Efficient Public Key Searchable Encryption Schemes from Standard Hard Lattice Problems for Cloud Computing

Lijun Qi[1,2] and Jincheng Zhuang[1,2,3]

[1] Key Laboratory of Cryptologic Technology and Information Security,
Ministry of Education, Shandong University, Qingdao 266237, China
[2] School of Cyber Science and Technology,
Shandong University, Qingdao 266237, China
[3] Quancheng Laboratory, Jinan 250103, China

Abstract. Cloud storage and computing offers significant convenience and management efficiency in the information era. Privacy protection is a major challenge in cloud computing. Public key encryption with keyword search (PEKS) is an ingenious tool for ensuring privacy and functionality in certain scenario, such as ensuring privacy for data retrieval appearing in the cloud computing. Despite many attentions received, PEKS schemes still face several challenges in practical applications, such as low computational efficiency, high end-to-end delay, vulnerability to inside keyword guessing attacks(IKGA) and key management defects in the multi-user environment.
In this work, we introduce three Ring-LWE/ISIS based PEKS schemes: (1) Our basic PEKS scheme achieves high level security in the standard model. (2) Our PAEKS scheme utilizes the sender's private key to generate an authentication when encrypting, which can resist IKGA. (3) Our IB-PAEKS scheme not only can resist IKGA, but also significantly reduces the complexity of key management in practical applications. Experimental results indicate that the first scheme provides lower end-to-end delay and higher computational efficiency compared to similar ones, and that our last two schemes can provide more secure properties with little additional overhead.

Keywords: Cloud computing, Searchable encryption, Authentication, Keyword guessing attacks, Ring-LWE/ISIS

1 Introduction

Cloud computing allows users to access unlimited resources via the network while being unrestricted by time and space. It is a new innovation in the information era after the Internet and computers, and has driven the development of Big Data, Internet of Things (IoT), Artificial Intelligence (AI) and other fields. Because of its emergence, the social working mode and business model are changing dramatically. Cloud computing is popular among enterprises and users for its many advantages such as virtualization, large scale, high scalability and flexibility. It has been integrated into all aspects of society, such as the healthcare industry. It not only provides convenient information storage, but also enables easy access and sharing of data. However, there are still many security threats and challenges associated with outsourcing data to cloud servers, such as data privacy and security, access control, virus and hacker attacks, etc. Privacy protection is a major challenge in cloud computing, which is related to the fact that enterprises and users can safely deliver their data to the cloud. To assure the privacy and security of stored data, users usually encrypt and store it in the cloud. However, in this environment, users will encounter the problem of being unable to search keywords in massive data, limiting the flexibility of file sharing in the cloud environment.

Searchable encryption can effectively support users to retrieve encrypted data in the cloud. In the symmetric setting, users can effectively retrieve encrypted data in the cloud through searchable encryption (SE) scheme proposed by Song et al. [28]. In the asymmetric setting, Boneh et al. [9] proposed a new variant called Public-key Encryption with Keyword Search(PEKS). PEKS aims to retrieve target encrypted data by searching specific keywords. It endorses any sender to dispatch encrypted data to the server, which comprises searchable ciphertext related to keywords. The receiver can search the encrypted data of the desired keyword.

PEKS is an ingenious mechanism to ensure privacy and functionality simultaneously for many applications in the cloud. Unfortunately, the widespread adoption of the PEKS scheme in practice has been hampered by some obstacles described below.

1. **Lack of Trapdoor Privacy.** PEKS cannot guarantee trapdoor privacy, i.e., an adversary can guess the information of keywords by initiating inside keyword guessing attack (IKGA) [11, 32]. Specifically, the malicious server can utilize the receiver's key to generate a ciphertext with arbitrary keywords. Thus, when a trapdoor is gained, the Test algorithm can verify the relationship between the trapdoor and the keyword.
2. **High End-to-End Delay.** Despite their strengths, most existing PEKS schemes incorporate high end-to-end delays, which may hamper their deployment in practice.
3. **Low Computational Efficiency.** One of the most challenging problems in practice is probably the efficiency of PEKS, including end-to-end delay and computational efficiency. In the PEKS scheme, the critical factor that affects the efficiency is the Test algorithm, because it needs to be executed for each keyword-file pair.
4. **Key Management Defects.** In a multi-user environment, the user's keys are randomly generated by the system, obviously increasing the complexity of key management.

To prevent trapdoor privacy leakage, Huang and Li [17] proposed public-key authenticated encryption with keyword search (PAEKS). This means encryption requires the sender's private key to generate authentication. This scheme fulfills both Ciphertext Indistinguishability(CI) and Trapdoor Privacy(TP) security. Later, Qin et al. [26] and Pan and Li [24] introduced Multi-Ciphertext Indistinguishability(MCI) and MTP(Multi-Trapdoor Privacy) to ensure ciphertext indistinguishability and trapdoor privacy in a multi keyword environment. With their pioneering work, numerous PAEKS programs have been presented. On the downside, some of them are deficient in meeting both MCI and MTP security. Since the introduction of the PEKS scheme in [9], many PEKS schemes have been constructed using bilinear maps (e.g., [17, 27, 34]) and other classical number theory tools (e.g., [5, 13]). However, with the emergence of quantum computers, most of them are vulnerable to quantum attacks. Therefore, there are some PEKS schemes based on hard problems believed to be quantum-resistant (e.g., [15, 18, 22, 30, 31, 35, 36]). Especially, Behnia et al. [6] presented two lattice-based PEKS schemes, which provide high computational efficiency and high level of security. But, these two PEKS schemes do not meet MTP. Zhang et al. [36] proposed a LWE-based PAEKS scheme to achieve higher security. Later, Liu et al. [19] indicated that Zhang et al.'s scheme is still susceptible to IKGA. They also proposed an LWE-based PAEKS to meet both MCI and MTP security. However, this requires some additional storage and efficiency overhead.

1.1 Our Contributions

This work introduces three lattice-based PEKS schemes which can not only provide high level security, but also achieve high computational efficiency both in theory and in practice. First, we use the transformation in [1] to transform Bert et al. IBE [8], and propose a PEKS scheme based on Ring-LWE/ISIS. Then we propose two extension schemes to achieve trapdoor privacy and predigest key management. More specifically, the main results are as follows.

1. **PEKS.** We propose an IND-CPA-secure and computationally consistent PEKS. Our PEKS is based on Ring-LWE/ISIS, which provides post-quantum security. The security is proved in the standard model, which is preferable to random oracle model.
2. **PAEKS.** To resist IKGA, we propose a PAEKS scheme. The security proof indicates that the scheme meets both MCI and MTP security.
3. **IB-PAEKS.** We propose an extended PAEKS scheme (IB-PAEKS) to predigest key management. We have modified the PAEKS scheme to encrypt keywords by identity. This scheme not only meets both MCI and MTP security, but also significantly reduces the complexity of key management in practical applications.

To compare our scheme with similar ones, we conduct experiments in terms of computational efficiency, storage requirements and end-to-end delay. The results show that the main performance advantages of our PEKS scheme is higher computational efficiency and lower end-to-end delay. Also, the experimental results indicate that our last two schemes can provide more secure properties with little additional overhead.

1.2 Paper Organization

The rest of the paper is organized as follows. Section 2 provides some definitions and theorems utilized in our scheme. Sections 3 describes our PEKS scheme and two extended schemes. Sections 4 describes parameters selection of our schemes and proves the security proof. Section 5 shows and analyzes the experimental results. Section 6 concludes the paper.

2 Preliminaries

Notations $\mathbf{a}, \mathbf{b}, \mathbf{x}, ...$ and $\mathbf{A}, \mathbf{T}, ...$ represent column vectors and matrices, respectively. $\|\mathbf{T}\|$ means the norm of matrix $\mathbf{T}$. $t \leftarrow D$ denotes t is sampled from distribution D, and $t \leftarrow U(S)$ denotes t is sampled from uniform distribution S. $\Lambda(\mathbf{B})$ represents the full-rank lattice generated by $\mathbf{B}$. The polynomial ring is $R_q = R/qR = \mathbb{Z}_q[x]/(x^n + 1)$, where n is a power of two and $q \equiv 1 \mod 2n$.

2.1 Tools and Definitions

Below, we describe some definitions and algorithms utilized in this work.

Definition 1 (Discrete Gaussian Distribution) *The discrete Gaussian distribution with the center* $\boldsymbol{y} \in \mathbb{R}^n$ *and* $\varsigma \in \mathbb{R}$ *over the lattice* Λ *is defined as follows:* $D_{\Lambda,\varsigma,\boldsymbol{y}} = \rho_{\varsigma,\boldsymbol{y}}(\boldsymbol{x})/\rho_{\varsigma,\boldsymbol{y}}(\Lambda)$*, where* $\rho_{\varsigma,\boldsymbol{y}}(\boldsymbol{x}) = \exp(-\pi\|\boldsymbol{x} - \boldsymbol{y}\|^2/\varsigma^2)$ *for all* $\boldsymbol{x} \in \mathbb{R}^n$*, and* $\rho_{\varsigma,\boldsymbol{y}}(\Lambda) = \Sigma_{x\in\Lambda}\rho_{\varsigma,\boldsymbol{y}}(\boldsymbol{x})$.

Definition 2 (Decision Ring-LWE Problem [21, 29]) *A Decision Ring-LWE$_{q,m,D_{R^m},\xi}$ problem instance is given $\boldsymbol{x} = (x_1, \cdots, x_m)^T \in R_q^m$ and $\boldsymbol{y} = \boldsymbol{x}s + \boldsymbol{e}$ where $s \leftarrow U(R_q)$, $e \leftarrow D_{R^m,\xi}$ and $\xi \in \mathbb{R}$, no PPT distinguisher can tell $(\boldsymbol{x}, \boldsymbol{y} = \boldsymbol{x}s + \boldsymbol{e})$ from $(\boldsymbol{x}, \boldsymbol{y})$ in the uniform distribution over $R_q^m \times R_q^m$.*

Definition 3 (Ring-ISIS [20, 25]) *A Ring-ISIS$_{q,m,\varepsilon}$ problem instance is given $\boldsymbol{x} = (x_1, \cdots, x_m)^T \in R_q^m$ and $y \in R_q$, discover a non-zero vector $\boldsymbol{s} = (s_1, \cdots, s_m)^T \in R^m$ satisfy $\boldsymbol{x}^T\boldsymbol{s} \equiv y \bmod q$, $0 < \|\boldsymbol{s}\| \leq \varepsilon$, where $\varepsilon \in \mathbb{R}$.*

Definition 4 (Encoding with Full-Rank Differences (FRD) [2]) *Given integer $n > 0$ and prime q. A function $H : \mathbb{Z}_q^n \to R_q$ is a FRD if:*

1. *The element $H(w) - H(t) \in R_q$ is invertible, for all distinct $w, t \in \mathbb{Z}_q^n$;*
2. *H is computed in polynomial time.*

The following lemmas give the lattice algorithm utilized in our schemes.

Lemma 1 (TrapGen [14, 23]) *Taking the Gaussian parameter σ, modulus q, and $h \in R_q$ as inputs, **TrapGen**(q, σ, h) algorithm chooses a uniformly random polynomial $\boldsymbol{x} \in R_q^{m-k}$ and outputs $\boldsymbol{y} = (\boldsymbol{x}^T, h\boldsymbol{g}^T - \boldsymbol{x}^T\boldsymbol{D})^T \in R_q^m$ with $\boldsymbol{D} \in R^{(m-k)\times k}$, which hides the structured vector $\boldsymbol{g} = (1, 2, 4, \cdots, 2^{k-1})^T \in R_q^k$.*

Lemma 2 (SamplePre [8]) *Taking $\boldsymbol{a} \in R_q^m$, a basis $\boldsymbol{D} \in R^{(m-k)\times k}$, Gaussian parameter μ, η, ς and $h, u \in R_q$ as inputs, **SamplePre** $(\boldsymbol{D}, \boldsymbol{a}, h, \mu, \eta, \varsigma, u)$ algorithm outputs $\boldsymbol{x} \in R_q^m$ which fulfills $\boldsymbol{a}^T\boldsymbol{x} = u$.*

Lemma 3 (DelTrap [23]) *Taking $\boldsymbol{a} \in R_q^m$, a basis $\boldsymbol{D} \in R^{(m-k)\times k}$, a real number $s \in \mathbb{R}$ and $h \in R_q$ as inputs, **DelTrap**$(\boldsymbol{a}, h, \boldsymbol{D}, s)$ algorithm computes $\boldsymbol{a}' = (\boldsymbol{a}^T, h\boldsymbol{g}^T)$ and outputs $\boldsymbol{D}'$ which fulfills $\boldsymbol{a}'[\boldsymbol{D}'\ \boldsymbol{I}] = h\boldsymbol{g}^T$.*

2.2 System Model

Below, we introduce the PEKS system (see Fig. 1) in the cloud computing environment. There are three entities: sender, receiver and cloud server,.

1. The sender extracts keywords from each file, encrypts the keywords with PEKS scheme, and encrypts the files utilizing the receiver's public key. Then the sender upload the searchable ciphertext and the encrypted file to the cloud.
2. To retrieve a file containing specific keywords in the cloud, the receiver compute a trapdoor, and upload it to the cloud server.
3. The cloud server detects each ciphertexts and decides whether it matches the trapdoors. If it matches, the cloud returns the associated file to the receiver.

Definition 5 (Public key Encryption with Keyword Search) *A PEKS scheme comprises four algorithms:*

– ***KeyGen**$(1^\lambda) \to (pk, sk)$: This algorithm takes as input the security parameter λ, and outputs the public key pk, the private key sk.*

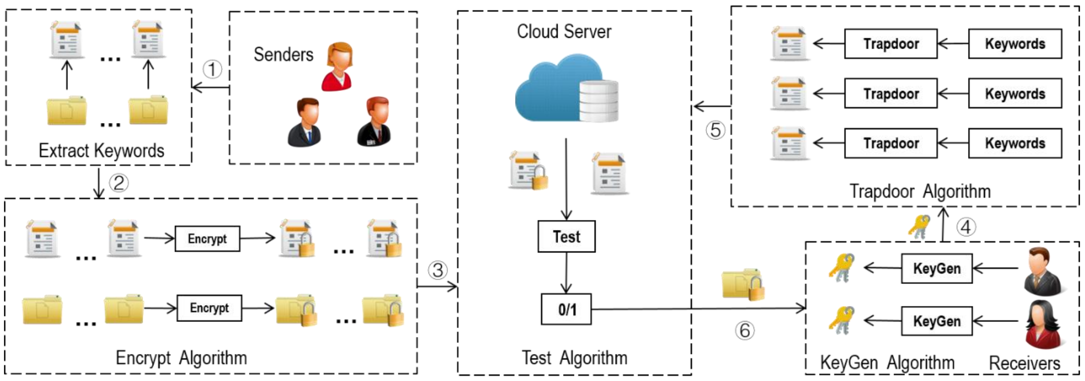


Fig. 1. The framework of PEKS, where it contains 6 steps, which are extracting keywords, encrypting keywords, sending ciphertext, generating trapdoor, sending trapdoor, and sending the results.

– ***Trapdoor***$(w, sk, pk)\rightarrow$ $\boldsymbol{t}_w$: *This algorithm takes as input a keyword* w, pk *and* sk, *and outputs a trapdoor* $\boldsymbol{t}_w$.
– ***Encrypt***$(w, pk)\rightarrow$ $\boldsymbol{s}_w$: *This algorithm takes as input* pk, *and* w, *outputs a searchable ciphertext* $\boldsymbol{s}_w$.
– ***Test***$(\boldsymbol{s}_w, \boldsymbol{t}_w)\rightarrow$ d: *This algorithm takes as input* $\boldsymbol{s}_w \leftarrow$***Encrypt***$(w, pk)$ *and* $\boldsymbol{t}_w \leftarrow$***Trapdoor*** (w', sk, pk), *and outputs* d, *which equal* 1 *if* $w = w'$, *and* 0 *otherwise.*

Recall that the transformation in [1] requires that the underlying IBE should be anonymous. Agrawal et al. [2] defined a strong privacy property that is stronger than anonymity in [1, 7], which is called indistinguishable from random(INDr-sID-CPA). This property requires that the challenge ciphertext cannot be distinguished from the random ciphertext, which means that anonymity and semantic security are satisfied.

Definition 6 (INDr-sID-CPA Security [2]) *INDr-sID-CPA security utilize a security game between a challenger* $\mathcal{P}$ *and an adversary* $\mathcal{Q}$ *(as shown in Table 1).* $\mathcal{Q}$*'s advantage in this game is defined as* $Adv_{IBE}(1^\lambda) = |\Pr[t' = t] - 1/2|$. *If* $Adv_{IBE}(1^\lambda)$ *is negligible for all PPT adversaries* $\mathcal{Q}$, *then IBE scheme satisfies INDr-sID-CPA security (see [2] for more details).*

CI(Ciphertext Indistinguishability) and Trapdoor Privacy(TP) are defined in the single challenge setting. CI means that if an adversary does not gain the trapdoors of w_0 and w_1, it is difficult for the adversary to distinguish the encryption of the keyword w_0 from that of w_1 (see [17] for more details). TP means that it is difficult for the adversary to distinguish the trapdoor of w_0 from that of w_1 (see [17] for more details). MCI and MTP are defined in the multi-challenge setting. If the adversary does not gain the trapdoor of these keywords, it is difficult for the adversary to distinguish the encryption of $\mathbf{w}_0 = (w_{0,1}, ..., w_{0,n})$ from that of $\mathbf{w}_1 = (w_{1,1}, ..., w_{1,n})$. MTP(Multi-Trapdoor Privacy) assures that it is difficult for the adversary to distinguish the trapdoor of $\mathbf{w}_0 = (w_{0,1}, ..., w_{0,n})$ from that of $\mathbf{w}_1 = (w_{1,1}, ..., w_{1,n})$. The following theorem is a property of MCI and MTP security we utilize. It is not difficult to deduce that this theorem is equally valid for the IB-PAEKS scheme.

Table 1. INDr-sID-CPA security game.

Phase	Adversary $\mathcal{Q}$		Challenger $\mathcal{P}$
Init	1) Pick the challenged identity id^*.		
Setup		$\xleftarrow[mpk]{}$	2) Run **Setup**$(1^\lambda) \rightarrow (mpk, msk)$.
Query phase 1	3) Dispatch private key queries to $\mathcal{C}$ on identity $id \neq id^*$.	$\xleftarrow[sk]{}$	4) Run **Extract**$(id, msk, mpk) \rightarrow sk$.
Challenge	5) Pick the challenge plaintext $m^* \in \mathbb{M}$, where $\mathbb{M}$ is the plaintext space.	$\xleftarrow[C^*]{}$	6) Select a random bit $t \in \{0,1\}$ and a random ciphertext $c \in \mathbb{C}$, where $\mathbb{C}$ is the ciphertext space. If $t = 1$, let the challenge ciphertext be $C^* :=$ **Enc**(m^*, id^*, mpk). If $t = 0$, let the challenge ciphertext be $C^* := c$.
Query phase 2	7) Dispatch additional private key queries on identity $id \neq id^*$.		8) Respond as in Query phase 1.
Guess	9) Output $t' \in \{0,1\}$, and wins if $t' = t$.		

Theorem 1 (MCI and MTP of PAEKS [19]) *If the PAEKS satisfies CI(TP, respectively) and its Encrypt(Trapdoor) algorithm is probabilistic, then this scheme meets MCI(MTP) security.*

Theorem 2 (Transformation [1]) *If the IBE scheme is anonymous, then it can be converted to a PEKS scheme, which is computationally consistent and satisfies IND-CPA security.*

2.3 Threat Model

We treat the sender and receiver as trusted entities and the cloud as "honest-but-curious". In our settings, the cloud performs the Test algorithm honestly and delivers the results to the receiver correctly. However, it cannot be excluded that the cloud server uses other methods to capture the plaintext information of the data. We consider the two threat models that have been used in [12, 16, 37].

- **Known ciphertext model.** The cloud server does not know other information except encrypting files, retrieving trapdoors and keyword ciphertext.
- **Known background model.** Compared to the known ciphertext model, the cloud server knows additional background information, such as information about dataset. Therefore, the server can launch keyword guessing attacks by exhausting keywords.

3 Concrete Constructions

In this section, we propose three Ring-LWE/ISIS based PEKS schemes. The first scheme is a basic scheme. To resist IKGA, the second scheme is a PAEKS scheme. To predigest key management, the third scheme is an extended PAEKS scheme (IB-PAEKS).

3.1 Construction of PEKS

In this subsection, we use the transformation in [1] to transform Bert et al. IBE [8], thus attain a PEKS based on Ring-LWE/ISIS. Recall that Theorem 2 requires that the IBE we utilized should be anonymous. We prove the anonymity of Bert et al. IBE [8] by Theorem 3. Our PEKS scheme includes the following four algorithms. The parameters of our PEKS scheme are $q, k, n, m, \mu, \alpha, \beta, \tau, \zeta$, and chosen as described later.

- **KeyGen**$(1^\lambda) \to (pk, sk)$. The receiver executes this algorithm to obtain its public and private key.
  1. Run $\mathbf{TrapGen}(q, \mu, h = 0) \to (\mathbf{a}, \mathbf{T})$, which fulfills $\mathbf{a} = (\mathbf{a}'^T, -\mathbf{a}'^T\mathbf{T})^T \in R_q^m$;
  2. Sample $u \leftarrow U(R_q)$;
  3. Return $pk = (\mathbf{a}, u) \in R_q^{m+1}, sk = \mathbf{T} \in R^{(m-k)\times k}$.
- **Trapdoor**$(w \in \mathbb{W}, sk = \mathbf{T}, pk = (\mathbf{a}, u)) \to \mathbf{t}_w$. The receiver executes this algorithm to obtain a retrieval trapdoor based on a predetermined keyword w, then uploads it to the server.
  1. Compute $h_w = H(w) \in R_q$, where $H(\cdot)$ is the FRD function mentioned above;
  2. Compute $\mathbf{a}_w = \mathbf{a}^T + (0, h_w\mathbf{g}^T)^T = (\mathbf{a}'^T, h_w\mathbf{g}^T - \mathbf{a}'^T\mathbf{T})^T \in R_q^m$, where $\mathbf{g} = (1, 2, 4, \cdots, 2^{k-1})^T \in R_q^k$;
  3. Run $\mathbf{SamplePre}(\mathbf{T}, \mathbf{a}_w, h_w, \zeta, \mu, \alpha, u) \to \mathbf{x} \in R^m$, which satisfies $\mathbf{a}_w^T\mathbf{x} = u$;
  4. Return $\mathbf{t}_w = \mathbf{x} \in R^m$.
- **Encrypt**$(w \in \mathbb{W}, pk = (\mathbf{a}, u)) \to \mathbf{s}_w$. The sender performs this algorithm to obtain a searchable ciphertext utilizing their private key, and uploads it along with the ciphertext file.
  1. Compute $h_w = H(w) \in R_q$, where $H(\cdot)$ is the FRD function mentioned above;
  2. Compute $\mathbf{a}_w = \mathbf{a}^T + (0, h_w\mathbf{g}^T)^T = (\mathbf{a}'^T, h_w\mathbf{g}^T - \mathbf{a}'^T\mathbf{T})^T \in R_q^m$, where $\mathbf{g} = (1, 2, 4, \cdots, 2^{k-1})^T \in R_q^k$;
  3. Select $s \leftarrow U(R_q), e_0 \leftarrow D_{R^{m-k},\tau}, e_1 \leftarrow D_{R^k,\beta}, e' \leftarrow D_{R,\tau}, c_1 \in R_2$;
  4. Compute $\mathbf{b} = \mathbf{a}_w s + (\mathbf{e}_0^T, \mathbf{e}_1^T)^T \in R_q^m$, and $c_2 = u \cdot s + e' + \lfloor q/2 \rfloor c_1 \in R_q$;
  5. Return $\mathbf{s}_w = (\mathbf{b}, c_2, c_1) \in R_q^{m+2}$.
- **Test**$(\mathbf{s}_w = (\mathbf{b}, c_2, c_1), \mathbf{t}_w = \mathbf{x}) \to d$. The server runs this algorithm to retrieve the searchable ciphertext matching the trapdoor, and sends all matching ciphertext files to the receiver.
  1. Compute $y = c_2 - \mathbf{b}^T\mathbf{t}_w = e' - (\mathbf{e}_0^T, \mathbf{e}_1^T)\mathbf{x} + \lfloor q/2 \rfloor c_1 \in R_q$;
  2. For each y_i, if y_i is closer to $\lfloor q/2 \rfloor$ than to 0, $y_i = 1$, otherwise $y_i = 0$;
  3. If $y = c_1$, $d = 1$, otherwise $d = 0$;
  4. Return d.

Correctness. Set the trapdoor be $\mathbf{t}_w = \mathbf{x} = (\mathbf{x}_0^T, \mathbf{x}_1^T)^T$ and the ciphertext be $\mathbf{s}_w = (\mathbf{b}, c_2, c_1)$. For the Test algorithm, we have $y = c_2 - \mathbf{b}^T\mathbf{t}_w = e' - \mathbf{e}_0^T\mathbf{x}_0 - \mathbf{e}_1^T\mathbf{x}_1 + \lfloor q/2 \rfloor c_1$. To decrypt correctly, the error term $e' - \mathbf{e}_0^T\mathbf{x}_0 - \mathbf{e}_1^T\mathbf{x}_1$ should satisfy $\|e' - \mathbf{e}_0^T\mathbf{x}_0 - \mathbf{e}_1^T\mathbf{x}_1\| < \lfloor q/4 \rfloor$. See Section 4.1 for parameter selection.

3.2 Extension to Resist IKGA

To resist IKGA, we propose a PAEKS scheme. This requires the sender to utilize its private key to generate an authentication when encrypting. An overview of PAEKS is shown in Fig. 2. Our PAEKS scheme includes the following five algorithms.

- **Setup**$(1^\lambda) \to sp$. The system runs this algorithm by inputting a security parameter λ, and initializes the global parameters sp.

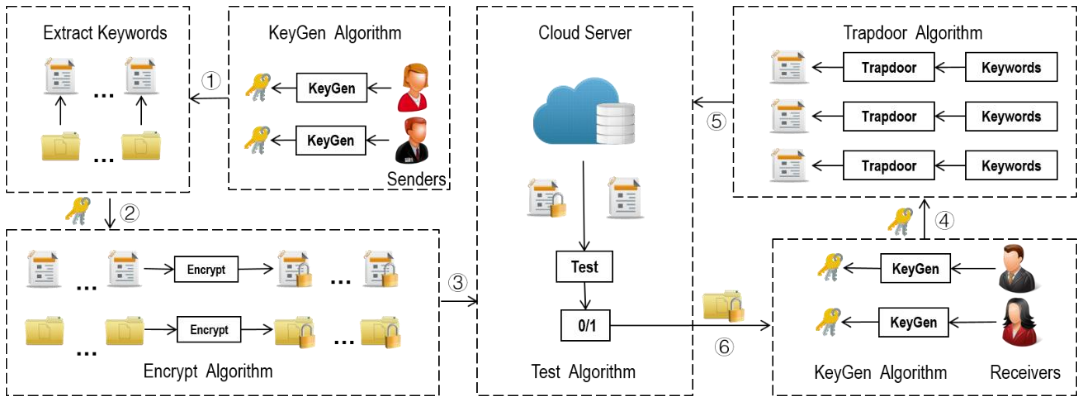


Fig. 2. The framework of PAEKS, where it contains 6 steps, which are extracting keywords, encrypting keywords, sending ciphertext, generating trapdoor, sending trapdoor, and sending the results.

1. Choose some security parameters $q, k, n, m \in \mathbb{Z}, \sigma, \alpha, \gamma, \tau, \zeta \in \mathbb{R}$;
2. Sample $u \leftarrow U(R_q)$;
3. Return $sp = (q, k, n, m, \mu, \alpha, \beta, \tau, \zeta, u)$.

– **KeyGen**$(1^\lambda) \rightarrow (pk, sk)$. The sender and receiver run this algorithm separately to obtain their own key pairs.
  1. Run **TrapGen**$(q, \mu, h = 0) \rightarrow (\mathbf{a}, \mathbf{T})$, which fulfills $\mathbf{a} = (\mathbf{a}'^T, -\mathbf{a}'^T\mathbf{T})^T \in R_q^m$;
  2. Return $pk = \mathbf{a} \in R_q^m, sk = \mathbf{T} \in R^{(m-k)\times k}$.

– **Trapdoor**$(w \in \mathbb{W}, sk_r = \mathbf{T}, pk_r = \mathbf{a}) \rightarrow \mathbf{t}_w$. The receiver executes this algorithm to obtain a retrieval trapdoor based on a predetermined keyword w, then uploads it to the server.
  1. Compute $h_w = H(w) \in R_q$, where $H(\cdot)$ is the FRD function mentioned above;
  2. Compute $\mathbf{a}_w = \mathbf{a}^T + (0, h_w\mathbf{g}^T)^T = (\mathbf{a}'^T, h_w\mathbf{g}^T - \mathbf{a}'^T\mathbf{T})^T \in R_q^m$, where $\mathbf{g} = (1, 2, 4, \cdots, 2^{k-1})^T \in R_q^k$;
  3. Run **SamplePre**$(\mathbf{T}, \mathbf{a}_w, h_w, \zeta, \mu, \alpha, u) \rightarrow \mathbf{x} \in R^m$, which satisfies $\mathbf{a}_w^T\mathbf{x} = u$;
  4. Return $\mathbf{t}_w = \mathbf{x} \in R^m$.

– **Encrypt**$(w \in \mathbb{W}, pk_r = \mathbf{a}, sk_s = \mathbf{T}_s) \rightarrow \mathbf{s}_w$. The sender performs this algorithm to obtain a searchable ciphertext utilizing their private key, and uploads it along with the ciphertext file.
  1. Compute $h_w = H(w) \in R_q$ and $\mathbf{a}_w = \mathbf{a}^T + (0, h_w\mathbf{g}^T)^T = (\mathbf{a}'^T, h_w\mathbf{g}^T - \mathbf{a}'^T\mathbf{T})^T \in R_q^m$, where $\mathbf{g} = (1, 2, 4, \cdots, 2^{k-1})^T \in R_q^k$;
  2. Select $s \leftarrow U(R_q), e_0 \leftarrow D_{R^{m-k},\tau}, e_1 \leftarrow D_{R^k,\beta}, e' \leftarrow D_{R,\tau}, c_1 \in R_2$;
  3. Compute $\mathbf{b} = \mathbf{a}_w s + (\mathbf{e}_0^T, \mathbf{e}_1^T)^T \in R_q^m$, and $c_2 = u \cdot s + e' + \lfloor q/2 \rfloor c_1 \in R_q$;
  4. Compute $h_{c_1} = H(c_1) \in R_q$, $\mathbf{a}_{s'} = \mathbf{a}_s^T + (0, h_{c_1}\mathbf{g}^T)^T$;
  5. Run **SamplePre**$(\mathbf{T}_s, \mathbf{a}_{s'}, h_{c_1}, \zeta, \sigma, \alpha, 0) \rightarrow \mathbf{c} \in R_q^m$, which satisfies $\mathbf{a}_{s'}^T\mathbf{c} = 0$.
  6. Return $\mathbf{s}_w = (\mathbf{b}, c_2, \mathbf{c}) \in R_q^{2m+1}$.

– **Test** $(\mathbf{s}_w = (\mathbf{b}, c_2, \mathbf{c}), \mathbf{t}_w = \mathbf{x}) \rightarrow d$. The server runs this algorithm to retrieve the searchable ciphertext matching the trapdoor, and sends all matching ciphertext files to the receiver.
  1. Compute $y = c_2 - \mathbf{b}^T\mathbf{t}_w = e' - (\mathbf{e}_0^T, \mathbf{e}_1^T)\mathbf{x} + \lfloor q/2 \rfloor c_1 \in R_q$;
  2. For each y_i, if y_i is closer to $\lfloor q/2 \rfloor$ than to 0, $y_i = 1$, otherwise $y_i = 0$;
  3. Compute $h = H(y)$ and $\mathbf{a}_{s'} = \mathbf{a}_s^T + (0, h\mathbf{g}^T)^T$;

4. If $\mathbf{a}_s^T\mathbf{c} = 0$, $d = 1$, otherwise $d = 0$;
5. Return d.

Correctness. This is as same as the basic PEKS scheme.

3.3 Extension to Predigest Key

In the PAEKS scheme, we observe that the user's key is randomly generated, which contains large-sized polynomials and matrices. This definitely increases the complexity of key management. To facilitate key management, we propose an extended PAEKS scheme (IB-PAEKS), which permits users to encrypt data utilizing their own identity. This scheme adds a Key Generator Center(KGC), which generates the user's private key according to their identity. In our settings, We presume that KGC is a trusted entity that transmits a valid private key to the user through a secure channel.

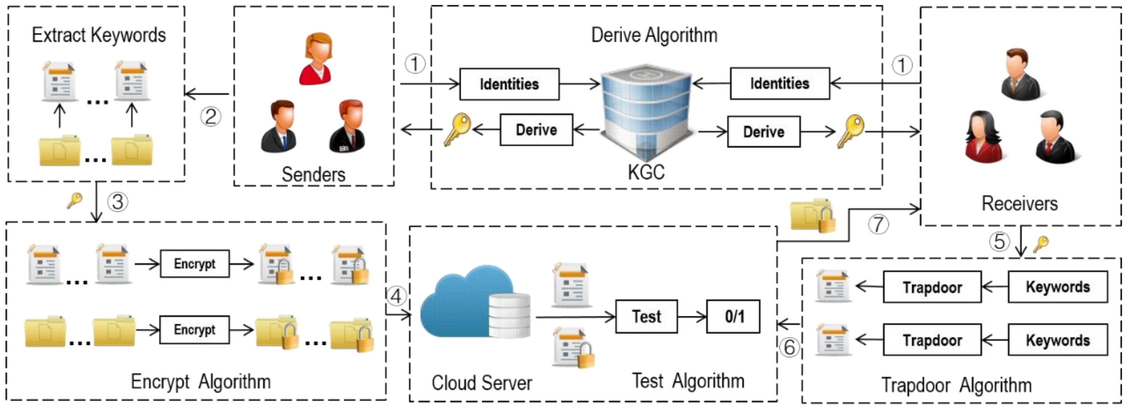


Fig. 3. The framework of IB-PAEKS, where it contains 7 steps, which are generating keys, extracting keywords, encrypting keywords, sending ciphertext, generating trapdoor, sending trapdoor, and sending the results.

An overview of IB-PAEKS is shown in Fig. 3. Our IB-PAEKS scheme includes the following five algorithms.

- **Setup**$(1^\lambda) \rightarrow (mpk, msk)$. The KGC runs this algorithm by entering security parameter λ to attain its public key and private key.
  1. Sample $u \leftarrow U(R_q)$;
  2. Run **TrapGen**$(q, \sigma, h = 0) \rightarrow (\mathbf{a}, \mathbf{T})$, which fulfills $\mathbf{a} = (\mathbf{a}'^T, -\mathbf{a}'^T\mathbf{T})^T$;
  3. Return $mpk = \mathbf{a} \in R_q^m, msk = \mathbf{T} \in R^{(m-k)\times k}$.
- **Derive**$(mpk, msk, id) \rightarrow sk_i$. The sender and receiver send their identity id to the KGC, and KGC executes this algorithm to generate their private key.
  1. Compute $h_{id} = H(id) \in R_q, \mathbf{a}_i = (\mathbf{a}, h_{id}\mathbf{g})^T \in R^{m+k}$;
  2. Run **DelTrap**$(\mathbf{a}, h_{id}, \mathbf{T}, s) \rightarrow \mathbf{T}_i \in R_q^m$;

3. Return $sk_i = \mathbf{T}_i \in R_q^m$.

- **Trapdoor**$(w \in \mathbb{W}, sk_r = \mathbf{T}_r, id_r) \rightarrow \mathbf{t}_w$. The receiver executes this algorithm to generate a retrieval trapdoor based on a predetermined keyword w, and uploads it to the server.
  1. Compute $h_{id} = H(id_r), \mathbf{a}_r = (\mathbf{a}, h_{id}\mathbf{g})^T = (\mathbf{a}'^T, -\mathbf{a}'^T\mathbf{T}, h_{id}\mathbf{g})^T$;
  2. Compute $h_w = H(w)$ and $\mathbf{a}_{r_w} = \mathbf{a}_r^T + (0, h_w\mathbf{g})^T = (\mathbf{a}, h_w\mathbf{g} + h_{id}\mathbf{g})^T$;
  3. Run **SamplePre**$(\mathbf{T}_r, \mathbf{a}_{r_w}, h_w + h_{id}, \zeta, \sigma, \alpha, u) \rightarrow \mathbf{x}$, which satisfies $\mathbf{a}_{r_w}^T\mathbf{x} = u$;
  4. Return $\mathbf{t}_w = \mathbf{x} \in R_q^{m+k}$.
- **Encrypt**$(w \in \mathbb{W}, id_r, id_s, sk_s = \mathbf{T}_s) \rightarrow \mathbf{s}_w$. The sender executes this algorithm to generate a searchable ciphertext utilizing their private key, and uploads it along with the ciphertext file.
  1. Compute $h_{id_s} = H(id_s), \mathbf{a}_s = (\mathbf{a}, h_{id_s}\mathbf{g})^T$;
  2. Compute $h_{id_r} = H(id_r), \mathbf{a}_r = (\mathbf{a}, h_{id_r}\mathbf{g})^T$;
  3. Compute $h_w = H(w)$ and $\mathbf{a}_{r_w} = \mathbf{a}_r^T + (0, h_w\mathbf{g})^T = (\mathbf{a}, h_w\mathbf{g} + h_{id_r}\mathbf{g})^T$;
  4. Select $s \leftarrow U(R_q), e_0 \leftarrow D_{R^{m-k},\tau}, e_1 \leftarrow D_{R^k,\gamma}, e_2 \leftarrow D_{R^k,\gamma}, e' \leftarrow D_{R,\tau}, c_1 \in R_2$;
  5. Compute $\mathbf{b} = \mathbf{a}_{r_w}s + (\mathbf{e}_0^T, \mathbf{e}_1^T, \mathbf{e}_2^T)^T \in R_q^{m+k}$, and $c_2 = u \cdot s + e' + \lfloor q/2 \rfloor c_1 \in R_q$;
  6. Compute $h_{c_1} = H(c_1) \in R_q$ and $\mathbf{a}_{s'} = \mathbf{a}_s^T + (0, h_{c_1}\mathbf{g}^T)^T$;
  7. Run **SamplePre**$(\mathbf{T}_s, \mathbf{a}_{s'}, h_{id_s} + h_{c_1}, \zeta, \sigma, \alpha, 0) \rightarrow \mathbf{c} \in R_q^{m+k}$, which satisfies $\mathbf{a}_{s'}^T\mathbf{c} = 0$;
  8. Return $\mathbf{s}_w = (\mathbf{b}, c_2, \mathbf{c}) \in R_q^{2m+2k+1}$.
- **Test**$(\mathbf{s}_w = (\mathbf{b}, c_2, \mathbf{c}), \mathbf{t}_w = \mathbf{x}, id_s) \rightarrow d$. The server runs this algorithm to retrieve the searchable ciphertext matching the trapdoor, and sends all matching ciphertext files to the receiver.
  1. Compute $y = c_2 - \mathbf{b}^T\mathbf{t}_w = e' - (\mathbf{e}_0^T, \mathbf{e}_1^T, \mathbf{e}_2^T)\mathbf{x} + \lfloor q/2 \rfloor c_1 \in R_q$;
  2. For each y_i, if y_i is closer to $\lfloor q/2 \rfloor$ than to 0, $y_i = 1$, otherwise $y_i = 0$;
  3. Compute $h = H(y)$ and $h_{id_s} = H(id_s), \mathbf{a}_s = (\mathbf{a}, h_{id_s}\mathbf{g})^T$;
  4. Compute $\mathbf{a}_{s'} = \mathbf{a}_s^T + (0, h\mathbf{g}^T)^T$;
  5. If $\mathbf{a}_s^T\mathbf{c} = 0$, $d = 1$, otherwise $d = 0$;
  6. Return d.

Correctness. Set the trapdoor be $\mathbf{t}_w = \mathbf{x} = (\mathbf{x}_0^T, \mathbf{x}_1^T, \mathbf{x}_2^T)^T$ and the ciphertext be $\mathbf{s}_w = (\mathbf{b}, c_2, \mathbf{c})$. For the Test algorithm, we have $y = c_2 - \mathbf{b}^T\mathbf{t}_w = e' - \mathbf{e}_0^T\mathbf{x}_0 - \mathbf{e}_1^T\mathbf{x}_1 - \mathbf{e}_2^T\mathbf{x}_2 + \lfloor q/2 \rfloor c_1$. To decrypt correctly, the error term $e' - \mathbf{e}_0^T\mathbf{x}_0 - \mathbf{e}_1^T\mathbf{x}_1 - \mathbf{e}_2^T\mathbf{x}_2$ should satisfy $\|e' - \mathbf{e}_0^T\mathbf{x}_0 - \mathbf{e}_1^T\mathbf{x}_1 - \mathbf{e}_2^T\mathbf{x}_2\| < \lfloor q/4 \rfloor$. See Section 4.1 for parameter selection.

4 Parameters Selection and Security Proof

4.1 Parameters Selection

To decrypt the IB-PAEKS scheme correctly, we modified the parameters. Table 2 shows the parameters set of IB-PAEKS scheme when $\lambda = 80$, which is also applicable to PEKS and PAEKS scheme. We utilize the BKZ lattice reduction algorithm cost model in [4, 10] to evaluate the core SVP hardness, where the time complexity of BKZ is $T = 2^{tb}$, where $t = 0.292$ is classical security, $t = 0.265$ is quantum security, $t = 0.2075$ is paranoid security, and b is the block size of BKZ. We utilize the LWE estimator[1] in [3] combined with our specified BKZ cost model to obtain our results.

[1] https://bitbucket.org/malb/lwe-estimator/src/master/, commit a2a6e84.

Table 2. Parameters set of our schemes for $\lambda = 80$.

Parameters	n	k	$\log q$	m	σ	α	τ	γ	ζ
Classic	512	62	62	64	6.5	14.5	6.5	22944.2	8373.5
Quantum	1024	62	62	64	3	6.7	3	6912.0	2482.6
Paranoid	1024	62	62	64	4.2	9.4	4.2	13547.52	4866.0

4.2 Security Proof of PEKS

According to the transformation of Abdalla et al. [1] , the anonymity of the IBE scheme necessitates to be proved. Below, we prove that Bert et al. IBE [8] satisfies INDr-sID-CPA security.

Theorem 3 *The IBE of Bert et al. [8] satisfies INDr-sID-CPA security supposing the Ring-LWE problem is hard.*

Proof. We describe three games to prove the INDr-sID-CPA security. Subsequently, by proving the indistinguishability between these games, we prove this theorem.

Game 0: The Game 0 is a game in Definition 6. In the Setup phase, $\mathcal{P}$ calls **TrapGen**$(q, \mu, h = 0) \to (\mathbf{a}, \mathbf{T})$.

Game 1: Except for the Setup phase, Game 1 is equal to Game 0. In the Setup phase, by adding challenge identity id^*, the generation of $\mathbf{a}$ is changed. $\mathcal{P}$ calls **TrapGen**$(q, \mu, \mathbf{a}', -h_{id^*}) \to (\mathbf{a}, \mathbf{T})$, where $\mathbf{a}' \leftarrow U(R_q^{m-k})$. $\mathbf{a}$ and $\mathbf{T}$ satisfy $\mathbf{a} = (\mathbf{a}'^T, -h_{id^*}\mathbf{g}^T - \mathbf{a}'^T\mathbf{T})^T$. In Query phase 1, $\mathcal{Q}$ sends private key queries to $\mathcal{P}$ on $id \neq id^*$. $\mathcal{P}$ responds by calling **Extract** (id, msk, mpk) algorithm. More specifically, $\mathcal{P}$ computes $\mathbf{a}_{id} = \mathbf{a}^T + (0, h_{id}\mathbf{g}^T) = (\mathbf{a}'^T, (h_{id} - h_{id^*})\mathbf{g}^T - \mathbf{a}'^T\mathbf{T})^T$. Then $\mathcal{P}$ calls **SamplePre**$(\mathbf{T}, \mathbf{a}_{id}, h_{id} - h_{id^*}, \zeta, \mu, \alpha, u) \to \mathbf{x}$, which fulfills $\mathbf{a}_{id}^T\mathbf{x} = u$. Recall that when $id = id^*$, $\mathbf{a}_{id} = (\mathbf{a}'^T, -\mathbf{a}'^T\mathbf{T})^T$, $\mathcal{P}$ stops responding $\mathcal{Q}$'s queries.

Indistinguishability between Game 0 and Game 1. The public parameter $\mathbf{a}$ consists of two parts: the first part $\mathbf{a}'^T$ is selected from R_q^{m-k}; and the last part, based on our chosen trapdoor instantiation, $\mathbf{a}'^T\mathbf{T} = (\Sigma_{i=1}^{m-k}a_i t_{i,1}, \Sigma_{i=1}^{m-k}a_i t_{i,1}, \cdots, \Sigma_{i=1}^{m-k}a_i t_{i,k})$ and uniform distribution are at least computationally indistinguishable. In the view of $\mathcal{Q}$, it cannot distinguish whether $\mathbf{a}$ comes from Game 0 or Game 1.

Game 2: Except for the ciphertext's selection, Game 2 is equal to Game 1. In Challenge phase, C^* is randomly selected from $R_q^m \times R_q$. Next, we prove that Game 1 is computationally indistinguishable from Game 2 by giving a reduction from Ring-LWE.

Indistinguishability between Game 1 and Game 2. Assuming that $\mathcal{Q}$ can differentiate Game 1 and Game 2, where the advantage of $\mathcal{Q}$ cannot be neglected, we can utilize an algorithm $\mathcal{P}$ to solve Ring-LWE. $\mathcal{P}$ does the following processes:

– **Init**: $\mathcal{P}$ obtains $m-k+1$ samples of decisional Ring-LWE instances (a_i, b_i), where $0 \le i \le m-k$. $\mathcal{P}$ obtains the challenge identity id^* sent by $\mathcal{Q}$.
– **Setup**: Set $\mathbf{a}' = (a_1, \cdots, a_{m-k})^T \in R_q^{m-k}$, $\mathbf{b}' = (b_1, \cdots, b_{m-k})^T \in R_q^{m-k}$ and $u = a_0$. Due to the Ring-LWE assumption, $\mathbf{a}'$ follows a uniform distribution. Then $\mathcal{P}$ runs **Setup**(1^λ) algorithm as in Game 1, where $(\mathbf{a}, \mathbf{T}) \leftarrow$ **TrapGen**$(q, \mu, \mathbf{a}', -h_{id^*})$. $\mathcal{P}$ sends $mpk = (\mathbf{a}, u)$ to $\mathcal{Q}$.
– **Query phase 1**: $\mathcal{Q}$ sends private key queries, $\mathcal{P}$ responds $\mathcal{Q}$'s queries.
– **Challenge**: $\mathcal{Q}$ selects plaintext $m \in \{0, 1\}$ as the challenge plaintext. $\mathcal{P}$ randomly chooses $t \in \{0, 1\}$ and sets $C^* = (\mathbf{b}^*, c^*)$, where $\mathbf{b}^* = (\mathbf{b}'^T, -\mathbf{b}'^T\mathbf{T} + \mathbf{e}'^T)^T$, $\mathbf{e}' \leftarrow D_{R^k,\rho}, \rho \in \mathbb{R}$, and $c^* = b_0 + \lfloor q/2 \rfloor m_t$. Then $\mathcal{P}$ returns C^* to $\mathcal{Q}$.

- **Query phase 2**: $\mathcal{Q}$ performs other queries, $\mathcal{P}$ answers queries which is identical to Query phase 1.
- **Guess**: $\mathcal{Q}$ guesses whether he interacts with the challenger of Game 1 or Game 2. $\mathcal{P}$ takes $\mathcal{Q}$'s guess as a reply to solve Ring-LWE problem.

If $m-k+1$ samples of decisional Ring-LWE instances (a_i, b_i) stem from Ring-LWE distribution, we get $b_0 = a_0 s + e_0, \mathbf{b}' = \mathbf{a}' s + \mathbf{r}$, where $s \in R_q$, $\mathbf{r} \leftarrow D_{R^{m-k},\tau}$, and $e_0 \leftarrow D_{R,\tau}, \tau \in \mathbb{R}$. Then, we have $\mathbf{b}^* = \mathbf{a}_{id^*}^T s + (\mathbf{r}^T, -\mathbf{r}^T\mathbf{T} + \mathbf{e}'^T)^T$, $c^* = a_0 s + e_0 + \lfloor q/2 \rfloor m_t = us + e_0 + \lfloor q/2 \rfloor m_t$.

First, we cannot distinguish the error term $-\mathbf{r}^T\mathbf{T} + \mathbf{e}'^T$ from sample in the distribution $D_{R^k,\gamma}$ for fixed $\mathbf{e}$, where $\gamma^2 = (\mu\|\mathbf{r}\|)^2 + \rho^2, \rho \in \mathbb{R}$. Hence, $\mathbf{b}^*$ is actually the first part $\mathbf{b}$ of C^* in Game 1. In addition, c^* is the second part c of C^* in Game 1. Therefore, the distribution of C^* is exactly equal to that in Game 1.

If $m-k+1$ samples of decisional Ring-LWE instances (a_i, b_i) stem from uniform distribution $R_q^m \times R_q$, then the distribution of C^* is exactly equal to that in Game 2.

Therefore, if the adversary $\mathcal{Q}$ can differentiate Game 1 and Game 2 whose advantage cannot be neglected, $\mathcal{P}$ can utilize $\mathcal{Q}$ to solve Ring-LWE with a non-negligible advantage.

4.3 Security Proof of PAEKS

Theorem 4 *Supposing the Ring-LWE problem is hard, our PAEKS scheme satisfies multi-ciphertext indistinguishability.*

Proof. The ciphertext indistinguishability of our PAEKS scheme relies on the basic PEKS scheme. In addition, Theorem 1 of [8] proved that the underlying IBE satisfied ciphertext indistinguishability under the hardness of Ring-LWE problem, and the encryption algorithm of our PAEKS scheme is probabilistic. Therefore, based on Theorem 1, our PAEKS scheme meets multi-ciphertext indistinguishability.

Theorem 5 *Supposing the Ring-ISIS problem is hard, our scheme satisfies multi-trapdoor privacy.*

Proof. First, we need to prove the trapdoor privacy of our scheme. Assuming that the PPT adversary $\mathcal{Q}$ can break trapdoor privacy of PAEKS, there is a challenger $\mathcal{P}$ to solve the Ring-ISIS problem.

- **Init**: $\mathcal{P}$ obtains $m-k$ samples of decisional Ring-ISIS instances $(\mathbf{a}', u)$ where $\mathbf{a}' = (a_1, \cdots, a_{m-k})^T \in R_q^{m-k}$. $\mathcal{Q}$ sets w^* as challenge keyword.
- **Setup**: $\mathcal{P}$ inputs the security parameter λ, calls $\mathbf{KeyGen}_s(mpk) \to (pk_s, sk_s)$ and $\mathbf{KeyGen}_r(mpk) \to (pk_r, sk_r)$ separately. In detail, $\mathcal{P}$ executes $\mathbf{TrapGen}(q, \sigma, \mathbf{a}', -h_{w^*}) \to (\mathbf{a}, \mathbf{T})$, and returns $pk = \mathbf{a}, sk = \mathbf{T}$. Finally, $\mathcal{P}$ sends mpk, pk_s, pk_r to $\mathcal{Q}$.
- **Query phase**: $\mathcal{Q}$ sends trapdoor or ciphertext queries to $\mathcal{P}$ for the keyword $w \neq w^*$.
  1. Trapdoor query: $\mathcal{P}$ calls $\mathbf{Trapdoor}(w, sk_r, pk_r) \to t_w$ to reply $\mathcal{Q}$'s queries. Specifically, $\mathcal{P}$ computes $h_w = H(w)$ and $\mathbf{a}_w = \mathbf{a}^T + (0, h_w\mathbf{g}^T)^T = (\mathbf{a}'^T, (h_w - h_{w^*})\mathbf{g}^T - \mathbf{a}'^T\mathbf{T})^T$, then runs $\mathbf{SamplePre}(\mathbf{T}, \mathbf{a}_w, h_w - h_{w^*}, \zeta, \sigma, \alpha, u) \to \mathbf{x}$, which satisfies $\mathbf{a}_w^T\mathbf{x} = u$.
  2. Ciphertext query: $\mathcal{P}$ randomly selects $c_1 \in \{0, 1\}$, computes $h_w = H(w)$ and $\mathbf{a}_w = \mathbf{a}^T + (0, h_w\mathbf{g}^T)^T$. $\mathcal{P}$ sets $C^* = (\mathbf{b}^*, c_2^*, \mathbf{e}^*)$, where $\mathbf{b}^* = \mathbf{a}_w s + (\mathbf{e}_0^T, \mathbf{e}_1^T)^T$, $c_2^* = u \cdot s + e' + \lfloor q/2 \rfloor c_1 \in R_q$ and $\mathbf{e}^* \in D_{R_q^m,\zeta}$ by running **SamplePre** algorithm.

– **Forge phase**: $\mathcal{Q}$ forges $\mathbf{x}^*$ for challenge keyword w^*, which satisfies $\mathbf{a}_{w^*}^T\mathbf{x}^* = u$. Then we have $\mathbf{a}'^T(\mathbf{I}_{m-k}, -\mathbf{T})\mathbf{x}^* = u$. Set $\mathbf{y} = (\mathbf{I}_{m-k}, -\mathbf{T})\mathbf{x}^*$, then we note that $\|\mathbf{T}\| \leq t\sigma\sqrt{(n-k)n}$, therefore $\|\mathbf{y}\| \leq (1 + t\sigma\sqrt{(n-k)n})t\zeta\sqrt{mn} = \beta$. Hence, $\mathbf{y}$ is a solution for Ring-ISIS instances $(\mathbf{a}, u)$.

Therefore, if $\mathcal{Q}$ can break trapdoor privacy of PAEKS, there is a challenger $\mathcal{P}$ to solve the Ring-ISIS. Besides, the trapdoor algorithm of our PAEKS scheme is probabilistic. Thus, our scheme satisfies multi-trapdoor privacy.

4.4 Security Proof of IB-PAEKS

Theorem 6 *Supposing the Ring-LWE problem is hard, our IB-PAEKS scheme satisfies multi-ciphertext indistinguishability.*

Proof. First, we necessitate to prove the CI security of our IB-PAEKS scheme. The proof is similar to the proof in Theorem 1 in [8], except that Game 3 is added between Game 1 and Game 2. Except for the Setup phase, Game 3 is equal to Game 1. In the Setup phase, by adding challenge keyword w^*, the generation of $\mathbf{a}$ is changed. the challenger $\mathcal{P}$ calls **TrapGen**$(q, \sigma, \mathbf{a}', -h_{w^*} - h_{id^*}) \rightarrow (\mathbf{a}, \mathbf{T})$, where $\mathbf{a}' \leftarrow U(R_q^{m-k})$. $\mathbf{a}$ and $\mathbf{T}$ satisfy $\mathbf{a} = (\mathbf{a}'^T, (-h_{w^*} - h_{id^*})\mathbf{g}^T - \mathbf{a}'^T\mathbf{T})^T$. If the adversary $\mathcal{Q}$ can distinguish between Game 1 and Game 3, then there exists an algorithm $\mathcal{P}$ can distinguish between $\mathbf{a}$ and a random vector from uniform distribution R_q^m. In $\mathcal{Q}$'s perspective, $\mathbf{a}$ and uniform distribution are at least computationally indistinguishable. Therefore, $\mathcal{Q}$ cannot distinguish whether $\mathbf{a}$ comes from Game 1 or Game 3. Next, the indistinguishability between Game 2 and game 3 is similar to the proof of Theorem 1 in [8], which will not be repeated here.

Besides, the Encrypt algorithm of our IB-PAEKS scheme is a probability algorithm. Therefore, our IB-PAEKS scheme satisfies multi-ciphertext indistinguishability.

Theorem 7 *Supposing the Ring-ISIS problem is hard, our IB-PAEKS scheme satisfies multi-trapdoor privacy.*

Proof. Assuming that the PPT adversary $\mathcal{Q}$ can break trapdoor privacy of IB-PAEKS, there exists a challenger $\mathcal{P}$ to solve the Ring-ISIS problem.

– **Init**: $\mathcal{P}$ obtains $m-k$ samples of decisional Ring-ISIS instances $(\mathbf{a}', u)$ where $\mathbf{a}' = (a_1, \cdots, a_{m-k})^T \in R_q^{m-k}$. $\mathcal{Q}$ sets w^* as challenge keyword and sets id_s, id_r as the identity of the sender and receiver respectively.
– **Setup**: $\mathcal{P}$ inputs the security parameter λ, calls **Setup**$(1^\lambda) \rightarrow (mpk, msk)$. In detail, $\mathcal{P}$ runs **TrapGen**$(q, \sigma, \mathbf{a}', 0) \rightarrow (\mathbf{a}, \mathbf{T})$, where $\mathbf{a} = (\mathbf{a}'^T, -\mathbf{a}'^T\mathbf{T})^T$. For the challenge keyword w^* and the challenge identity id_r, $\mathcal{P}$ sets $\mathbf{a} = (\mathbf{a}, -h_{w^*}\mathbf{g}^T - h_{id_r}\mathbf{g}^T)$. Then $\mathcal{P}$ runs **Derive**$(mpk, msk, id_s) \rightarrow sk_s$, and **Derive** $(mpk, msk, id_r) \rightarrow sk_r$. Finally, $\mathcal{P}$ sends mpk, id_s, id_r to $\mathcal{Q}$.
– **Query phase 1:** $\mathcal{Q}$ sends trapdoor or ciphertext queries to $\mathcal{P}$ for the keyword $w \neq w^*$.
   1. Trapdoor query: $\mathcal{P}$ calls **Trapdoor**$(w, sk_r, pk_r) \rightarrow t_w$ to reply $\mathcal{Q}$'s queries. Specifically, $\mathcal{P}$ computes $h_w = H(w), h_{id} = H(id_r)$ and $\mathbf{a}_w = (\mathbf{a}^T, (h_{id} + h_w)\mathbf{g}^T)^T$, then runs **SamplePre**$(\mathbf{T}, \mathbf{a}_w, h_w - h_{w^*}, \zeta, \sigma, \alpha, u) \rightarrow \mathbf{x}$, which satisfies $\mathbf{a}_w^T\mathbf{x} = u$.
   2. Ciphertext query: $\mathcal{P}$ calls **Encrypt**$(w, pk_r, pk_s, sk_s) \rightarrow s_w$ to reply $\mathcal{Q}$'s queries.
– **Forge phase**: $\mathcal{Q}$ forges $\mathbf{x}^*$ for challenge keyword w^*, which satisfies $\mathbf{a}_{w^*}^T\mathbf{x}^* = u$. Then we get $\mathbf{a}'^T(\mathbf{I}_{m-k}, -\mathbf{T})\mathbf{x}'^* = u$, where $\mathbf{x}'^* \in R_q^m$ is the first m dimension of $\mathbf{x}^*$. Set $\mathbf{y} = (\mathbf{I}_{m-k}, -\mathbf{T})\mathbf{x}'^*$, then we note that $\|\mathbf{T}\| \leq t\sigma\sqrt{(n-k)n}$, therefore $\|\mathbf{y}\| \leq (1 + t\sigma\sqrt{(n-k)n})t\zeta\sqrt{mn} = \beta$. Hence, $\mathbf{y}$ is a solution for Ring-ISIS instances $(\mathbf{a}, u)$.

Therefore, if $\mathcal{Q}$ can break trapdoor privacy of IB-PAEKS, there exists a challenger $\mathcal{P}$ to solve the Ring-ISIS problem. Besides, the trapdoor algorithm of our IB-PAEKS scheme is a probabilistic algorithm. Thus, our scheme satisfies multi-trapdoor privacy.

5 Performance Evaluation

In this section, we compare our schemes with other PEKS/PAEKS schemes (e.g. [6, 17]). We give the experimental results in terms of storage requirements, computational efficiency and end-to-end delay. As for commercial hardware, we use an AMD Ryzen 7 4700U laptop with Radeon Graphics 2.0 GHz CPU and 16 GB RAM. As for server, we utilize a Dell T630 with Intel(R) Xeon(R) CPU E5-2650 v3 at 2.30 GHz and 128 SSD.

Table 3 shows the security comparison of our schemes with PEKS/PAEKS schemes of [17, 33, 6, 26, 36, 19]. Most schemes achieve MCI security and resist quantum attacks, only the scheme of [19] and our PAEKS, IB-PAEKS scheme implement MTP security. Except for the schemes of [19], BOYb and our schemes, which are implemented in the standard model, the other schemes are implemented in ROM. We notice that only QCH+20 and our IB-PAEKS scheme introduce identity-based techniques to simplify public key management.

Table 3. Security comparison of PEKS schemes.

Schemes	CI	MCI	TP	MTP	QR	PP	Model	Assumption
HL [17]	Yes	No	Yes	No	No	No	ROM	DBDH&mDLN
QCH [26]	Yes	Yes	Yes	No	No	Yes	ROM	CBDH
BOYa [6]	Yes	Yes	No	No	Yes	No	ROM	NTRU
BOYb [6]	Yes	Yes	No	No	Yes	No	Standard	LWE
ZXW [36]	Yes	Yes	No	No	Yes	No	ROM	LWE
LTT [19]	Yes	Yes	Yes	Yes	Yes	No	Standard	LWE
Our PEKS	Yes	Yes	No	No	Yes	No	Standard	Ring-LWE&Ring-ISIS
Our PAEKS	Yes	Yes	Yes	Yes	Yes	No	Standard	Ring-LWE&Ring-ISIS
Our IB-PAEKS	Yes	Yes	Yes	Yes	Yes	Yes	Standard	Ring-LWE&Ring-ISIS

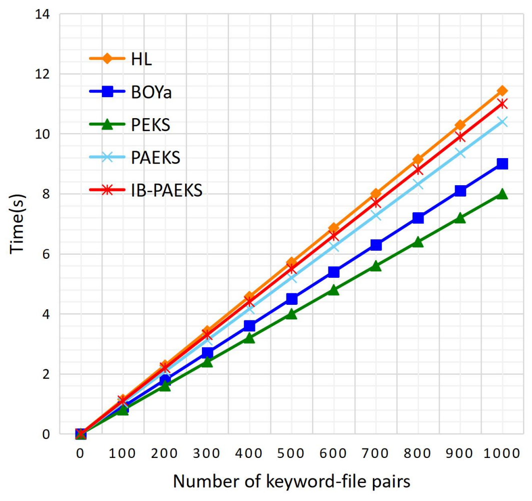
(M)CI: (Multi-)ciphertext indistinguishability. (M)TP: (Multi-)trapdoor privacy.
QR: Quantum resistant. PP: Predigest key.

5.1 Storage Requirements

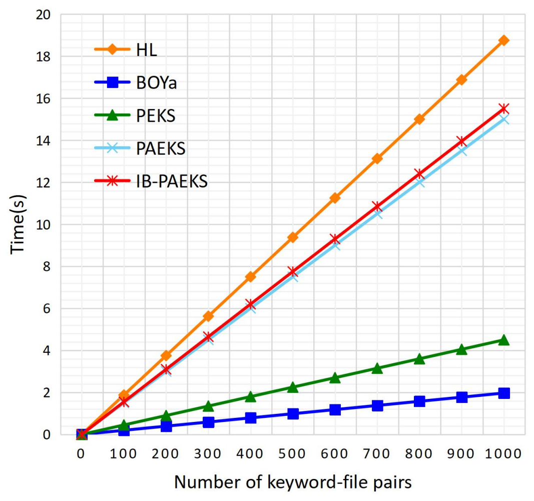
The estimation of storage requirements for implementing lattice-based PEKS schemes is described in Table 4 and Table 5 respectively. For $\lambda = 80$, we set $n = 512, |q| = 62$ for our schemes. To resist IKGA, our PAEKS requires additional ciphertext storage compared to the PEKS scheme. BOYa scheme has a major storage advantage, which is based on NTRU and requires a small storage overhead. Compared our scheme with it, our schemes has a smaller private key size. Compared to LWE-based schemes, our scheme has a better advantage in public key, private key and ciphertext storage.

5.2 Computational Efficiency

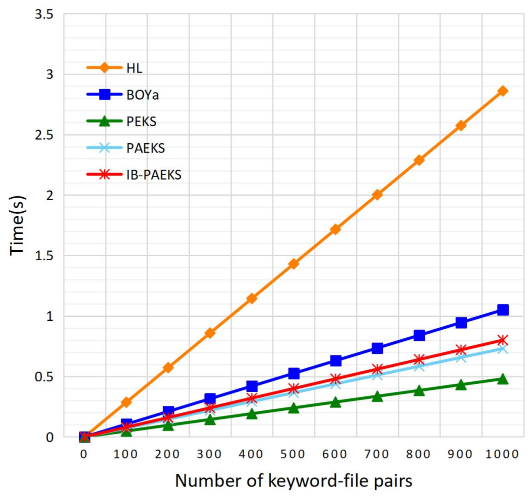
We compared the computational efficiency of our scheme with HL and BOYa. The results are shown in Fig. 4. These schemes have similar trapdoor generation times, with our PEKS being slightly faster



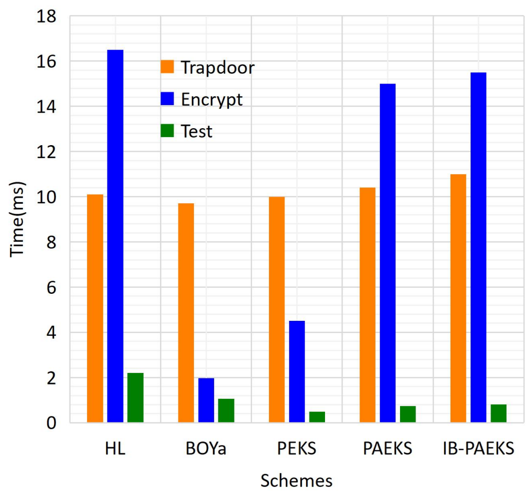
(a) Time of generating trapdoors.



(b) Time of encryption keywords.



(c) Time of testing keywords.



(d) Average running time of each algorithm.

Fig. 4. Computational efficiency of PEKS schemes.

Table 4. Storage requirements of lattice-based PEKS schemes.

Schemes	Public Key	Private Key	Ciphertext	Trapdoor
BOYa [6]	$n\|q\|$	$4n^2\|q\|$	$3n\|q\|$	$2n\|q\|$
BOYb [6]	$m\|q\|((\iota+2)n+1)$	$n^2\|q\|$	$\epsilon(\|q\|+2n\|q\|+1)$	$2n\|q\|$
ZXW [36]	$nm\|q\|$	$nm\|q\|$	$(\iota+n\iota+n)\|q\|$	$n\|q\|$
LTT [19]	$(2m+n\epsilon+nm(\iota+2))\|q\|$	$(n+mk+n^2)\|q\|$	$\epsilon(\|q\|+2n\|q\|+1)$	$2n\|q\|$
Our PEKS	$n\|q\|(\|q\|+3)$	$2n\|q\|$	$n\|q\|(\|q\|+3)$	$n\|q\|(\|q\|+2)$
Our PAEKS	$n\|q\|(\|q\|+3)$	$2n\|q\|$	$n\|q\|(2\|q\|+5)$	$n\|q\|(\|q\|+2)$
Our IB-PAEKS	$n\|q\|(2\|q\|+2)$	$n\|q\|(\|q\|+2)$	$n\|q\|(6\|q\|+5)$	$n\|q\|(2\|q\|+2)$

ϵ, m : They are related to the security parameter. n: The dimension.
$|q|$: The bit length of the modules. ι : The length of keyword.

Table 5. Storage requirements of lattice-based PEKS schemes when $\lambda = 80$.

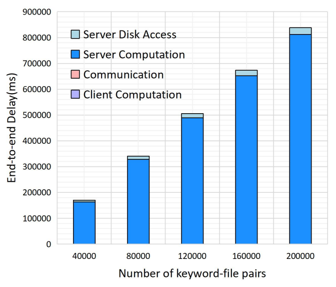
Schemes	Public Key (Kb)	Private Key (Kb)	Ciphertext (Kb)	Trapdoor (Kb)
BOYa [6]	11.5	23,552	34.5	23
BOYb [6]	351,111	1,114,699	2,286	229
ZXW [36]	29,259	29,259	1,257	114
LTT [19]	352,257	29,403	2,286	229
Our PEKS	2,015	62	2,015	1,984
Our PAEKS	2,015	62	3,999	1,984
Our IB-PAEKS	3,906	1,984	11,687	3,906

than the other schemes. We notice that for the timing of the encryption algorithm, our timing is longer than BOYa. For each search query, the Test algorithm executes once on every keyword-file pair. Hence, the efficiency of Test algorithm is especially important, which leads to computational overhead. Note that the Test time depends on the security level, average length of the keywords and the hardware. In our environment, if the security level is 80 bits and the average keyword length is 4 letters, then the average Test time of our PEKS scheme is 0.21ms; if the security level is 192 bits and the average keyword length is 4 letters, then the average Test time of our PEKS scheme is 0.99ms. If the average keyword length is 4 letters and the security level is 80 bits, the result is gathered in Fig. 4. Compared with BOYa, the efficiency of Test algorithm is improved by about 8 times for 80-bit security, which gives us an advantage in end-to-end delay.

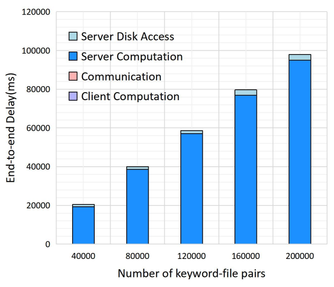
5.3 End-to-end Delay

End-to-end delay is extremely important for practical application of PEKS schemes. Since the end-to-end delay of LWE-based PEKS schemes(e.g. BOYb, ZXW, LTT) is higher, we compare our PEKS scheme with NTRU-based PEKS (BOYa [6]). Based on the complete implementation, the simulation results are shown in Fig. 5.

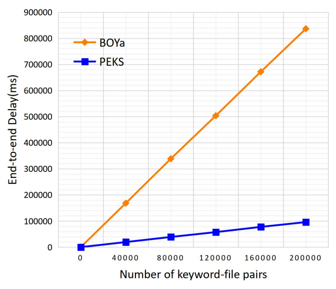
We test the end-to-end delay, including client computation, communication, server computation and server disk access. The server computation time is the time to execute the Test algorithm. The Test algorithm needs to be run once for each keyword-data pair, which is linear with the amount of keyword-data pairs, and the server computing time accounts for the total cost of end-to-end delay. The server disk access is the process that the server traverses all files and takes out all matching files. This is linear with the amount of keyword-data pairs, but it is more efficient than the Test algorithm. To sum up, in BOYa, it accounts for 3% of the total time. In our PEKS, it accounts for 1% of the total time. We have experimented with databases of different measurements on the



(a) End-to-end delay of BOYa [6].



(b) End-to-end delay of PEKS.



(c) End-to-end delay comparison.

Fig. 5. End-to-end delay comparison.

same server. As the amount of keyword-file pairs increases, the gap between our scheme and BOYa increases gradually. For keyword-data pairs up to 200000, where the average keyword length is 8-10 letters, the end-to-end delay of our scheme is 8 times lower than BOYa.

6 Conclusion

In this paper, we construct three PEKS schemes based on Ring-LWE/ISIS. The basic scheme not only enjoys high computational efficiency, but also supplies low end-to-end delay. Compared our basic scheme with LWE-based PEKS schemes (e.g.ZXW [36], LTT [19], BOYb [6]), our PEKS scheme has higher advantages in storage requirements. Compared with the NTRU-based PEKS scheme(BOYa [6]) proved secure in the ROM, our basic scheme ensures high computational efficiency, and provides high level of security. For 80-bit security, the efficiency of our Test algorithm is improved by about 8 times for certain parameters. As a result, the end-to-end delay of our scheme is 8 times lower than BOYa. The two extension schemes provide more secure properties. To resist IKGA, the second scheme is a PAEKS scheme, the security proof indicates that the scheme meets both MCI and MTP security. To predigest key management, the third scheme is an extended PAEKS scheme (IB-PAEKS). This scheme not only meets both MCI and MTP security, but significantly reduces the complexity of key management in practical applications. In summary, our schemes provide a feasible solution for the PEKS scheme, which supplies high computational efficiency, low end-to-end delay and anti-quantum attacks.

Acknowledgements

This work was partially supported by the National Key Research and Development Program of China (No. 2021YFB3100200, 2018YFA0704702), the Major Basic Research Project of Natural Science Foundation of Shandong Province, China (No. ZR202010220025).

References

1. Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., Malone-Lee, J., Neven, G., Paillier, P., Shi, H.: Searchable encryption revisited:consistency properties, relation to anonymous IBE, and extensions. In: Advances in Cryptology – CRYPTO 2005. pp. 205–222 (2005)
2. Agrawal, S., Boneh, D., Boyen, X.: Efficient lattice (H)IBE in the standard model. In: Advances in Cryptology – EUROCRYPT 2010. pp. 553–572 (2010)
3. Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. In: Journal of Mathematical Cryptology. vol. 9, p. 169C203 (2015)
4. Alkim, E., Ducas, L., Pöppelmann, T., Schwabe, P.: Post-quantum key Exchange—A new hope. In: 25th USENIX Security Symposium (USENIX Security 16). pp. 327–343. USENIX Association (Aug 2016)
5. Anada, H., Kanaoka, A., Matsuzaki, N., Watanabe, Y.: Key-updatable public-key encryption with keyword search: Models and generic constructions. In: Information Security and Privacy. pp. 341–359 (2018)
6. Behnia, R., Ozmen, M.O., Yavuz, A.A.: Lattice-based public key searchable encryption from experimental perspectives. IEEE Transactions on Dependable and Secure Computing 17(6), 1269–1282 (2020)
7. Bellare, M., Boldyreva, A., Desai, A., Pointcheval, D.: Key-privacy in public-key encryption. In: Advances in Cryptology — ASIACRYPT 2001. pp. 566–582 (2001)
8. Bert, P., Fouque, P.A., Roux-Langlois, A., Sabt, M.: Practical implementation of Ring-SIS/LWE based signature and IBE. In: Post-Quantum Cryptography. vol. 10786, pp. 271–291 (2018)
9. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Advances in Cryptology - EUROCRYPT 2004. pp. 506–522 (2004)
10. Bos, J., Costello, C., Ducas, L., Mironov, I., Naehrig, M., Nikolaenko, V., Raghunathan, A., Stebila, D.: Frodo: Take off the ring! practical, quantum-secure key exchange from lwe. Tech. Rep. MSR-TR-2016-1137 (September 2016)
11. 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: Secure Data Management. pp. 75–83 (2006)
12. Dai, H., Yang, M., Yang, G., Xiang, Y., Hu, Z., Wang, H.: A keyword-grouping inverted index based multi-keyword ranked search scheme over encrypted cloud data. IEEE Transactions on Sustainable Computing pp. 1–1 (2021)
13. Di Crescenzo, G., Saraswat, V.: Public key encryption with searchable keywords based on jacobi symbols. In: Progress in Cryptology – INDOCRYPT 2007. pp. 282–296 (2007)
14. Genise, N., Micciancio, D.: Faster gaussian sampling for trapdoor lattices with arbitrary modulus. In: Advances in Cryptology – EUROCRYPT 2018. vol. 10820, pp. 174–203 (2018)
15. Gu, C., Zheng, Y., Kang, F., Xin, D.: Keyword search over encrypted data in cloud computing from lattices in the standard model. In: Cloud Computing and Big Data. vol. 9106, pp. 335–343 (2015)
16. Hozhabr, M., Asghari, P., Javadi, H.H.S.: Dynamic secure multi-keyword ranked search over encrypted cloud data. Journal of Information Security and Applications 61, 102902 (2021)
17. Huang, Q., Li, H.: An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks. Information Sciences 403, 1–14 (2017)
18. Kuchta, V., Markowitch, O.: Multi-authority distributed attribute-based encryption with application to searchable encryption on lattices. In: Paradigms in Cryptology – Mycrypt 2016. Malicious and Exploratory Cryptology. pp. 409–435 (2017)
19. Liu, Z.Y., Tseng, Y.F., Tso, R., Mambo, M., Chen, Y.C.: Public-key authenticated encryption with keyword search: Cryptanalysis, enhanced security, and quantum-resistant instantiation. Cryptology ePrint Archive (2021)
20. Lyubashevsky, V., Micciancio, D.: Generalized compact knapsacks are collision resistant. In: Automata, Languages and Programming. vol. 4052, pp. 144–155 (2006)
21. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Advances in Cryptology – EUROCRYPT 2010. pp. 1–23 (2010)

22. Mao, Y., Fu, X., Guo, C., Wu, G.: Public key encryption with conjunctive keyword search secure against keyword guessing attack from lattices. Transactions on Emerging Telecommunications Technologies 30(11), e3531 (Nov 2019)
23. Micciancio, D., Peikert, C.: Trapdoors for lattices: Simpler, tighter, faster, smaller. In: Advances in Cryptology – EUROCRYPT 2012. pp. 700–718 (2012)
24. Pan, X., Li, F.: Public-key authenticated encryption with keyword search achieving both multi-ciphertext and multi-trapdoor indistinguishability. Journal of Systems Architecture 115, 102075 (2021)
25. Peikert, C., Rosen, A.: Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In: Theory of Cryptography. vol. 3876, pp. 145–166 (2006)
26. Qin, B., Chen, Y., Huang, Q., Liu, X., Zheng, D.: Public-key authenticated encryption with keyword search revisited: Security model and constructions. Information Sciences 516, 515–528 (2020)
27. Rhee, H.S., Park, J.H., Susilo, W., Lee, D.H.: Trapdoor security in a searchable public-key encryption scheme with a designated tester. Journal of Systems and Software 83(5), 763–771 (May 2010)
28. Song, D.X., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: Proceeding 2000 IEEE Symposium on Security and Privacy. S P 2000. pp. 44–55 (2000)
29. Stehlé, D., Steinfeld, R., Tanaka, K., Xagawa, K.: Efficient public key encryption based on ideal lattices. In: Advances in Cryptology – ASIACRYPT 2009. pp. 617–635 (2009)
30. Xu, L., Yuan, X., Steinfeld, R., Wang, C., Xu, C.: Multi-writer searchable encryption: an LWE-based realization and implementation. In: Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security. pp. 122–133 (2019)
31. Yang, Y., Zheng, X., Chang, V., Tang, C.: Semantic keyword searchable proxy re-encryption for postquantum secure cloud storage. Concurrency and Computation: Practice and Experience 29(19), e4211 (Oct 2017)
32. Yau, W.C., Heng, S.H., Goi, B.M.: Off-line keyword guessing attacks on recent public key encryption with keyword search schemes. In: Autonomic and Trusted Computing. pp. 100–105 (2008)
33. Yu, X., Xu, C., Xu, L., Wang, Y.: Lattice-based searchable encryption scheme against inside keywords guessing attack. Computers, Materials & Continua 64(2), 1107–1125 (2020)
34. Yu, Y., Ni, J., Yang, H., Mu, Y., Susilo, W.: Efficient public key encryption with revocable keyword search, efficient public key encryption with revocable keyword search. Security and Communication Networks 7(2), 466–472 (Feb 2014)
35. Zhang, X., Xu, C., Mu, L., Zhao, J.: Identity-based encryption with keyword search from lattice assumption. China Communications 15(4), 164–178 (Apr 2018)
36. 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 Transactions on Dependable and Secure Computing 18(3), 1019–1032 (2021)
37. Zhong, H., Li, Z., Cui, J., Sun, Y., Liu, L.: Efficient dynamic multi-keyword fuzzy search over encrypted cloud data. Journal of Network and Computer Applications 149, 102469 (2020)