Public Key Authenticated Encryption with Keyword Search from LWE

Leixiao Cheng [1,2] and Fei Meng [⋆,3,4]

[1]School of Mathematics, Shandong University, Jinan 250100, China
[2]School of Cyber Science and Technology, Shandong University, Qingdao, China
[3]Yanqi Lake Beijing Institute of Mathematical Science and Applications, Beijing, China
[4]Yau Mathematical Sciences Center, Tsinghua University, Beijing, China

Abstract. Public key encryption with keyword search (PEKS) inherently suffers from the inside keyword guessing attack. To resist against this attack, Huang et al. proposed the public key authenticated encryption with keyword search (PAEKS), where the sender not only encrypts a keyword, but also authenticates it. To further resist against quantum attacks, Liu et al. proposed a generic construction of PAEKS and the first quantum-resistant PAEKS instantiation based on lattices. Later, Emura pointed out some issues in Liu et al.'s construction and proposed a new generic construction of PAEKS. The basic construction methodology of Liu et al. and Emura is the same, i.e., each keyword is converted into an extended keyword using the shared key calculated by a word-independent smooth projective hash functions (SPHF), and PEKS is used for the extended keyword.
In this paper, we first analyze the schemes of Liu et al. and Emura, and point out some issues regarding their construction and security model. In short, in their lattice-based instantiations, the sender and receiver use a lattice-based word independent SPHF to compute the same shared key to authenticate keywords, leading to a super-polynomial modulus q; their generic constructions need a trusted setup assumption or the designated-receiver setting; Liu et al. failed to provide convincing evidence that their scheme satisfies their claimed security.
Then, we propose two new lattice-based PAEKS schemes with totally different construction methodology from Liu et al. and Emura. Specifically, in our PAEKS schemes, instead of using the shared key calculated by SPHF, the sender and receiver achieve keyword authentication by using their own secret key to sample a set of short vectors related to the keyword. In this way, the modulus q in our schemes could be of polynomial size, which results in much smaller size of the public key, ciphertext and trapdoor. In addition, our schemes need neither a trusted setup assumption nor the designated-receiver setting. Finally, our schemes can be proven secure in stronger security model, and thus provide stronger security guarantee for both ciphertext privacy and trapdoor privacy.

Keywords: Public Key Authenticated Encryption · Keyword Search · Inside Keyword Guessing Attack · LWE.

⋆ Corresponding author, mengfei_sdu@163.com

1 Introduction

Boneh et al. [7] proposed the first public key encryption with keyword search (PEKS) scheme. In PEKS, the sender encrypts ciphertext keyword ck as ciphertext Ct and uploads Ct along with encrypted files to the cloud server; to retrieve encrypted files containing the specific target keyword tk, the receiver generates the trapdoor Tr of tk and submits Tr to the server; For each Ct in the cloud, the server runs the test algorithm to check whether Ct and Tr embed with the same keyword. If so, it returns the encrypted files corresponding to Ct to the receiver. The security of the PEKS system [7] requires that an attacker cannot derive any information of the ciphertext keyword from the ciphertext.

Later, Byun et al. [8] showed the inherent weakness of PEKS, that is, the information of target keyword in trapdoor could be extracted by the keyword guessing attack (KGA). In detail, given a trapdoor Tr, an attacker launches this attack as follows: (1) It picks a guessing keyword gk and encrypts gk as ciphertext Ct_{gk}; (2) It runs the test algorithm to check whether Tr and Ct_{gk} contain the same keyword. If so, it returns gk; otherwise, it returns to step (1). As discussed in [8], in practical applications, the keyword space usually has low entropy. For example, there are only 225,000 words in Merriam-Webster's collegiate dictionary [26]. Hence, KGA is feasible in real-world scenarios. If the KGA is launched by the server itself, we call it inside KGA (IKGA).

To resist against the IKGA, Huang et al. [15] proposed the public key authenticated encryption with keyword search (PAEKS). In the PAEKS system, the sender (resp., receiver) encrypts and authenticates the ciphertext keyword ck (resp., target keyword tk) with the public key of both parties and its own secret key to obtain the ciphertext Ct (resp., trapdoor Tr). The server runs the test algorithm to check if Ct and Tr embed with the same keyword and if so, returns the encrypted file indexed by Ct to the receiver. The novelty of the PAEKS system is that the sender not only encrypts the ciphertext keyword but also uses his own secret key SK_S to authenticate it, so that the server is unable to generate a valid ciphertext to issue IKGA. The basic security model of the PAEKS scheme [15] requires ciphertext indistinguishability (CI security) and trapdoor indistinguishability (TI security).

However, some recent results have shown that this basic security model can be enhanced. In specific, Noroozi et al. [20] found that [15] is insecure in the multi-user setting; Qin et al. [22] found that [15] does not satisfy the multi-ciphertext indistinguishability (MCI security). Pan et al. [21] claimed that they propose a PAEKS scheme with both MCI security and multi-trapdoor indistinguishability (MTI security) in the one-user setting. However, Cheng et al. [10] found that Pan et al.'s MCI security is broken and the proof of MTI security has a serious mistake. Later, Qin et al. [23] proposed a PAEKS scheme with TI security and cipher-keyword indistinguishability against fully chosen keyword to cipher-keyword attacks (fully CI security). In general, for ciphertext privacy, the security model of PAEKS has been enhanced from the CI security to MCI security, and further to fully CI security. However, when it comes to trapdoor privacy, it is still stagnant, failing to achieve any improvement beyond TI security.

All aforementioned PAEKS schemes are vulnerable to quantum attacks [25]. Recently, Zhang et al. [27, 28] proposed two lattice-based PEKS schemes and claimed that these schemes are resistant to IKGA. Liu et al. [16] proposed a generic construction of PAEKS scheme. But Liu et al. [17] found that neither [27] nor [28] is resistant to IKGA, and that [16] does not follow the syntax of PAEKS since it needs a trusted authority to help users generate their secret keys. Besides, Liu et al. [17] introduced a new generic construction of PAEKS by adopting a word-independent smooth project hash function (SPHF) [12] and a PEKS as building blocks. Furthermore, they proposed a lattice-based PAEKS instantiation by employing the SPHF [6] and PEKS [5], claiming that it was the first quantum-resistant PAEKS scheme with MCI and MTI security. Later, Emura [13] proposed another generic construction of PAEKS scheme. The construction methodology of Liu et al. [17] and Emura [13] is the same, i.e., each keyword is converted into an extended keyword using the shared key calculated by word-independent SPHF, and PEKS is used for the extended keyword.

1.1 Our Contributions

In this paper, we propose two lattice-based PAEKS schemes. Our contributions are reflected in the following three aspects:

- We use totally different techniques to authenticate keyword in the construction of lattice-based PAEKS scheme. Specifically, in previous lattice-based PAEKS schemes [17, 13], the sender and receiver need to calculate the same shared key using word independent SPHF to authenticate keywords, which leads to a super-polynomial modulus q (see Sect.4). Instead, in our PAEKS schemes, the secret key of the sender (resp., receiver) is a short basis, and the sender (resp., receiver) uses his own short basis to sample a set of short vectors related to the keyword to achieve keyword authentication. In this way, $q = \mathsf{poly}(\lambda)$ suffices for the correctness and security analysis of our schemes, which results in much smaller size of public key, ciphertext and trapdoor.
- Our schemes do not require a trusted setup assumption to ensure security. Furthermore, our schemes do not need to apply the designated-receiver setting, where the sender needs to generate a unique public/secret key pair for each designated receiver and the computational and storage burden of the sender scales linearly with the total number of designated-receivers. Therefore, in our schemes, the sender avoids heavy key storage burden.
- Our schemes can be proven secure in stronger security models (see Sect.3.2). Specifically, our first scheme achieves the selective version of fully CI security and target-keyword indistinguishability against fully chosen keyword to target-keyword attacks (fully TI security) in the one-user setting under the standard model; our second scheme can be proven fully CI and fully TI secure in multi-user setting under the random oracle model. Compare with existing PAEKS schemes, our schemes provide stronger security guarantee for both ciphertext privacy and trapdoor privacy.

1.2 Technical Overview

In a PAEKS system, the sender (resp., receiver) not only encrypts a ciphertext keyword (resp., target keyword), but also authenticates it. In the previous PAEKS schemes [22, 23, 17, 13], the sender and receiver would implicitly run an one-round key exchange protocol to calculate the shared key K (using both parties' public keys and their own secret key) for keyword authentication and encryption. Specifically, for those pairing-based PAEKS schemes [22, 23], the sender and receiver compute the shared key K using the well-known Diffie-Hellman key exchange, and then directly use K to encrypt and authenticate the keyword. For those lattice-based PAEKS instantiations [13, 17], the sender and receiver use the word independent SPHF [6] to calculate the same shared key K, and authenticate the keyword by expanding the keyword with K, then PEKS [5] is used for the expanded keyword. Unfortunately, computing the shard key via lattice-based SPHF leads to a super-polynomial modulus q (see Sect.4 for details).

Technically speaking, our new lattice-based PAEKS schemes use totally different construction methodology from previous PAEKS schemes [22, 23, 17, 13]. In our scheme, the sender and receiver no longer calculate the shared key to encrypt and authenticate keywords. Our intuition is to use a variant of the lattice-based anonymous identity-based encryption (IBE) [1] to achieve encryption and authentication. In detail, for keyword encryption, we treat the keyword as an identity and encrypt it by running the encryption algorithm of the IBE scheme. For keyword authentication, we take the user's secret key and keyword as the master secret key and identity in IBE, respectively, then authenticate the keyword by running the key generation algorithm of the IBE scheme. Consequently, the modulus q in our schemes could be of polynomial size.

Following the above methodology, we construct our lattice-based PAEKS scheme as follows. First of all, the sender and receiver respectively run the setup algorithm of IBE to generate the sender's public/secret key pair as $PK_S = (\mathbf{A}, \mathbf{A}_w, \mathbf{U}_S)/SK_S = \mathbf{T_A}$ and the receiver's public/secret key pair as $PK_R = (\mathbf{B}, \mathbf{B}_w)/SK_R = \mathbf{T_B}$, where $\mathbf{A}, \mathbf{A}_w, \mathbf{B}, \mathbf{B}_w \leftarrow \mathbb{Z}_q^{n\times m}$, $\mathbf{U}_S \leftarrow \mathbb{Z}_q^{n\times n}$, $\mathbf{T_A}$ and $\mathbf{T_B}$ are short bases. Then, the PAEKS, Trapdoor and Test algorithms are as follows:

PAEKS. Given PK_R, PK_S, SK_S and a ciphertext keyword ck, the sender does

1. Ciphertext keyword encryption.
$$\mathbf{C}_u = \mathbf{U}_S^\top \mathbf{S}_S + \mathsf{error}_S, \quad \mathbf{C}_{ck} = (\mathbf{A}\|\mathbf{B}\|\mathbf{A}_w + H(ck)\mathbf{G})^\top \mathbf{S}_S + \mathsf{error}'_S.$$
2. Ciphertext keyword authentication. Use the sender's secret key $\mathbf{T_A}$ to generate a matrix $\mathbf{E}_S$ such that each column of $\mathbf{E}_S$ is a short vector and $[\mathbf{A}\|\mathbf{B}\|\mathbf{B}_w + H(ck)\mathbf{G}]\mathbf{E}_S = \mathbf{U}_S$.

Finally, output the PAEKS ciphertext as $Ct = (\mathbf{C}_u, \mathbf{C}_{ck}, \mathbf{E}_S)$.

Trapdoor. Given PK_R, PK_S, SK_R and a target keyword tk, the receiver does

1. Target keyword encryption.
$$\mathbf{T}_u = \mathbf{U}_S^\top \mathbf{S}_R + \mathsf{error}_R, \quad \mathbf{T}_{tk} = (\mathbf{A}\|\mathbf{B}\|\mathbf{B}_w + H(tk)\mathbf{G})^\top \mathbf{S}_R + \mathsf{error}'_R.$$

2. Target keyword authentication. Use the receiver's secret key $\mathbf{T_B}$ to generate a matrix $\mathbf{E}_R$ such that each column of $\mathbf{E}_R$ is a short vector and $[\mathbf{A}\|\mathbf{B}\|\mathbf{A}_w + H(tk)\mathbf{G}]\mathbf{E}_R = \mathbf{U}_S$.

Finally, output the PAEKS trapdoor as $Tr = (\mathbf{T}_U, \mathbf{T}_{tk}, \mathbf{E}_R)$.

Test. Given $Ct = (\mathbf{C}_u, \mathbf{C}_{ck}, \mathbf{E}_S)$ and $Tr = (\mathbf{T}_u, \mathbf{T}_{tk}, \mathbf{E}_R)$, if $ck = tk$, then the server can check that each entry in $\mathbf{C}_u$ (resp., $\mathbf{T}_u$) is close to the corresponding entry in $\mathbf{E}_R^\top \mathbf{C}_{ck}$ (resp., $\mathbf{E}_S^\top \mathbf{T}_{tk}$), i.e.,

$$(\mathbf{C}_u = \mathbf{U}_S^\top \mathbf{S}_S + \mathsf{error}_S) \approx (\mathbf{E}_R^\top \mathbf{C}_{ck} = \mathbf{U}_S^\top \mathbf{S}_S + \mathbf{E}_R^\top \cdot \mathsf{error}'_S),$$
$$(\mathbf{T}_u = \mathbf{U}_S^\top \mathbf{S}_R + \mathsf{error}_R) \approx (\mathbf{E}_S^\top \mathbf{T}_{tk} = \mathbf{U}_R^\top \mathbf{S}_S + \mathbf{E}_R^\top \cdot \mathsf{error}'_R).$$

Unfortunately, so far, the above PAEKS construction can provide neither ciphertext privacy nor trapdoor privacy. In details, given $Ct = (\mathbf{C}_u, \mathbf{C}_{ck}, \mathbf{E}_S)$, $(\mathbf{C}_u, \mathbf{C}_{ck})$ will not leak the information of ck, since the underlying IBE [1] satisfies ciphertext anonymity, i.e., an attacker cannot extract the id from the ciphertext without the secret key corresponding to id. However, $\mathbf{E}_S$ cannot hide ck, since [1] doesn't achieve secret key anonymity. In other words, an attacker can extract the id from the corresponding secret key. Specifically, the attacker picks a guessing keyword ck', then decides whether the guess is correct by checking if

$$[\mathbf{A}\|\mathbf{B}\|\mathbf{B}_w + H(ck')\mathbf{G}]\mathbf{E}_S \overset{?}{=} \mathbf{U}_S. \tag{1}$$

For the same reason, $\mathbf{E}_R$ in trapdoor Tr cannot hide the target keyword tk.

It seems that if the underlying lattice-based IBE achieves both ciphertext anonymity and secret key anonymity, then $\mathbf{E}_S$ (resp., $\mathbf{E}_R$) will not expose ck (resp., tk). Unfortunately, we are not aware of any lattice-based IBE providing both properties. In fact, there is an inherent conflict between secret key anonymity of IBE and the public key setting: given an IBE secret key sk_{id} of id, an attacker can pick a guessing identity id' and generate the guessing IBE ciphertext $ct_{id'}$ under id' using public parameter, then decrypt $ct_{id'}$ by using sk_{id}. If the decryption succeeds, then the attacker knows that $id = id'$.

Now, in order to hide ck (resp., tk) in $\mathbf{E}_S$ (resp., $\mathbf{E}_R$), we try to make the attacker lose the ability to guess keywords using Equ.(1), as shown below:

PAEKS. The sender generates $\mathbf{C}_u, \mathbf{C}_{ck}$ as above, then samples $\mathbf{E}_S$ such that $[\mathbf{A}\|\mathbf{B}\|\mathbf{B}_w + H(ck)\mathbf{G}]\mathbf{E}_S = \mathbf{C}_u$. Finally, it erases $\mathbf{C}_u$ and outputs the PAEKS ciphertext as $Ct = (\mathbf{C}_{ck}, \mathbf{E}_S)$.

Trapdoor. The receiver generates $\mathbf{T}_u, \mathbf{T}_{tk}$ as above, then samples $\mathbf{E}_R$ such that $[\mathbf{A}\|\mathbf{B}\|\mathbf{A}_w + H(tk)\mathbf{G}]\mathbf{E}_R = \mathbf{T}_u$. Finally, it erases $\mathbf{T}_u$ and outputs the PAEKS trapdoor as $Tr = (\mathbf{T}_{tk}, \mathbf{E}_R)$.

Test. Given $Ct = (\mathbf{C}_{ck}, \mathbf{E}_S)$ and $Tr = (\mathbf{T}_{tk}, \mathbf{E}_R)$, if $ck = tk$, the server can check that each entry in $\mathbf{E}_R^\top \mathbf{C}_{ck}$ is close to the corresponding entry in $\mathbf{T}_{tk}^\top \mathbf{E}_S$, since

$$\mathbf{E}_R^\top \mathbf{C}_{ck} \approx \mathbf{E}_R^T \cdot (\mathbf{A}\|\mathbf{B}\|\mathbf{A}_w + H(ck)\mathbf{G})^\top \mathbf{S}_S \approx \left(\mathbf{S}_R^\top \mathbf{U}_S^\top\right) \cdot \mathbf{S}_S,$$
$$\mathbf{T}_{tk}^\top \mathbf{E}_S \approx \mathbf{S}_R^\top (\mathbf{A}\|\mathbf{B}\|\mathbf{B}_w + H(tk)\mathbf{G}) \cdot \mathbf{E}_S \approx \mathbf{S}_R^\top \cdot \left(\mathbf{U}_S^\top \mathbf{S}_S\right).$$

In this way, because $\mathbf{C}_u$ (resp., $\mathbf{T}_u$) is kept secret from the attacker, he/she can no longer extract ck (resp., tk) from $\mathbf{E}_S$ (resp., $\mathbf{E}_R$) by picking a guessing ciphertext keyword ck' (resp., guessing target keyword tk') and checking whether

$$[\mathbf{A}\|\mathbf{B}\|\mathbf{B}_w + H(ck')\mathbf{G}]\mathbf{E}_S \overset{?}{=} \mathbf{C}_u \ \Big(\text{resp., } [\mathbf{A}\|\mathbf{B}\|\mathbf{B}_w + H(tk')\mathbf{G}]\mathbf{E}_R \overset{?}{=} \mathbf{T}_u\Big).$$

So far, we have introduced the main idea of our lattice-based PAEKS schemes. In our concrete PAEKS schemes in Sect.5 and Sect.6, we will choose two matrices $\mathbf{D}_A, \mathbf{D}_B \in \mathbb{Z}_q^{n\times m}$ additionally, and add $\mathbf{D}_A$ into PK_S and $\mathbf{D}_B$ into PK_R respectively. Meanwhile, the ciphertext and trapdoor will be modified accordingly. These additional matrices serve for the security proof of our schemes.

2 Preliminaries

For a vector $\boldsymbol{t}$, let $\|\mathbf{t}\|$ denote its ℓ_2 norm. For a matrix $\mathbf{T} \in \mathbb{Z}^{n\times m}$, let $\mathbf{T}[i,j]$ denote its (i,j)-th entry, let $\|\mathbf{T}\|$ denote the maximum length of its column vectors, let $\widetilde{\mathbf{T}}$ denote its Gram-Schmidt orthogonalization, and let $s_1(\mathbf{T}) := \sup_{\|\boldsymbol{u}\|=1} \|\mathbf{T}\boldsymbol{u}\|$. Let $[\mathbf{A}\|\mathbf{B}]$ and $[\mathbf{A};\mathbf{B}]$ denote horizontal and vertical concatenation of vector and/or matrices, respectively. Let $\mathcal{D}_{\Lambda,\sigma}$ represent the standard discrete Gaussian distribution over Λ with Gaussian parameter σ.

Background on lattices. Let $\mathbf{B} = \{\boldsymbol{b}_1 \cdots \boldsymbol{b}_m\} \subset \mathbb{R}^m$ consist of m linearly independent vectors. The m-dimensional full-rank lattice Λ generated by the *basis* $\mathbf{B}$ is the set $\Lambda = \mathcal{L}(\mathbf{B}) := \{\sum_{i=1}^m x_i\boldsymbol{b}_i \mid x_i \in \mathbb{Z}\}$. For any integers n, m and $q \geq 2$, a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and a vector $\boldsymbol{u} \in \mathbb{Z}_q^n$, we define $\mathcal{L}_q^{\perp}(\mathbf{A}) := \{\boldsymbol{z} \in \mathbb{Z}^m : \mathbf{A}\cdot\boldsymbol{z} = \mathbf{0}_n \mod q\}$ and $\mathcal{L}_q^{\boldsymbol{u}}(\mathbf{A}) := \{\boldsymbol{z} \in \mathbb{Z}^m : \mathbf{A}\cdot\boldsymbol{z} = \boldsymbol{u} \mod q\}$.

Discrete Gaussian Distribution. Let $\mathcal{D}_{\Lambda,\sigma}$ represent the standard discrete Gaussian distribution over Λ with Gaussian parameter σ. We recall some properties of the discrete Gaussian distribution. Let Λ be an m-dimensional lattice. For any vector $\boldsymbol{c} \in \mathbb{R}^m$ and any parameter $\sigma \in \mathbb{R}_{>0}$, define $\rho_{\sigma,\boldsymbol{c}}(\boldsymbol{x}) = \exp(-\pi\frac{\|\boldsymbol{x}-\boldsymbol{c}\|^2}{\sigma^2})$ and $\rho_{\sigma,\boldsymbol{c}}(\Lambda) = \sum_{\boldsymbol{x}\in\Lambda}\rho_{\sigma,\boldsymbol{c}}(\boldsymbol{x})$. The *discrete Gaussian distribution* over Λ with center $\boldsymbol{c}$ and Gaussian parameter σ is $\mathcal{D}_{\Lambda,\sigma,\boldsymbol{c}} = \frac{\rho_{\sigma,\boldsymbol{c}}(\boldsymbol{y})}{\rho_{\sigma,\boldsymbol{c}}(\Lambda)}$ for $\forall \boldsymbol{y} \in \Lambda$. If $\boldsymbol{c} = \mathbf{0}$, we conveniently use ρ_σ and $\mathcal{D}_{\Lambda,\sigma}$. We recall some properties of the discrete Gaussian distribution.

Lemma 1 ([14]). *Let $n, m, q > 0$ denote integers with q a prime, $m \geq 2n\lceil\log q\rceil$. Let σ denote a real with $\sigma \geq \omega(\sqrt{\log m})$. For $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}$ and $\boldsymbol{e} \leftarrow \mathcal{D}_{\mathbb{Z}^m,\sigma}$, the distribution of $\boldsymbol{u} = \mathbf{A}\boldsymbol{e} \mod q$ is statistically close to uniform over $\mathbb{Z}_q^n$. For a fixed $\boldsymbol{u} \in \mathbb{Z}_q^n$, the conditional distribution of $\boldsymbol{e} \leftarrow \mathcal{D}_{\mathbb{Z}^m,\sigma}$, given $\mathbf{A}\boldsymbol{e} = \boldsymbol{u} \mod q$ for an $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}$ is $\mathcal{D}_{\Lambda_q^{\boldsymbol{u}}(\mathbf{A}),\sigma}$ with all but a negligible probability.*

Lemma 2 ([14]). *Λ is an m-dimensional lattice and $\mathbf{T}$ is its basis. Assume $\sigma \geq \|\widetilde{\mathbf{T}}\| \cdot \omega(\sqrt{\log m})$, then $\Pr[\|\boldsymbol{x}\| > \sigma\sqrt{m} : \boldsymbol{x} \leftarrow \mathcal{D}_{\Lambda,\sigma}] \leq \mathsf{negl}(m)$.*

Sampling Algorithms. We review some trapdoor generation algorithms and sampling algorithms in the following lemmas.

Lemma 3. *Let $n, m, q > 0$ be positive integers with q a prime,*

- [3, 19, 2] *there's a PPT algorithm* $\mathsf{TrapGen}$ *that when $m \geq 6n\lceil \log q \rceil$, outputs a pair $(\mathbf{A}, \mathbf{T_A}) \in \mathbb{Z}_q^{n\times m} \times \mathbb{Z}^{m\times m}$ such that $\mathbf{A}$ is full rank and statistically close to uniform and $\mathbf{T_A}$ is a basis for $\Lambda_q^{\perp}(\mathbf{A})$ satisfying $\|\widetilde{\mathbf{T_A}}\| = O(\sqrt{n \log q})$.*
- [19] *when $m \geq n\lceil \log q \rceil$, there exists a fixed full rank matrix $\mathbf{G} \in \mathbb{Z}_q^{n\times m}$ such that the lattice $\Lambda_q^{\perp}(\mathbf{G})$ has a basis $\mathbf{T_G} \in \mathbb{Z}^{m\times m}$ with $\|\widetilde{\mathbf{T_G}}\| \leq \sqrt{5}$.*

The algorithms in the following lemma can be extended from a vector $\boldsymbol{u}$ to a matrix $\mathbf{U}$ by processing each column of $\mathbf{U}$ separately then combining the results.

Lemma 4. *Let integers $q > 2$ and $m > n$, then we have*

- [14] *a PPT algorithm* $\mathsf{SamplePre}(\mathbf{A}, \mathbf{T_A}, \boldsymbol{u}, \sigma)$ *that inputs a full rank matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, a basis $\mathbf{T_A} \in \mathbb{Z}^{m\times m}$ of $\Lambda_q^{\perp}(\mathbf{A})$, a vector $\boldsymbol{u} \in \mathbb{Z}_q^n$, and a Gaussian parameter $\sigma > \|\widetilde{\mathbf{T_A}}\| \cdot \omega(\sqrt{\log m})$, outputs a vector $\boldsymbol{e} \in \mathbb{Z}^m$ distributed statistically close to $\mathcal{D}_{\Lambda_q^{\boldsymbol{u}}(\mathbf{A}),\sigma}$.*
- [1] *a PPT algorithm* $\mathsf{SampleLeft}(\mathbf{A}, \mathbf{B}, \boldsymbol{u}, \mathbf{T_A}, \sigma)$ *that inputs a full rank matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, a matrix $\mathbf{B} \in \mathbb{Z}_q^{n\times \bar{m}}$, a vector $\boldsymbol{u} \in \mathbb{Z}_q^n$, a basis $\mathbf{T_A} \in \mathbb{Z}^{m\times m}$ of $\Lambda_q^{\perp}(\mathbf{A})$, and a Gaussian parameter $\sigma > \|\widetilde{\mathbf{T_A}}\| \cdot \omega(\sqrt{\log(m+\bar{m})})$, outputs a vector $\boldsymbol{e} \in \mathbb{Z}^{m+\bar{m}}$ distributed statistically close to $\mathcal{D}_{\Lambda_q^{\boldsymbol{u}}([\mathbf{A}\|\mathbf{B}]),\sigma}$.*
- [1] *a PPT algorithm* $\mathsf{SampleRight}(\mathbf{A}, \mathbf{B}, \mathbf{R}, \boldsymbol{u}, \mathbf{T_B}, \sigma)$ *that inputs matrices $\mathbf{A}, \mathbf{B} \in \mathbb{Z}_q^{n\times m}$, where $\mathbf{B}$ is full rank, a matrix $\mathbf{R} \in \mathbb{Z}_q^{m\times m}$, a vector $\boldsymbol{u} \in \mathbb{Z}_q^n$, a basis $\mathbf{T_B}$ of $\Lambda_q^{\perp}(\mathbf{B})$, and a Gaussian parameter $\sigma > \|\widetilde{\mathbf{T_B}}\| \cdot s_1(\mathbf{R}) \cdot \omega(\sqrt{\log m})$, outputs $\boldsymbol{e} \in \mathbb{Z}^{2m}$ distributed statistically close to $\mathcal{D}_{\Lambda_q^{\boldsymbol{u}}([\mathbf{A}\|\mathbf{AR}+\mathbf{B}]),\sigma}$.*

In this paper, we set $\sigma_r = \sqrt{n \log q} \cdot \omega(\sqrt{\log m})$, let $\mathcal{D}^{m\times m}$ denote the distribution on matrices in $\mathbb{Z}^{m\times m}$ defined as $(\mathcal{D}_{\mathbb{Z}^m,\sigma_r})^m$, conditioned on the resulting matrix being $\mathbb{Z}_q$-invertible. For a m-dimension lattice $\Lambda_q^{\perp}(\mathbf{A})$ and $i = 1, 2, \ldots, m$, sample $\boldsymbol{c} \leftarrow \mathcal{D}_{\Lambda_q^{\perp}(\mathbf{A}),\sigma}$ repeatedly until $\boldsymbol{c}$ is linearly independent of $\{\boldsymbol{c}_1, \cdots, \boldsymbol{c}_{i-1}\}$ and set $\boldsymbol{c}_i \leftarrow \boldsymbol{c}$, then transform $\{\boldsymbol{c}_1, \cdots, \boldsymbol{c}_m\}$ into a basis $\mathbf{T_A}$ of $\Lambda_q^{\perp}(\mathbf{A})$ using Lemma 7.1 in [18]. The distribution of this $\mathbf{T_A}$ is denoted as $\mathcal{D}(\Lambda_q^{\perp}(\mathbf{A}), \sigma)$.

Lemma 5 ([9]). *There is an efficient algorithm* $\mathsf{SampleRwithBasis}(\mathbf{A}, \sigma) \to (\mathbf{R}, \mathbf{T}_{\mathbf{A}\cdot\mathbf{R}^{-1}})$ *that takes as input a full rank matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and a Gaussian parameter $\sigma \geq \sqrt{n \log q} \cdot \omega(\sqrt{\log m})$, outputs an $\mathbb{Z}_q$-invertible matrix $\mathbf{R} \in \mathbb{Z}^{m\times m}$ sampled from a distribution statistically close to $\mathcal{D}^{m\times m}$, and a basis $\mathbf{T}_{\mathbf{A}\cdot\mathbf{R}^{-1}} \in \mathbb{Z}^{m\times m}$ distributed statistically close to $\mathcal{D}(\Lambda_q^{\perp}(\mathbf{A}\cdot\mathbf{R}^{-1}), \sigma)$.*

The following lemma can be obtained by generalizing Lemma 13 in [1].

Lemma 6 (leftover hash lemma [1]). *Let $\mathcal{H} = \{h : X \to Y\}_{h\in\mathcal{H}}$ be a universal hash family. Let $f : X \to Z$ be some function. Let U_Y denote a uniform independent random variable in Y. Let $T_1, \ldots, T_k$ be independent random variables taking values in X, let $\gamma := \max_{i=1}^{k}(\max_{t_i} \Pr[T_i = t_i])$, then $\triangle\Big((h, h(T_1), f(T_1), \ldots, h(T_k), f(T_k))\ ;\ (h, U_Y^{(1)}, f(T_1), \ldots, U_Y^{(k)}, f(T_k))\Big) \le \frac{k}{2} \cdot \sqrt{\gamma \cdot |Y| \cdot |Z|}$. Specifically, suppose $q > 2$ is prime, $m > (n + \bar{n}) \log q + \omega(\log n)$, $k = poly(n)$. Let $\mathbf{A}$, $\mathbf{B}$, $\mathbf{R}$ be matrices chosen uniformly from $\mathbb{Z}_q^{n\times m}$, $\mathbb{Z}_q^{n\times k}$, $\{-1, 1\}^{m\times k} \bmod q$, respectively. For all matrices $\mathbf{W}$ in $\mathbb{Z}_q^{m\times\bar{n}}$, the distribution $(\mathbf{A}, \mathbf{AR}, \mathbf{R}^\top\mathbf{W})$ is statistically close to the distribution $(\mathbf{A}, \mathbf{B}, \mathbf{R}^\top\mathbf{W})$.*

Lemma 7. *Let $\mathbf{R}$ be a $m \times k$ matrix chosen at random from $\{-1, 1\}^{m\times k}$, then there exists a universal constant C such that $\Pr[s_1(\mathbf{R}) > C\sqrt{m + k}] < e^{-m}$.*

Full Rank Difference Encoding (FRD) We use an encoding function to map keywords in $\mathbb{Z}_q^n$ to matrices in $\mathbb{Z}_q^{n\times n}$.

Definition 1 ([1, 11]). *Let q be a prime and n a positive integer. A function $H : \mathbb{Z}_q^n \to \mathbb{Z}_q^{n\times n}$ is a full-rank difference (FRD) map if: for all distinct $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_q^n$, the matrix $H(\mathbf{x}) - H(\mathbf{y})$ is full rank and H is computable in polynomial time.*

Learning with Errors (LWE) Assumption. In security proof, we need the LWE assumption as follows, which can be obtained by combining [24, 4]. For an $\alpha \in (0, 1)$ and a prime q, let $\bar{\Psi}_\alpha$ denote the distribution over $\mathbb{Z}_q$ of the random variable $\lfloor qX \rceil \bmod q$, where X is a normal random variable with mean 0 and standard deviation $\frac{\alpha}{\sqrt{2\pi}}$.

Assumption 1 ([24, 4]) *Consider a prime q, integers $n, \bar{n}, m$, a real $\alpha \in (0, 1)$ such that $\alpha q > 2\sqrt{n}$, and a PPT algorithm $\mathcal{A}$, the advantage for the LWE problem $\mathsf{LWE}_{n,\bar{n},m,q,\alpha}$ of $\mathcal{A}$ is defined as $|\mathsf{Pr}[\mathcal{A}(\mathbf{A}, \mathbf{A}^\top\mathbf{S}+\mathbf{E}) = 1] - \mathsf{Pr}[\mathcal{A}(\mathbf{A}, \mathbf{V}) = 1]|$, where $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}$, $\mathbf{S} \leftarrow \bar{\Psi}_\alpha^{n\times\bar{n}}$, $\mathbf{E} \leftarrow \bar{\Psi}_\alpha^{m\times\bar{n}}$, $\mathbf{V} \leftarrow \mathbb{Z}_q^{m\times\bar{n}}$. The LWE assumption holds if the above advantage is negligible for all PPT $\mathcal{A}$.*

Lemma 8 ([1]). *Let $\boldsymbol{t}$ be some vector in $\mathbb{Z}^m$ and let $\boldsymbol{x} \leftarrow \bar{\Psi}_\alpha^m$, then the quantity $\boldsymbol{t}^\top\boldsymbol{x}$ treated as an integer in $[0, q-1]$ satisfies $|\boldsymbol{t}^\top\boldsymbol{x}| \le \|\boldsymbol{t}\|q\alpha\omega(\log m) + \|\boldsymbol{t}\|\sqrt{m}/2$ with all but negligible probability in m.*

3 System and Security Models of PAEKS

3.1 System Model of PAEKS

There are three entities (sender, receiver, server) in the PAEKS system.

- $\mathsf{Setup}(1^\lambda) \to PP$: Given a security parameter λ, the algorithm generates the global public parameter PP.

- $\mathsf{KeyGen}_S(PP) \to (PK_S, SK_S)$: Given PP, the sender runs this algorithm to generate its public/secret key pair (PK_S, SK_S).
- $\mathsf{KeyGen}_R(PP) \to (PK_R, SK_R)$: Given PP, the receiver runs this algorithm to generate its public/secret key pair (PK_R, SK_R).
- $\mathsf{PAEKS}(PK_R, PK_S, SK_S, ck) \to Ct$: Given PK_R, PK_S, SK_S and a ciphertext keyword ck, the sender generates the ciphertext Ct embedded with ck.
- $\mathsf{Trapdoor}(PK_R, PK_S, SK_R, tk) \to Tr$: Given PK_R, PK_S, SK_R and a target keyword tk, the receiver generates the trapdoor Tr of tk.
- $\mathsf{Test}(Ct, Tr) \to 0\, or\, 1$: Given Ct, Tr, the server runs this algorithm to check whether Ct and Tr embed with the same keyword. If so, it outputs 1; otherwise 0.

3.2 Security Model of PAEKS

We introduce two security models for PAEKS. The first model that captures ciphertext privacy, i.e., the ciphertext indistinguishability against fully chosen keyword to cipher-keyword attacks (fully CI security), is defined by [23]. The second model that captures trapdoor privacy, i.e., the trapdoor indistinguishability against fully chosen keyword to target-keyword attacks (fully TI security), was roughly mentioned in [23], and here we formally define it.

Fully CI security.

- **Setup**: Given a security parameter, the challenger $\mathcal{C}$ runs the Setup algorithm to generate PP, then runs $\mathsf{KeyGen_S}, \mathsf{KeyGen_R}$ to generate the challenge sender's key pair (PK_S, SK_S) and the challenge receiver's key pair (PK_R, SK_R) respectively, finally sends PP, PK_S, PK_R to $\mathcal{A}$.
- **Phase 1**: $\mathcal{A}$ can submit polynomial queries to the ciphertext oracle $\mathcal{O}_C$ and the trapdoor oracle $\mathcal{O}_T$ as follows.

  $\mathcal{O}_C$**:** Given a receiver's public key $\widetilde{PK}_R$ (not necessarily the challenge receiver's public key PK_R) and a ciphertext keyword ck, $\mathcal{C}$ returns to $\mathcal{A}$ with the ciphertext $Ct \leftarrow \mathsf{PAEKS}(\widetilde{PK}_R, PK_S, SK_S, ck)$.

  $\mathcal{O}_T$**:** Given a sender's public key $\widetilde{PK}_S$ (not necessarily the challenge sender's public key PK_S) and a target keyword tk, $\mathcal{C}$ returns to $\mathcal{A}$ with the trapdoor $Tr \leftarrow \mathsf{Trapdoor}(PK_R, \widetilde{PK}_S, SK_R, tk)$.
- **Challenge**: $\mathcal{A}$ chooses two challenge ciphertext keywords ck_0^* and ck_1^* with the restriction that neither (PK_S, ck_0^*) nor (PK_S, ck_1^*) has been queried on $\mathcal{O}_T$. Then, $\mathcal{C}$ picks a bit $\theta \in \{0,1\}$ randomly, and returns to $\mathcal{A}$ with the challenge ciphertext $Ct^* \leftarrow \mathsf{PAEKS}(PK_R, PK_S, SK_S, ck_\theta^*)$.
- **Phase 2**: It is the same as Phase 1 with the restriction mentioned in the Challenge phase.
- **Output**: $\mathcal{A}$ outputs a guess bit θ' for θ. If $\theta' = \theta$, then $\mathcal{A}$ wins. $\mathcal{A}$'s advantage in winning the game is defined as $Adv_{\mathcal{A}}^{FullyCI}(\lambda) = \left| \Pr[\theta' = \theta] - \frac{1}{2} \right|$.

Definition 2. *If no PPT adversary can win the above fully CI security game with a non-negligible advantage, then the PAEKS scheme is fully CI secure.*

Fully TI security.

- **Setup**: It is the same as Setup phase in the fully CI security game.
- **Phase 1**: It is the same as Phase 1 in the fully CI security game.
- **Challenge**: The adversary $\mathcal{A}$ chooses two challenge keywords tk_0^* and tk_1^* with the restriction that neither (PK_R, tk_0^*) nor (PK_R, tk_1^*) has been queried on $\mathcal{O}_C$. Then, the challenger $\mathcal{C}$ picks a bit $\theta \in \{0,1\}$ randomly, and sends to $\mathcal{A}$ the challenge trapdoor $Tr_\theta^* \leftarrow \text{Trapdoor}(PK_R, PK_S, SK_R, tk_\theta^*)$.
- **Phase 2**: It is the same as Phase 1 with the restriction mentioned in the Challenge phase.
- **Output**: $\mathcal{A}$ outputs a guess bit θ' for θ. If $\theta' = \theta$, then $\mathcal{A}$ wins. $\mathcal{A}$'s advantage in winning the game is defined as $Adv_{\mathcal{A}}^{FullyTI}(\lambda) = \left| \Pr[\theta' = \theta] - \frac{1}{2} \right|$.

Definition 3. *If no PPT adversary can win the above fully TI security game with a non-negligible advantage, then the PAEKS scheme is fully TI secure.*

One/Multi-User Setting. If we limit $\widetilde{PK}_R = PK_R$ and $\widetilde{PK}_S = PK_S$ in the above security games, i.e., the adversary can only obtain ciphertexts from the challenge sender to the challenge receiver by oracle $\mathcal{O}_C$ and trapdoors from the challenge receiver to the challenge sender by oracle $\mathcal{O}_T$, then we call it the *one-user setting*; Otherwise, we call it the *multi-user setting.*
Selectively Fully CI/TI Security. If the adversary $\mathcal{A}$ has to initiate the challenge ciphertext keywords ck_0^* and ck_1^* (resp., challenge target keywords tk_0^* and tk_1^*) in the setup phase before being given PP in the fully CI (resp., fully TI) security game, we call it *selectively fully CI (resp.,selectively fully TI) security.*
CI/TI, MCI/MTI Security. The CI security is similar to the fully CI security, except that the adversary can submit neither (PK_R, ck_0^*) nor (PK_R, ck_1^*) to the oracle $\mathcal{O}_C$. If the adversary is allowed to submit two challenge keyword tuples $(ck_{0,i}^*)_{i\in[1,I]}$ and $(ck_{1,i}^*)_{i\in[1,I]}$ instead of two challenge keywords ck_0^*, ck_1^* in the CI security model, then it is called the MCI security. Similarly, the TI security and MTI security can be defined.
Relation between fully CI/fully TI and MCI/MTI security. As it was shown in [23], the fully CI security implies the MCI security. Similarly, we can prove that the fully TI security implies the MTI security and we omit the proof due to space limitations.

4 Analysis of Liu et al. [17] and Emura [13]

Both Liu et al. [17] and Emura [13] provided generic constructions for the PAEKS scheme. The basic methodology of [17] and [13] is the same, i.e., each keyword is converted into an extended keyword using the shared key calculated by the word-independent SPHF, and PEKS is used for the extended keyword[1].

[1] The difference is that [17] use SPHF twice, while [17] uses SPHF once, and the consistency definition defined in [17] is stronger than that of [17].

In this section, we analyze the issues in [17] and [13] from the perspective of the modulus q in the lattice-based instantiations, system model, and security model.

Modulus q in the lattice-based instantiations. In the lattice-based instantiations of Liu et al. [17] and Emura [13], in order for the sender and receiver to correctly compute the same shared key using word-independent SPHF, the modulus q should be super-polynomial. The details are as follows.

In [17], the instantiation uses the word-independent SPHF (with approximate correctness) based on LWE in [6] to compute the shared key. Specifically, let $\mathsf{R}(x) = \lfloor 2x/q \rceil \mod 2$ be a deterministic rounding function, let $m = O(n \log q)$, let $\mathbf{A}_u \in \mathbb{Z}_q^{n\times m}$, let $t = \sqrt{mn} \cdot \omega(\sqrt{\log n})$, let $s \geq \eta_\epsilon(\Lambda^\perp(\mathbf{A}_u))$ for some $\epsilon = \mathsf{negl}(n)$, take $s = O(\sqrt{n})$ (see Lemma 2.11 [6]). The sender picks $\boldsymbol{k}_S \leftarrow \mathcal{D}^m_{\mathbb{Z},s}$, $\boldsymbol{s}_{S,i} \leftarrow \mathbb{Z}_q^n$, $\boldsymbol{e}_{S,i} \leftarrow \mathcal{D}^m_{\mathbb{Z},t}$, and computes part of its public/secret key pair as

$$\mathsf{pk}_S = \left(\boldsymbol{p}_S = \mathbf{A}_u^T \boldsymbol{k}_S,\ \{\boldsymbol{c}_{S,i} = \mathbf{A}_u \cdot \boldsymbol{s}_{S,i} + \boldsymbol{e}_{S,i}\}_i\right), \mathsf{sk}_S = (\boldsymbol{k}_S,\ \{\boldsymbol{s}_{S,i}\}_i)\,.$$

The receiver picks $\boldsymbol{k}_R \leftarrow \mathcal{D}^m_{\mathbb{Z},s}$, $\boldsymbol{s}_{R,i} \leftarrow \mathbb{Z}_q^n$, $\boldsymbol{e}_{R,i} \leftarrow \mathcal{D}^m_{\mathbb{Z},t}$, and computes part of its public/secret key pair as

$$\mathsf{pk}_R = \left(\boldsymbol{p}_R = \mathbf{A}_u \boldsymbol{k}_R,\ \{\boldsymbol{c}_{R,i} = \mathbf{A}_u^T \cdot \boldsymbol{s}_{R,i} + \boldsymbol{e}_{R,i}\}_i\right), \mathsf{sk}_R = (\boldsymbol{k}_R,\ \{\boldsymbol{s}_{R,i}\}_i)\,.$$

Then, in the PAEKS algorithm, for $i = 1, \ldots, \kappa$, the sender computes

$$h_{S,i} \leftarrow \mathsf{R}(\boldsymbol{c}_{R,i}^\top \cdot \boldsymbol{k}_S \bmod q),\ \ p_{S,i} \leftarrow \mathsf{R}(\boldsymbol{s}_{S,i}^\top \cdot \boldsymbol{p}_R \bmod q),$$

and $y_{S,i} = h_{S,i} \cdot p_{S,i}$, then sets $\boldsymbol{y}_S = y_{S,1}y_{S,2}\cdots y_{S,\kappa} \in \{0,1\}^\kappa$ as the shared key. In the $\mathsf{Trapdoor}$ algorithm, for $i = 1, \ldots, \kappa$, the receiver computes

$$h_{R,i} \leftarrow \mathsf{R}(\boldsymbol{c}_{S,i}^\top \cdot \boldsymbol{k}_R \bmod q) \text{ and } p_{R,i} \leftarrow \mathsf{R}(\boldsymbol{s}_{R,i}^\top \cdot \boldsymbol{p}_S \bmod q).$$

and $y_{R,i} = h_{R,i} \cdot p_{R,i}$, then sets $\boldsymbol{y}_R = y_{R,1}y_{R,2}\cdots y_{R,\kappa} \in \{0,1\}^\kappa$ as the shared key. Note that

$$\begin{aligned}\boldsymbol{c}_{R,i}^\top \cdot \boldsymbol{k}_S &= (\boldsymbol{s}_{R,i}^\top \mathbf{A}_u + \boldsymbol{e}_{R,i}^\top) \cdot \boldsymbol{k}_S = \boldsymbol{s}_{R,i}^\top \cdot \boldsymbol{p}_S + \boldsymbol{e}_{R,i}^\top \cdot \boldsymbol{k}_S \approx \boldsymbol{s}_{R,i}^\top \cdot \boldsymbol{p}_S,\\ \boldsymbol{c}_{S,i}^\top \cdot \boldsymbol{k}_R &= (\boldsymbol{s}_{S,i}^\top \mathbf{A}_u + \boldsymbol{e}_{S,i}^\top) \cdot \boldsymbol{k}_R = \boldsymbol{s}_{S,i}^\top \cdot \boldsymbol{p}_R + \boldsymbol{e}_{S,i}^\top \cdot \boldsymbol{k}_R \approx \boldsymbol{s}_{S,i}^\top \cdot \boldsymbol{p}_R,\end{aligned}$$

where $\boldsymbol{k}_R, \boldsymbol{k}_S, \boldsymbol{e}_{S,i}, \boldsymbol{e}_{S,i}$ are short vectors, $\boldsymbol{s}_{R,i}^\top \cdot \boldsymbol{p}_S$ and $\boldsymbol{s}_{S,i}^\top \cdot \boldsymbol{p}_R$ are almost uniform over $\mathbb{Z}_q$. Therefore, $h_{S,i} = p_{R,i}$ and $h_{R,i} = p_{S,i}$ hold except with probability $\approx 2|\boldsymbol{e}_{R,i}^\top \cdot \boldsymbol{k}_S|/q$ and $\approx 2|\boldsymbol{e}_{S,i}^\top \cdot \boldsymbol{k}_R|/q$, respectively. Here

$$|\boldsymbol{e}_{R,i}^\top \cdot \boldsymbol{k}_S|, |\boldsymbol{e}_{S,i}^\top \cdot \boldsymbol{k}_R| \leq 2t\sqrt{m} \cdot s\sqrt{m} \leq O(m^{1.5} n \cdot \omega(\sqrt{\log n})).$$

The sender and receiver must calculate the same shared key with overwhelming probability, otherwise the correctness of the scheme fails. Therefore, it requires that $h_{S,i} = p_{R,i}$ and $p_{S,i} = h_{R,i}$ for each $i \in [1, \kappa]$, except with negligible probability (i.e., $2^{-\hat{n}}$ for some $\hat{n} = \Theta(\lambda)$). That is,

$$\frac{2|\boldsymbol{e}_{R,i}^\top \cdot \boldsymbol{k}_S|}{q}, \frac{2|\boldsymbol{e}_{S,i}^\top \cdot \boldsymbol{k}_R|}{q} \leq \frac{O(m^{1.5} n \cdot \omega(\sqrt{\log n}))}{q} \leq 2^{-\hat{n}}.$$

Hence, [17]'s instantiation requires an extra parameter constraint

$$q \geq 2^{\hat{n}} \cdot O(m^{1.5} n \cdot \omega(\sqrt{\log n})). \tag{2}$$

Therefore, q is super-polynomial in the security parameter λ because $\hat{n} = \Theta(\lambda)$.

Emura [13] didn't give a specific instantiation, but claimed that it's feasible to initiate with the word-independent SPHF (with statistical correctness) based on LWE in [6]. However, as it was claimed in [6], "*We remark that our word-independent SPHF uses a super-polynomial modulus q to get statistical correctness. It seems hard to construct such an SPHF for a polynomial modulus, as a word-independent SPHF for an IND-CPA encryption scheme directly yields a one-round key exchange and we do not know of any lattice-based one-round key exchange using a polynomial modulus*".

System model. In the previous PAEKS system model [15, 20, 22, 23], the Setup algorithm is defined by inputting a security parameter, and outputting the public parameter PP. The key generation algorithm GenKey_S (resp., GenKey_R), which takes PP as input, is defined to generate sender's (resp., receiver's) key pair. However, as shown in [13], [17] needs a completely trusted Setup to run the key generation algorithm of a public key encryption $(\mathsf{pk}_{\mathsf{PKE}}, \mathsf{dk}_{\mathsf{PKE}}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$, then only outputs $\mathsf{pk}_{\mathsf{PKE}}$ and "erases" $\mathsf{dk}_{\mathsf{PKE}}$, otherwise $\mathsf{dk}_{\mathsf{PKE}}$ can be used to break the underlying membership problem. To avoid the trusted setup assumption, Emura [13] adopted the designated-receiver setting, where the sender inputs a receiver's public key to generate its own public/secret key pair. In this case, the sender needs to generate and store a corresponding public/secret key pair for each designated receiver. Thus, the computation and storage burden of the sender scales linearly with the number of designated receivers.

Security Model. In Theorem 3.3 of Liu et al. [17], they claimed that a PAEKS with CI and TI security achieves MCI (resp. MTI) security if its PAEKS (resp. $\mathsf{Trapdoor}$) algorithm is probabilistic.

However, Emura [13] showed that probabilistic $\mathsf{Trapdoor}$ algorithm is not sufficient to support MTI security and hence Liu et al. [17] did not provide a convincing proof to the MTI security of their scheme. This is because probabilistic $\mathsf{Trapdoor}$ algorithm does not guarantee trapdoor unlinkability that hides information whether two trapdoors contain the same keyword or not.

In fact, Liu et al. [17] failed to provide a convincing proof for the MCI security as well. Similarly, probabilistic PAEKS algorithm is only a necessary condition for achieving MCI security, but not a sufficient condition, since it cannot support ciphertext unlinkability that hides the information whether two ciphertexts contain the same keyword or not. For example, the PAEKS scheme [15] with CI security has a probabilistic PAEKS algorithm, but [22] has proved that [15] cannot achieve MCI security.

5 Our First PAEKS Scheme

Now, we construct our first lattice-based PAEKS scheme that can be proven selectively fully CI/TI secure under the one-user setting in the standard model.

$\mathsf{Setup}(\lambda)$**:** On input a security parameter λ, this algorithm sets the primitive matrix $\mathbf{G}$ with the public trapdoor $\mathbf{T_G}$ (see Lemma 3), chooses a full-rank difference map $H : \mathbb{Z}_q^n \to \mathbb{Z}_q^{n\times n}$, sets the parameters n, m, q, α, σ as specified in Sect.5.2. Then, it returns $PP = \{n, m, q, \alpha, \sigma, H, \mathbf{G}\}$.

$\mathsf{KeyGen}_S(PP) \to (PK_S, SK_S)$**:** On input PP, the sender goes as follows.

1. Run $(\mathbf{A}, \mathbf{T_A}) \leftarrow \mathsf{TrapGen}(n, m, q)$.
2. Choose $\mathbf{U}_S \leftarrow \mathbb{Z}_q^{n\times n}$ and $\mathbf{D}_A, \mathbf{A}_w \leftarrow \mathbb{Z}_q^{n\times m}$.
3. Return $PK_S = (\mathbf{A}, \mathbf{U}_S, \mathbf{D}_A, \mathbf{A}_w)$ and $SK_S = \mathbf{T_A}$.

$\mathsf{KeyGen}_R(PP) \to (PK_R, SK_R)$**:** On input PP, the receiver goes as follows.

1. Run $(\mathbf{B}, \mathbf{T_B}) \leftarrow \mathsf{TrapGen}(n, m, q)$.
2. Choose $\mathbf{D}_B, \mathbf{B}_w \leftarrow \mathbb{Z}_q^{n\times m}$.
3. Return $PK_R = (\mathbf{B}, \mathbf{D}_B, \mathbf{B}_w)$ and $SK_R = \mathbf{T_B}$.

$\mathsf{PAEKS}(PK_R, PK_S, SK_S, ck) \to Ct$**:** On input PK_R, PK_S, SK_S and a ciphertext keyword ck, the sender picks $\mathbf{S}_S \leftarrow \bar{\Psi}_\alpha^{n\times l_S}$ and works as follows.

1. Compute $\mathbf{C}_a = \mathbf{A}^\top \mathbf{S}_S + \mathbf{E}_A$, where $\mathbf{E}_A \leftarrow \bar{\Psi}_\alpha^{m\times l_S}$.
2. Compute $\mathbf{C}_b = \mathbf{B}^\top \mathbf{S}_S + \mathbf{R}_B^\top \mathbf{E}_A$, where $\mathbf{R}_B \leftarrow \{-1, 1\}^{m\times m}$.
3. Compute $\mathbf{C}_d = \mathbf{D}_A^\top \mathbf{S}_S + \mathbf{R}_D^\top \mathbf{E}_A$, where $\mathbf{R}_D \leftarrow \{-1, 1\}^{m\times m}$.
4. Compute $\mathbf{C}_w = [\mathbf{A}_w + H(ck)\mathbf{G}]^\top \mathbf{S}_S + \mathbf{R}_{A,w}^\top \mathbf{E}_A$, where $\mathbf{R}_{A,w} \leftarrow \{-1, 1\}^{m\times m}$.
5. Compute $\mathbf{C}_u = \mathbf{U}_S^\top \mathbf{S}_S + \mathbf{E}_U$, where $\mathbf{E}_U \leftarrow \bar{\Psi}_\alpha^{n\times l_S}$.
6. Sample $\mathbf{E}_S \leftarrow \mathsf{SampleLeft}(\mathbf{A}, \mathbf{B}\|\mathbf{D}_B\|[\mathbf{B}_w + H(ck)\mathbf{G}], \mathbf{C}_u, \mathbf{T_A}, \sigma)$ such that $\mathbf{E}_S \in \mathbb{Z}^{4m\times l_S}$ and $[\mathbf{A}\|\mathbf{B}\|\mathbf{D}_B\|\mathbf{B}_w + H(ck)\mathbf{G}]\mathbf{E}_S = \mathbf{C}_u$.
7. Return $Ct = (\mathbf{C}_a, \mathbf{C}_b, \mathbf{C}_d, \mathbf{C}_w, \mathbf{E}_S)$ as the ciphertext.

$\mathsf{Trapdoor}(PK_R, PK_S, SK_R, tk) \to Tr$**:** Given PK_R, PK_S, SK_R and a trapdoor keyword tk, the receiver picks $\mathbf{S}_R \leftarrow \bar{\Psi}_\alpha^{n\times l_R}$, then works as follows.

1. Compute $\mathbf{T}_a = \mathbf{A}^\top \mathbf{S}_R + \mathbf{E}'_A$, where $\mathbf{E}'_A \leftarrow \bar{\Psi}_\alpha^{m\times l_R}$.
2. Compute $\mathbf{T}_b = \mathbf{B}^\top \mathbf{S}_R + \mathbf{R}'^\top_B \mathbf{E}'_A$, where $\mathbf{R}'_B \leftarrow \{-1, 1\}^{m\times m}$.
3. Compute $\mathbf{T}_d = \mathbf{D}_B^\top \mathbf{S}_R + \mathbf{R}'^\top_D \mathbf{E}'_A$, where $\mathbf{R}'_D \leftarrow \{-1, 1\}^{m\times m}$.
4. Compute $\mathbf{T}_w = [\mathbf{B}_w + H(tk)\mathbf{G}]^\top \mathbf{S}_R + \mathbf{R}_{B,w}^\top \mathbf{E}'_A$, where $\mathbf{R}_{B,w} \leftarrow \{-1, 1\}^{m\times m}$.
5. Compute $\mathbf{T}_u = \mathbf{U}_S \mathbf{S}_R + \mathbf{E}'_U$, where $\mathbf{E}'_U \leftarrow \bar{\Psi}_\alpha^{n\times l_R}$.
6. Sample $\mathbf{E}_R \leftarrow \mathsf{SampleLeft}(\mathbf{B}, \mathbf{A}\|\mathbf{D}_A\|[\mathbf{A}_w + H(tk)\mathbf{G}], \mathbf{T}_U, \mathbf{T_B}, \sigma)$ such that $\mathbf{E}_R \in \mathbb{Z}^{4m\times l_R}$ and $[\mathbf{B}\|\mathbf{A}\|\mathbf{D}_A\|\mathbf{A}_w + H(tk)\mathbf{G}]\mathbf{E}_R = \mathbf{T}_u$.
7. Return $Tr = (\mathbf{T}_a, \mathbf{T}_b, \mathbf{T}_d, \mathbf{T}_w, \mathbf{E}_R)$ as the trapdoor.

$\mathsf{Test}(Ct, Tr)$**:** Given Ct and Tr, the server computes $\mathbf{R} = \mathbf{E}_R^\top[\mathbf{C}_b; \mathbf{C}_a; \mathbf{C}_d; \mathbf{C}_w] - [\mathbf{T}_a^\top \| \mathbf{T}_b^\top \| \mathbf{T}_d^\top \| \mathbf{T}_w^\top]\mathbf{E}_S$ and checks whether $|\mathbf{R}[i, j]| < \lfloor q/4 \rfloor$ for each $i \in [1, l_R]$, $j \in [1, l_S]$. If so, it returns 1, otherwise 0.

Correctness. If $ck = tk$, then we have

$$\begin{aligned}
\mathbf{R} =&\mathbf{E}_R^\top[\mathbf{C}_b;\mathbf{C}_a;\mathbf{C}_d;\mathbf{C}_w] - [\mathbf{T}_a^\top\|\mathbf{T}_b^\top\|\mathbf{T}_d^\top\|\mathbf{T}_w^\top]\mathbf{E}_S\\
=&\mathbf{E}_R^\top\,[\mathbf{B}\|\mathbf{A}\|\mathbf{D}_A\|\mathbf{A}_w + H(ck)\mathbf{G}]^\top\mathbf{S}_S + \mathbf{E}_R^\top\,[\mathbf{R}_B\|\mathbf{I}\|\mathbf{R}_D\|\mathbf{R}_{A,w}]^\top\mathbf{E}_A\\
&-\mathbf{S}_R^\top\,[\mathbf{A}\|\mathbf{B}\|\mathbf{D}_B\|\mathbf{B}_w + H(tk)\mathbf{G}]\mathbf{E}_S - \mathbf{E}_A'^\top\,[\mathbf{I}\|\mathbf{R}_B'\|\mathbf{R}_D'\|\mathbf{R}_{B,w}]\mathbf{E}_S\\
=&\mathbf{T}_u^\top\mathbf{S}_S - \mathbf{S}_R^\top\mathbf{C}_u + \mathbf{E}_R^\top\,[\mathbf{R}_B\|\mathbf{I}\|\mathbf{R}_D\|\mathbf{R}_{A,w}]^\top\mathbf{E}_A - \mathbf{E}_A'^\top\,[\mathbf{I}\|\mathbf{R}_B'\|\mathbf{R}_D'\|\mathbf{R}_{B,w}]\mathbf{E}_S\\
=&\mathbf{S}_R^\top\mathbf{U}_S^\top\mathbf{S}_S - \mathbf{S}_R^\top\mathbf{U}_S^\top\mathbf{S}_S + \mathbf{error} = \mathbf{error},
\end{aligned}$$

$\mathbf{error} = \mathbf{E}_R^\top\,[\mathbf{R}_B\|\mathbf{I}\|\mathbf{R}_D\|\mathbf{R}_{A,w}]^\top\mathbf{E}_A - \mathbf{E}_A'^\top\,[\mathbf{I}\|\mathbf{R}_B'\|\mathbf{R}_D'\|\mathbf{R}_{B,w}]\mathbf{E}_S + \mathbf{E}_U'^\top\mathbf{S}_S - \mathbf{S}_R^\top\mathbf{E}_U$. By Lemmas 2, 4, 7, 8, we have $|\,(\mathbf{E}_U'^\top\mathbf{S}_S)[i,j]\,|, |\,(\mathbf{S}_R^\top\mathbf{E}_U)[i,j]\,| \le \sqrt{n}(\alpha q)^2\omega(\log n)$, and $|\left(\mathbf{E}_R^\top\,[\mathbf{R}_B\|\mathbf{I}\|\mathbf{R}_D\|\mathbf{R}_{A,w}]^\top\mathbf{E}_A\right)[i,j]|, |\left(\mathbf{E}_A'^\top\,[\mathbf{I}\|\mathbf{R}_B'\|\mathbf{R}_D'\|\mathbf{R}_{B,w}]\mathbf{E}_S\right)[i,j]| \le \sigma\cdot mq\alpha\omega(\log m) + O(\sigma m^{3/2})$. Therefore, $|\mathbf{error}[i,j]| \le \sigma mq\alpha\omega(\log m) + O(\sigma m^{3/2})$ for every $i \in [1,l_R]$, $j \in [1,l_S]$.

5.1 Security proof

Intuitively, during the selectively fully CI security proof, the simulator will choose $\mathbf{A}$ randomly, run $(\mathbf{D}_B,\mathbf{T}_{\mathbf{D}_B}) \leftarrow \mathsf{TrapGen}(n,m,q)$ and set $\mathbf{A}_w = \mathbf{A}\mathbf{R}_{A,w}^* - H(ck^*)\mathbf{G}$. For a ciphertext query of keyword ck (even the challenge keyword ck^*), the simulator can compute $\mathbf{E}_S$ using $\mathbf{T}_{\mathbf{D}_B}$, and answer *any* ciphertext query successfully. For a trapdoor query of keyword $tk \ne ck^*$, note that $\mathbf{A}_w + H(tk)\mathbf{G} = \mathbf{A}\mathbf{R}_{B,w}^* + (H(tk) - H(ck^*))\mathbf{G}$, so the simulator can use $\mathbf{T}_\mathbf{G}$ to compute $\mathbf{E}_R$ and answer the trapdoor query. Therefore, our first scheme can be proven selectively fully CI secure. The details are as follows.

Theorem 1. *If the decisional-LWE problem is hard, then our first PAEKS scheme is selectively fully CI secure in one-user setting in the standard model.*

Proof. We define a series of games between a simulator $\mathcal{B}$ and an adversary $\mathcal{A}$ who plays the selectively fully CI security game. At the beginning of each game, $\mathcal{A}$ chooses two challenge cipher-keywords ck_0^* and ck_1^*. The first and last games are the real security game with challenge ciphertext $\mathsf{PAEKS}(PK_R, PK_S, SK_S, ck_0^*)$ and $\mathsf{PAEKS}(PK_R, PK_S, SK_S, ck_1^*)$, respectively. In other games, we use algorithms $\widehat{\mathsf{KeyGen}}_S$, $\widehat{\mathsf{KeyGen}}_R$, $\widehat{\mathsf{PAEKS}}^*$, $\widehat{\mathsf{PAEKS}}$ and $\widehat{\mathsf{Trapdoor}}$ alternatively. During these games, the adversary $\mathcal{A}$ can query neither ck_0^* nor ck_1^* on $\mathcal{O}_T$.

- Game_0**:** $\mathcal{B}$ runs the Setup, KeyGen_S, KeyGen_R algorithms to setup the system, answers $\mathcal{A}$'s ciphertext queries and trapdoor queries using the PAEKS and $\mathsf{Trapdoor}$ algorithms respectively, generates the challenge ciphertext using the PAEKS algorithm with ck_0^*.
- Game_1**:** Let $ck^* = ck_0^*$, $\mathcal{B}$ runs the Setup, $\widehat{\mathsf{KeyGen}}_S$, $\widehat{\mathsf{KeyGen}}_R$ to setup the system, and answers ciphertext queries and trapdoor queries using the $\widehat{\mathsf{PAEKS}}$ and $\widehat{\mathsf{Trapdoor}}$ algorithms respectively. Then $\mathcal{B}$ generates the challenge ciphertext using the $\widehat{\mathsf{PAEKS}}^*$ algorithm.

Game_2**:** Let $ck^* = ck_0^*$, $\mathcal{B}$ runs the Setup, $\widehat{\mathsf{KeyGen}}_S$, $\widehat{\mathsf{KeyGen}}_R$ to setup the system, and answers ciphertext queries and trapdoor queries using the $\widehat{\mathsf{PAEKS}}$ and $\widehat{\mathsf{Trapdoor}}$ algorithms respectively. Then $\mathcal{B}$ generates the challenge ciphertext by selecting $\mathbf{C}_a, \mathbf{C}_b, \mathbf{C}_w, \mathbf{C}_u$ uniformly random and samples $\mathbf{E}_S$ from $\mathcal{D}_{\Lambda_q^{\mathbf{C}_u}([\mathbf{A}\|\mathbf{B}\|\mathbf{D}_B\|\mathbf{B}_w+H(ck_0^*)\mathbf{G}]),\sigma}$.

Game_3**:** Let $ck^* = ck_1^*$, $\mathcal{B}$ runs the Setup, $\widehat{\mathsf{KeyGen}}_S$, $\widehat{\mathsf{KeyGen}}_R$ to setup the system, and answers ciphertext queries and trapdoor queries using $\widehat{\mathsf{PAEKS}}$ and $\widehat{\mathsf{Trapdoor}}$ algorithms respectively. Then $\mathcal{B}$ generates the challenge ciphertext by selecting $\mathbf{C}_a, \mathbf{C}_b, \mathbf{C}_w, \mathbf{C}_u$ uniformly random and samples $\mathbf{E}_S$ from $\mathcal{D}_{\Lambda_q^{\mathbf{C}_u}([\mathbf{A}\|\mathbf{B}\|\mathbf{D}_B\|\mathbf{B}_w+H(ck_1^*)\mathbf{G}]),\sigma}$.

Game_4**:** Let $ck^* = ck_1^*$, $\mathcal{B}$ runs Setup, $\widehat{\mathsf{KeyGen}}_S$, $\widehat{\mathsf{KeyGen}}_R$ to setup the system, and answers ciphertext queries and trapdoor queries using $\widehat{\mathsf{PAEKS}}$ and $\widehat{\mathsf{Trapdoor}}$ algorithms respectively. Then $\mathcal{B}$ generates the challenge ciphertext using the $\widehat{\mathsf{PAEKS}}^*$ algorithm.

Game_5**:** $\mathcal{B}$ runs the Setup, KeyGen_S, KeyGen_R algorithms to setup the system, answers $\mathcal{A}$'s ciphertext queries and trapdoor queries using the PAEKS and $\mathsf{Trapdoor}$ algorithms respectively, generates the challenge ciphertext using the PAEKS algorithm with ck_1^*.

Now, we define the algorithms $\widehat{\mathsf{KeyGen}}_S$, $\widehat{\mathsf{KeyGen}}_R$, $\widehat{\mathsf{PAEKS}}^*$, $\widehat{\mathsf{PAEKS}}$, $\widehat{\mathsf{Trapdoor}}$.

$\widehat{\mathsf{KeyGen}}_S(PP, ck^*)$**:** On input PP and ck^*,

1. Choose $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}$ and $\mathbf{U}_S \leftarrow \mathbb{Z}_q^{n\times n}$.
2. Choose $\mathbf{R}_D^*, \mathbf{R}_{A,w}^* \leftarrow \{-1,1\}^{m\times m}$, set $\mathbf{D}_A = \mathbf{A}\mathbf{R}_D^*$, $\mathbf{A}_w = \mathbf{A}\mathbf{R}_{A,w}^* - H(ck^*)\mathbf{G}$.
3. Return $PK_S = (\mathbf{A}, \mathbf{U}_S, \mathbf{D}_A, \mathbf{A}_w)$, $SK_S = (\mathbf{R}_D^*, \mathbf{R}_{A,w}^*)$.

$\widehat{\mathsf{KeyGen}}_R(PP, PK_S, ck^*)$**:** On input PP, PK_S and ck^*,

1. Run $(\mathbf{D}_B, \mathbf{T}_{\mathbf{D}_B}) \leftarrow \mathsf{TrapGen}(n, m, q)$.
2. Choose $\mathbf{R}_B^*, \mathbf{R}_{B,w}^* \leftarrow \{-1,1\}^{m\times m}$, set $\mathbf{B} = \mathbf{A}\mathbf{R}_B^*$, $\mathbf{B}_w = \mathbf{A}\mathbf{R}_{B,w}^* - H(ck^*)\mathbf{G}$.
3. Return $PK_R = (\mathbf{B}, \mathbf{D}_B, \mathbf{B}_w)$, $SK_R = (\mathbf{R}_B^*, \mathbf{R}_{B,w}^*, \mathbf{T}_{\mathbf{D}_B})$.

$\widehat{\mathsf{PAEKS}}^*(PK_R, PK_S, SK_R, SK_S, ck^*)$**:** On input PK_R, PK_S, SK_R, SK_S and ck^*, pick $\mathbf{S}_S \leftarrow \bar{\Psi}_\alpha^{n\times l_S}$, do the following

1. Compute $\mathbf{C}_a = \mathbf{A}^\top \mathbf{S}_S + \mathbf{E}_A$, where $\mathbf{E}_A \leftarrow \bar{\Psi}_\alpha^{m\times l_S}$.
2. Compute $\mathbf{C}_b = \mathbf{B}^\top \mathbf{S}_S + \mathbf{R}_B^{*T}\mathbf{E}_A = \mathbf{R}_B^{*T}(\mathbf{A}^\top \mathbf{S}_S + \mathbf{E}_A)$.
3. Compute $\mathbf{C}_d = \mathbf{D}_A^\top \mathbf{S}_S + \mathbf{R}_D^{*T}\mathbf{E}_A = \mathbf{R}_D^{*T}(\mathbf{A}^\top \mathbf{S}_S + \mathbf{E}_A)$.
4. Compute $\mathbf{C}_w = [\mathbf{A}_w + H(ck^*)\mathbf{G}]^\top \mathbf{S}_S + \mathbf{R}_{A,w}^{*T}\mathbf{E}_A = \mathbf{R}_{A,w}^{*T}(\mathbf{A}^\top \mathbf{S}_S + \mathbf{E}_A)$.
5. Compute $\mathbf{C}_u = \mathbf{U}_S^\top \mathbf{S}_S + \mathbf{E}_U$, where $\mathbf{E}_U \leftarrow \bar{\Psi}_\alpha^{n\times l_S}$.
6. Sample $\bar{\mathbf{E}}_S \leftarrow \mathsf{SampleLeft}(\mathbf{D}_B, \mathbf{A}\|\mathbf{B}\|[\mathbf{B}_w + H(ck^*)\mathbf{G}], \mathbf{C}_u, \mathbf{T}_{\mathbf{D}_B}, \sigma)$ such that $[\mathbf{D}_B\|\mathbf{A}\|\mathbf{B}\|\mathbf{B}_w + H(ck^*)\mathbf{G}]\bar{\mathbf{E}}_S = [\mathbf{D}_B\|\mathbf{A}\|\mathbf{B}\|\mathbf{A}\mathbf{R}_{B,w}^*]\bar{\mathbf{E}}_S = \mathbf{C}_u$. Parse $\bar{\mathbf{E}}_S = (\mathbf{E}_{S,2}; \mathbf{E}_{S,3}; \mathbf{E}_{S,1}; \mathbf{E}_{S,4})$ and set $\mathbf{E}_S = (\mathbf{E}_{S,1}; \mathbf{E}_{S,2}; \mathbf{E}_{S,3}; \mathbf{E}_{S,4})$.
7. Return the ciphertext $Ct^* = (\mathbf{C}_a, \mathbf{C}_b, \mathbf{C}_d, \mathbf{C}_w, \mathbf{E}_S)$.

$\widehat{\mathsf{PAEKS}}(PK_R, PK_S, SK_R, SK_S, ck)$**:** Given PK_R, PK_S, SK_R, SK_S and ck, compute $\mathbf{C}_a, \mathbf{C}_b, \mathbf{C}_d, \mathbf{C}_w, \mathbf{C}_u$ as in the PAEKS algorithm, then

1. Sample $\bar{\mathbf{E}}_S \leftarrow \mathsf{SampleLeft}(\mathbf{D}_B, \mathbf{A}\|\mathbf{B}\|[\mathbf{B}_w + H(ck)\mathbf{G}], \mathbf{C}_u, \mathbf{T}_{\mathbf{D}_B}, \sigma)$ such that $\bar{\mathbf{E}}_S \in \mathbb{Z}^{4m\times l_S}$ and $[\mathbf{D}_B\|\mathbf{A}\|\mathbf{B}\|\mathbf{B}_w + H(ck)\mathbf{G}]\bar{\mathbf{E}}_S = \mathbf{C}_u$. Parse $\bar{\mathbf{E}}_S = (\mathbf{E}_{S,3}; \mathbf{E}_{S,1}; \mathbf{E}_{S,2}; \mathbf{E}_{S,4})$ and set $\mathbf{E}_S = (\mathbf{E}_{S,1}; \mathbf{E}_{S,2}; \mathbf{E}_{S,3}; \mathbf{E}_{S,4})$.
2. Return the ciphertext $Ct = (\mathbf{C}_a, \mathbf{C}_b, \mathbf{C}_d, \mathbf{C}_w, \mathbf{E}_S)$.

$\widehat{\mathsf{Trapdoor}}(PK_R, PK_S, SK_R, SK_S, tk)$**:** On input PK_R, PK_S, SK_R, SK_S and tk, if $tk \in \{ck_0^*, ck_1^*\}$, return $\perp$; else compute $\mathbf{T}_a, \mathbf{T}_b, \mathbf{T}_d, \mathbf{T}_w, \mathbf{T}_u$ as in the $\mathsf{Trapdoor}$ algorithm. Observe that $\mathbf{A}_w + H(tk) = \mathbf{A}\mathbf{R}^*_{A,w} + (H(tk) - H(ck^*))\mathbf{G}$, do the following

1. Sample $(\mathbf{E}_{R,1}; \mathbf{E}_{R,3}) \leftarrow \mathcal{D}_{\mathbb{Z}^{2m\times l_R},\sigma}$, compute $\mathbf{T}'_u = \mathbf{T}_u - [\mathbf{B}\|\mathbf{D}_A]\cdot(\mathbf{E}_{R,1}; \mathbf{E}_{R,3})$, sample $(\mathbf{E}_{R,2}; \mathbf{E}_{R,4}) \leftarrow \mathsf{SampleRight}(\mathbf{A}, (H(ck^*) - H(tk))\mathbf{G}, \mathbf{R}^*_{A,w}, \mathbf{T}'_u, \mathbf{T}_{\mathbf{G}}, \sigma)$ and set $\mathbf{E}_R = (\mathbf{E}_{R,1}; \mathbf{E}_{R,2}; \mathbf{E}_{R,3}; \mathbf{E}_{R,4})$.
2. Return trapdoor $Tr = (\mathbf{T}_a, \mathbf{T}_b, \mathbf{T}_d, \mathbf{T}_w, \mathbf{E}_R)$.

Next, we gradually prove that games $(\mathsf{Game}_i, \mathsf{Game}_{i+1})$ are indistinguishable for $i = 0, 1, 2, 3, 4$.

Lemma 9. *For $i = 0, 4$, Game_i and Game_{i+1} are statistically indistinguishable.*

Proof. We only proof the case when $i = 0$, the proof for $i = 4$ is similar. First, we prove that PK_S, PK_R and challenge ciphertext returned by the $\widehat{\mathsf{KeyGen}}_S$, $\widehat{\mathsf{KeyGen}}_R$ and $\widehat{\mathsf{PAEKS}}^*$ are statistically close to those returned by the KeyGen_S, KeyGen_R and PAEKS. Specifically, by Lemma 3, the distribution of $\mathbf{A}, \mathbf{D}_B$ in Game_0 and Game_1 are statistically indistinguishable. By Lemmas 3 and 6, the distribution of $\big(\mathbf{A}, (\mathbf{D}_A, \mathbf{A}_w, \mathbf{B}, \mathbf{B}_w), (\mathbf{R}_D^{*T}, \mathbf{R}_{A,w}^{*T}, \mathbf{R}_B^{*T}, \mathbf{R}_{B,w}^{*T})\mathbf{E}_A\big)$ in Game_0 and Game_1 are statistically indistinguishable. Therefore, the distributions of PK_S and PK_R in these two games are indistinguishable. By Lemma 4, the distribution of $\mathbf{E}_S$ in Game_0 and Game_1 are statistically indistinguishable. Therefore, the distributions of challenge ciphertext in these two games are indistinguishable.

Next, we only need to show that ciphertexts and trapdoors output by the $\widehat{\mathsf{PAEKS}}$ and $\widehat{\mathsf{Trapdoor}}$ are statistically close to those returned by the PAEKS and $\mathsf{Trapdoor}$. By Lemma 4, the distributions of $\mathbf{E}_S$ and $\mathbf{E}_R$ in Game_0 and Game_1 are statistically indistinguishable, which concludes the proof. □

Lemma 10. *Assume the decision LWE problem is hard, then Game_i and Game_{i+1} are computationally indistinguishable, for $i = 1, 3$.*

Proof. We proof the case when $i=1$. The proof for $i=3$ is similar.

- **Setup**: The adversary $\mathcal{A}$ chooses challenge sender/receiver and two challenge ciphertext keywords ck_0^*, ck_1^*. The simulator $\mathcal{B}$ runs Setup, returns PP to its challenger, who gives $(\mathbf{A}'\|\mathbf{U}', \mathbf{V}'_A\|\mathbf{V}_{U'}) \in \mathbb{Z}_q^{n\times(m+n)} \times \mathbb{Z}_q^{(m+n)\times n}$ to $\mathcal{B}$. Let $\mathbf{A} = \mathbf{A}'$, $\mathbf{U}_S = \mathbf{U}'$, the simulator $\mathcal{B}$ runs $\widehat{\mathsf{KeyGen}}_S$ and $\widehat{\mathsf{KeyGen}}_R$ with $ck^* = ck_0^*$ and returns PP, PK_S/PK_R of challenge sender/receiver to $\mathcal{A}$.
- **Phase 1 and 2**: Run algorithms $\widehat{\mathsf{PAEKS}}$ and $\widehat{\mathsf{Trapdoor}}$ to answer queries on ciphertext oracle $\mathcal{O}_C$ and trapdoor oracle $\mathcal{O}_T$, respectively.

– **Challenge**: Let $\mathbf{C}_a = \mathbf{V}_{A'}$, $\mathbf{C}_b = \mathbf{R}_B^{*T}\mathbf{V}_{A'}$, $\mathbf{C}_d = \mathbf{R}_D^{*T}\mathbf{V}_{A'}$, $\mathbf{C}_w = \mathbf{R}_{A,w}^{*T}\mathbf{V}_{A'}$, $\mathbf{C}_u = \mathbf{V}_{U'}$ (using $\mathbf{R}_B^{*T} \in SK_R$, $\mathbf{R}_D^{*T}, \mathbf{R}_{A,w}^{*T} \in SK_S$), then work as in $\widehat{\mathsf{PAEKS}}^*$ to generate $\mathbf{E}_S$ and output the challenge ciphertext $Ct^* = (\mathbf{C}_a, \mathbf{C}_b, \mathbf{C}_d, \mathbf{C}_w, \mathbf{E}_S)$.

If $(\mathbf{A}', \mathbf{V}'_A)$ and $(\mathbf{U}', \mathbf{V}_{U'})$ are real LWE tuples, $\mathcal{B}$ simulates Game_1. Otherwise, we claim that $\mathcal{B}$ simulates Game_2, which concludes the proof. This is because the quantities $\mathbf{A}(\mathbf{R}_B^{*T} \| \mathbf{R}_D^{*T} \| \mathbf{R}_{A,w}^{*T} \| \mathbf{R}_{B,w}^{*T})$ and $(\mathbf{R}_B^{*T} \| \mathbf{R}_D^{*T} \| \mathbf{R}_{A,w}^{*T})\mathbf{C}_a$ are independent uniformly random samples by Lemma 6. Besides, the distribution of $\mathbf{E}_S$ in Game_2 is indistinguishable from $\mathcal{D}_{\Lambda_q^{\mathbf{C}_u}([\mathbf{A}\|\mathbf{B}\|\mathbf{D}_B\|\mathbf{B}_w + H(ck_0^*)\mathbf{G}]),\sigma}$ by Lemma 1. $\square$

Lemma 11. Game_2 *and* Game_3 *are statistically indistinguishable.*

Proof. Game_2 and Game_3 differ in two places: (1) $\mathbf{A}_w = \mathbf{A}\mathbf{R}_{A,w}^* - H(ck_0^*) \cdot \mathbf{G}$, $\mathbf{B}_w = \mathbf{A}\mathbf{R}_{B,w}^* - H(ck_0^*) \cdot \mathbf{G}$ in Game_2 while $\mathbf{A}_w = \mathbf{A}\mathbf{R}_{A,w}^* - H(ck_1^*) \cdot \mathbf{G}$, $\mathbf{B}_w = \mathbf{A}\mathbf{R}_{B,w}^* - H(ck_1^*) \cdot \mathbf{G}$ in Game_3. By Lemma 6, the distributions of $\mathbf{A}_w$ and $\mathbf{B}_w$ are statistically indistinguishable in these two games; (2) $\mathbf{E}_S$ is chosen from $\mathcal{D}_{\Lambda_q^{\mathbf{C}_u}([\mathbf{A}\|\mathbf{B}\|\mathbf{D}_B\|\mathbf{B}_w + H(ck_0^*)\mathbf{G}]),\sigma} = \mathcal{D}_{\Lambda_q^{\mathbf{C}_u}([\mathbf{A}\|\mathbf{B}\|\mathbf{D}_B\|\mathbf{A}\mathbf{R}_{B,w}^*]),\sigma}$ in Game_2, while from $\mathcal{D}_{\Lambda_q^{\mathbf{C}_u}([\mathbf{A}\|\mathbf{B}\|\mathbf{D}_B\|\mathbf{B}_w + H(ck_1^*)\mathbf{G}]),\sigma} = \mathcal{D}_{\Lambda_q^{\mathbf{C}_u}([\mathbf{A}\|\mathbf{B}\|\mathbf{D}_B\|\mathbf{A}\mathbf{R}_{B,w}^*]),\sigma}$ in Game_3. Obviously, the distribution of $\mathbf{E}_S$ is statistically indistinguishable in these two games, which concludes the proof. $\square$

Proof of Theorem 1. If there is a PPT adversary $\mathcal{A}$ wins the selectively fully CI security game with a non-negligible advantage, then $\mathcal{A}$ can distinguish Game_i and Game_{i+1} for some $i \in [0, 4]$. By Lemma 9 and 11, $i \neq 0, 4$ and $i \neq 2$. Thus, by Lemma 10, $\mathcal{A}$ can be used to solve the LWE problem. $\square$

Theorem 2. *If the decisional-LWE problem is hard, then our first PAEKS scheme is selectively fully TI secure in one-user setting in the standard model.*

Proof. This proof is similar to that of Theorem 1, since the constructions of algorithms PAEKS and $\mathsf{Trapdoor}$ are almost symmetric. $\square$

5.2 Parameter Selection of Our First PAEKS Scheme

Now, to satisfy the correctness and make the security proof work, we need that

(1) $\sigma m q \alpha \omega(\log m) + O(\sigma m^{3/2}) \leq \lfloor q/4 \rfloor$ for correctness,
(2) $m \geq 6n\lceil \log q \rceil$ to ensure $\mathsf{TrapGen}$ works and $\mathbf{T_G}$ exists (Lemma 3),
(3) $\alpha q > 2\sqrt{n}$ for the hardness of LWE (Assumption 1),
(4) $\sigma > O(n\lceil \log q \rceil) \cdot \omega(\sqrt{\log 4m})$ for $\mathsf{SampleLeft}, \mathsf{SampleRight}$ (Lemma 4),
(5) $m > (n+1)\log q + \omega(\log n)$ for the leftover hash lemma (Lemma 6).

Let δ be a real such that $n^\delta > \lceil \log q \rceil = O(\log n)$, then we set

$$m = 6n^{1+\delta}, \quad \sigma = m \cdot \omega(\sqrt{\log n}), \quad q = m^{2.5} \cdot \omega(\sqrt{\log n}), \quad \alpha = [m^2 \cdot \omega(\sqrt{\log n})]^{-1}.$$

6 Our Second PAEKS Scheme

Now, we construct our second lattice-based PAEKS scheme that can be proven fully CI/TI secure under the multi-user setting in the random oracle model.

$\mathsf{Setup}(\lambda)$**:** On input a security parameter λ, this algorithm chooses a hash function $H : \{0,1\}^* \to \mathbb{Z}_q^{m\times m}$ modeled as the random oracle, sets parameters n, m, q, α, σ as specified in Sect.6.2, then outputs $PP = \{n, m, q, \alpha, \sigma, H\}$.

$\mathsf{KeyGen}_S(PP) \to (PK_S, SK_S)$**:** On input PP, the sender goes as follows.
1. Run $(\mathbf{A}, \mathbf{T_A}) \leftarrow \mathsf{TrapGen}(n, m, q)$.
2. Choose $\mathbf{U}_S \leftarrow \mathbb{Z}_q^{n\times n}$ and $\mathbf{D}_A, \mathbf{A}_w \leftarrow \mathbb{Z}_q^{n\times m}$.
3. Return $PK_S = (\mathbf{A}, \mathbf{U}_S, \mathbf{D}_A, \mathbf{A}_w)$ and $SK_S = \mathbf{T_A}$.

$\mathsf{KeyGen}_R(PP) \to (PK_R, SK_R)$**:** On input PP, the receiver goes as follows.
1. Run $(\mathbf{B}, \mathbf{T_B}) \leftarrow \mathsf{TrapGen}(n, m, q)$.
2. Choose $\mathbf{D}_B, \mathbf{B}_w \leftarrow \mathbb{Z}_q^{n\times m}$.
3. Return $PK_R = (\mathbf{B}, \mathbf{D}_B, \mathbf{B}_w)$ and $SK_R = \mathbf{T_B}$.

$\mathsf{PAEKS}(PK_R, PK_S, SK_S, ck) \to Ct$**:** On input PK_R, PK_S, SK_S and a ciphertext keyword ck, the sender picks $\mathbf{S}_S \leftarrow \bar{\Psi}_\alpha^{n\times l_S}$ and works as follows.
1. Compute $\mathbf{C}_w = \left[\mathbf{A}_w \cdot H(PK_S, PK_R, ck)^{-1}\right]^\top \mathbf{S}_S + \mathbf{E}_w$, $\mathbf{E}_w \leftarrow \bar{\Psi}_\alpha^{m\times l_S}$.
2. Compute $\mathbf{C}_a = \mathbf{A}^\top \mathbf{S}_S + \mathbf{R}_A^\top \mathbf{E}_w$, where $\mathbf{R}_A \leftarrow \{-1,1\}^{m\times m}$.
3. Compute $\mathbf{C}_b = \mathbf{B}^\top \mathbf{S}_S + \mathbf{R}_B^\top \mathbf{E}_w$, where $\mathbf{R}_B \leftarrow \{-1,1\}^{m\times m}$.
4. Compute $\mathbf{C}_d = \mathbf{D}_A^\top \mathbf{S}_S + \mathbf{R}_D^\top \mathbf{E}_w$, where $\mathbf{R}_D \leftarrow \{-1,1\}^{m\times m}$.
5. Compute $\mathbf{C}_u = \mathbf{U}_S^\top \mathbf{S}_S + \mathbf{E}_U$, where $\mathbf{E}_U \leftarrow \bar{\Psi}_\alpha^{n\times l_S}$.
6. Sample $\mathbf{E}_S \leftarrow \mathsf{SampleLeft}(\mathbf{A}, \mathbf{B}\|\mathbf{D}_B\|[\mathbf{B}_w \cdot H(PK_S, PK_R, ck)^{-1}], \mathbf{C}_u, \mathbf{T_A}, \sigma)$ such that $[\mathbf{A}\|\mathbf{B}\|\mathbf{D}_B\|\mathbf{B}_w \cdot H(PK_S, PK_R, ck)^{-1}] \cdot \mathbf{E}_S = \mathbf{C}_u$.
7. Return $Ct = (\mathbf{C}_a, \mathbf{C}_b, \mathbf{C}_d, \mathbf{C}_w, \mathbf{E}_S)$ as the ciphertext.

$\mathsf{Trapdoor}(PK_R, PK_S, SK_R, tk) \to Tr$**:** On input PK_R, PK_S, SK_R and a target keyword tk, the receiver picks $\mathbf{S}_R \leftarrow \bar{\Psi}_\alpha^{n\times l_R}$, then works as follows.
1. Compute $\mathbf{T}_w = \left[\mathbf{B}_w \cdot H(PK_S, PK_R, tk)^{-1}\right]^\top \mathbf{S}_R + \mathbf{E}'_w$, $\mathbf{E}'_w \leftarrow \bar{\Psi}_\alpha^{m\times l_R}$.
2. Compute $\mathbf{T}_a = \mathbf{A}^\top \mathbf{S}_R + \mathbf{R}_A'^\top \mathbf{E}'_w$, where $\mathbf{R}'_A \leftarrow \{-1,1\}^{m\times m}$.
3. Compute $\mathbf{T}_b = \mathbf{B}^\top \mathbf{S}_R + \mathbf{R}_B'^\top \mathbf{E}'_w$, where $\mathbf{R}'_B \leftarrow \{-1,1\}^{m\times m}$.
4. Compute $\mathbf{T}_d = \mathbf{D}_B^\top \mathbf{S}_R + \mathbf{R}_D'^\top \mathbf{E}'_w$, where $\mathbf{R}'_D \leftarrow \{-1,1\}^{m\times m}$.
5. Compute $\mathbf{T}_u = \mathbf{U}_S \mathbf{S}_R + \mathbf{E}'_U$, where $\mathbf{E}'_U \leftarrow \bar{\Psi}_\alpha^{n\times l_R}$.
6. Sample $\mathbf{E}_R \leftarrow \mathsf{SampleLeft}(\mathbf{B}, \mathbf{A}\|\mathbf{D}_A\|[\mathbf{A}_w \cdot H(PK_S, PK_R, tk)^{-1}], \mathbf{T}_u, \mathbf{T_B}, \sigma)$ such that $[\mathbf{B}\|\mathbf{A}\|\mathbf{D}_A\|\mathbf{A}_w \cdot H(PK_S, PK_R, tk)^{-1}] \cdot \mathbf{E}_R = \mathbf{T}_u$.
7. Return $Tr = (\mathbf{T}_a, \mathbf{T}_b, \mathbf{T}_d, \mathbf{T}_w, \mathbf{E}_R)$ as the trapdoor.

$\mathsf{Test}(Ct, Tr)$**:** Given Ct and Tr, the server computes $\mathbf{R} = \mathbf{E}_R^\top[\mathbf{C}_b; \mathbf{C}_a; \mathbf{C}_d; \mathbf{C}_w] - [\mathbf{T}_a^\top\|\mathbf{T}_b^\top\|\mathbf{T}_d^\top\|\mathbf{T}_w^\top]\mathbf{E}_S$ and checks whether $|\mathbf{R}[i,j]| < \lfloor q/4 \rfloor$ for each $i \in [1, l_R]$, $j \in [1, l_S]$. If so, it returns 1, otherwise 0.

Correctness. If $ck = tk$, then we have

$$\begin{aligned}
\mathbf{R} =& \mathbf{E}_R^\top[\mathbf{C}_b; \mathbf{C}_a; \mathbf{C}_d; \mathbf{C}_w] - [\mathbf{T}_a^\top\|\mathbf{T}_b^\top\|\mathbf{T}_d^\top\|\mathbf{T}_w^\top]\mathbf{E}_S \\
=& \mathbf{E}_R^\top \left[\mathbf{B}\|\mathbf{A}\|\mathbf{D}_A\|\mathbf{A}_w \cdot H(PK_S, PK_R, ck)^{-1}\right]^\top \mathbf{S}_S + \mathbf{E}_R^\top \left[\mathbf{R}_B\|\mathbf{R}_A\|\mathbf{R}_D\|\mathbf{I}\right]^\top \mathbf{E}_w \\
& - \mathbf{S}_R^\top \left[\mathbf{A}\|\mathbf{B}\|\mathbf{D}_B\|\mathbf{B}_w \cdot H(PK_S, PK_R, tk)^{-1}\right]\mathbf{E}_S - \mathbf{E}_w'^\top \left[\mathbf{R}'_A\|\mathbf{R}'_B\|\mathbf{R}'_D\|\mathbf{I}\right]\mathbf{E}_S \\
=& \mathbf{T}_u^\top \mathbf{S}_S - \mathbf{S}_R^\top \mathbf{C}_u + \mathbf{E}_R^\top \left[\mathbf{R}_B\|\mathbf{R}_A\|\mathbf{R}_D\|\mathbf{I}\right]^\top \mathbf{E}_w - \mathbf{E}_w'^\top \left[\mathbf{R}'_A\|\mathbf{R}'_B\|\mathbf{R}'_D\|\mathbf{I}\right]\mathbf{E}_S \\
=& \mathbf{S}_R^\top \mathbf{U}_S^\top \mathbf{S}_S - \mathbf{S}_R^\top \mathbf{U}_S^\top \mathbf{S}_S + \mathbf{error} = \mathbf{error},
\end{aligned}$$

$\mathbf{error} = \mathbf{E}_R^\top [\mathbf{R}_B \| \mathbf{R}_A \| \mathbf{R}_D \| \mathbf{I}]^\top \mathbf{E}_w - \mathbf{E}_w'^\top [\mathbf{R}_A' \| \mathbf{R}_B' \| \mathbf{R}_D' \| \mathbf{I}] \mathbf{E}_S + \mathbf{E}_U'^\top \mathbf{S}_S - \mathbf{S}_R^\top \mathbf{E}_U$. Similar to the analysis in Sect.5, we have $|\mathbf{error}[i,j]| \leq \sigma m q \alpha \omega(\log m) + O(\sigma m^{3/2})$.

6.1 Security proof

Intuitively, in the fully CI security proof, the simulator will run the algorithm $(\mathbf{D}_B, \mathbf{T}_{\mathbf{D}_B}) \leftarrow \mathsf{TrapGen}(n,m,q)$, and answers *any* ciphertext query as in Sect.5.1. For a trapdoor query of $tk \neq ck^*$, the simulator will run $(\mathbf{R}_{1,j}, \mathbf{T}_{\mathbf{A}_w \cdot \mathbf{R}_{1,j}^{-1}}) \leftarrow \mathsf{SampleRwithBasis}(\mathbf{A}_w, \sigma)$ and set $H(PK_S, PK_R, kw_j) = \mathbf{R}_{1,j}$. In this case, since $\mathbf{A}_w \cdot H(PK_S, PK_R, kw_j)^{-1} = \mathbf{A}_w \cdot \mathbf{R}_{1,j}^{-1}$, the simulator can use $\mathbf{T}_{\mathbf{A}_w \cdot \mathbf{R}_{1,j}^{-1}}$ to compute $\mathbf{E}_R$ and answer the trapdoor query. Therefore, our second scheme can be proven fully CI secure. The details are as follows.

Theorem 3. *If the decisional-LWE problem is hard, then our second PAEKS scheme is fully CI secure in multi-user setting under random oracle model.*

Proof. If there's a PPT adversary $\mathcal{A}$ that breaks the fully CI security of our second scheme with a non-negligible advantage, then we can construct a PPT algorithm $\mathcal{B}$ that solves the LWE problem with a non-negligible probability.

Setup: The adversary $\mathcal{A}$ chooses the challenge sender/receiver and send them to $\mathcal{B}$. $\mathcal{B}$ runs the Setup algorithm, returns PP to its challenger, who gives samples $(\mathbf{A}_0 \| \mathbf{U}, \mathbf{V}_0 \| \mathbf{V}) \in \mathbb{Z}_q^{n \times (m+n)} \times \mathbb{Z}_q^{(m+n) \times n}$ back to $\mathcal{B}$. Then $\mathcal{B}$ picks $\mathbf{B}_0 \leftarrow \mathbb{Z}_q^{n \times m}$, $\mathbf{R}_A^*, \mathbf{R}_B^*, \mathbf{R}_D^* \leftarrow \{-1,1\}^{m \times m}$, $\mathbf{R}_{A,w}^* \leftarrow \mathcal{D}^{m \times m}$, a hash function $H : \{0,1\}^* \to \mathbb{Z}_q^{m \times m}$, runs $(\mathbf{D}_B, \mathbf{T}_{D_B}) \leftarrow \mathsf{TrapGen}(n,m,q)$, sets $\mathbf{U}_S = \mathbf{U}$, $\mathbf{A} = \mathbf{A}_0 \cdot \mathbf{R}_A^*$, $\mathbf{D}_A = \mathbf{A}_0 \cdot \mathbf{R}_D^*$, $\mathbf{A}_w = \mathbf{A}_0 \mathbf{R}_{A,w}^*$, $\mathbf{B} = \mathbf{A}_0 \mathbf{R}_B^*$, $\mathbf{B}_w = \mathbf{B}_0 \mathbf{R}_{A,w}^*$, sends PP, $PK_S = (\mathbf{A}, \mathbf{U}_S, \mathbf{D}_A, \mathbf{A}_w)$, $PK_R = (\mathbf{B}, \mathbf{D}_B, \mathbf{B}_w)$ to $\mathcal{A}$.

Phase 1: $\mathcal{A}$ is allowed to submit polynomial queries to the hash oracle $\mathcal{O}_H$, the ciphertext oracle $\mathcal{O}_C$ and the trapdoor oracle $\mathcal{O}_T$. For simplicity, we assume that (1) $\mathcal{A}$ doesn't submit a query to $\mathcal{O}_H$ repeatedly; (2) before submitting a query $(\widetilde{PK}_R, kw_j)$ to $\mathcal{O}_C$ (resp. $(\widetilde{PK}_S, kw_j)$ to $\mathcal{O}_T$), $\mathcal{A}$ must have submit $(PK_S, \widetilde{PK}_R, kw_j)$ (resp. $(\widetilde{PK}_S, PK_R, kw_j)$ to $\mathcal{O}_H$. In response to $\mathcal{A}$'s queries, $\mathcal{B}$ does as follows:

$\mathcal{O}_H$**:** $\mathcal{B}$ maintains lists L_1 of tuples $(PK_S, PK_R, kw_j, \mathbf{R}_{1,j}, \mathbf{T}_{\mathbf{A}_w \cdot \mathbf{R}_{1,j}^{-1}})$ and L_2 of tuples $(\widetilde{PK}_S, \widetilde{PK}_R, kw_j, \mathbf{R}_{2,j}, \mathbf{T}_{\mathbf{A}_w \cdot \mathbf{R}_{2,j}^{-1}})$, which are initially empty. Assume $\mathcal{A}$ submits q_1 times of queries in form of (PK_S, PK_R, kw) to hash oracle $\mathcal{O}_H$. $\mathcal{B}$ randomly selects $j^* \in [1, q_1]$. For a $\mathcal{O}_H$ query of $(\widetilde{PK}_S, \widetilde{PK}_R, kw_j)$,

1. If $(\widetilde{PK}_S, \widetilde{PK}_R) = (PK_S, PK_R)$, then if this is the j^*'th query, $\mathcal{B}$ sets $H(PK_S, PK_R, kw_{j^*}) = \mathbf{R}_{A,w}^*$ and adds $(PK_S, PK_R, kw_{j^*}, \mathbf{R}_{A,w}^*, \perp)$ into L_1; else $\mathcal{B}$ runs $(\mathbf{R}_{1,j}, \mathbf{T}_{\mathbf{A}_w \cdot \mathbf{R}_{1,j}^{-1}}) \leftarrow \mathsf{SampleRwithBasis}(\mathbf{A}_w, \sigma)$, sets $H(PK_S, PK_R, kw_j) = \mathbf{R}_{1,j}$ and adds $(PK_S, PK_R, kw_j, \mathbf{R}_{1,j}, \mathbf{T}_{\mathbf{A}_w \cdot \mathbf{R}_{1,j}^{-1}})$ into L_1, returns $H(PK_S, PK_R, kw_j)$ to $\mathcal{A}$.

2. Else $(\widetilde{PK}_S, \widetilde{PK}_R) \neq (PK_S, PK_R)$, $\mathcal{B}$ runs algorithm $(\mathbf{R}_{2,j}, \mathbf{T}_{\widetilde{\mathbf{A}_w} \cdot \mathbf{R}_{2,j}^{-1}}) \leftarrow$ $\mathsf{SampleRwithBasis}(\widetilde{\mathbf{A}_w}, \sigma)$, sets $H(\widetilde{PK}_S, \widetilde{PK}_R, kw_j) = \mathbf{R}_{2,j}$, then adds $(\widetilde{PK}_S, \widetilde{PK}_R, kw_j, \mathbf{R}_{2,j}, \mathbf{T}_{\widetilde{\mathbf{A}_w} \cdot \mathbf{R}_{2,j}^{-1}})$ into L_2, sends $\mathbf{R}_{2,j}$ to $\mathcal{A}$.

$\mathcal{O}_C$**:** For $\mathcal{O}_C$ query of $(\widetilde{PK}_R, kw_j)$, $\mathcal{B}$ retrieves $H(PK_S, \widetilde{PK}_R, kw_j)$ from list $L_1 \cup L_2$, then computes $\mathbf{C}_w, \mathbf{C}_a, \mathbf{C}_b, \mathbf{C}_d, \mathbf{C}_u$ as in the PAEKS algorithm, and does as follows:
1. Sample $\bar{\mathbf{E}}_S \leftarrow \mathsf{SampleLeft}(\mathbf{D}_B, \mathbf{A}\|\mathbf{B}\|[\mathbf{B}_w + H(kw_j)\mathbf{G}], \mathbf{C}_u, \mathbf{T}_{\mathbf{D}_B}, \sigma)$ such that $\bar{\mathbf{E}}_S \in \mathbb{Z}^{4m \times l_S}$ and $[\mathbf{D}_B\|\mathbf{A}\|\mathbf{B}\|\mathbf{B}_w + H(kw_j)\mathbf{G}]\bar{\mathbf{E}}_S = \mathbf{C}_u$. Parse $\bar{\mathbf{E}}_S = (\mathbf{E}_{S,3}; \mathbf{E}_{S,1}; \mathbf{E}_{S,2}; \mathbf{E}_{S,4})$ and set $\mathbf{E}_S = (\mathbf{E}_{S,1}; \mathbf{E}_{S,2}; \mathbf{E}_{S,3}; \mathbf{E}_{S,4})$.
2. Return the ciphertext $Ct = (\mathbf{C}_a, \mathbf{C}_b, \mathbf{C}_d, \mathbf{C}_w, \mathbf{E}_S)$.

$\mathcal{O}_T$**:** For $\mathcal{O}_T$ query of $(\widetilde{PK}_S, kw_j)$, $\mathcal{B}$ firstly retrieves $H(\widetilde{PK}_S, PK_R, kw_j)$ and $\mathbf{T}_{\widetilde{\mathbf{A}_w} \cdot H(\widetilde{PK}_S, PK_R, kw_j)^{-1}}$ from list $L_1 \cup L_2$. If $H(\widetilde{PK}_S, PK_R, kw_j) = \mathbf{R}^*_{A,w}$, then $\mathcal{B}$ aborts; Else, it computes $\mathbf{T}_w, \mathbf{T}_a, \mathbf{T}_b, \mathbf{T}_d, \mathbf{T}_u$ as in the $\mathsf{Trapdoor}$ algorithm, then does as follows:
1. Sample $(\mathbf{E}_{R,1}; \mathbf{E}_{R,2}; \mathbf{E}_{R,3}) \leftarrow \mathcal{D}_{\mathbb{Z}^{3m \times l_R}, \sigma}$, then compute $\mathbf{T}'_u = \mathbf{T}_u - [\mathbf{B}\|\mathbf{A}\|\mathbf{D}_A] \cdot (\mathbf{E}_{R,1}; \mathbf{E}_{R,2}; \mathbf{E}_{R,3})$, then sample $\mathbf{E}_{R,4} \leftarrow \mathsf{SamplePre}(\widetilde{\mathbf{A}_w} \cdot H(\widetilde{PK}_S, PK_R, kw_j)^{-1}, \mathbf{T}'_u, \mathbf{T}_{\widetilde{\mathbf{A}_w} \cdot H(\widetilde{PK}_S, PK_R, kw_j)^{-1}}, \sigma)$ and set $\mathbf{E}_R = (\mathbf{E}_{R,1}; \mathbf{E}_{R,2}; \mathbf{E}_{R,3}; \mathbf{E}_{R,4})$.
2. Return trapdoor $Tr = (\mathbf{T}_a, \mathbf{T}_b, \mathbf{T}_d, \mathbf{T}_w, \mathbf{E}_R)$.

Challenge: $\mathcal{A}$ selects two challenge cipher-keywords ck_0^* and ck_1^* with the restriction that none of (PK_S, ck_0^*) and (PK_S, ck_1^*) have been queried on $\mathcal{O}_T$. $\mathcal{B}$ retrieves $H(PK_S, PK_R, ck_0^*)$ and $H(PK_S, PK_R, ck_1^*)$ from list L_1. Then, if $H(PK_S, PK_R, ck_i^*) \neq \mathbf{R}^*_{A,w}$ for $i \in \{0, 1\}$, $\mathcal{B}$ aborts. Otherwise, there is a ck_β^*, $\beta \in \{0, 1\}$, such that $H(PK_S, PK_R, ck_\beta^*) = \mathbf{R}^*_{A,w}$. $\mathcal{B}$ generates the challenge ciphertext corresponding with ck_β^* as follows:
1. Set $\mathbf{C}^*_w = \mathbf{V}_0$, compute $\mathbf{C}^*_a = (\mathbf{R}^*_A)^\top \mathbf{V}_0$, $\mathbf{C}^*_b = (\mathbf{R}^*_B)^\top \mathbf{V}_0$, $\mathbf{C}^*_d = (\mathbf{R}^*_D)^\top \mathbf{V}_0$.
2. Set $\mathbf{C}^*_u = \mathbf{V}$.
3. Sample $(\mathbf{E}^*_{S,1}; \mathbf{E}^*_{S,2}; \mathbf{E}^*_{S,4}) \leftarrow \mathcal{D}_{\mathbb{Z}^{3m \times l_S}, \sigma}$, then compute $\mathbf{V}' = \mathbf{V} - [\mathbf{A}\|\mathbf{B}\|\mathbf{B}_0] \cdot (\mathbf{E}^*_{S,1}; \mathbf{E}^*_{S,2}; \mathbf{E}^*_{S,4})$, sample $\mathbf{E}^*_{S,3} \leftarrow \mathsf{SamplePre}(\mathbf{D}_B, \mathbf{V}', \mathbf{T}_{\mathbf{D}_B}, \sigma)$, then set $\mathbf{E}^*_S = (\mathbf{E}^*_{S,1}; \mathbf{E}^*_{S,2}; \mathbf{E}^*_{S,3}; \mathbf{E}^*_{S,4})$.
4. Return the challenge ciphertext $Ct^* = (\mathbf{C}^*_a, \mathbf{C}^*_b, \mathbf{C}^*_d, \mathbf{C}^*_w, \mathbf{E}^*_S)$.

Phase2: This phase is the same as Phase 1 with the same restriction defined in the challenge phase.

Output: $\mathcal{A}$ outputs a guess bit β' of β. If $\mathcal{A}$'s guess is correct, then $\mathcal{B}$ claims that he is given the real LWE samples; Otherwise, random samples.

Since $\mathbf{A}_0, \mathbf{B}_0$ are uniform, $\mathbf{R}^*_{A,w} \leftarrow \mathcal{D}^{m \times m}$, by Lemma 6, $\mathbf{A}_w, \mathbf{B}_w$ are statistically close to uniform. By Lemma 3, the distribution of $\mathbf{D}_B$ is statistical to that in the real game. Since $\mathbf{R}^*_A, \mathbf{R}^*_B, \mathbf{R}^*_D \leftarrow \{-1, 1\}^{m \times m}$, by Lemma 6, $\mathbf{A}, \mathbf{B}, \mathbf{D}_A$ are statistically close to uniform even given $\mathbf{R}^{*\top}_A \mathbf{E}_w, \mathbf{R}^{*\top}_B \mathbf{E}_w, \mathbf{R}^{*\top}_D \mathbf{E}_w$. Therefore, the distribution of public keys in the simulation is statistical to that in the real game.

According to Lemma 1, 4 and 5, the output distributions of the ciphertext oracle $\mathcal{O}_C$ and trapdoor oracle $\mathcal{O}_T$ are statistical to those in the real game.

If $(\mathbf{A}_0 \| \mathbf{U}, \mathbf{V}_0 \| \mathbf{V})$ are real LWE samples, then $\mathbf{C}_w^* = \mathbf{V}_0 = \mathbf{A}_0^\top \mathbf{S}_S + \mathbf{E}_w = (\mathbf{A}_w \cdot H(PK_S, PK_R, ck_\beta^*)^{-1})^\top \mathbf{S}_S + \mathbf{E}_w$, $\mathbf{C}_u^* = \mathbf{V} = \mathbf{U}^\top \mathbf{S}_S + \mathbf{E}_U = \mathbf{U}_S^\top \mathbf{S}_S + \mathbf{E}_U$. Along the way, it is easy to check that $\mathbf{C}_a^*, \mathbf{C}_b^*, \mathbf{C}_d^*$ are the same as $\mathcal{A}$'s view in the real game. By Lemma 4, the distribution of $\mathbf{E}_S$ is the same as $\mathcal{A}$'s view in the real game. Else if $(\mathbf{A}_0 \| \mathbf{U}, \mathbf{V}_0 \| \mathbf{V})$ are random, then $\mathbf{C}_w^*, \mathbf{C}_u^*$ are uniform. Since $\mathbf{E}_S$ is distributed statistically close to$\mathcal{D}_{\Lambda_q^{\mathbf{C}_u^*}([\mathbf{A}\|\mathbf{B}\|\mathbf{D}_B\|\mathbf{B}_0]),\sigma}$, $\mathcal{A}$'s probability in guessing β successfully is 1/2. Hence, if $\mathcal{A}$ breaks the fully-CI security with a non-negligible advantage ϵ, then $\mathcal{B}$ can solve the decisional-LWE problem with advantage $\epsilon/2q_1$. □

Theorem 4. *If decisional-LWE problem is hard, then our second PAEKS scheme is fully TI secure in multi-user setting under random oracle model.*

Proof. This proof is similar to that of Theorem 3, since the constructions of algorithms PAEKS and Trapdoor are almost symmetric.

6.2 Parameter Selection of Our Second PAEKS Scheme

Now, to satisfy the correctness and make the security proof work, we need that

– $\sigma m q \alpha \omega(\log m) + O(\sigma m^{3/2}) \leq \lfloor q/4 \rfloor$ for correctness,
– $m \geq 6n\lceil \log q \rceil$ to ensure TrapGen works (Lemma 3),
– $\alpha q > 2\sqrt{n}$ for the hardness of LWE (Assumption 1),
– $\sigma > O(n\lceil \log q \rceil)\omega(\sqrt{\log 4m})$ for SampleLeft, SamplePre, security (Lemma 4,5),
– $m > 2n \log q + \omega(\log n)$ for the leftover hash lemma (Lemma 6).

Let δ be a real such that $n^\delta > \lceil \log q \rceil = O(\log n)$, then we set $m = 6n^{1+\delta}$, $\sigma = m \cdot \omega(\sqrt{\log n})$, $q = m^{2.5} \cdot \omega(\sqrt{\log n})$, $\alpha = [m^2 \cdot \omega(\sqrt{\log n})]^{-1}$.

7 Comparison

In this section, we compare our lattice-based PAEKS schemes with other PAEKS schemes in terms of security properties, parameters and communication cost.

In Table 1, we discuss the security properties of various PAEKS schemes. Among them, PAEKS schemes [15, 20, 22, 23] cannot resist quantum attacks; [17] relies on the trusted Setup assumption and fails to provide convincing evidence for their MTI security and MCI security, while [13] relies on the designated-receiver setting, as shown in Sect.4. Our first PAEKS scheme can be proven selectively fully CI and selectively fully TI secure in the one-user setting, and our second scheme can be proven fully CI and fully TI secure in the multi-user setting. In a nutshell, our schemes provide stronger security guarantee for ciphertext privacy and trapdoor privacy than previous PAEKS schemes.

In Table 2, we compare the parameter selection for the LWE-based PAEKS instantiation in [17] with those in our schemes[2]. Note that LTT+22 [17]'s instantiation needs to satisfy an extra parameter constraint $q \geq 2^{\hat{n}} \cdot O(m^{1.5} n \cdot$

[2] We omit the comparison with Eumra22 [13] because it does not provide a concrete lattice-based instantiation.

Table 1. Comparison of security properties for various PAEKS schemes.

Schemes	CI	TI	MCI	MTI	Fully CI	Fully TI	Multi-user setting	Neither trusted Setup nor designated-user setting	QR
HL17 [15]	$\surd$	$\surd$	$\times$	$\times$	$\times$	$\times$	$\times$	$\surd$	$\times$
NE19 [20]	$\surd$	$\surd$	$\times$	$\times$	$\times$	$\times$	$\surd$	$\surd$	$\times$
QCH+20 [22]	$\surd$	$\surd$	$\surd$	$\times$	$\times$	$\times$	$\times$	$\surd$	$\times$
QCZ+21 [23]	$\surd$	$\surd$	$\surd$	$\times$	$\surd$	$\times$	$\surd$	$\surd$	$\times$
LTT+22 [17]	$\surd$	$\surd$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\surd$
Eumra22 [13]	$\surd$	$\surd$	$\surd$	$\times$	$\surd$	$\times$	$\surd$	$\times$	$\surd$
Ours1	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\times$	$\surd$	$\surd$
Ours2	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$

Note that "Ours1" is proved in the selective version as described in Sect.3.2. QR: quantum resistance.

Table 2. Comparison of parameters selection of lattice-based PAEKS schemes

Schemes	m	σ	α	modulus q
LTT+22 [17]	$6n^{1+\delta}$	$m \cdot l \cdot \omega(\sqrt{\log n})$	$[2^{\hat{n}} \cdot m^2 \cdot \omega(\sqrt{\log n})]^{-1}$	$2^{\hat{n}} \cdot m^{2.5} \cdot \omega(\sqrt{\log n})$
Ours1	$6n^{1+\delta}$	$m \cdot \omega(\sqrt{\log n})$	$[m^2 \cdot \omega(\sqrt{\log n})]^{-1}$	$m^{2.5} \cdot \omega(\sqrt{\log n})$
Ours2	$6n^{1+\delta}$	$m \cdot \omega(\sqrt{\log n})$	$[m^2 \cdot \omega(\sqrt{\log n})]^{-1}$	$m^{2.5} \cdot \omega(\sqrt{\log n})$

δ is a real such that $n^{\delta} > \lceil \log q \rceil$; l is the output length of a secure hash function; $2^{-\hat{n}}$ is the failure probability to compute the shared key as described in Sect.4.

Table 3. Comparison of communication costs of lattice-based PAEKS schemes

Schemes	PK_S Size	PK_R Size	Ct Size	Tr Size
LTT+22 [17]	$(n + \kappa m) \log q$	$(2n + \kappa m + (l+2)nm) \log q$	$\rho((1+2m) \log q + 1)$	$2m \log q$
Ours1	$(3mn + n^2) \log q$	$3mn \log q$	$8ml_s \log q$	$8ml_r \log q$
Ours2	$(3mn + n^2) \log q$	$3mn \log q$	$8ml_s \log q$	$8ml_r \log q$

The underlying lattice-based PEKS [5] used by [17] provides ρ-bit security. κ is the length of the shared key generated by the SPHF [6] as described in Sect.4.

$\omega(\sqrt{\log n})$) (see Equ.(2) in Sect.4), while our schemes don't require this, so the modulus q in our schemes could be of polynomial size. By contrast, the modulus q in our schemes is polynomial. Specifically, in [17], set $n = 256$, $\hat{n} = 128$ and $l = 256$, then $m \approx 2^{18}$, $\sigma \approx 2^{26}$, $\alpha \approx 2^{-164}$, $q \approx 2^{173}$. In our schemes, set $n = 256$, then $m \approx 2^{16}$, $\sigma \approx 2^{16}$, $\alpha \approx 2^{-35}$, $q \approx 2^{43}$.

In Table 3, we compare the communication cost of our schemes and the LWE-based PAEKS instantiation in [17]. For [17], set $n = 256$, $m = 2^{18}$, $q = 2^{173}$, $\rho = 128$, $\kappa = 128$, $l = 256$, then the size of public key of sender/receiver is about 708614(Kb)/366350347(Kb), and the size of ciphertext/trapdoor is about 1417219(Kb)/11072(Kb). For our schemes, set $n = 256$, $m = 2^{16}$, $q = 2^{43}$, $l_s = 32$, $l_r = 4$, then the size of the public key of sender/receiver is about 264536(Kb)/264192(Kb), and the size of ciphertext/trapdoor is about 88064(Kb)/11008(Kb). Hence, the communication overhead of our schemes is much smaller than that of [17].

8 Conclusion

In this paper, we propose two lattice-based PAEKS schemes, with totally different construction methodology from previous lattice-based PAEKS schemes. Our first scheme satisfies selectively fully CI/TI security in one-user setting under the standard model. Our second scheme can be proven fully CI/TI secure in multi-user setting under the random oracle model. Compared with the existing lattice-based PAEKS schemes, our schemes not only provide stronger security guarantee for ciphertext privacy and trapdoor privacy, but also greatly reduce the communication overhead.

Acknowledgments. This work is supported by the National Key Research and Development Program of China (Grant No.2021YFB3100200).

References

1. Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In *EUROCRYPT 2010*, pages 553–572, 2010.
2. Miklós Ajtai. Generating hard instances of the short basis problem. In *ICALP'99*, pages 1–9, 1999.
3. Joël Alwen and Chris Peikert. Generating shorter bases for hard random lattices. *Theory Comput. Syst.*, 48(3):535–553, 2011.
4. Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *CRYPTO 2009*, volume 5677, pages 595–618. Springer, 2009.
5. Rouzbeh Behnia, Muslum Ozgur Ozmen, and Attila Altay Yavuz. Lattice-based public key searchable encryption from experimental perspectives. *IEEE Transactions on Dependable and Secure Computing*, 17(6):1269–1282, 2020.
6. Fabrice Benhamouda, Olivier Blazy, Léo Ducas, and Willy Quach. Hash proof systems over lattices revisited. In *PKC 2018*, volume 10770, pages 644–674. Springer, 2018.
7. Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In *EUROCRYPT 2004*, pages 506–522, 2004.
8. Jin Wook Byun, Hyun Suk Rhee, Hyun-A Park, and Dong Hoon Lee. Off-line keyword guessing attacks on recent keyword search schemes over encrypted data. In *SDM 2006*, pages 75–83.
9. David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. *J. Cryptology*, 25(4):601–639, 2012.
10. Leixiao Cheng and Fei Meng. Security analysis of pan et al.'s "public-key authenticated encryption with keyword search achieving both multi-ciphertext and multi-trapdoor indistinguishability". *J. Syst. Archit.*, 119:102248, 2021.
11. Ronald Cramer and Ivan Damgård. On the amortized complexity of zero-knowledge protocols. In *CRYPTO 2009*, pages 177–191, 2009.
12. Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *Advances in Cryptology — EUROCRYPT 2002*, pages 45–64. Springer Berlin Heidelberg, 2002.

13. Keita Emura. Generic construction of public-key authenticated encryption with keyword search revisited: Stronger security and efficient construction. In *APKC '22*, pages 39–49. ACM, 2022.
14. Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC 2008*, pages 197–206, 2008.
15. Qiong Huang and Hongbo Li. An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks. *Inf. Sci.*, 403:1–14, 2017.
16. Zi-Yuan Liu, Yi-Fan Tseng, Raylin Tso, Masahiro Mambo, and Yu-Chi Chen. Public-Key Authenticated Encryption with Keyword Search: A Generic Construction and Its Quantum-Resistant Instantiation. *The Computer Journal*, 2021.
17. Zi-Yuan Liu, Yi-Fan Tseng, Raylin Tso, Masahiro Mambo, and Yu-Chi Chen. Public-key authenticated encryption with keyword search: Cryptanalysis, enhanced security, and quantum-resistant instantiation. In *ASIA CCS '22*, pages 423–436. ACM, 2022.
18. Daniele Micciancio and Shafi Goldwasser. *Complexity of lattice problems - a cryptograhic perspective*, volume 671. Springer, 2002.
19. Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT 2012*, pages 700–718, 2012.
20. Mahnaz Noroozi and Ziba Eslami. Public key authenticated encryption with keyword search: revisited. *IET Inf. Secur.*, 13(4):336–342, 2019.
21. Xiangyu Pan and Fagen Li. Public-key authenticated encryption with keyword search achieving both multi-ciphertext and multi-trapdoor indistinguishability. *J. Syst. Archit.*, 115:102075, 2021.
22. Baodong Qin, Yu Chen, Qiong Huang, Ximeng Liu, and Dong Zheng. Public-key authenticated encryption with keyword search revisited: Security model and constructions. *Inf. Sci.*, 516:515–528, 2020.
23. Baodong Qin, Hui Cui, Xiaokun Zheng, and Dong Zheng. Improved security model for public-key authenticated encryption with keyword search. In *ProvSec*, volume 13059, pages 19–38. Springer, 2021.
24. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6):34:1–34:40, 2009.
25. Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. pages 124–134. IEEE Computer Society, 1994.
26. Chang Yan-Cheng and Mitzenmacher Michael. Privacy preserving keyword searches on remote encrypted data. pages 442–455, 2005.
27. Xiaojun Zhang, Yao Tang, Huaxiong Wang, Chunxiang Xu, Yinbin Miao, and Hang Cheng. Lattice-based proxy-oriented identity-based encryption with keyword search for cloud storage. *Inf. Sci.*, 494:193–207, 2019.
28. Xiaojun Zhang, Chunxiang Xu, Huaxiong Wang, Yuan Zhang, and Shixiong Wang. FS-PEKS: lattice-based forward secure public-key encryption with keyword search for cloud-assisted industrial internet of things. *IEEE Trans. Dependable Secur. Comput.*, 18(3):1019–1032, 2021.