $\mathbf{T}$
$\lvert v\rvert$	The bit length of element v
$\lVert\mathbf{v}\rVert, \lVert\mathbf{V}\rVert$	The Euclidean norm of $\mathbf{v}$ and $\mathbf{V}$
$\mathsf{negl}(\cdot), \mathsf{poly}(\cdot)$	Negligible function, polynomial function
PPT	Probabilistic polynomial-time
R	Ring $R = \mathbb{Z}[X]/X^n + 1$
$\phi(\cdot)$	Embedding function: $\phi : R \to \mathbb{Z}^n$
rot	Ring homomorphism $\mathsf{rot} : R \to \mathbb{Z}^{n\times n}$

Definition 1 (Lattice). *We say that a m-dimension lattice $\mathit{\Lambda}$ generated by a basis $\mathbf{B} = [\mathbf{b}_1|\cdots|\mathbf{b}_n] \in \mathbb{R}^{m\times n}$ is defined by*

$$\mathit{\Lambda}(\mathbf{B}) = \mathit{\Lambda}(\mathbf{b}_1, \cdots, \mathbf{b}_n) = \left\{\sum_{i=1}^{n} \mathbf{b}_i a_i | a_i \in \mathbb{Z}\right\},$$

where $\mathbf{b}_1, \cdots, \mathbf{b}_n \in \mathbb{R}^m$ are n linear independent vectors.

In addition, for a prime q, a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, and a vector $\mathbf{u} \in \mathbb{Z}_q^n$, we can define the following three sets [17,1]:

- $\mathit{\Lambda}_q := \{\mathbf{e} \in \mathbb{Z}^m \mid \exists \mathbf{s} \in \mathbb{Z}^n \text{ where } \mathbf{A}\mathbf{s} = \mathbf{e} \bmod q\}$.
- $\mathit{\Lambda}_q^\perp := \{\mathbf{e} \in \mathbb{Z}^m \mid \mathbf{A}\mathbf{e} = 0 \bmod q\}$.
- $\mathit{\Lambda}_q^{\mathbf{u}} := \{\mathbf{e} \in \mathbb{Z}^m \mid \mathbf{A}\mathbf{e} = \mathbf{u} \bmod q\}$.

2.2 Discrete Gaussian Distributions

For any vector $\mathbf{c} \in \mathbb{R}^n$ and any positive real number s, we define the following two notations:

- $\rho_{s,\mathbf{c}}(\mathbf{x}) = \exp\left(-\pi\frac{\|\mathbf{x}-\mathbf{c}\|^2}{s^2}\right)$.
- $\rho_{s,\mathbf{c}}(\mathit{\Lambda}) = \sum_{\mathbf{x}\in\mathit{\Lambda}} \rho_{s,\mathbf{c}}(\mathbf{x})$.

The discrete Gaussian distribution over the lattice $\mathit{\Lambda}$ with center $\mathbf{c}$ and parameter s can then be defined as $D_{\mathit{\Lambda},s,\mathbf{c}}(\mathbf{x}) = \rho_{s,\mathbf{c}}(\mathbf{x})/\rho_{s,\mathbf{c}}(\mathit{\Lambda})$ for any $\mathbf{x} \in \mathit{\Lambda}$. Note that we usually omit $\mathbf{c}$ if $\mathbf{c}$ is 0.

2.3 Rings and Ideal Lattices

Here, we briefly introduce rings and ideal lattices, as formulated in previous studies [29,30]. Let n be a power of 2. The ring can then be defined as $R = \mathbb{Z}[X]/\Phi_m(X)$, where $\Phi_m(X) = X^n + 1$ is the mth cyclotomic polynomial and $m = 2n$. Furthermore, for some integer q, we use R_q to denote $R/qR = \mathbb{Z}[X]/(q, \Phi_m(X))$. Because we can consider the coefficients in R as the elements on $\mathbb{Z}^n$, we define a embedding function $\phi : \mathbb{R} \to \mathbb{Z}^n$, which maps $b = \sum_{i=0}^{n-1} \beta_i X^i \in R$ to $[\beta_0, \beta_1, \cdots, \beta_{n-1}] \in \mathbb{Z}^n$. Furthermore, we can expend the embedding function and define the ring homomorphism $\mathsf{rot} : R \to \mathbb{Z}^{n\times n}$, where the ith row in $\mathbb{Z}^{n\times n}$ is $\phi(b \cdot X^{i-1} \bmod \Phi_m(X)) \in \mathbb{Z}^n$. We also use $s_1(\mathbf{R}) := \max_{\|\mathbf{z}\|_2} \|\mathbf{z} \cdot \mathsf{rot}(\mathbf{R})\|_2$ to define the largest singular value of $\mathbf{R} \in R^{s\times t}$.

Ring Learning with Errors Assumption The security of our instantiation is based on the famous lattice hard assumption, which is the ring learning with errors (RLWE) assumption, first introduced by Lybashevsky *et al.* [29,30].

Definition 2 (Ring Learning with Errors Assumption [21]). *Let λ be a security parameter. Given $n = n(\lambda), k = k(n)$, a prime integer $q = q(n) > 2$, an error distribution $\chi = \chi(n)$ over R_q, an advantage for the RLWE problem of $\mathcal{A}$ is defined as follows.*

$$\mathbf{Adv}_{\mathcal{A}}^{\mathsf{RLWE}_{n,k,q,\chi}} = \left|\mathbf{Pr}[\mathcal{A}(\{a_i, v_i\}_{i=1}^k) \to 1] - \mathbf{Pr}[\mathcal{A}(\{a_i, a_i s + e_i\}_{i=1}^k) \to 1]\right|,$$

where $a_1, \cdots, a_k, v_1, \cdots, v_k, s \xleftarrow{\$} R_q$ and $e_1, \cdots, e_k \xleftarrow{\chi}$. We say that $\mathsf{RLWE}_{n,k,q,\chi}$ assumption holds if for all PPT $\mathcal{A}$, $\mathbf{Adv}_{\mathcal{A}}^{\mathsf{RLWE}_{n,k,q,\chi}}$ is negligible.

Trapdoor Functions over Rings The following recalls two important trapdoor functions in the "ring setting" [21] that are used in our instantiation.

Lemma 1 ($\mathsf{TrapGen}$) [33]). *Let n be a power of 2, q be a prime larger than $4n$ such that $q \equiv 3 \bmod 8$, and $b, \rho \in \mathbb{Z}^+$ satisfying $\rho < \frac{1}{2}\sqrt{q/n}$. There is a randomized polynomial time algorithm $\mathsf{TrapGen}(1^n, 1^k, q, \rho)$ that outputs a vector $\mathbf{a} \in R_q^k$ and a matrix $\mathbf{T_a} \in R^{k\times k}$ when $k \geq 2\log_\rho q$. Here, $\mathsf{rot}(\mathbf{a}^T)^T \in \mathbb{Z}_q^{n\times nk}$ is a full-rank matrix and $\mathsf{rot}(\mathbf{T_a}) \in \mathbb{Z}_q^{nk\times nk}$ is a basis for $\Lambda^\perp(\mathsf{rot}(\mathbf{a}^T)^T)$. Besides, $\mathbf{a}$ is close to uniform and $\|\mathsf{rot}(\mathbf{T_a}\|_{\mathsf{GS}}) = O(b\rho \cdot \sqrt{n \log_\rho q})$.*

Lemma 2 ($\mathsf{SampleLeft}$ [9]). *Let n be a power of 2, q be a prime larger than $4n$ such that $q \equiv 3 \bmod 8$, and $b, \rho \in \mathbb{Z}^+$ satisfying $\rho < \frac{1}{2}\sqrt{q/n}$. Given $\mathbf{a}, \mathbf{b} \in R_q^k$ where $\mathsf{rot}(\mathbf{a}^T)^T, \mathsf{rot}(\mathbf{b}^T)^T$ are full-rank, an element $u \in R_q$, a matrix $\mathbf{T_a} \in R^{k\times k}$ such that $\mathsf{rot}(\mathbf{T_a}) \in \mathbb{Z}^{nk\times nk}$ is a basis for $\Lambda^\perp(\mathsf{rot}(\mathbf{a}^T)^T)$, and a Gaussian parameter $\sigma > \|\mathsf{rot}(\mathbf{T_a})\|_{\mathsf{GS}} \cdot \omega(\sqrt{\log nk})$, there is a randomized polynomial time algorithm $\mathsf{SampleLeft}(\mathbf{a}, \mathbf{b}, u, \mathbf{T_a}, \sigma)$ that outputs a vector $\mathbf{e} \in R^{2k}$ sampled from the distribution that close to $D^{\mathsf{coeff}}_{\Lambda^\perp_{\Phi(u)}([\mathsf{rot}(\mathbf{a}^T)^T|\mathsf{rot}(\mathbf{b}^T)^T]),\sigma}$.*

Homomorphic Computation We recall the $\mathsf{PubEval}_d : (R_q^k)^d \to R_q^k$ function used in our instantiation to hash identities to R_q^k. This function can be defined as

$$\mathsf{PubEval}_d(\mathbf{b}_1, \cdots, \mathbf{b}_d) = \begin{cases} \mathbf{b}_1 & \text{if } d = 1; \\ \mathbf{b}_1 \cdot \mathbf{g}_b^{-1}\left(\mathsf{PubEval}_{d-1}(\mathbf{b}_2, \cdots, \mathbf{b}_d)\right) & \text{if } d \geq 2. \end{cases}$$

In this definition, $d \in \mathbb{N}, \mathbf{b}_1, \cdots, \mathbf{b}_d \in R_q^k$. Moreover, $\mathbf{g}_b^{-1}(\cdot)$ is a deterministic polynomial time algorithm [33] that takes the input $\mathbf{u} \in R_q^k$ to output $\mathbf{R} \in [-b, b]_R^{k\times k}$ such that $\mathbf{g}_b\mathbf{R} = \mathbf{u}$, where $\mathbf{g}_b = [1|b|\cdots|b^{k'-1}|\mathbf{0}] \in R_q^k$ is a gadget matrix for $b \in \mathbb{Z}^+$ and $k \geq k' = \lfloor \log_b q \rfloor$.

3 Building Blocks

In this section, we recall three crucial cryptographic primitives, namely two-tier IBKEM, IBE, and PRG, which are used as the building blocks in our generic construction.

3.1 Two-tier IBKEM

A two-tier IBKEM Ω comprises the five algorithms: $(\mathsf{Setup}, \mathsf{Extract}, \mathsf{Enc}_1, \mathsf{Enc}_2, \mathsf{Dec})$ along with an identity space IDS, ciphertext space CS, and symmetric key space KS. These algorithms are described as follows.

- $\mathsf{Setup}(1^\lambda) \to (\mathsf{msk}, \mathsf{mpk})$: This is the *setup* algorithm that takes the security parameter 1^λ as its input and outputs a master private key msk and a master public key mpk.
- $\mathsf{Extract}(\mathsf{msk}, \mathsf{id} \in IDS) \to \mathsf{sk}_{\mathsf{id}}$: This is the *extraction* algorithm that takes the two inputs of a master private key msk and identity $\mathsf{id} \in IDS$ and outputs a private key $\mathsf{sk}_{\mathsf{id}}$ for the identity.
- $\mathsf{Enc}_1(\mathsf{mpk}) \to (\mathsf{ct}, \mathsf{r})$: This is the *first encapsulation* algorithm that takes the input of a master public key mpk and outputs a ciphertext $\mathsf{ct} \in CS$ and a randomness r.
- $\mathsf{Enc}_2(\mathsf{mpk}, \mathsf{id}, \mathsf{r}) \to \mathsf{k}/\perp$: This is the *second encapsulation* algorithm that takes the three inputs of a master public key mpk, identity id, and randomness r and outputs either a symmetric key $\mathsf{k} \in KS$ or the reject symbol $\perp$.
- $\mathsf{Dec}(\mathsf{sk}_{\mathsf{id}}, \mathsf{id}, \mathsf{ct}) \to \mathsf{k}/\perp$: This is the *decryption* algorithm that takes the three inputs of a private key $\mathsf{sk}_{\mathsf{id}}$, identity id, and ciphertext ct and outputs either symmetric key $\mathsf{k} \in KS$ or a reject symbol $\perp$.

Definition 3 (Correctness of 2-tier IBKEM). *A two-tier IBKEM Ω is correct if for all security parameters 1^λ, all master key pairs $(\mathsf{msk}, \mathsf{mpk})$ output by $\mathsf{Setup}(1^\lambda)$, all private keys $\mathsf{sk}_{\mathsf{id}}$ for identity id output by $\mathsf{Extract}(\mathsf{msk}, \mathsf{id})$, all $(\mathsf{ct}, \mathsf{r})$ pairs output by $\mathsf{Enc}_1(\mathsf{mpk})$, and all k values output by $\mathsf{Enc}_2(\mathsf{mpk}, \mathsf{id}, \mathsf{r})$, the following equation holds:*

$$\mathbf{Pr}[\mathsf{Dec}(\mathsf{sk}_{\mathsf{id}}, \mathsf{id}, \mathsf{ct}) = \mathsf{k}] = 1 - \mathsf{negl}(\lambda).$$

The basis security requirement of two-tier IBKEM is IND-CPA, which ensures that no PPT adversary can distinguish whether the challenge ciphertext is generated from the Enc_1 and Enc_2 algorithm or is randomly chosen from the ciphertext space CS. This security requirement can be modeled by the following security game played between an adversary $\mathcal{A}$ and a challenger $\mathcal{B}$.

Game - IND-CPA:

- **Initialization.** The challenger $\mathcal{B}$ first runs $(\mathsf{msk}, \mathsf{mpk}) \leftarrow \mathsf{Setup}(1^\lambda)$. $\mathcal{B}$ then sends the master public key mpk to $\mathcal{A}$ and keeps the master private key msk secret.
- **Phase 1.** The adversary $\mathcal{A}$ is given access to query the extract oracle with any identity id, and $\mathcal{B}$ returns a valid private key $\mathsf{sk}_{\mathsf{id}}$ for identity id by using $\mathsf{Extract}$ algorithm.
- **Challenge.** $\mathcal{A}$ submits $\mathcal{B}$ an identity id^* that has not been queried to extract oracle in **Phase 1**. $\mathcal{B}$ randomly selects a bit $b \in \{0, 1\}$. If $b = 0$, $\mathcal{B}$ generate a true ciphertext by using Enc_1 and Enc_2. Otherwise, $\mathcal{B}$ randomly selects a ciphertext from the ciphertext space. $\mathcal{B}$ then returns the ciphertext as a challenge to $\mathcal{A}$.
- **Phase 2.** $\mathcal{A}$ can continue querying the extract oracle as **Phase 1**. The only restriction is that $\mathcal{A}$ cannot query the extract oracle with the identity id^*.
- **Guess.** $\mathcal{A}$ outputs a bit $b' \in \{0, 1\}$.

The advantage of $\mathcal{A}$ is defined as

$$\mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{\Omega,\mathcal{A}}(\lambda) = |\mathbf{Pr}[b = b'] - \tfrac{1}{2}|.$$

Definition 4 (IND-CPA Security of two-tier IBKEM). *A two-tier IBKEM scheme Ω is IND-CPA secure if for all PPT adversaries $\mathcal{A}$, $\mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{\Omega,\mathcal{A}}(\lambda)$ is negligible.*

3.2 IBE

An IBE scheme Ψ comprises four algorithms $(\mathsf{Setup}, \mathsf{Extract}, \mathsf{Enc}, \mathsf{Dec})$ along with an identity space IDS, ciphertext space CS, and plaintext space PS, described as follows.

- $\mathsf{Setup}(1^\lambda) \to (\mathsf{msk}, \mathsf{mpk})$: This is the *setup* algorithm that takes the security parameter 1^λ as its input and outputs a master private key msk and master public key mpk.
- $\mathsf{Extract}(\mathsf{msk}, \mathsf{id}) \to \mathsf{sk}_{\mathsf{id}}$: This is the *extraction* algorithm that takes the two inputs of a master private key msk and identity $\mathsf{id} \in IDS$ and outputs a private key $\mathsf{sk}_{\mathsf{id}}$ for the identity.

– $\mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, \mathsf{m}) \to \mathsf{ct}_{\mathsf{id}}$: This is the *encryption* algorithm that takes the three inputs of a master public key mpk, identity id, and plaintext $\mathsf{m} \in PS$ and outputs a ciphertext $\mathsf{ct}_{\mathsf{id}} \in CS$.

– $\mathsf{Dec}(\mathsf{sk}_{\mathsf{id}}, \mathsf{ct}_{\mathsf{id}}) \to \mathsf{m}$: This is the *decryption* algorithm that takes the two inputs of a private key $\mathsf{sk}_{\mathsf{id}}$ (for identity id) and ciphertext $\mathsf{ct}_{\mathsf{id}}$ and outputs a plaintext $\mathsf{m} \in PS$.

Definition 5 (Correctness of IBE). *An IBE Ψ is correct if, for all security parameters 1^λ, all master key pairs $(\mathsf{msk}, \mathsf{mpk})$ output by $\mathsf{Setup}(1^\lambda)$, all private keys $\mathsf{sk}_{\mathsf{id}}$ for identity id output by $\mathsf{Extract}(\mathsf{msk}, \mathsf{id})$, and all ciphertexts $(\mathsf{ct}_{\mathsf{id}})$ output by $\mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, \mathsf{m})$, the following equation holds:*

$$\mathbf{Pr}[\mathsf{Dec}(\mathsf{sk}_{\mathsf{id}}, \mathsf{ct}_{\mathsf{id}}) = \mathsf{m}] = 1 - \mathsf{negl}(\lambda).$$

The basis requirement of IBE is indistinguishability against chosen plaintext attacks. However, our instantiation requires a stronger security requirement called indistinguishability and anonymity against chosen plaintext and chosen identity attacks (IND-ANON-ID-CPA). IND-ANON-ID-CPA security ensures that no PPT adversary can retrieve any information pertaining to the identity and the message from a challenge ciphertext, as modelled by the following game.

Game - IND-ANON-ID-CPA:

– **Initialization.** The challenger $\mathcal{B}$ first runs $(\mathsf{msk}, \mathsf{mpk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and then sends the master public key mpk to $\mathcal{A}$ and keeps master private key msk secret.

– **Phase 1.** The adversary $\mathcal{A}$ is given access to query the extract oracle with any identity id, and $\mathcal{B}$ returns a valid private key $\mathsf{sk}_{\mathsf{id}}$ for identity id by using the $\mathsf{Extract}$ algorithm.

– **Challenge.** $\mathcal{A}$ submits $\mathcal{B}$ two messages $\mathsf{m}_0^*, \mathsf{m}_1^*$ and two identities $\mathsf{id}_0^*, \mathsf{id}_1^*$ that have not been queried to extract the oracle. $\mathcal{B}$ randomly chooses a bit $b \in \{0, 1\}$ and then computes $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{id}_b^*, \mathsf{m}_b^*)$. Finally, $\mathcal{B}$ returns the challenge ciphertext ct^* to $\mathcal{A}$.

– **Phase 2.** $\mathcal{A}$ can continue querying the oracle per **Phase 1**. The only restriction is that $\mathcal{A}$ cannot query the extract oracle with id_0^* and id_1^*.

– **Guess.** $\mathcal{A}$ outputs a bit $b' \in \{0, 1\}$.

The advantage of $\mathcal{A}$ is defined as

$$\mathbf{Adv}_{\Psi,\mathcal{A}}^{\mathsf{IND\text{-}ANON\text{-}ID\text{-}CPA}}(\lambda) = |\mathbf{Pr}[b = b'] - \tfrac{1}{2}|.$$

Definition 6 (IND-ANON-ID-CPA Security of IBE). *An IBE scheme Ψ is IND-ANON-ID-CPA secure if $\mathbf{Adv}_{\Psi,\mathcal{A}}^{\mathsf{IND\text{-}ANON\text{-}ID\text{-}CPA}}(\lambda)$ is negligible for all PPT adversaries $\mathcal{A}$.*

For analytical convenience, in this work, we consider an IBE to be anonymous if the IBE is IND-ANON-ID-CPA secure.

3.3 Pseudorandom Generator (PRG)

Informally, suppose that a distribution $\mathcal{D}$ is pseudorandom if no PPT distinguisher that can distinguish a string s is either selected from the distribution $\mathcal{D}$ or randomly selected from a uniform distribution. We provide the following definition of the pseudorandom generator in [22].

Definition 7 (Pseudorandom Generator). *Let $F : \{0, 1\}^n \to \{0, 1\}^m$ be a deterministic PPT algorithm, where $n' = \mathsf{poly}(n)$ and $m > n$. We say that F is a pseudorandom generator the following two conditions are satisfied:*

– *Expansion: For every n, it holds that $m > n$.*

– *Pseudorandomness: For all PPT distinguishers $\mathcal{D}$,*

$$|\mathbf{Pr}[\mathcal{D}(r) = 1] - \mathbf{Pr}[\mathcal{D}(F(s)) = 1]| \leq \mathsf{negl}(n),$$

where $r \xleftarrow{\$} \{0, 1\}^m$ and seed $s \xleftarrow{\$} \{0, 1\}^n$.

4 PAEKS

In this section we introduce the system model and the security requirements of PAEKS.

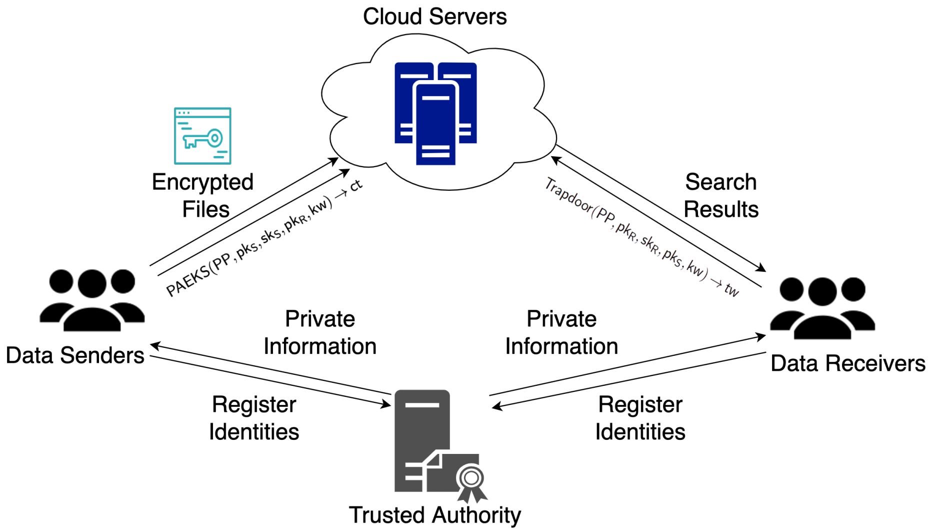

Fig. 2. System model for the proposed PAEKS scheme.

4.1 System Model

A PAEKS has four entities: a trusted authority, data sender, data receiver, and cloud server (Fig 2). In practice, the data sender and data receiver register their identity with the trusted authority and obtain their public/private key pairs. A PAEKS scheme Π comprises six algorithms: $(\mathsf{Setup}, \mathsf{KeyGen_S}, \mathsf{KeyGen_S}, \mathsf{PAEKS}, \mathsf{Trapdoor}, \mathsf{Test})$ together with a keyword space W, which are detailed as follows.

- $\mathsf{Setup}(1^\lambda) \to (\mathsf{PP}, \mathsf{msk})$: This is the *setup* algorithm that takes the security parameter 1^λ as input, and outputs a system parameter PP and a master private key msk. Note that the master private key is hold by trusted authority.
- $\mathsf{KeyGen_S}(\mathsf{PP}, \mathsf{msk}, \mathsf{id_S}) \to (\mathsf{pk_S}, \mathsf{sk_S})$: This is the *data sender key generation* algorithm that interacts between data sender and trusted authority. It takes a system parameter PP, master private key msk, and an identity $\mathsf{id_S}$ as input, and outputs data sender's public key $\mathsf{pk_S}$ and private key $\mathsf{sk_S}$.
- $\mathsf{KeyGen_R}(\mathsf{PP}, \mathsf{msk}, \mathsf{id_R}) \to (\mathsf{pk_R}, \mathsf{sk_R})$: This is the *data receiver key generation* algorithm that interacts between data receiver and trusted authority. It takes a system parameter PP, master private key msk, and an identity $\mathsf{id_R}$ as input, and outputs data receiver's public key $\mathsf{pk_R}$ and private key $\mathsf{sk_R}$.
- $\mathsf{PAEKS}(\mathsf{PP}, \mathsf{pk_S}, \mathsf{sk_S}, \mathsf{pk_R}, \mathsf{kw}) \to \mathsf{ct}$: This is the *authenticated encryption* algorithm that takes a system parameter PP, data sender's public key $\mathsf{pk_S}$ and private key $\mathsf{sk_S}$, data receiver's public key $\mathsf{pk_R}$, and a keyword $\mathsf{kw} \in W$, and outputs a searchable ciphertext ct.
- $\mathsf{Trapdoor}(\mathsf{PP}, \mathsf{pk_R}, \mathsf{sk_R}, \mathsf{pk_S}, \mathsf{kw}) \to \mathsf{tw}$: This is the *trapdoor* algorithm that takes a system parameter PP, data receiver's public key $\mathsf{pk_R}$ and private key $\mathsf{sk_R}$, data sender's public key $\mathsf{pk_S}$, and a keyword $\mathsf{kw} \in \mathsf{W}$, and outputs a trapdoor tw.
- $\mathsf{Test}(\mathsf{PP}, \mathsf{ct}, \mathsf{tw}) \to 1/0$: This is the *test* algorithm that takes a system parameter PP, searchable ciphertext ct, and a trapdoor tw, and outputs 1 if ct and tw correspond the same keyword; outputs 0, otherwise.

Definition 8 (Correctness of PAEKS). *A PAEKS scheme Π is correct if, for all security parameters 1^λ, all system parameter/master private key pairs $(\mathsf{PP}, \mathsf{msk})$ output by $\mathsf{Setup}(1^\lambda)$, all data sender $\mathsf{id_S}$'s key pairs $(\mathsf{pk_S}, \mathsf{sk_S})$ output by $\mathsf{KeyGen_S}(\mathsf{PP}, \mathsf{msk}, \mathsf{id_S})$, all data receiver $\mathsf{id_R}$'s key pairs $(\mathsf{pk_R}, \mathsf{sk_R})$ output by $\mathsf{KeyGen_R}(\mathsf{PP}, \mathsf{msk}, \mathsf{id_R})$, all searchable ciphertexts ct output by $\mathsf{PAEKS}(\mathsf{PP}, \mathsf{pk_S}, \mathsf{sk_S}, \mathsf{pk_R}, \mathsf{kw})$, and all trapdoors tw output by $\mathsf{Trapdoor}(\mathsf{PP}, \mathsf{pk_R}, \mathsf{sk_R}, \mathsf{pk_S}, \mathsf{kw})$, the following equation holds:*

$$\mathsf{Test}(\mathsf{PP}, \mathsf{ct}, \mathsf{tw}) = \begin{cases} 1, & \textit{if } \mathsf{ct}, \mathsf{tw} \textit{ contains the same } \mathsf{kw}; \\ 0, & \textit{otherwise.} \end{cases}$$

4.2 Security Requirements

The basic secure requirement of the PAEKS scheme is IND-CKA and IND-IKGA. Specifically, IND-CKA and IND-IKGA security ensures that no PPT adversary can obtain any information regarding the keyword from the searchable ciphertext and keyword, respectively. We follow the method of [34] to model the aforementioned two security requirements in the multi-user context by using two security games featuring interaction between the adversary $\mathcal{A}$ and challenger $\mathcal{B}$. Because the malicious insider has more power than the malicious outsider has, we only consider the IND-IKGA in this work. Note that we use $\mathsf{id_U}$, $\mathsf{pk_U}$, and $\mathsf{sk_U}$ to denote some user U's identity, public key, and private key, respectively.

Game - IND-CKA:

– **Initialization.** The challenger $\mathcal{B}$ first runs $(\mathsf{PP}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$. The algorithm then chooses two identities $\mathsf{id_S}, \mathsf{id_R}$ and runs $(\mathsf{pk_S}, \mathsf{sk_S}) \leftarrow \mathsf{KeyGen_S}(\mathsf{PP}, \mathsf{msk}, \mathsf{id_S})$ and $(\mathsf{pk_R}, \mathsf{sk_R}) \leftarrow \mathsf{KeyGen_R}(\mathsf{PP}, \mathsf{msk}, \mathsf{id_R})$. Finally, $\mathcal{B}$ sends the system parameter PP, data sender's public key $\mathsf{pk_S}$, and data receiver's public key $\mathsf{pk_R}$ to $\mathcal{A}$ while keeping secret the master private key msk, data sender's private key $\mathsf{sk_S}$, and data sender's private key $\mathsf{sk_R}$.

– **Phase 1.** $\mathcal{A}$ can make polynomially many queries to oracles $\mathcal{O}_{\mathsf{PKGen_S}}$, $\mathcal{O}_{\mathsf{PKGen_R}}$, $\mathcal{O}_{\mathsf{Trapdoor}}$, and $\mathcal{O}_{\mathsf{PAEKS}}$, $\mathcal{B}$ then responds as follows.
  - $\mathcal{O}_{\mathsf{PKGen_S}}(\mathsf{id_U})$: $\mathcal{B}$ runs $(\mathsf{pk_U}, \mathsf{sk_U}) \leftarrow \mathsf{KeyGen_S}(\mathsf{PP}, \mathsf{msk}, \mathsf{id_U})$. Then, $\mathcal{B}$ returns $\mathsf{pk_U}$ to $\mathcal{A}$, and keeps $\mathsf{sk_U}$ secret.
  - $\mathcal{O}_{\mathsf{PKGen_R}}(\mathsf{id_U})$: $\mathcal{B}$ runs $(\mathsf{pk_U}, \mathsf{sk_U}) \leftarrow \mathsf{KeyGen_R}(\mathsf{PP}, \mathsf{msk}, \mathsf{id_U})$. Then, $\mathcal{B}$ returns $\mathsf{pk_U}$ to $\mathcal{A}$, and keeps $\mathsf{sk_U}$ secret.
  - $\mathcal{O}_{\mathsf{PAEKS}}(\mathsf{kw}, \mathsf{pk_U})$: $\mathcal{B}$ computes $\mathsf{ct} \leftarrow \mathsf{PAEKS}(\mathsf{PP}, \mathsf{pk_U}, \mathsf{sk_U}, \mathsf{pk_R}, \mathsf{kw})$ and returns ct to $\mathcal{A}$.
  - $\mathcal{O}_{\mathsf{Trapdoor}}(\mathsf{kw}, \mathsf{pk_U})$: $\mathcal{B}$ computes $\mathsf{tw} \leftarrow \mathsf{Trapdoor}(\mathsf{PP}, \mathsf{pk_U}, \mathsf{sk_U}, \mathsf{pk_S}, \mathsf{kw})$ and returns tw to $\mathcal{A}$.

– **Challenge.** After the end of **Phase 1**, $\mathcal{A}$ outputs two keywords $\mathsf{kw}_0^*, \mathsf{kw}_1^* \in W$ with the restriction that $(\mathsf{kw}_0^*, \mathsf{pk_R})$ and $(\mathsf{kw}_1^*, \mathsf{pk_R})$ have not been queried to oracles $\mathcal{O}_{\mathsf{PAEKS}}$ and $\mathcal{O}_{\mathsf{Trapdoor}}$ in **Phase 1**. $\mathcal{B}$ first chooses a random bit $b \in \{0, 1\}$ and then returns $\mathsf{ct}^* = (\Psi.\mathsf{ct}^*, \mathsf{h}) \leftarrow \mathsf{PAEKS}(\mathsf{PP}, \mathsf{pk_S}, \mathsf{sk_S}, \mathsf{pk_R}, \mathsf{kw}_b^*)$ to $\mathcal{A}$.

– **Phase 2.** $\mathcal{A}$ can continue to make queries, as was the case in **Phase 1**. The only restriction is that $\mathcal{A}$ cannot make any query to $\mathcal{O}_{\mathsf{PAEKS}}$ on $(\mathsf{kw}_i^*, \mathsf{pk_S})$ and to $\mathcal{O}_{\mathsf{Trapdoor}}$ on $(\mathsf{kw}_i^*, \mathsf{pk_R})$ for $i = 0, 1$.

– **Guess.** $\mathcal{A}$ outputs its guess $b' \in \{0, 1\}$.
  The advantage of $\mathcal{A}$ is defined as

$$\mathbf{Adv}_{\Pi,\mathcal{A}}^{\mathsf{IND\text{-}CKA}}(\lambda) = |\mathbf{Pr}[b = b'] - \tfrac{1}{2}|.$$

Definition 9 (IND-CKA security of PAEKS). *A PAEKS scheme Ω is IND-CKA secure if for all PPT adversaries $\mathcal{A}$, $\mathbf{Adv}_{\Pi,\mathcal{A}}^{\mathsf{IND\text{-}CKA}}(\lambda)$ is negligible.*

Game - IND-IKGA:

– **Initialization.** The challenger $\mathcal{B}$ first runs $(\mathsf{PP}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and then runs $(\mathsf{pk_S}, \mathsf{sk_S}) \leftarrow \mathsf{KeyGen_S}(\mathsf{PP}, \mathsf{msk})$ and then runs $(\mathsf{pk_R}, \mathsf{sk_R}) \leftarrow \mathsf{KeyGen_R}(\mathsf{PP}, \mathsf{msk})$. Finally, $\mathcal{B}$ sends the system parameter PP, data sender's public key $\mathsf{pk_S}$, and data receiver's public key $\mathsf{pk_R}$ to $\mathcal{A}$ while keeping secret the master private key msk, data sender's private key $\mathsf{sk_S}$, and data sender's private key $\mathsf{sk_R}$.

– **Phase 1.** $\mathcal{A}$ can make polynomially many queries to oracles $\mathcal{O}_{\mathsf{PKGen_S}}$, $\mathcal{O}_{\mathsf{PKGen_R}}$, $\mathcal{O}_{\mathsf{Trapdoor}}$, and $\mathcal{O}_{\mathsf{PAEKS}}$ with any keyword kw and any user's public key $\mathsf{pk_U}$, $\mathcal{B}$ then responds as follows.
  - $\mathcal{O}_{\mathsf{PKGen_S}}(\mathsf{id_U})$: $\mathcal{B}$ runs $(\mathsf{pk_U}, \mathsf{sk_U}) \leftarrow \mathsf{KeyGen_S}(\mathsf{PP}, \mathsf{msk}, \mathsf{id_U})$. Then, $\mathcal{B}$ returns $\mathsf{pk_U}$ to $\mathcal{A}$, and keeps $\mathsf{sk_U}$ secret.
  - $\mathcal{O}_{\mathsf{PKGen_R}}(\mathsf{id_U})$: $\mathcal{B}$ runs $(\mathsf{pk_U}, \mathsf{sk_U}) \leftarrow \mathsf{KeyGen_R}(\mathsf{PP}, \mathsf{msk}, \mathsf{id_U})$. Then, $\mathcal{B}$ returns $\mathsf{pk_U}$ to $\mathcal{A}$, and keeps $\mathsf{sk_U}$ secret.
  - $\mathcal{O}_{\mathsf{PAEKS}}(\mathsf{kw}, \mathsf{pk_U})$: $\mathcal{B}$ computes $\mathsf{ct} \leftarrow \mathsf{PAEKS}(\mathsf{PP}, \mathsf{pk_U}, \mathsf{sk_U}, \mathsf{pk_R}, \mathsf{kw})$ and returns ct to $\mathcal{A}$.
  - $\mathcal{O}_{\mathsf{Trapdoor}}(\mathsf{kw}, \mathsf{pk_U})$: $\mathcal{B}$ computes $\mathsf{tw} \leftarrow \mathsf{Trapdoor}(\mathsf{PP}, \mathsf{pk_U}, \mathsf{sk_U}, \mathsf{pk_S}, \mathsf{kw})$ and returns tw to $\mathcal{A}$.

– **Challenge.** After the end of **Phase 1**, $\mathcal{A}$ outputs two keywords $\mathsf{kw}_0^*, \mathsf{kw}_1^* \in W$ with the restriction that $(\mathsf{kw}_0^*, \mathsf{pk_R})$ and $(\mathsf{kw}_1^*, \mathsf{pk_R})$ have not been queried to oracles $\mathcal{O}_{\mathsf{PAEKS}}$ and $\mathcal{O}_{\mathsf{Trapdoor}}$ in **Phase 1**. $\mathcal{B}$ first selects a random bit $b \in \{0, 1\}$ and then returns $\mathsf{tw}^* \leftarrow \mathsf{Trapdoor}(\mathsf{PP}, \mathsf{pk_R}, \mathsf{sk_R}, \mathsf{pk_S}, \mathsf{kw}_b^*)$ to $\mathcal{A}$.

- **Phase 2.** $\mathcal{A}$ can continue to make queries, as was the case in **Phase 1**. The only restriction is that $\mathcal{A}$ cannot make any query to $\mathcal{O}_{\mathsf{PAEKS}}$ on $(\mathsf{kw}_i^*, \mathsf{pk_S})$ and to $\mathcal{O}_{\mathsf{Trapdoor}}$ on $(\mathsf{kw}_i^*, \mathsf{pk_R})$ for $i = 0, 1$.
- **Guess.** $\mathcal{A}$ outputs its guess $b' \in \{0, 1\}$.
  The advantage of $\mathcal{A}$ is defined as

$$\mathbf{Adv}_{\Pi,\mathcal{A}}^{\mathsf{IND\text{-}IKGA}}(\lambda) = |\mathbf{Pr}[b = b'] - \tfrac{1}{2}|.$$

Definition 10 (IND-IKGA security of PAEKS). *A PAEKS scheme Ω is IND-IKGA secure if for all PPT adversaries $\mathcal{A}$, $\mathbf{Adv}_{\Pi,\mathcal{A}}^{\mathsf{IND\text{-}IKGA}}(\lambda)$ is negligible.*

5 Generic PAEKS Construction

We now construct our generic PAEKS. Specifically, we demonstrate how a PAEKS scheme can be constructed by combing an anonymous IBE, PRG, and two-tier IBKEM.

The high-level conception of our construction is that through two-tier IBKEM, the data sender and data receiver can obtain the shared key shk without interaction. The data sender and data receiver each use this shared key to extend the keyword by computing $\mathsf{f} \leftarrow \mathsf{F}(\mathsf{kw}\|\mathsf{shk})$, where F is PRG. Rather than using the original keyword kw, the data sender and data receiver use the extended keyword f to generate a ciphertext and trapdoor, respectively. The data sender takes f as an "identity" to generate a ciphertext for the data receiver by using an anonymous IBE. The data receiver can extract a private key for identity f and take this private key as the corresponding trapdoor. By using this trapdoor, the cloud server can search for the ciphertext containing the keyword kw. In addition, because the ciphertext and trapdoor are using the output of PRG as the identity and because the IBE is anonymous, PPT adversaries cannot obtain any information regarding the keyword from the ciphertext and trapdoor.

To construct a PAEKS scheme Π = (Setup, $\mathsf{KeyGen_S}$, $\mathsf{KeyGen_R}$, PAEKS, $\mathsf{Trapdoor}$, Test) with the keyword space W, we use the following cryptosystems as the building block. Let Ψ = (Setup, $\mathsf{Extract}$, Enc, Dec) be an anonymous IBE scheme with the identity space $\Psi.IDS$, ciphertext space $\Psi.CS$, and plaintext space $\Psi.PS$. Let $\Omega = (\mathsf{Setup}, \mathsf{Extract}, \mathsf{Enc}_1, \mathsf{Enc}_2, \mathsf{Dec})$ be a two-tier IBKEM scheme with the identity space $\Omega.IDS$, ciphertext space $\Omega.CS$, and symmetric key space $\Omega.KS$. In addition, let $\mathsf{F} : \mathcal{X} \rightarrow \mathcal{Y}$ be a PRG that maps $\mathcal{X}$ to $\mathcal{Y}$, where $\mathcal{X} = \{\mathsf{kw}\|\mathsf{shk} \mid \mathsf{kw} \in W \wedge \mathsf{shk} \in \Omega.KS\}$ and $\mathcal{Y} = \Psi.IDS$. The generic construction is detailed in the subsequent section. Note that although our construction is based on identity-based cryptosystems, the entire construction remains in the public key setting.

- $\mathsf{Setup}(1^\lambda) \rightarrow (\mathsf{PP}, \mathsf{msk})$: Given a security parameter 1^λ, this algorithm runs as follows.
  1. Choose a proper PRG $\mathsf{F} : \mathcal{X} \rightarrow \mathcal{Y}$.
  2. Choose a secure hash function $\mathsf{H} : \{0,1\}^\alpha \rightarrow \{0,1\}^\beta$, where $\alpha, \beta \in \mathbb{Z}^+$.
  3. Generate $(\Omega.\mathsf{msk}, \Omega.\mathsf{mpk}) \leftarrow \Omega.\mathsf{Setup}(1^\lambda)$.
  4. Output system parameter $\mathsf{PP} := (\lambda, \Omega.\mathsf{mpk}, \mathsf{H}, \mathsf{F})$ and master private key $\mathsf{msk} := \Omega.\mathsf{msk}$. Note that msk is kept secret by the trusted authority.
- $\mathsf{KeyGen_S}(\mathsf{PP}, \mathsf{msk}, \mathsf{id_S}) \rightarrow (\mathsf{pk_S}, \mathsf{sk_S})$: Given a system parameter $\mathsf{PP} = (\lambda, \Omega.\mathsf{mpk}, \mathsf{H}, \mathsf{F})$, a master private key $\mathsf{msk} = \Omega.\mathsf{msk}$, and a data sender's identity $\mathsf{id_S} \in \Omega.IDS$, data sender and trusted authority interact as follows.
  1. The data sender registers the identity $\mathsf{id_S}$ to trusted authority which return $\Omega.\mathsf{sk}_{\mathsf{id_S}} \leftarrow \Omega.\mathsf{Extract}(\Omega.\mathsf{msk}, \mathsf{id_S})$ and $(\Omega.\mathsf{ct_S}, \Omega.\mathsf{r_S}) \leftarrow \Omega.\mathsf{Enc}_1(\mathsf{mpk})$.
  2. Output data sender's public key $\mathsf{pk_S} := (\mathsf{id_S}, \Omega.\mathsf{ct_S})$ and private key $\mathsf{sk_S} := (\Omega.\mathsf{sk}_{\mathsf{id_S}}, \Omega.\mathsf{r_S})$.
- $\mathsf{KeyGen_R}(\mathsf{PP}, \mathsf{msk}, \mathsf{id_R}) \rightarrow (\mathsf{pk_R}, \mathsf{sk_R})$: Given a system parameter $\mathsf{PP} = (\lambda, \Omega.\mathsf{mpk}, \mathsf{H}, \mathsf{F})$, a master private key $\mathsf{msk} = \Omega.\mathsf{msk}$, and a data receiver's identity $\mathsf{id_R} \in \Omega.IDS$, data receiver and trusted authority interact as follows.
  1. The data receiver registers this identity $\mathsf{id_R}$ to trusted authority which return $\Omega.\mathsf{sk}_{\mathsf{id_R}} \leftarrow \Omega.\mathsf{Extract}(\Omega.\mathsf{msk}, \mathsf{id_R})$ and $(\Omega.\mathsf{ct_R}, \Omega.\mathsf{r_R}) \leftarrow \Omega.\mathsf{Enc}_1(\mathsf{mpk})$.
  2. Compute $(\Psi.\mathsf{mpk}, \Psi.\mathsf{msk}) \leftarrow \Psi.\mathsf{Setup}(1^\lambda)$.
  3. Output data receiver's public key $\mathsf{pk_R} := (\mathsf{id_R}, \Omega.\mathsf{ct_R}, \Psi.\mathsf{mpk})$ and private key $\mathsf{sk_R} := (\Omega.\mathsf{sk}_{\mathsf{id_R}}, \Omega.\mathsf{r_R}, \Psi.\mathsf{msk})$.

- $\mathsf{PAEKS}(\mathsf{PP}, \mathsf{pk_S}, \mathsf{sk_S}, \mathsf{pk_R}, \mathsf{kw}) \rightarrow \mathsf{ct}$: Given a system parameter $\mathsf{PP} = (\lambda, \Omega.\mathsf{mpk}, \mathsf{H}, \mathsf{F})$, a data sender's public key $\mathsf{pk_S} = (\mathsf{id_S}, \Omega.\mathsf{ct_S})$ and private key $\mathsf{sk_S} = (\Omega.\mathsf{sk}_{\mathsf{id_S}}, \Omega.\mathsf{r_S})$, a data receiver's public key $\mathsf{pk_R} = (\Omega.\mathsf{id_R}, \Omega.\mathsf{ct_R}, \Psi.\mathsf{mpk})$, and a keyword $\mathsf{kw} \in W$, data sender works as follows.
  1. Compute $\mathsf{k}_{\mathsf{id_S},\mathsf{id_R}} \leftarrow \Omega.\mathsf{Dec}(\Omega.\mathsf{sk}_{\mathsf{id_S}}, \mathsf{id_S}, \Omega.\mathsf{ct_R})$.
  2. Compute $\mathsf{k}_{\mathsf{id_R},\mathsf{id_S}} \leftarrow \Omega.\mathsf{Enc}_2(\Omega.\mathsf{mpk}, \mathsf{id_R}, \Omega.\mathsf{r_S})$.
  3. Compute $\mathsf{shk} \leftarrow \mathsf{k}_{\mathsf{id_S},\mathsf{id_R}} \oplus \mathsf{k}_{\mathsf{id_R},\mathsf{id_S}}$, where $\oplus$ is an operation compatible with the key space.
  4. Compute $\mathsf{f} \leftarrow \mathsf{F}(\mathsf{kw}\|\mathsf{shk})$.
  5. Choose a random $\mathsf{r} \xleftarrow{\$} \Psi.PS$ and compute $\Psi.\mathsf{ct_{kw}} \leftarrow \Psi.\mathsf{Enc}(\Psi.\mathsf{mpk}, \mathsf{f}, \mathsf{r})$.
  6. Compute $\mathsf{h} = \mathsf{H}(\Psi.\mathsf{ct_{kw}}, \mathsf{r})$.
  7. Output a searchable ciphertext $\mathsf{ct} := (\Psi.\mathsf{ct_{kw}}, \mathsf{h})$.
- $\mathsf{Trapdoor}(\mathsf{PP}, \mathsf{pk_R}, \mathsf{sk_R}, \mathsf{pk_S}, \mathsf{kw}) \rightarrow \mathsf{tw}$: Given a system parameter $\mathsf{PP} = (\lambda, \Omega.\mathsf{mpk}, \mathsf{H}, \mathsf{F})$, a data receiver's public key $\mathsf{pk_R} = (\mathsf{id_R}, \Omega.\mathsf{ct_R}, \Psi.\mathsf{mpk})$ and private key $\mathsf{sk_R} = (\Omega.\mathsf{sk}_{\mathsf{id_R}}, \Omega.\mathsf{r_R}, \Psi.\mathsf{msk})$, a data sender's public key $\mathsf{pk_S} = (\mathsf{id_S}, \Omega.\mathsf{ct_S})$, and a keyword $\mathsf{kw} \in W$, data receiver works as follows.
  1. Compute $\mathsf{k}_{\mathsf{id_R},\mathsf{id_S}} \leftarrow \Omega.\mathsf{Dec}(\Omega.\mathsf{sk}_{\mathsf{id_R}}, \mathsf{id_R}, \Omega.\mathsf{ct_S})$.
  2. Compute $\mathsf{k}_{\mathsf{id_S},\mathsf{id_R}} \leftarrow \Omega.\mathsf{Enc}_2(\Omega.\mathsf{mpk}, \mathsf{id_S}, \Omega.\mathsf{r_R})$.
  3. Compute $\mathsf{shk} \leftarrow \mathsf{k}_{\mathsf{id_R},\mathsf{id_S}} \oplus \mathsf{k}_{\mathsf{id_S},\mathsf{id_R}}$, where $\oplus$ is an operation compatible with the key space.
  4. Compute $\mathsf{f} \leftarrow \mathsf{F}(\mathsf{kw}\|\mathsf{shk})$.
  5. Compute $\Psi.\mathsf{sk_{kw}} \leftarrow \Psi.\mathsf{Extract}(\Psi.\mathsf{msk}, \mathsf{f})$.
  6. Output a trapdoor $\mathsf{tw} := \Psi.\mathsf{sk_{kw}}$ for keyword kw.
- $\mathsf{Test}(\mathsf{PP}, \mathsf{ct}, \mathsf{tw})$: Given a system parameter $\mathsf{PP} = (\lambda, \Omega.\mathsf{mpk}, \mathsf{H}, \mathsf{F})$, a searchable ciphertext $\mathsf{ct} = (\Psi.\mathsf{ct_{kw}}, \mathsf{h})$, and a trapdoor $\mathsf{tw} = \Psi.\mathsf{sk_{kw}}$ for keyword kw, cloud server works as follows.
  1. Compute $\mathsf{r} \leftarrow \Psi.\mathsf{Dec}(\Psi.\mathsf{sk_{kw}}, \Psi.\mathsf{ct_{kw}})$.
  2. Output 1 if $\mathsf{H}(\Psi.\mathsf{ct}, \mathsf{r}) = \mathsf{h}$; outputs 0, otherwise.

Correctness. Notably, the data sender and data receiver rely on the underlying two-tier IBKEM to exchange an extended keyword and the extended keyword acts as an identity in the underlying IBE scheme. Therefore, the proposed construction is correct if and only if the underlying anonymous IBE and two-tier IBKEM are correct.

6 Security Proofs

The following provides two security proofs to show that our generic construction is IND-KGA secure and IND-IKGA secure under standard model.

Theorem 1. *The proposed PAEKS scheme Π is IND-CKA secure if the underlying IBE scheme Ψ is IND-ANON-ID-CPA secure.*

Proof (Proof of Theorem 1). If adversary $\mathcal{A}$ can win the IND-CKA game with a non-negligible advantage, then challenger $\mathcal{B}$ can win the IND-ANON-ID-CPA game of the underlying IBE scheme Ψ with a non-negligible advantage. Their interaction is as follows.

- **Initialization.** Given the security parameter 1^λ, $\mathcal{B}$ first chooses the proper secure hash function H and pseudorandom generator F and invokes the IND-ANON-ID-CPA game of Ψ to obtain $\Psi.\mathsf{mpk}$. Next, $\mathcal{B}$ executes the following steps.
  - Compute $(\Omega.\mathsf{msk}, \Omega.\mathsf{mpk}) \leftarrow \Omega.\mathsf{Setup}(1^\lambda)$.

- Choose $\mathsf{id_S}$ and $\mathsf{id_R}$ from $\Omega.IDS$.
- Compute $\Omega.\mathsf{sk}_{\mathsf{id_S}} \leftarrow \Omega.\mathsf{Extract}(\Omega.\mathsf{msk}, \mathsf{id_S})$ and $\Omega.\mathsf{sk}_{\mathsf{id_R}} \leftarrow \Omega.\mathsf{Extract}(\Omega.\mathsf{msk}, \mathsf{id_R})$.
- Compute $(\Omega.\mathsf{ct_S}, \Omega.\mathsf{r_S}) \leftarrow \Omega.\mathsf{Enc}_1(\mathsf{mpk})$ and $(\Omega.\mathsf{ct_R}, \Omega.\mathsf{r_R}) \leftarrow \Omega.\mathsf{Enc}_1(\mathsf{mpk})$.

Finally, $\mathcal{B}$ sends the data sender's public key $\mathsf{pk_S} = (\mathsf{id_S}, \Omega.\mathsf{ct_S})$, data receiver's public key $\mathsf{pk_R} = (\mathsf{id_R}, \Omega.\mathsf{ct_R}, \Psi.\mathsf{mpk})$, and system parameter $\mathsf{PP} = (\lambda, \Omega.\mathsf{mpk}, \mathsf{H}, \mathsf{F})$ to $\mathcal{A}$, and keeps master private key $\mathsf{msk} = (\Omega.\mathsf{msk}, \Omega.\mathsf{sk}_{\mathsf{id_S}}, \Omega.\mathsf{sk}_{\mathsf{id_R}})$ secret.

– **Phase 1.** $\mathcal{A}$ can make polynomially many queries to oracles $\mathcal{O}_{\mathsf{PKGen_S}}(\mathsf{id_U})$, $\mathcal{O}_{\mathsf{PKGen_R}}(\mathsf{id_U})$, $\mathcal{O}_{\mathsf{Trapdoor}}(\mathsf{kw}, \mathsf{pk_U})$, and $\mathcal{O}_{\mathsf{PAEKS}}(\mathsf{kw}, \mathsf{pk_U})$, $\mathcal{B}$ then responds as follows.
  - $\mathcal{O}_{\mathsf{PKGen_S}}(\mathsf{id_U})$: $\mathcal{B}$ first computes $\Omega.\mathsf{sk}_{\mathsf{id_U}} \leftarrow \Omega.\mathsf{Extract}(\Omega.\mathsf{msk}, \mathsf{id_U})$ and $(\Omega.\mathsf{ct_U}, \Omega.\mathsf{r_U}) \leftarrow \Omega.\mathsf{Enc}_1(\mathsf{mpk})$. $\mathcal{B}$ then returns $\mathsf{pk_U} = (\mathsf{id_U}, \Omega.\mathsf{ct_U})$ to $\mathcal{A}$ and keeps $\mathsf{sk_U} = (\Omega.\mathsf{sk}_{\mathsf{id_U}}, \mathsf{r_U})$ secret.
  - $\mathcal{O}_{\mathsf{PKGen_R}}(\mathsf{id_U})$: $\mathcal{B}$ first computes $\Omega.\mathsf{sk}_{\mathsf{id_U}} \leftarrow \Omega.\mathsf{Extract}(\Omega.\mathsf{msk}, \mathsf{id_U})$ and $(\Omega.\mathsf{ct_U}, \Omega.\mathsf{r_U}) \leftarrow \Omega.\mathsf{Enc}_1(\mathsf{mpk})$. $\mathcal{B}$ also computes $(\Psi.\mathsf{mpk}, \Psi.\mathsf{msk}) \leftarrow \Psi.\mathsf{Setup}(1^\lambda)$. Finally, $\mathcal{B}$ returns $\mathsf{pk_U} = (\mathsf{id_U}, \Omega.\mathsf{ct_U}, \Psi.\mathsf{mpk})$ to $\mathcal{A}$ and keeps $\mathsf{sk_U} = (\Omega.\mathsf{sk}_{\mathsf{id_U}}, \Omega.\mathsf{r_U}, \Psi.\mathsf{msk})$ secret.
  - $\mathcal{O}_{\mathsf{PAEKS}}(\mathsf{kw}, \mathsf{pk_U})$: $\mathcal{B}$ first computes $\mathsf{k}_{\mathsf{id_S},\mathsf{id_U}} \leftarrow \Omega.\mathsf{Dec}(\Omega.\mathsf{sk}_{\mathsf{id_S}}, \mathsf{id_S}, \Omega.\mathsf{ct_U})$ and $\mathsf{k}_{\mathsf{id_U},\mathsf{id_S}} \leftarrow \Omega.\mathsf{Enc}_2(\Omega.\mathsf{mpk}, \mathsf{id_U}, \Omega.\mathsf{r_S})$. Then, $\mathcal{B}$ computes $\mathsf{shk} \leftarrow \mathsf{k}_{\mathsf{id_S},\mathsf{id_U}} \oplus \mathsf{k}_{\mathsf{id_U},\mathsf{id_S}}$ and computes $\mathsf{f} \leftarrow \mathsf{F}(\mathsf{kw}\|\mathsf{shk})$. Next, $\mathcal{B}$ randomly chooses $\mathsf{r} \leftarrow \{0,1\}^*$, computes $\Psi.\mathsf{ct_{kw}} \leftarrow \Psi.\mathsf{Enc}(\Psi.\mathsf{mpk}, \mathsf{f}, \mathsf{r})$ and computes $\mathsf{h} = \mathsf{H}(\Psi.\mathsf{ct_{kw}}, \mathsf{r})$. Finally, $\mathcal{B}$ returns $\mathsf{ct} = (\Psi.\mathsf{ct_{kw}}, \mathsf{h})$ to $\mathcal{A}$.
  - $\mathcal{O}_{\mathsf{Trapdoor}}(\mathsf{kw}, \mathsf{pk_U})$: $\mathcal{B}$ first computes $\mathsf{k}_{\mathsf{id_R},\mathsf{id_U}} \leftarrow \Omega.\mathsf{Dec}(\Omega.\mathsf{sk}_{\mathsf{id_R}}, \mathsf{id_R}, \Omega.\mathsf{ct_U})$ and $\mathsf{k}_{\mathsf{id_U},\mathsf{id_R}} \leftarrow \Omega.\mathsf{Enc}_2(\Omega.\mathsf{mpk}, \mathsf{id_U}, \Omega.\mathsf{r_R})$. Then, $\mathcal{B}$ computes $\mathsf{shk} \leftarrow \mathsf{k}_{\mathsf{id_R},\mathsf{id_U}} \oplus \mathsf{k}_{\mathsf{id_U},\mathsf{id_R}}$ and computes $\mathsf{f} \leftarrow \mathsf{F}(\mathsf{kw}\|\mathsf{shk})$. Next, $\mathcal{B}$ invokes $\Psi.\mathsf{Extract}$ oracle of the IND-ANON-ID-CPA game on f, and is given $\Psi.\mathsf{sk_{kw}}$. Finally, $\mathcal{B}$ returns a trapdoor $\mathsf{tw} = \Psi.\mathsf{sk_{kw}}$ to $\mathcal{A}$.

– **Challenge.** After the end of **Phase 1**, $\mathcal{A}$ outputs two keywords $\mathsf{kw}_0^*, \mathsf{kw}_1^* \in W$ with the following restriction: for $i = 0, 1$, $(\mathsf{kw}_i^*, \mathsf{pk_S})$ and $(\mathsf{kw}_i^*, \mathsf{pk_R})$ have not been queried to oracles $\mathcal{O}_{\mathsf{PAEKS}}$ and $\mathcal{O}_{\mathsf{Trapdoor}}$ in **Phase 1**, respectively. $\mathcal{B}$ then selects a bit $b \in \{0,1\}$ and runs the subsequent steps.
  1. Compute $\mathsf{k}_{\mathsf{id_S},\mathsf{id_R}} \leftarrow \Omega.\mathsf{Dec}(\Omega.\mathsf{sk}_{\mathsf{id_S}}, \mathsf{id_S}, \Omega.\mathsf{ct_R})$.
  2. Compute $\mathsf{k}_{\mathsf{id_R},\mathsf{id_S}} \leftarrow \Omega.\mathsf{Enc}_2(\Omega.\mathsf{mpk}, \mathsf{id_R}, \Omega.\mathsf{r_S})$.
  3. Compute $\mathsf{shk} \leftarrow \mathsf{k}_{\mathsf{id_S},\mathsf{id_R}} \oplus \mathsf{k}_{\mathsf{id_R},\mathsf{id_S}}$.
  4. Compute $\mathsf{f}_0 \leftarrow \mathsf{F}(\mathsf{kw}_0^*\|\mathsf{shk})$ and $\mathsf{f}_1 \leftarrow \mathsf{F}(\mathsf{kw}_1^*\|\mathsf{shk})$.
  5. Invoke the Challenge phase of the IND-ANON-ID-CPA game on $(\mathsf{f}_0, \mathsf{f}_1, \mathsf{r})$, where r is randomly chosen from $\{0,1\}^*$, and is given $\Psi.\mathsf{ct}^*$.
  6. Compute $\mathsf{h} = \mathsf{H}(\Psi.\mathsf{ct}^*, \mathsf{r})$.
  7. Return $\mathsf{ct}^* = (\Psi.\mathsf{ct}^*, \mathsf{h})$ to $\mathcal{A}$.

– **Phase 2.** $\mathcal{A}$ can continue to make queries, as was the case in **Phase 1**. The only restriction is that $\mathcal{A}$ cannot make any query to $\mathcal{O}_{\mathsf{PAEKS}}$ and $\mathcal{O}_{\mathsf{Trapdoor}}$ regarding $(\mathsf{kw}_i^*, \mathsf{pk_S})$ and $(\mathsf{kw}_i^*, \mathsf{pk_R})$, respectively.

– **Guess.** $\mathcal{A}$ outputs its guess b'. Then, $\mathcal{B}$ follows $\mathcal{A}$'s answer and outputs b'.

Regardless of whether $\Psi.\mathsf{ct}^*$ is generated from f_0 or f_1, from $\mathcal{A}$'s perspective, $\mathsf{ct}^* = (\Psi.\mathsf{ct}^*, \mathsf{h})$ is a valid searchable ciphertext. Thus, $\mathcal{A}$ can whether distinguish $\Psi.\mathsf{ct}^*$ is generated from f_0 or f_1 and win the IND-CKA game with non-negligible advantage. Then, $\mathcal{B}$ can follow $\mathcal{A}$'s answer to win the IND-ANON-ID-CPA of the underlying IBE scheme Ψ with the non-negligible advantage. Therefore, we have

$$\mathbf{Adv}_{\Pi,\mathcal{A}}^{\mathsf{IND\text{-}CKA}}(\lambda) \leq \mathbf{Adv}_{\Psi,\mathcal{B}}^{\mathsf{IND\text{-}ANON\text{-}ID\text{-}CPA}}(\lambda).$$

Theorem 2. *The proposed PAEKS scheme Π is IND-IKGA secure if the underlying pseudorandom generator* F *satisfies pseudorandomness.*

Proof (Proof of Theorem 2). Let $\mathcal{A}$ be a PPT adversary that attacks the IND-IKGA security of the PAEKS scheme Π with advantage $\mathbf{Adv}_{\Pi,\mathcal{A}}^{\mathsf{IND-IKGA}}(\lambda)$. We prove Theorem 2 through the following two games, where we define E_i to be the event that $\mathcal{A}$ wins Game_i.

Game_0: This is the original IND-IKGA game, defined in Section 4. The simulation of this game is similar to Theorem 1, except for the challenge phase.

- **Challenge.** After the end of **Phase 1**, $\mathcal{A}$ outputs two keywords $\mathsf{kw}_0^*, \mathsf{kw}_1^* \in W$ with the following restriction: for $i = 0, 1$, $(\mathsf{kw}_i^*, \mathsf{pk_R})$ and $(\mathsf{kw}_i^*, \mathsf{pk_R})$ have not been queried to oracles $\mathcal{O}_{\mathsf{PAEKS}}$ and $\mathcal{O}_{\mathsf{Trapdoor}}$ in **Phase 1**, respectively. $\mathcal{B}$ then runs the following steps:
  1. Random choose a bit $\beta \in \{0, 1\}$.
  2. Compute $\mathsf{k}_{\mathsf{id_R},\mathsf{id_S}} \leftarrow \Omega.\mathsf{Dec}(\Omega.\mathsf{sk}_{\mathsf{id_R}}, \mathsf{id_R}, \Omega.\mathsf{ct_S})$.
  3. Compute $\mathsf{k}_{\mathsf{id_S},\mathsf{id_R}} \leftarrow \Omega.\mathsf{Enc}_2(\Omega.\mathsf{mpk}, \mathsf{id_S}, \Omega.\mathsf{r_R})$.
  4. Compute $\mathsf{shk} \leftarrow \mathsf{k}_{\mathsf{id_R},\mathsf{id_S}} \oplus \mathsf{k}_{\mathsf{id_S},\mathsf{id_R}}$, where $\oplus$ is an operation compatible with the key space.
  5. Compute $\mathsf{f} \leftarrow \mathsf{F}(\mathsf{kw}_\beta \| \mathsf{shk})$.
  6. Return a challenge trapdoor $\mathsf{tw}^* \leftarrow \Psi.\mathsf{Extract}(\Psi.\mathsf{msk}, \mathsf{f})$ to $\mathcal{A}$.

By the definition,

$$\mathbf{Adv}_{\Pi,\mathcal{A}}^{\mathsf{IND-KGA}}(\lambda) = \left|\mathbf{Pr}[\mathsf{E}_0] - \tfrac{1}{2}\right|.$$

Game_1: In this game, we make the following minor conceptual change to the aforementioned game. In the challenge phase, the challenger $\mathcal{B}$ substitutes the value $\mathsf{ct}^* \leftarrow \Psi.\mathsf{Enc}(\mathsf{pk_R}, \mathsf{f}, \mathsf{r})$ with $\mathsf{ct}^* \leftarrow \Psi.\mathsf{Enc}(\mathsf{pk_R}, \mathsf{f}', \mathsf{r})$, where f' is randomly selected from the output space $\mathcal{Y}$ of the underlying pseudorandom generator $\mathcal{Y}$. Then, according to Lemma 3, because the pseudorandom generator F satisfies pseudorandomness, no distinguisher can distinguish f from f' with non-negligible probability:

$$|\mathbf{Pr}[\mathsf{E}_0] - \mathbf{Pr}[\mathsf{E}_1]| = \mathbf{Adv}_{\mathsf{F},\mathcal{B}}^{\mathsf{PRG}}(\lambda).$$

Lemma 3. *For all PPT algorithms $\mathcal{A}_1$, $|\mathbf{Pr}[\mathsf{E}_0] - \mathbf{Pr}[\mathsf{E}_1]|$ is negligible if the underlying pseudorandom generator F satisfies pseudorandomness.*

Proof (Proof of Lemma 3). If $\mathcal{A}_1$ can win the IND-IKGA game with non-negligible advantage, then there exists a challenger $\mathcal{B}$ that can win the pseudorandom game of the underlying pseudorandom generator with non-negligible advantage. $\mathcal{B}$ constructs a hybrid game, interacting with $\mathcal{A}_1$ as follows. Given a challenge string $T \in \mathcal{Y}$ and the description of a pseudorandom generator F', $\mathcal{B}$ constructs a hybrid game, interacting with $\mathcal{A}_1$ as follows.

- **Initialization.** $\mathcal{B}$ chooses the public parameter following the proposed construction, with the following exception: rather than selecting a proper pseudorandom generator from the pseudorandom generator family, $\mathcal{B}$ sets F' as a system parameter. $\mathcal{B}$ then follows the previous game to generate the system parameter params, data sender's key pair $(\mathsf{pk_S}, \mathsf{sk_S})$, and data receiver's key pair $(\mathsf{pk_R}, \mathsf{sk_R})$. Finally, $\mathcal{B}$ sends $(\mathsf{PP}, \mathsf{pk_S}, \mathsf{pk_R})$ to $\mathcal{A}$ and keeps $(\mathsf{msk}, \mathsf{sk_S}, \mathsf{sk_R})$ secret.
- **Phase 1.** $\mathcal{A}_1$ can make polynomially many queries to oracles as was the case in a previous game.
- **Challenge.** After the end of **Phase 1**, $\mathcal{A}_1$ outputs two keywords $\mathsf{kw}_0^*, \mathsf{kw}_1^* \in W$ with the following restriction: for $i = 0, 1$, $(\mathsf{kw}_i^*, \mathsf{pk_S})$ and $(\mathsf{kw}_i^*, \mathsf{pk_R})$ have not been queried to oracles $\mathcal{O}_{\mathsf{PAEKS}}$ and $\mathcal{O}_{\mathsf{Trapdoor}}$ in **Phase 1**, respectively. $\mathcal{B}$ then runs the subsequent steps.
  1. Set $\mathsf{f}^* = T$.
  2. Compute $\mathsf{tw}^* = \Psi.\mathsf{Extract}(\Psi.\mathsf{msk}, \mathsf{f}^*)$.
  3. Return tw^* to $\mathcal{A}_1$.
- **Phase 2.** $\mathcal{A}$ can continue to make queries, same as in **Phase 1**. The only restriction is that $\mathcal{A}_1$ cannot make any query to $\mathcal{O}_{\mathsf{PAEKS}}$ on $(\mathsf{kw}_i^*, \mathsf{pk_S})$ and $\mathcal{O}_{\mathsf{Trapdoor}}$ on $(\mathsf{kw}_i^*, \mathsf{pk_R})$, for $i = 0, 1$.

– **Guess.** $\mathcal{A}_1$ outputs its guess b'.

If $\mathcal{T}$ is generated from F', $\mathcal{B}$ provides the view of Game_0 to $\mathcal{A}_1$; if T is a random string sampled from $\mathcal{Y}$, then $\mathcal{B}$ provides the view of Game_1 to $\mathcal{A}_1$. Hence, if $|\mathbf{Pr}[\mathsf{E}_0] - \mathbf{Pr}[\mathsf{E}_1]|$ is non-negligible, $\mathcal{B}$ has a non-negligible advantage against the pseudorandom generator security game. Therefore, the advantage of $\mathcal{A}_1$ is

$$|\mathbf{Pr}[\mathsf{E}_0] - \mathbf{Pr}[\mathsf{E}_1]| = \mathbf{Adv}^{\mathsf{PRG}}_{\mathsf{F},\mathcal{B}}(\lambda).$$

Lemma 4. $\mathbf{Pr}[\mathsf{E}_1] = 0$.

Proof (Proof of Lemma 4). The proof of this lemma is intuitive. Because the trapdoor tw^* contains no information regarding the keyword, the adversary can only return b' by guessing.

Combining Lemmas 3 and 4, we can conclude that the advantage of $\mathcal{A}$ in winning the IND-IKGA game is

$$\begin{aligned}\mathbf{Adv}^{\mathsf{IND-KGA}}_{\Pi,\mathcal{A}}(\lambda) &= \left|[\mathsf{E}_0] - \frac{1}{2}\right| \\ &= \left|\mathbf{Pr}[\mathsf{E}_1] + \mathbf{Adv}^{\mathsf{PRG}}_{\mathsf{F},\mathcal{B}}(\lambda)\right| \\ &\leq \mathbf{Adv}^{\mathsf{PRG}}_{\mathsf{F},\mathcal{B}}(\lambda).\end{aligned}$$

This completes the proof.

Table 2. Comparison of Security Properties with Other PAEKS Schemes

Schemes	IKGAs	Quantum-resistance	Security
HL17 [20]	✗	✗	ROM
HMZKL17 [18]	✓	✗	ROM
NE18 [34]	✓	✗	ROM
LHSYS19 [24]	✓	✗	ROM
WZMKH19 [40]	✓	✗	ROM
LLYSTH19 [25]	✓	✗	ROM
QCHLZ20 [36]	✓	✗	ROM
PSE20 [35]	✓	✗	ROM
Ours	✓	✓	SM

✓: the scheme supports the corresponding feature.
✗: the scheme fails in supporting the corresponding feature.
ROM: random oracle model; SM: standard model.

Table 3. Experimentation Platform Information

Description	Data
CPU	AMD Ryzen 5-2600 3.4GHz
CPU processor number	6
Operation system	Ubuntu 18.04
Linux kernel version	5.3.0-59-generic
Random access memory	16.3GB
Solid state disk	232.9GB

7 Concrete Instantiation

In this section we give a concrete instantiation by adopting Katsumata and Yamada's adaptively anonymous IBE [21], which is secure under the RLWE assumption. In other words, we first follow the idea in [5] to tweak [21] and obtain a two-tier IBKEM. We then combine this two-tier IBKEM with Katsumata and Yamada's IBE [21] to instantiate a quantum-resistant PAEKS scheme. The instantiation is comprehensively detailed in the subsequent section.

Table 4. Notations of Operations and Their Running Time

Notations	Operations	Running time (ms)
T_H	Hash-to-point	59.423115
T_{BP}	Bilinear pairing	7.526549
T_{SM}	Scalar multiplication over point	4.094151
T_{GM}	General multiplication over point	0.022241
T_{EX}	Modular exponentiation over point	4.153842
T_{PA}	Addition over point	0.019013
T_{HA}	General hash function	0.008724
T_{PRG}	Pseudorandom generation	0.005132
T_{PRM}	Multiplication over polynomial ring	0.003416
T_{PRA}	Addition over polynomial ring	0.001813
T_{SAM}	$\mathsf{SampleLeft}$ function	6.219831

Table 5. Comparison of Needing Operations with Other PAEKS Schemes

Schemes	Ciphertext generation	Trapdoor generation	Testing
HL17 [20]	$T_H + 3T_{EX} + T_{GM}$	$T_H + T_{BP} + T_{EX}$	$2T_{BP} + T_{GM}$
HMZKL17 [18]	$T_H + 3T_{BP} + 5T_{SM} + 2T_{PA} + 2T_{HA}$	$T_H + T_{BP} + 3T_{SM} + 2T_{PA} + 2T_{HA}$	$2T_{BP} + 2T_{SM} + T_{GM} + 2T_{PA} + 2T_{HA}$
NE18 [34]	$T_H + 3T_{EX} + T_{GM}$	$T_H + T_{BP} + T_{EX}$	$2T_{BP} + T_{GM}$
LHSYS19 [24]	$2T_H + 2T_{BP} + 3T_{EX}$	$4T_H + T_{BP} + T_{GM}$	$2T_{BP} + T_{GM} + 2T_{EX}$
WZMKH19 [40]	$T_H + 6T_{SM} + 2T_{PA} + 2T_{HA}$	$T_H + +T_{BP} + 9T_{SM} + 4T_{PA} + T_{HA}$	$2T_{BP} + 4T_{SM} + T_{EX} + 2T_{PA} + T_{HA}$
LLYSTH19 [25]	$T_H + 3T_{SM} + T_{PA}$	$T_H + T_{BP} + 4T_{SM} + 2T_{PA}$	$2T_{BP} + 2T_{SM} + T_{GM} + 2T_{PA}$
QCHLZ20 [36]	$3T_H + 2T_{BP} + 3T_{EX} + T_{HA}$	$3T_H + T_{BP} + 2T_{EX}$	$T_H + T_{BP}$
PSE20 [35]	$T_H + T_{BP} + 3T_{SM} + 2T_{HA}$	$T_H + T_{BP} + T_{SM} + T_{HA}$	$T_{SM} + T_{HA}$
Ours	$2T_{HA} + T_{PRG}$ $+(6k+7)T_{PRM} + (2k+2)T_{PRA}$	$T_{HA} + T_{PRG}$ $+(4k+4)T_{PRM} + T_{SAM}$	$T_{HA} + (2k+1)T_{PRM}$

– $\mathsf{Setup}(1^\lambda)$: Given a security parameter 1^λ, this algorithm runs as follows.
  1. Choose a proper PRG $\mathsf{F} : \{0,1\}^{\kappa+n} \to \{0,1\}^{\kappa+n+1}$.
  2. Choose three secure hash functions:
     - $\mathsf{H}_1 : \{0,1\}^{\kappa+n+1} \to \{0,1\}^{\kappa}$.
     - $\mathsf{H}_2 : \{0,1\}^{2k+1+n} \to \{0,1\}^{\kappa}$.
  3. Run $(\mathbf{a}, \mathbf{T_a}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q, \rho)$, where $\mathbf{a} \in R_q^k$, and $\mathbf{T} \in R^{k\times k}$
  4. Pick $u \xleftarrow{\$} R_q, \mathbf{b}_0, \mathbf{b}_{i,j} \xleftarrow{\$} R_q^k$ for $(i,j) \in [d] \times [\ell]$.
  5. Choose a deterministic function $\mathsf{H} : \{0,1\}^\kappa \to R_q^k$ defined as $\mathsf{H}(m) = \mathbf{b}_0 + \sum_{(j_1,\cdots,j_d)\in S(m)} \mathsf{PubEval}_d(\mathbf{b}_{1,j_1}, \cdots, \mathbf{b}_{d,j_d}) \in R_q^k$, where S is an efficiently computable injective map that maps a message $m \in \{0,1\}^\kappa$ to a subset $S(m)$ of $[1,\ell]^d$, where $\ell = \lceil \kappa^{1/d} \rceil$.
  6. Outputs $\mathsf{PP} = (\mathbf{a}, \mathbf{b}_0, \{\mathbf{b}_{i,j}\}_{(i,j)\in[d]\times[\ell]}, u, \mathsf{H})$ and master private key $\mathsf{msk} = \mathbf{T}$. Note that msk is kept secret by the trusted authority.

– $\mathsf{KeyGen_S}(\mathsf{PP}, \mathsf{msk}, \mathsf{id_S})$: Given a system parameter PP, a master private key msk, and an identity $\mathsf{id_S} \in \{0,1\}^\kappa$, data sender and trusted authority interact as follows.
  1. Data sender registers his/her identity $\mathsf{id_S}$ to trusted authority. The trusted authority computes the following steps.
     (a) Choose $s_S, v_S \xleftarrow{\$} R_q$, and $\mathbf{x}_{S,1}, \mathbf{x}_{S,2} \xleftarrow{\$} (D^{\mathsf{coeff}}_{\mathbb{Z}^n,\alpha'})^k$.
     (b) Compute $\mathbf{p}_S = s_S[\mathbf{a}|\mathsf{H}(\mathsf{id_S})] + [\mathbf{x}_{S,1}|\mathbf{x}_{S,2}] \in R_q^{2k}$.

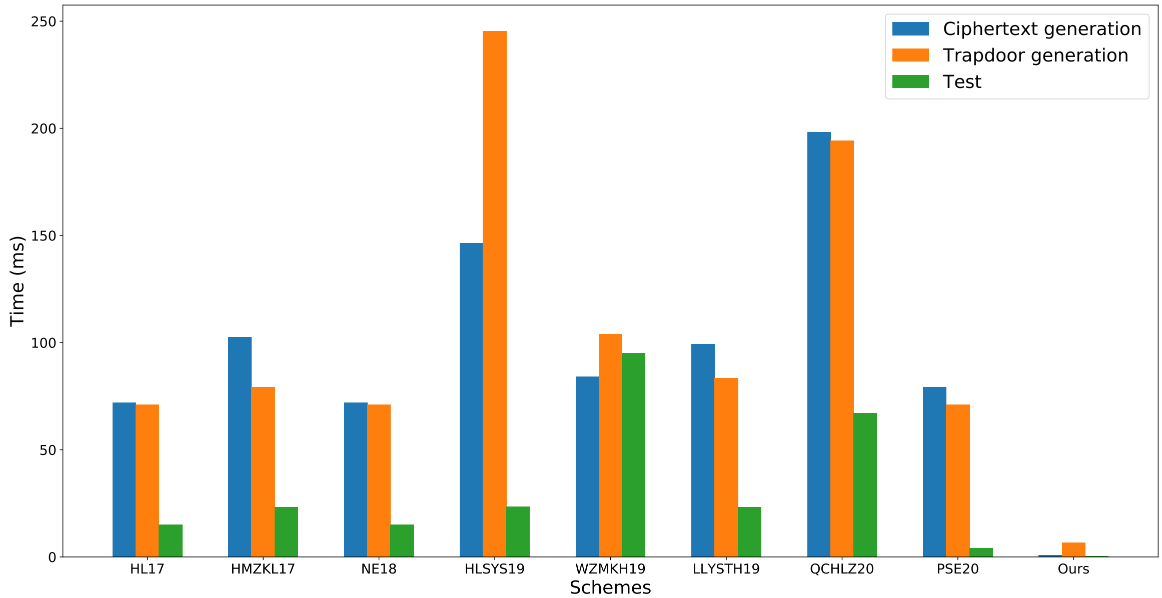

Fig. 3. Comparison of computational costs with other PAEKS schemes.

   (c) Compute $\mathbf{e}_S \leftarrow \mathsf{SampleLeft}(\mathbf{a}, \mathsf{H}(\mathsf{id_S}), u, \mathbf{T_a}, \sigma)$.

   (d) Return $(s_S, \mathbf{p}_S, \mathbf{e}_S)$ to data sender.

2. Data sender outputs his/her public key $\mathsf{pk_S} = (\mathsf{id_S}, \mathbf{p}_S, v_S)$ and private key $\mathsf{sk_S} = (\mathbf{e}_S, s_S)$.

– $\mathsf{KeyGen_R}(\mathsf{PP}, \mathsf{msk}, \mathsf{id_R})$: Given a system parameter PP, a master private key msk, and an identity $\mathsf{id_R} \in \{0,1\}^\kappa$, data receiver and trusted authority interact as follows.
  1. Data sender registers his/her identity $\mathsf{id_R}$ to trusted authority. The trusted authority computes the following steps.
     (a) Choose $s_R, v_R \xleftarrow{\$} R_q$, and $\mathbf{x}_{R,1}, \mathbf{x}_{R,2} \xleftarrow{\$} (D^{\mathsf{coeff}}_{\mathbb{Z}^n,\alpha'})^k$.

     (b) Compute $\mathbf{p}_R = s_R[\mathbf{a}|\mathsf{H}(\mathsf{id_S})] + [\mathbf{x}_{R,1}|\mathbf{x}_{R,2}] \in R_q^{2k}$.

     (c) Compute $\mathbf{e}_R \leftarrow \mathsf{SampleLeft}(\mathbf{a}, \mathsf{H}(\mathsf{id_R}), u, \mathbf{T}, \sigma)$.

     (d) Return $(s_R, \mathbf{p}_R, \mathbf{e}_R)$ to data sender.

  2. Data receiver then computes $(\mathbf{a}_R, \mathbf{T}_{\mathbf{a}R}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q, \rho)$.

  3. Pick $u_R \xleftarrow{\$} R_q, \mathbf{b}_{R,0}, \mathbf{b}_{R,i,j} \xleftarrow{\$} R_q^k$ for $(i,j) \in [d] \times [\ell]$.

  4. Choose a deterministic function $\mathsf{H}_R : \{0,1\}^\kappa \to R_q^k$ defined as $\mathsf{H}_R(m) = \mathbf{b}_{R,0} + \sum_{(j_1,\cdots,j_d)\in S(m)} \mathsf{PubEval}_d(\mathbf{b}_{R,1,j_1}, \cdots, \mathbf{b}_{R,d,j_d}) \in R_q^k$, where S is an efficiently computable injective map that maps a message $m \in \{0,1\}^\kappa$ to a subset $S(m)$ of $[1,\ell]^d$, where $\ell = \lceil \kappa^{1/d} \rceil$.

  5. Data sender outputs his/her public key $\mathsf{pk_R} = (\mathsf{id_R}, \mathbf{p}_R, v_R, \mathbf{a}_R, \mathbf{b}_{R,0}, \{\mathbf{b}_{R,i,j}\}_{(i,j)\in[d]\times[\ell]}, u_R, \mathsf{H}_R)$ and private key $\mathsf{sk_R} = (\mathbf{e}_R, s_R, \mathbf{T}_R)$.

Table 6. Comparison of Communication Costs with Other PAEKS Schemes

Schemes	Ciphertext overhead	Trapdoor overhead
HL17 [20]	$2\lvert\mathbb{G}_1\rvert$	$\lvert\mathbb{G}_T\rvert$
HMZKL17 [18]	$\lvert\mathbb{G}_1\rvert$	$\lvert\mathbb{G}_T\rvert$
NE18 [34]	$\lvert\mathbb{G}_1\rvert$	$\lvert\mathbb{G}_T\rvert$
LHSYS19 [24]	$2\lvert\mathbb{G}_1\rvert + \lvert\mathbb{G}_T\rvert$	$\lvert\mathbb{G}_1\rvert + \lvert\mathbb{G}_T\rvert$
WZMKH19 [40]	$2\lvert\mathbb{G}_1\rvert$	$2\lvert\mathbb{G}_1\rvert + \lvert\mathbb{G}_T\rvert$
LLYSTH19 [25]	$\lvert\mathbb{G}_1\rvert$	$\lvert\mathbb{G}_T\rvert$
QCHLZ20 [36]	$\lvert\mathbb{G}_1\rvert + \lvert p\rvert$	$\lvert\mathbb{G}_T\rvert$
PSE20 [35]	$2\lvert\mathbb{G}_1\rvert$	$\lvert\mathbb{G}_T\rvert$
Ours	$3nk\lvert q\rvert + n$	$2nk\lvert\sigma\rvert$

n: a power of 2.
p, q: module.
$\mathbb{G}_1, \mathbb{G}_T$: cyclic group.
k: $\mathsf{poly}(n)$.
σ: Gaussian parameter.

- $\mathsf{PAEKS}(\mathsf{PP}, \mathsf{pk_S}, \mathsf{sk_S}, \mathsf{pk_R}, \mathsf{kw})$: Given a system parameter PP, data sender's public key $\mathsf{pk_S}$ and private key $\mathsf{SK_S}$, data receiver's public key $\mathsf{pk_R}$, and a keyword $\mathsf{kw} \in \{0,1\}^k$, data sender runs the following steps.
  1. $\mathsf{k}_{S,1} = (\lfloor (2/q) \cdot \phi(v_R - s_S[\mathbf{a}|\mathsf{H}(\mathsf{id_S})]) \rceil) \bmod 2) \in \{0,1\}^n$.
  2. $\mathsf{k}_{S,2} = (\lfloor (2/q) \cdot \phi(v_S - \mathbf{p}_S\mathbf{e}_S^T) \rceil) \bmod 2) \in \{0,1\}^n$.
  3. $\mathsf{shk}_S \leftarrow \mathsf{k}_{S,1} \oplus \mathsf{k}_{S,2}$.
  4. Compute $\mathsf{pseudo\_kw} \leftarrow \mathsf{H}_1(\mathsf{F}(\mathsf{kw}\|\mathsf{shk}_S))$.
  5. Choose a random $\xi \xleftarrow{\$} \{0,1\}^n$.
  6. Choose $s_S, v_S \xleftarrow{\$} R_q$, $x_{S,0} \xleftarrow{\$} D^{\mathsf{coeff}}_{\mathbb{Z}^n,\alpha q}$ and $\mathbf{x}_{S,1}, \mathbf{x}_{S,2} \xleftarrow{\$} (D^{\mathsf{coeff}}_{\mathbb{Z}^n,\alpha'})^k$.
  7. Compute $c_0 = s_S u_R + x_{S,0} + \lfloor q/2 \rceil \cdot \xi$, $\mathbf{c}_1 = s_S[\mathbf{a}_R|\mathsf{H}_R(\mathsf{pseudo\_kw})] + [\mathbf{x}_{S,1}|\mathbf{x}_{S,2}]$.
  8. Compute $\mathsf{h} = \mathsf{H}_2(c_0, \mathbf{c}_1, b)$.
  9. Output a searchable ciphertext $\mathsf{ct} = (c_0, \mathbf{c}_1, \mathsf{h})$.
- $\mathsf{Trapdoor}$: Given a system parameter PP, data receiver's public key $\mathsf{pk_R}$ and private key $\mathsf{sk_R}$, data sender's public key $\mathsf{pk_S}$, and a keyword $\mathsf{kw} \in \{0,1\}^k$, data receiver runs the following steps.
  1. $\mathsf{k}_{R,1} = (\lfloor (2/q) \cdot \phi(v_R - \mathbf{p}_R\mathbf{e}_R^T) \rceil) \bmod 2) \in \{0,1\}^n$.
  2. $\mathsf{k}_{R,2} = (\lfloor (2/q) \cdot \phi(v_S - s_R[\mathbf{a}|\mathsf{H}(\mathsf{id_R})]) \rceil) \bmod 2) \in \{0,1\}^n$.
  3. $\mathsf{shk}_R \leftarrow \mathsf{k}_{R,1} \oplus \mathsf{k}_{R,2}$.
  4. Compute $\mathsf{pseudo\_kw} \leftarrow \mathsf{H}_1(\mathsf{F}(\mathsf{kw}\|\mathsf{shk}_R))$.
  5. Compute $\mathbf{t} \leftarrow \mathsf{SampleLeft}(\mathbf{a}_R, \mathsf{H}_R(\mathsf{pseudo\_kw}), u_R, \mathbf{T}_R, \sigma)$.
  6. Output a trapdoor $\mathsf{tw} = \mathbf{t}$.
- Test: Given a system parameter PP, a searchable ciphertext ct, ans a trapdoor tw, cloud server works as follows.
  1. Compute $\xi' = (\lfloor (2/q) \cdot \phi(c_0 - \mathbf{c}_1\mathbf{t}^T) \rceil \bmod 2)$.
  2. If $\mathsf{H}_2(c_0, \mathbf{c}_1, \xi') = \mathsf{h}$, output 1; otherwise, output 0.

7.1 Parameter Selection

Due to the correctness and the security of our instantiation is based on [21], we follow the parameter selection in [21]; that is, the following conditions should be hold:

- the error term is less than $\alpha q\omega(\sqrt{\log n}) + \sqrt{nk}\alpha'\sigma\omega(\sqrt{\log nk}$ such that the instantiation has negligible decryption error, where $\alpha q\omega(\sqrt{\log n}) + \sqrt{nk}\alpha'\sigma\omega(\sqrt{\log nk} \leq q/5$ ([21] Lemma 10).
- $\rho < 1/2\sqrt{q/n}$ and $k \geq 2\log_\rho q$ such that $\mathsf{TrapGen}$ can work (Lemma 1).
- $k \geq \lceil \log_b q \rceil$ such that the gadget matrix $\mathbf{g}_b$ can be defined ([21] Lemma 5).
- $\sigma > O(b\rho \cdot \sqrt{n\log_\rho q} \cdot \omega(\sqrt{\log nk})$ and $\sigma < s_1(\mathbf{R})\sqrt{b^2+1} \cdot \omega(\sqrt{\log n})$, where $s_1(\mathbf{R}) \leq \tau \cdot \kappa\rho\sqrt{n}(\sqrt{k} + \omega(\sqrt{n}))\left((cn)^{d-1} + bnk\frac{(cn)^{d-1}-1}{cn-1}\right)$ for some absolute constant τ, such that $\mathsf{SampleLeft}$ and $\mathsf{SampleRight}$ can work (Lemma 2, [21] Eq. 37).
- $\alpha' > 2\alpha q(s_1(\mathbf{R})+1), \alpha q > \omega(\sqrt{\log nk})$, and $\alpha q \geq n^{3/2}k^{1/4}\omega(\log^{9/4} n)$ such that the IBE scheme in [35] is secure ([21] Theorem 1).

For our instantiation, we set the concrete parameters as Type 2 IBE in [21], *i.e.,* $k = 8d+12, q = n^{2d+3}, b = \rho = n^{\frac{1}{4}}, \sigma = n^d \cdot \omega(\log n), \alpha = n^{-2d-\frac{3}{8}} \cdot \omega(\log^2 n)^{-1}$, and $\alpha' = n^{d+\frac{5}{2}} \cdot \omega(\log^{\frac{3}{4}} n)^{-1}$ to ensure that our instantiation is correct.

8 Comparison and Analysis

To the best of our knowledge, although existing PAEKS schemes [18,34,24,36,35,40,25] can defend against IKGAs, these schemes cannot defend against quantum attacks because the security of these schemes are based on the discrete logarithm assumption. In this section, we first compare our proposed instantiation with these existing schemes with respect to their security properties. We then compared these schemes with respect to their computational and communication complexities.

Table 2 lists the results of our comparison between our instantiation and its counterpart PAEKS schemes with respect to their security properties. Because our instantiation inherits the security of [17,1], it can be considered to be based on the lattice hard assumption. In other words, only our instantiation has the ability to resist quantum attacks and IKGAs simultaneously.

We subsequently conducted such a comparison with respect to computational complexity when generating searchable ciphertexts and trapdoors. For simplicity, we only considered the time-consuming operations listed in Table 4. Experiments simulating these operations were performed on a PC; the efficiency of the methods are detailed in Table 3. In particular, the operations of T_H, T_{BP}, T_{SM}, T_{GM}, T_{EX}, and T_{PA} were obtained by using a pairing-based cryptography library (PBC)—under Type-A pairing with a 160-bit group order and a 2048-bit group element for $\mathbb{G}_1$ and $\mathbb{G}_T$ [28]. T_{PRM}, T_{PRA}, and T_{SAM} were simulated using an NFLlib libarary [2] with the parameters $d = 2$, $n = 512$, and $k = 30$. Moreover, T_{PRG} was obtained using the AES-256 algorithm[3], and T_{HA} was simulated using the SHA3-256 algorithm[4]. The computational costs for the methods are compared in Table 5. The results indicate that our instantiation took the least time to generate the ciphertext and trapdoor as well as to perform tests; such speed was due to our method not requiring any time-consuming operations, such as bilinear pairing and hash-to-point.

Additionally, we also conducted such a comparison with respect to communication complexity (which was indicated by the size of the ciphertext and trapdoor). The comparison results are detailed in Table 6. For the pairing-based schemes, the pairing operation is represented by $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$, where $\mathbb{G}_1$ and $\mathbb{G}_T$ are 2048-bit elements. Moreover, because the group order of the pairing is 160 bit, $|p| = 160$. For our instantiation, $n = 512$, $k = 30$, $|\sigma| = 20$, and $|q| = 63$. To ensure security, our instantiation must be set in high dimensions. Therefore, in contrast to its counterpart schemes, our instantiation yielded ciphertext and trapdoor sizes of $3nk|q| + n = 2903040$ bits and $2nk|\sigma| = 614400$ bits, respectively.

[3] `https://github.com/kokke/tiny-AES-c`
[4] `https://github.com/brainhub/SHA3IUF`

9 Conclusion

In this work, we introduced a new method for constructing a generic PAEKS scheme, which is secure against IND-KGA and IND-IKGAs under multi-user context in standard model. In addition, we provided a lattice-based concrete instantiation based on the lattice hard assumption. Compared with current PAEKS schemes, our instantiation is not only the first PAEKS scheme that is quantum-resistant and secure under standard model but also the most efficient scheme with respect to computational cost.

Recently, PAEKS schemes [24,36] consider the designated testability of PAEKS, which ensures that only designated server can execute the test function. Hence, even when an adversary monitors the communication channel between a user and the cloud for the encrypted ciphertext and its corresponding trapdoor, they cannot learn any information regarding the user's search pattern. We intend to investigate the aforementioned problem in our future work.

Acknowledgment

This research was supported by the Ministry of Science and Technology, Taiwan (ROC), under Project Numbers MOST 108-2218-E-004-001-, MOST 108-2218-E-004-002-MY2, MOST 109-2218-E-011-007-.

References

1. Agrawal, S., Boneh, D., Boyen, X.: Efficient Lattice (H)IBE in the Standard Model. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110. pp. 553–572. Springer, Berlin, Heidelberg (2010)
2. Aguilar-Melchor, C., Barrier, J., Guelton, S., Guinet, A., Killijian, M.O., Lepoint, T.: NFLlib: NTT-based Fast Lattice Library. In: Sako, K. (ed.) CT-RSA 2016. LNCS, vol. 9610. pp. 341–356. Springer Cham (2016)
3. Arute, F., Arya, K., Babbush, R., Bacon, D., Bardin, J.C., Barends, R., Biswas, R., Boixo, S., Brandao, F.G., Buell, D.A., et al.: Quantum Supremacy using a Programmable Superconducting Processor. Nature **574**(7779), 505–510 (2019). https://doi.org/10.1038/s41586-019-1666-5
4. Bellare, M., Boldyreva, A., Palacio, A.: An Uninstantiable Random-oracle-model Scheme for a Hybrid-encryption Problem. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. pp. 171–188. Springer, Berlin, Heidelberg (2004)
5. Blazy, O., Chevalier, C.: Non-Interactive Key Exchange from Identity-Based Encryption. In: ARES 2018. Association for Computing Machinery (2018). https://doi.org/10.1145/3230833.3230864
6. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public Key Encryption with Keyword Search. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027. pp. 506–522. Springer, Berlin, Heidelberg (2004)
7. Byun, J.W., Rhee, H.S., Park, H.A., Lee, D.H.: Off-line Keyword Guessing Attacks on Recent Keyword Search Schemes over Encrypted Data. In: Jonker, W., Petkovic, M. (eds.) SDM 2006. LNCS, vol. 4165. pp. 75–83. Springer, Berlin, Heidelberg (2006). https://doi.org/10.1007/11844662_6
8. Canetti, R., Goldreich, O., Halevi, S.: The Random Oracle Methodology, Revisited. J. ACM **51**(4), 557–594 (2004)
9. Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai Trees, or How to Delegate a Lattice Basis. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110. pp. 523–552. Springer Berlin Heidelberg (2010)
10. Chen, B., Wu, L., Zeadally, S., He, D.: Dual-Server Public-Key Authenticated Encryption With Keyword Search. IEEE Trans. Cloud Comput. (2019), (early access)
11. Chen, R., Mu, Y., Yang, G., Guo, F., Huang, X., Wang, X., Wang, Y.: Server-aided public key encryption with keyword search. IEEE Trans. on Information Forensics and Security **11**(12), 2833–2842 (2016). https://doi.org/10.1109/TIFS.2016.2599293
12. Chen, R., Mu, Y., Yang, G., Guo, F., Wang, X.: A New General Framework for Secure Public Key Encryption with Keyword Search. In: Foo, E., Stebila, D. (eds.) ACISP 2015. LNCS, vol. 9144. pp. 59–76. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-19962-7_4
13. Chen, R., Mu, Y., Yang, G., Guo, F., Wang, X.: Dual-server Public-key Encryption with Keyword Search for Secure Cloud Storage. IEEE Trans. on Information Forensics and Security **11**(4), 789–798 (2015). https://doi.org/10.1109/TIFS.2015.2510822
14. Chen, Y.C., Xie, X., Wang, P.S., Tso, R.: Witness-based Searchable Encryption with Optimal Overhead for Cloud-edge Computing. Future Gener. Comput. Syst. **100**, 715–723 (2019). https://doi.org/https://doi.org/10.1016/j.future.2019.05.038
15. Fang, L., Susilo, W., Ge, C., Wang, J.: A Secure Channel Free Public Key Encryption with Keyword Search Scheme without Random Oracle. In: Garay, J., Miyaji, A., Otsuka, A. (eds.) CANS 2009. LNCS, vol. 5888. pp. 248–258. Springer, Berlin, Heidelberg (2009)
16. Fang, L., Susilo, W., Ge, C., Wang, J.: Public Key Encryption with Keyword Search Secure against Keyword Guessing Attacks without Random Oracle. Inf. Sci. **238**, 221–241 (2013). https://doi.org/10.1016/j.ins.2013.03.008
17. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for Hard Lattices and New Cryptographic Constructions. In: STOC 2008. pp. 197–206. Association for Computing Machinery (2008)

18. He, D., Ma, M., Zeadally, S., Kumar, N., Liang, K.: Certificateless Public Key Authenticated Encryption with Keyword Search for Industrial Internet of Things. IEEE Trans. on Ind. Informatics **14**(8), 3618–3627 (2017)
19. Huang, K., Tso, R.: Provable Secure Dual-server Public Key Encryption with Keyword Search. In: 2017 IEEE 2nd International Verification and Security Workshop (IVSW). pp. 39–44. IEEE (2017)
20. Huang, Q., Li, H.: An Efficient Public-key Searchable Encryption Scheme Secure against Inside Keyword Guessing Attacks. Inf. Sci. **403**, 1–14 (2017). https://doi.org/10.1016/j.ins.2017.03.038
21. Katsumata, S., Yamada, S.: Partitioning via Non-linear Polynomial Functions: More compact IBEs from Ideal Lattices and Bilinear Maps. In: Cheon, J., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10032. pp. 682–712. Springer, Berlin, Heidelberg (2016)
22. Katz, J., Lindell, Y.: Introduction to Modern Cryptography. Chapman and Hall/CRC (2014)
23. Leurent, G., Nguyen, P.Q.: How Risky is the Random-oracle Model? In: Halevi, S. (ed.) CRYPTO 2009. pp. 445–464. Springer, Berlin, Heidelberg (2009)
24. Li, H., Huang, Q., Shen, J., Yang, G., Susilo, W.: Designated-server Identity-based Authenticated Encryption with Keyword Search for Encrypted Emails. Inf. Sci. **481**, 330–343 (2019)
25. Liu, X., Li, H., Yang, G., Susilo, W., Tonien, J., Huang, Q.: Towards Enhanced Security for Certificateless Public-Key Authenticated Encryption with Keyword Search. In: Steinfeld, R., Yuen, T. (eds.) ProvSec 2019. LNCS, vol. 11821. pp. 113–129. Springer Cham (2019)
26. Liu, Z.Y., Tseng, Y.F., Tso, R.: Cryptanalysis of "FS-PEKS: Lattice-based forward secure public-key encryption with keyword search for cloud-assisted industrial Internet of things". Cryptology ePrint Archive, Report 2020/651 (2020), `https://eprint.iacr.org/2020/651`
27. Liu, Z.Y., Tseng, Y.F., Tso, R., Mambo, M.: Designated-ciphertext searchable encryption. Cryptology ePrint Archive, Report 2019/1418 (2019), `https://eprint.iacr.org/2019/1418`
28. Lynn, B.: PBC Library the Pairing-cryptography Library (2014)
29. Lyubashevsky, V., Peikert, C., Regev, O.: On Ideal Lattices and Learning with Errors over Rings. In: Gibert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110. pp. 1–23. Springer, Berlin, Heidelberg (2010)
30. Lyubashevsky, V., Peikert, C., Regev, O.: On Ideal Lattices and Learning with Errors over Rings. J. ACM **60**(6), 1–35 (2013)
31. Ma, S., Mu, Y., Susilo, W., Yang, B.: Witness-based Searchable Encryption. Inf. Sci. **453**, 364–378 (2018). https://doi.org/10.1016/j.ins.2018.04.012
32. Mao, Y., Fu, X., Guo, C., Wu, G.: Public Key Encryption with Conjunctive Keyword Search Secure against Keyword Guessing Attack from Lattices. Trans. Emerg. Telecommun. Technolo. **30**(11), e3531 (2019). https://doi.org/10.1002/ett.3531
33. Micciancio, D., Peikert, C.: Trapdoors for Lattices: Simpler, Tighter, Faster, Smaller. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237. pp. 700–718. Springer, Berlin, Heidelberg (2012)
34. Noroozi, M., Eslami, Z.: Public Key Authenticated Encryption with Keyword Search: Revisited. IET Information Security **13**(4), 336–342 (2018)
35. Pakniat, N., Shiraly, D., Eslami, Z.: Certificateless Authenticated Encryption with Keyword Search: Enhanced Security Model and a Concrete Construction for Industrial IoT. J. Inf. Secur. Appl. **53**, 102525 (2020)
36. Qin, B., Chen, Y., Huang, Q., Liu, X., Zheng, D.: Public-key Authenticated Encryption with Keyword Search Revisited: Security Model and Constructions. Inf. Sci. **516**, 515–528 (2020)
37. Rhee, H.S., Park, J.H., Susilo, W., Lee, D.H.: Trapdoor Security in a Searchable Public-key Encryption Scheme with a Designated Tester. J. Syst. Softw. **83**(5), 763–771 (2010). https://doi.org/10.1016/j.jss.2009.11.726
38. Shor, P.W.: Algorithms for Quantum Computation: Discrete Logarithms and Factoring. In: FOCS 1994. pp. 124–134. IEEE (1994). https://doi.org/10.1109/SFCS.1994.365700
39. Shor, P.W.: Polynomial-time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. SIAM Review **41**(2), 303–332 (1999). https://doi.org/10.1137/S0097539795293172
40. Wu, L., Zhang, Y., Ma, M., Kumar, N., He, D.: Certificateless Searchable Public Key Authenticated Encryption with Designated Tester for Cloud-assisted Medical Internet of Things. Ann. of Telecommunications **74**(7-8), 423–434 (2019)
41. Zhang, X., Xu, C., Wang, H., Zhang, Y., Wang, S.: FS-PEKS: Lattice-based Forward Secure Public-key Encryption with Keyword Search for Cloud-assisted Industrial Internet of Things. IEEE Trans. Dependable Secur. Comput. (2019). https://doi.org/10.1109/TDSC.2019.2914117, (early access)