Lattice-Based Public Key Encryption with Keyword Search

Rouzbeh Behnia[*] Muslum Ozgur Ozmen[*] Attila A. Yavuz[*]

Abstract

Public key Encryption with Keyword Search (PEKS) aims in mitigating the impacts of data privacy versus utilization dilemma by allowing *any user in the system* to send encrypted files to the server to be searched by a receiver. The receiver can retrieve the encrypted files containing specific keywords by providing the corresponding trapdoors of these keywords to the server. Despite their merits, the existing PEKS schemes introduce a high end-to-end delay that may hinder their adoption in practice. Moreover, they do not scale well for large security parameters and provide no post-quantum security promise. In this paper, we propose novel lattice-based PEKS schemes that offer a high computational efficiency along with better security assurances than that of existing alternatives. Specifically, our NTRU-PEKS scheme achieves 18 times lower end-to-end delay than the most efficient pairing-based alternatives. Our LWE-PEKS offers provable security in the standard model with a reduction to the worst-case lattice problems. We fully implemented our NTRU-PEKS on embedded devices with a deployment on real cloud infrastructures to demonstrate its effectiveness.

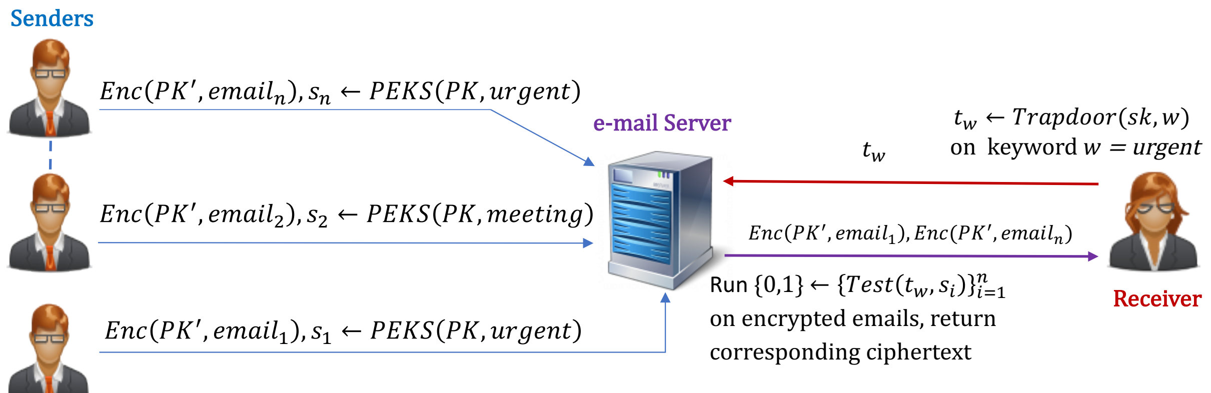
Keywords: applied cryptography, Public Key Encryption with Keyword Search (PEKS), lattice-based cryptography, internet of things

1 Introduction

Cloud computing has significantly impacted the computing infrastructure and enabled a large pool of applications. For example, data outsourcing [1] permits small/medium size businesses to increase data availability by minimizing the management and maintenance costs. Data outsourcing, despite its merits, raises significant data privacy concerns for clients. Traditional encryption techniques can be used to overcome such privacy concerns. However, standard encryption does not permit search capabilities on the encrypted data. Therefore, a significant amount of research is focused on Searchable Encryption (SE) technologies that can be used to efficiently address this problem. There are two main branches of SE where each is tailored for a distinct set of application.

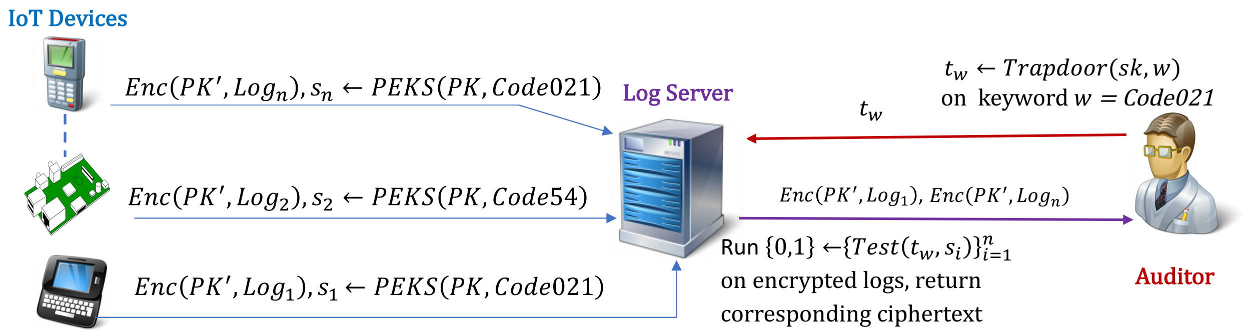
Dynamic Searchable Symmetric Encryption (DSSE) (e.g., [2, 3, 4]) provides search capabilities on encrypted data for private data outsourcing applications (e.g., data storage on the cloud), in which the client uses *her own private key* to encrypt and then search on *her own data* over the cloud. Public Key Encryption with Keyword Search (PEKS) schemes [5] allow *any* client to encrypt data with specified keywords under the public key of a *designed receiver*. The designed receiver, Alice, can then use her private key to generate and send *trapdoors* for her desired keywords, and enable the server to search on the encrypted data to retrieve the files that are

[*]Oregon State University, {behniar,ozmenmu,attila.yavuz}@oregonstate.edu



Senders
$Enc(PK', email_n), s_n \leftarrow PEKS(PK, urgent)$
e-mail Server
$t_w \leftarrow Trapdoor(sk, w)$
on keyword $w = urgent$
t_w
$Enc(PK', email_2), s_2 \leftarrow PEKS(PK, meeting)$
$Enc(PK', email_1), Enc(PK', email_n)$
Run $\{0,1\} \leftarrow \{Test(t_w, s_i)\}_{i=1}^{n}$
on encrypted emails, return
corresponding ciphertext
Receiver
$Enc(PK', email_1), s_1 \leftarrow PEKS(PK, urgent)$

Figure. 1-a PEKS scheme in secure email system



IoT Devices
$Enc(PK', Log_n), s_n \leftarrow PEKS(PK, Code021)$
Log Server
$t_w \leftarrow Trapdoor(sk, w)$
on keyword $w = Code021$
t_w
$Enc(PK', Log_2), s_2 \leftarrow PEKS(PK, Code54)$
$Enc(PK', Log_1), Enc(PK', Log_n)$
Run $\{0,1\} \leftarrow \{Test(t_w, s_i)\}_{i=1}^{n}$
on encrypted logs, return
corresponding ciphertext
Auditor
$Enc(PK', Log_1), s_1 \leftarrow PEKS(PK, Code021)$

Figure. 1-b PEKS scheme in privacy-preserving audit logging system

Figure 1: Potential applications of PEKS schemes

associated with the keyword. PEKS is well suited for distributed applications (e.g., e-mail, audit logging for Internet of Things, etc.) where a large number of users/entities generate encrypted data to be retrieved by a receiver. *The focus of this paper is on PEKS schemes.*

In Figure 1, we depict two system models as the potential applications of PEKS schemes. Figure 1-a illustrates an application of PEKS schemes which is considered as the main motive for the initial proposal of PEKS schemes in [5]. Alice, who is assumed to have a number of devices (e.g., cellphone, desktop), wants her e-mails to be routed to her devices based on the keywords associated to them. For instance, when the sender, Bob, sends her an e-mail with keyword "*urgent*", the e-mail should be routed to her cellphone. To achieve this, after encrypting the e-mail content with a conventional public key encryption scheme, Bob uses a PEKS scheme to encrypt the keyword "*urgent*" and sends it together with the encrypted e-mail to the e-mail server. Alice can then use her private key to generate the trapdoor corresponding to keyword "$w = urgent$" and ask the server to retrieve all the e-mails associated with w.

Another important application of PEKS schemes for storing private log files is illustrated in Figure 1-b. For Internet of Things (IoT) applications, PEKS schemes can enable a heterogeneous set of devices to send their encrypted log files concatenated with a searchable ciphertext of distinct keywords to an untrusted storage server. To analyze the log files, an auditor can use his private key to generate trapdoors and enable the server to search and return the files that are associated with the target keyword.

1.1 Research Gap

Since their introduction in [5], several PEKS schemes with a variety of features have been proposed (e.g., [6, 7, 8, 9]). However, the wide adoption of PEKS schemes in practice has been hindered due to a number of obstacles:

• High End-to-End Delay: The most computationally expensive part of the PEKS is generally the search phase, which requires the execution of a *test algorithm* for each keyword in the database during the search process. The existing PEKS schemes (e.g., [5, 7]) introduce a significant end-to-end computation delay due to their reliance on heavy pairing computations, which require at least one pairing operation per item, in total linear number of pairing calls with respect to the number of keyword-file pairs in the database, for each search query. Therefore, providing a computational efficient test algorithm is a critical requirement to minimize the end-to-end delay.

• Lack of Long-term Security: It is a highly desirable property for data storage applications to offer long-term data security measures. However, achieving a high level of security for an extended duration of time requires continuous increment in key sizes, which results in a substantial increase in computation overhead for conventional cryptographic primitives (e.g., ECC, RSA). Furthermore, the emergence of quantum computers will render most of the conventional asymmetric cryptography primitives unsafe, and therefore, there is a great merit in devising PEKS schemes that can provide a post-quantum security promise.

• Lack of Full-Fledged Implementations: While a number of DSSE schemes have been fully implemented on real data (and are publicly accessible, e.g., [10, 11]), to the best of our knowledge, no full-fledged implementation (with a real dataset) of PEKS schemes is publicly available to this date. Hence, there is a need of providing a full-fledge implementation of PEKS schemes on actual cloud platforms to measure important performance factors (e.g., communication delay, disk access time) that cannot be precisely captured with mere "cost estimations".

1.2 Our Contribution

Towards addressing the aforementioned research gaps, we developed two lattice-based PEKS schemes with a post-quantum security promise and presented a full-fledged implementation of our efficient lattice-based PEKS scheme and its pairing-based counterpart. We outline our contributions as follows:

• New Lattice-Based PEKS Schemes: In the initial proposal of PEKS in [5], Boneh et al. showed how to derive a PEKS scheme from an Identity-Based Encryption (IBE). Abdalla et al. [12] specified the requirements for the underlying IBE scheme to ensure the security and correctness of the derived PEKS. In this paper, we propose two PEKS schemes based on lattices-based tools. (i) Our first scheme is referred to as NTRU-PEKS, which harnesses Ducas et al.'s IBE scheme [13], and it meets the all requirements provided by [12] to ensure the provable security in Random Oracle Model (ROM). (ii) Our second scheme is referred to as LWE-PEKS, which leverages IBE constructions in [14, 15] and [16] to offer the first provable secure lattice-based PEKS in the standard model (to the best of our knowledge). We prove the security and consistency of our PEKS schemes and suggest parameter sizes to avoid potential consistency errors (with an overwhelming probability).

• High Computational Efficiency: Our NTRU-PEKS scheme offers significant computational efficiency advantages over the existing PEKS schemes. This is achieved by harnessing the latest efforts in improving the efficiency of the lattice-based schemes, ring-LWE [17] and fast arithmetic operations (e.g., Fast Fourier Transform) over polynomial rings $\mathbb{Z}[x]/(x^N + 1)$. As it is shown in Table 1, our scheme has significantly more efficient Test and PEKS algorithms than those in [5, 18], which are currently considered as the most efficient PEKS alternatives [19]. The efficiency of Test algorithm is of vital importance, since it is executed by the server linear with the total number keyword-file pairs in the database. The efficiency of the PEKS algorithm facilitates the implementation of PEKS schemes on battery-limited IoT devices. Due to its computational efficiency, despite having larger ciphertext sizes, we showed with experiments on actual cloud deployments that, NTRU-PEKS achieves a superior end-to-end response time compared to its counterparts (see Section 4).

• Long-term Security: Both of our constructions are based on lattice-based tools that provide long-term security and are currently considered secure against quantum computers. It is worth noting that lattice-based schemes also have a substantially smoother performance response to increased key sizes compared to conventional cryptographic primitives (e.g., ECC, RSA).

• Full-Fledged Implementation on Cloud Infrastructure: We provide a full-fledged implementation of our NTRU-PEKS scheme and its most efficient pairing-based counterpart on a real cloud computing infrastructure with Enron e-mail dataset. We chose Amazon Web Server (AWS) as the server in our system, a commodity hardware and an ARM Cortex-A53 as the client machines. One can easily see the importance of a full-fledged implementation when comparing the benchmark provided from the simulation results in Table 5 and the benchmark results of the full implementation in Section 4. Detailed experimental results are further explained in Section 4. We also open-sourced our implementations for public testing and wide adoption (please see Section 4).

Differences Between this Article and its Preliminary Version in [20]: In this article, we put forth a new lattice-based PEKS scheme in the standard model and present the first full-fledged implementation of our NTRU-PEKS scheme along with its counterparts on actual cloud environments: (i) We introduced the first LWE-based PEKS scheme in the standard model (LWE-PEKS) with a security reduction to the worst-case lattice-based problems. (ii) We presented full-fledged implementations of the NTRU-PEKS scheme and its most efficient counterpart [5] (with Enron e-mail dataset) on commodity hardware, AWS and ARM Cortex-A53 (as an embedded device). (iii) Based on our full-fledged implementations, we provided a more precise performance analysis of our NTRU-PEKS scheme and its counterpart in [5]. We also highlighted differences between the results of the simulated implementation and our full-fledged implementation.

2 Preliminaries

In this section, we provide definitions and notations that are used by our schemes. For the sake of compliance, we try to use similar notation as in [13] and [14].

Notations. $a \xleftarrow{\$} \mathcal{X}$ denotes that a is randomly selected from distribution $\mathcal{X}$. H_i for $i \in \{1, \ldots, n\}$ denotes a hash function which is perceived to behave as a random oracle in this paper. $\mathcal{A}^{\mathcal{O}_1,\ldots,\mathcal{O}_n}(.)$ denotes algorithm $\mathcal{A}$ that is provided with access to oracles $\mathcal{O}_1, \ldots, \mathcal{O}_n$. The norm of a vector $\mathbf{v}$ is denoted by $\|\mathbf{v}\|$. $\lceil x \rfloor$ rounds x to the closest integer. $x \triangleq y$ means x is defined as y. The function $\gcd(x, y)$ returns the greatest common divisor of values x and y.

2.1 Integer Lattices

Let $\mathbf{B} = [\mathbf{b_1}|\ldots|\mathbf{b_m}] \in \mathbb{R}^{m\times m}$ be an $m \times m$ matrix whose columns are linearly independent vectors $\mathbf{b_1}, \ldots, \mathbf{b_m} \in \mathbb{R}^m$. The m-dimensional full-rank lattice Λ generated by B is the set,

$$\Lambda = \mathcal{L}(\mathbf{B}) = \left\{ \mathbf{y} \in \mathbb{R}^m : \exists s \in \mathbb{Z}^m, \mathbf{y} = \mathbf{Bs} = \sum_{i=1}^{m} \mathbf{s_i b_i} \right\}$$

Definition 1. *For a prime q, $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and $\mathbf{u} \in \mathbb{Z}_q^n$, define:*
$\Lambda_q(\mathbf{A}) := \{\mathbf{e} \in \mathbb{Z}^m : \exists \mathbf{s} \in \mathbb{Z}_q^n$ *where* $\mathbf{A}^\top \mathbf{s} = \mathbf{e} \mod q\}$
$\Lambda_q^\top(\mathbf{A}) := \{\mathbf{e} \in \mathbb{Z}^m : \mathbf{Ae} = 0 \mod q\}$
$\Lambda_q^u(\mathbf{A}) := \{\mathbf{e} \in \mathbb{Z}^m : \mathbf{Ae} = \mathbf{u} \mod q\}$

2.2 NTRU Lattices

Ajtai [21] introduced the Short Integer Solution (SIS) problem and demonstrated the connection between average-case SIS problem and worst-case problems over lattices. Hoffstein et al. [22] proposed a very efficient public key encryption scheme based on NTRU lattices. Regev [17] introduced the Learning with Error (LWE) problem. The SIS and LWE problems have been used as the building blocks of many lattice-based schemes.

NTRU encryption works over rings of polynomials $\mathcal{R} \triangleq \mathbb{Z}[x]/(x^N+1)$ and $\mathcal{R}' \triangleq \mathbb{Q}[x]/(x^N+1)$ which are parametrized with N as a power-of-two integer. (x^N+1) is irreducible, therefore, $\mathcal{R}'$ is a cyclotomic field. For $f = \sum_{i=0}^{N-1} f_i x^i$ and $g = \sum_{i=0}^{N-1} g_i x^i$ as polynomials in $\mathbb{Q}[x]$, fg denotes polynomial multiplication in $\mathbb{Q}[x]$ while $f * g \triangleq fg \mod (x^N+1)$ is referred to as convolution product. For an N-dimension anti-circulant matrix $\mathcal{A}_N$ we have $\mathcal{A}_N(f) + \mathcal{A}_N(g) = \mathcal{A}_N(f+g)$, and $\mathcal{A}_N(f) \times \mathcal{A}_N(g) = (f * g)$.

Definition 2. *For prime integer q and $f, g \in \mathcal{R}$, $h = g * f^{-1} \mod q$, the NTRU lattice with h and q is $\Lambda_{h,q} = \{(u,v) \in \mathcal{R}^2 : u + v * h = 0 \mod q\}$. $\Lambda_{h,q}$ is a full-rank lattice generated by $\mathcal{A}_{h,q} = \begin{pmatrix} \mathcal{A}_N(h) & \mathbf{I}_N \\ q\mathbf{I}_N & 0_N \end{pmatrix}$, where $\mathbf{I}$ is an identity matrix.*

Note that one can generate this basis using a single polynomial $h \in \mathcal{R}_q$. However, the lattice generated from $\mathcal{A}_{h,q}$ has a large orthogonal defect which results in inefficiency of standard lattice operations. As proposed by [23], another basis (which is much more orthogonal) can be efficiently [13] generated by selecting $F, G \in \mathcal{R}$ and computing $f * G - g * F = q$. The new base $\mathbf{B}_{f,g} = \begin{pmatrix} \mathcal{A}(g) & -\mathcal{A}(f) \\ \mathcal{A}(G) & -\mathcal{A}(F) \end{pmatrix}$ generates the same lattice $\Lambda_{h,q}$.

2.3 Tools and Definitions

Definition 3. *(Gram-Schmidt norm [24]) Given $\mathbf{B} = (\mathbf{b}_i)_{i\in I}$ as a finite basis and $\tilde{\mathbf{B}} = (\tilde{\mathbf{b}_i})_{i\in I}$ as its Gram-Schmidt orthogonalization, the Gram-Schmidt norm of $\mathbf{B}$ is $\left\|\tilde{\mathbf{B}}\right\| = \max_{i\in I} \|\mathbf{b}_i\|$.*

Definition 4. *(Statistical Distance [14]) Given two random variables X and Y taking values in a finite set $\mathcal{S}$, the statistical distance is defined as:*

$$\Delta(X,Y) = \frac{1}{2}\sum_{s\in\mathcal{S}} |\Pr[X=s] - \Pr[Y=s]|$$

X is said to be $\delta-$uniform over $\mathcal{S}$ if $\Delta(X,Y) \leq \delta$.

Using Gaussian sampling, Gentry et al. [24] proposed a technique to use a short basis as trapdoor without disclosing any information about the short basis and prevent attacks similar as in [25].

Definition 5. *An N-dimensional Gaussian function $\rho_{\sigma,c} : \mathbb{R} \rightarrow (0,1])$ is defined as $\rho_{\sigma,c}(x) \triangleq \exp(-\frac{\|x-c\|^2}{2\sigma^2})$. Given a lattice $\Lambda \subset \mathbb{R}^n$, the discrete Gaussian distribution over Λ is $D_{\Lambda,s,c}(\mathbf{x}) = \frac{\rho_{\sigma,c}(\mathbf{x})}{\rho_{\sigma,c}(\Lambda)}$ for all $\mathbf{x} \in \Lambda$.*

If we pick a noise vector over a Gaussian distribution with the radius not smaller than the *smoothing parameter* [26], and reduce the vector to the fundamental parallelepiped of our lattice, the resulting distribution is close to uniform. We formally define this parameter through the following definition.

Definition 6. *(Smoothing Parameter [26]) For any N-dimensional lattice Λ, its dual Λ^* and $\epsilon > 0$, the smoothing parameter $\eta_\epsilon(\Lambda)$ is the smallest $s > 0$ such that $\rho_{1/s\sqrt{2\pi},0}(\Lambda^* \setminus 0) \leqslant \epsilon$. A scaled version of the smoothing parameter is defined in [13] as $\eta_\epsilon^{'} = \frac{1}{\sqrt{2\pi}}\eta_\epsilon(\Lambda)$.*

Gentry et al. [24] defined a requirement on the size of σ related to the smoothing parameter. In [13], Ducas et al. showed that using Kullback-Leibler divergence, the required width of σ can be reduced by a factor of $\sqrt{2}$. Based on [27, 24, 13], for positive integers n, λ, $\epsilon \leqslant 2^{-\lambda/2}/(4\sqrt{2N})$, any basis $\mathbf{B} \in \mathbb{Z}^{N\times N}$ and any target vector $\mathbf{c} \in \mathbb{Z}^{1\times n}$, the algorithm ($\mathbf{v}_0 \leftarrow$ `Gaussian-Sampler`$(\mathbf{B}, \sigma, \mathbf{c})$) as defined in [24, 13] is such that $\Delta(D_{\Lambda(\mathbf{B}),\sigma,\mathbf{c}}, \mathbf{v}_0) < 2^{-\lambda}$.

In this paper, we will use the same algorithm in our `Trapdoor` algorithm of our NTRU-PEKS scheme.

Definition 7. *(Decision LWE Problem) Given $\mathcal{R} \triangleq \mathbb{Z}[x]/(x^N + 1)$ and an error distribution χ over $\mathbb{R}$. For s as a random secret ring element, uniformly random a_i's $\in \mathcal{R}$ and small error elements $e_i \in \chi$, the decision LWE problem asks to distinguish between samples of the form $(a_i, a_i s + e_i)$ and randomly selected $(a_i, b_i) \in \mathcal{R} \times \mathcal{R}$.*

In [17], Regev has shown that for a certain error distribution χ, denoted $\bar{\Psi}_\alpha$, the LWE problem is as hard as the worst-case SIVP and GapSVP under quantum reduction.

Definition 8. *(Distribution of $\bar{\Psi}_\alpha$ [17, 28]) For $\alpha \in (0,1)$ and a prime q, $\bar{\Psi}_\alpha$ denotes the distribution over $\mathbb{Z}_q$ of the random variable $\lfloor qX \rceil \mod q$ is a normal random variable with mean 0 and the standard deviation $\frac{\alpha}{\sqrt{2\pi}}$.*

Theorem 1. *(From LWE to Worst-case Lattice Problems [17]) For $\alpha \in (0,1)$, and prime q s.t. $q > 2\sqrt{n}/\alpha$ there is an efficient algorithm that solve LWE$_{p,\bar{\Psi}_\alpha}$, then there is an efficient, possibly quantum, algorithm that approximates the decision version of SVP and the SIVP within $\tilde{\mathcal{O}}(n/\alpha)$ factor, in the worst case.*

Proof. Please refer to [17, Theorem 1.1]. □

Definition 9. *(A tool for computing Gram-Schmidt norm [13]) Let $f \in \mathcal{R}'$, we denote $\bar{f}$ as a unique polynomial in $f \in \mathcal{R}'$ such that $\mathcal{A}(f)^T = \mathcal{A}(\bar{f})$. If $f(x) = \sum_{i=0}^{N-1} f_i x^i$, then $\bar{f}(x) = f_0 - \sum_{i=1}^{N-1} f_{N-i}x^i$.*

Definition 10. *[29, Lemma 7.1] Given Λ as an $m-$dimensional lattice, there exists a deterministic polynomial time algorithm that, given an arbitrary basis of Λ and a full-rank set $S \in \Lambda$, where $S = s_1, s_2, \ldots, s_m$, returns a short basis T where $\|\tilde{T}\| \leq \|\tilde{S}\|$ and $\|T\| \leq \|S\|\sqrt{m}/2$.*

2.4 Identity-Based Encryption

Definition 11. *An IBE scheme is a tuple of four algorithms IBE* = $(\texttt{Setup}, \texttt{Extract}, \texttt{Enc}, \texttt{Dec})$ *defined as follows.*

- $(mpk, msk) \leftarrow \texttt{Setup}(1^k)$: On the input of the security parameter(s), this algorithm publishes system-wide public parameters *params*, outputs the master public key *mpk* and the master secret key *msk*.
- $sk \leftarrow \texttt{Extract}(id, msk, mpk)$: On the input of a user's identity $id \in \{0,1\}^*$, *mpk*, and *msk*, this algorithm outputs the user's private key *sk*.
- $c \leftarrow \texttt{Enc}(m, id, mpk)$: On the input of a message $m \in \{0,1\}^*$, identity *id*, and *mpk*, this algorithm outputs a ciphertext *c*.
- $m \leftarrow \texttt{Dec}(c, sk)$: On the input of a ciphertext *c*, the receiver's private key *sk* and *mpk*, this algorithm recovers the message *m* from the ciphertext *c*.

Following the work of [12], the following definition defines anonymity in the sense of [30].

Definition 12. *Anonymity under chosen plaintext attack (IBE-ANO-RE-CPA) for an IBE scheme is defined as follows. Given an IBE scheme, the* `KeyQuery` *oracle as defined below, we associate a bit* $b \in \{0,1\}$ *to the adversary* $\mathcal{A}$ *in the following experiment.*

```
KeyQuery(id)
  $idSet \leftarrow idSet \cup id$
  return sk
```

```
Experiment $Exp_{IBE,\mathcal{A}}^{IBE\text{-}ANO\text{-}RE\text{-}CPA\text{-}b}(1^k)$

  $idSet \leftarrow \emptyset$, $(mpk, msk) \xleftarrow{\$} \texttt{Setup}(1^k)$
  for a random oracle $H$
  $(id_0, id_1, m) \leftarrow \mathcal{F}^{\texttt{KeyQuery}(.),H}(find, mpk)$
  $c \leftarrow \texttt{Enc}^H(m, id_b, mpk)$
  $b' \leftarrow \mathcal{F}^{\texttt{KeyQuery}(.),H}(guess, c)$
  if $\{id_0, id_1\} \cap idSet = \emptyset$ return $b'$ else, return 0
```

$\mathcal{A}$'s advantage in the above experiment is defined as:

$$Adv_{IBE,\mathcal{A}}^{IBE\text{-}ANO\text{-}RE\text{-}CPA}(1^k) = \Pr[Exp_{IBE,\mathcal{A}}^{IBE\text{-}ANO\text{-}RE\text{-}CPA\text{-}1}(1^k) = 1] - \Pr[Exp_{IBE,\mathcal{A}}^{IBE\text{-}ANO\text{-}RE\text{-}CPA\text{-}0}(1^k) = 0]$$

2.5 Public Key Encryption with Keyword Search

A PEKS scheme consists of the following algorithms.

Definition 13. *A PEKS scheme is a tuple of four algorithms PEKS* = $(\texttt{KeyGen}, \texttt{PEKS}, \texttt{Trapdoor}, \texttt{Test})$ *defined as follows.*

- $(pk, sk) \leftarrow \texttt{KeyGen}(1^k)$: On the input of the security parameter(s), this algorithm outputs the public and private key pair (pk, sk).

- $s_w \leftarrow \texttt{PEKS}(pk, w)$: On the input of user's public key pk and a keyword $w \in \{0,1\}^*$, this algorithm outputs a searchable ciphertext s_w.
- $t_w \leftarrow \texttt{Trapdoor}(sk, w)$: On the input of a user's private key sk and a keyword $w \in \{0,1\}^*$, this algorithm outputs a trapdoor t_w.
- $d \leftarrow \texttt{Test}(t_w, s_w)$: On the input of a trapdoor $t_w = \texttt{Trapdoor}(sk, w')$ and a searchable ciphertext $s_w = \texttt{PEKS}(pk, w)$, this algorithm outputs a decision bit $d = 1$ if $w = w'$, and $d = 0$ otherwise.

Definition 14. *Keyword indistinguishability against an adaptive chosen-keyword attack (IND-CKA) is defined as follows. Given a PEKS scheme, and the* `TdQuery` *oracle as defined below, we associate a bit $b \in \{0,1\}$ to the adversary $\mathcal{A}$ in the following experiment.*

```
$\texttt{TdQuery}(w)$
  $wSet \leftarrow wSet \cup w$
  $t_w \leftarrow \texttt{Trapdoor}(w, sk, pk)$
  $return\ t_w$
```

```
$Experiment\ Exp^{PEKS\text{-}IND\text{-}CKA\text{-}b}_{PEKS,\mathcal{A}}(1^k)$
  $wSet \leftarrow \emptyset$, $(pk, sk) \leftarrow \texttt{KeyGen}(1^k)$
  for a random oracle $H$
  $(w_0, w_1) \leftarrow \mathcal{A}^{\texttt{TdQuery}(.),H}(find, pk)$
  $s_w \leftarrow \texttt{PEKS}^H(pk, w_b)$
  $b' \leftarrow \mathcal{A}^{\texttt{TdQuery}(.),H}(guess, s_w)$
  $if\ \{w_0, w_1\} \cap wSet = \emptyset\ return\ b'\ else,\ return\ 0$
```

$\mathcal{A}$'s advantage in the above experiment is defined as

$$Adv^{PEKS\text{-}IND\text{-}CKA}_{PEKS,\mathcal{A}}(1^k) = \Pr[Exp^{PEKS\text{-}IND\text{-}CKA\text{-}1}_{PEKS,\mathcal{A}}(1^k) = 1] - \Pr[Exp^{PEKS\text{-}IND\text{-}CKA\text{-}0}_{PEKS,\mathcal{A}}(1^k) = 0]$$

.

2.6 Consistency of PEKS

Due to the properties of NTRU-based encryption scheme, and following the work of [31], we investigate the consistency of our scheme from two aspects, namely, right-keyword consistency and adversary-based consistency [12]. Right-keyword consistency implies the success of a search query to retrieve records associated with keyword w for which the `PEKS` algorithm had computed a searchable ciphertext. On the other hand, we define the adversary-based consistency [12] as follows.

Definition 15. *Adversary-based consistency of a PEKS scheme is defined in the following experiment.*

```
$Experiment\ Exp^{PEKS\text{-}Consist}_{PEKS,\mathcal{A}}(1^k)$
  $(pk, sk) \leftarrow \texttt{KeyGen}(1^k)$
  $for\ a\ random\ oracle\ H$
  $(w_0, w_1) \leftarrow \mathcal{A}^H(pk)$, $s_{w_0} \leftarrow \texttt{PEKS}^H(pk, w_0)$
  $t_{w_1} \leftarrow \texttt{Trapdoor}^H(pk, w_1)$
  if $w_0 \neq w_1$ and $[\texttt{Test}^H(pk, t_{w_1}, s_{w_0}) = 1]$ return 1 else,
  $return\ 0$
```

$\mathcal{A}$'s advantage in the above experiment is defined as:

$$Adv^{PEKS\text{-}Consist}_{PEKS,\mathcal{A}}(1^k) = \Pr[Exp^{PEKS\text{-}Consist}_{PEKS,\mathcal{A}}(1^k) = 1].$$

We note that for schemes with security in the standard model, the random oracle H is eliminated in Definitions 12, 14 and 15 .

3 Proposed Schemes

In this section, we first propose our scheme based on NTRU lattices (NTRU-PEKS) which enjoys from highly efficient `Test` and `PEKS` algorithms and then put forth our scheme in the standard model (i.e., LWE-PEKS) which provides a high level of security.

3.1 PEKS Scheme from NTRU Lattices

In this section, we present our highly efficient NTRU-PEKS scheme that consists of the following algorithms.

$\underline{(pk, sk) \leftarrow \texttt{KeyGen}(q, N)}$: Given a power-of-two integer N and a prime q, this algorithm works as follows.

1) Compute $\sigma_f \leftarrow 1.17\sqrt{\frac{q}{2N}}$ and select $f, g \leftarrow \mathcal{D}_{N,\sigma_f}$ to compute $\left\|\tilde{\mathbf{B}}_{f,g}\right\|$ and $Norm \leftarrow \max(\|(g, -f)\|, \left\|\left(\frac{q\bar{f}}{f*\bar{f}+g*\bar{g}}, \frac{q\bar{g}}{f*\bar{f}+g*\bar{g}}\right)\right\|)$. If $Norm < 1.17\sqrt{q}$, proceed to the next step. Otherwise, if $Norm \geq 1.17\sqrt{q}$, this process is repeated by sampling new f and g.

2) Using extended euclidean algorithm, compute $\rho_f, \rho_g \in \mathcal{R}$ and $\mathcal{R}_f, \mathcal{R}_g \in \mathbb{Z}$ such that $\rho_f \cdot f = \mathcal{R}_f \mod (x^N + 1)$ and $\rho_g \cdot g = \mathcal{R}_g \mod (x^N + 1)$. Note that if $\gcd(\mathcal{R}_f, \mathcal{R}_g) \neq 1$ or $\gcd(\mathcal{R}_f, q) \neq 1$, start from the previous step by sampling new f and g.

3) Using extended euclidean algorithm, compute $u, v \in \mathbb{Z}$ such that $u \cdot \mathcal{R}_f + v \cdot \mathcal{R}_g = 1$. Compute $F \leftarrow q \cdot v \cdot \rho_g$, $G \leftarrow q \cdot u \cdot \rho_f$ and $k \leftarrow \lfloor \frac{F*\bar{f}+G*\bar{g}}{f*\bar{f}+g*\bar{g}} \rceil \in \mathcal{R}$ and reduce F and G by computing $F \leftarrow F - k * f$ and $G \leftarrow G - k * g$.

4) Finally, compute $h = g * f^{-1} \mod q$ and $\mathbf{B} = \begin{pmatrix} \mathcal{A}(g) & -\mathcal{A}(f) \\ \mathcal{A}(G) & -\mathcal{A}(F) \end{pmatrix}$ and output $(pk \leftarrow h, sk \leftarrow \mathbf{B})$.

$\underline{s_w \leftarrow \texttt{PEKS}(pk, w)}$: Given cryptographic hash functions $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_q^N$ and $H_2 : \{0,1\}^N \times \{0,1\}^N \rightarrow \mathbb{Z}_q^N$, the receiver's public key pk and a keyword $w \in \{0,1\}^*$ to be encrypted, the sender performs as follows.

1. Compute $t \leftarrow H_1(w)$ and pick $\mathbf{r}, \mathbf{e_1}, \mathbf{e_2} \xleftarrow{\$} \{-1, 0, 1\}^N$, $k \xleftarrow{\$} \{0,1\}^N$.
2. Compute $\mathbf{c_0} \leftarrow \mathbf{r} * \mathbf{h} + \mathbf{e_1} \in \mathcal{R}_q$ and $\mathbf{c_1} \leftarrow \mathbf{r} * \mathbf{t} + \mathbf{e_2} + \left\lfloor \frac{q}{2} \right\rfloor \mathbf{k} \in \mathcal{R}_q$.
3. Finally, the algorithm outputs $s_w = \langle c_0, c_1, H_2(k, c_1) \rangle$.

$\mathbf{t_w} \leftarrow \texttt{Trapdoor}(sk, w)$: Given the receiver's private key sk, and a keyword $w \in \{0,1\}^*$, the receiver computes $t \leftarrow H_1(w)$ and using the sampling algorithm $\texttt{Gaussian-Sampler}(\mathbf{B}, \sigma, (t, 0))$, samples s and $\mathbf{t_w}$ such that $\mathbf{s} + \mathbf{t_w} * \mathbf{h} = t$.
$d \leftarrow \texttt{Test}(pk, t_w, s_w)$: On the input of a receiver's public key pk, a trapdoor t_w and a searchable ciphertext $s_w = \langle A, B, H_2(k, B)\rangle$, this algorithm computes $y \leftarrow \lfloor \frac{\mathbf{c_1} - \mathbf{c_0} * \mathbf{t_w}}{q/2} \rceil$ and outputs $d = 1$ if $H_2(\mathbf{y}, \mathbf{c_1}) = H_2(\mathbf{k}, \mathbf{c_1})$ and $d = 0$, otherwise.

3.1.1 Completeness and Consistency

In this section, we show the completeness and consistency of NTRU-PEKS.

Lemma 1. *Given a public-private key pair $(h, B) \leftarrow \texttt{KeyGen}(q, N)$, a searchable ciphertext $s_w \leftarrow \texttt{PEKS}(pk, w)$, and a trapdoor generated by the receiver $t_w \leftarrow \texttt{Trapdoor}(sk, w)$ our proposed scheme is complete.*

Proof. To show the completeness of our scheme for $s_w = \langle \mathbf{c_0}, \mathbf{c_1}, H_2(k, \mathbf{c_1})\rangle$, the $\texttt{Test}$ algorithm should return 1 when $\lfloor \frac{\mathbf{c_1} - \mathbf{c_0} * \mathbf{t_w}}{q/2} \rceil = \mathbf{k}$. To affirm this, we work as follows.

$$\begin{aligned}\mathbf{c_1} - \mathbf{c_0} * \mathbf{t_w} &= (\mathbf{r} * \mathbf{t} + \mathbf{e_2} + \left\lfloor \frac{q}{2} \right\rfloor \mathbf{k}) - (\mathbf{r} * \mathbf{h} + \mathbf{e_1}) * \mathbf{t_w} \in \mathcal{R}_q \\ &= \mathbf{r} * \mathbf{s} + \mathbf{e_2} + \lfloor \frac{q}{2} \rfloor \mathbf{k} - \mathbf{t_w} * \mathbf{e_1}\end{aligned}$$

Given $\mathbf{r}, \mathbf{e_1}, \mathbf{e_2}, \mathbf{t_w}$ and $\mathbf{s}$ are all short vectors (due to the parameters of our sampling algorithm), all the coefficients of $\mathbf{r} * \mathbf{s} + \mathbf{e_2} - \mathbf{t_w} * \mathbf{e_1}$ will be in $(-\frac{q}{4}, \frac{q}{4})$, and therefore, $\lfloor \frac{\mathbf{c_1} - \mathbf{c_0} * \mathbf{t_w}}{q/2} \rceil = \mathbf{k}$. □

To address right-keyword consistency issues related to the decryption error of encryption over NTRU lattices, we need to make sure that all the coefficients of $\mathbf{z} = \mathbf{r} * \mathbf{s} + \mathbf{e_2} - \mathbf{e_1} * \mathbf{tw}$ are in the range $(-\frac{q}{4}, \frac{q}{4})$ and $q \approx 2^{24}$ for $\kappa = 80$ and $q \approx 2^{27}$ for $\kappa = 192$.

Theorem 2. *The NTRU-PEKS scheme is consistent in the sense of Definition 11.*

Proof. Upon inputting q and N, the challenger $\mathcal{C}$ initiates the experiment $(h, \mathbf{B}) \leftarrow \texttt{KeyGen}(q, N)$. It passes h to the adversary $\mathcal{A}$ and keeps $\mathbf{B}$ secret.

- $(w_0, w_1) \leftarrow \mathcal{A}^{H_1}(pk)$: $\mathcal{A}$ sends $\mathcal{C}$ two keywords (w_0, w_1).
- $s_{w_0} \leftarrow \texttt{PEKS}^H(pk, w_b)$: $\mathcal{C}$ computes $\mathbf{c_0} = \mathbf{r} * \mathbf{h} + \mathbf{e_1}$ and $\mathbf{c_1} = \mathbf{r} * \mathbf{H}(\mathbf{w_0}) + \mathbf{e_2} + \left\lfloor \frac{q}{2} \right\rfloor \mathbf{k}$ for a random selection of $\mathbf{r}, \mathbf{e_1}, \mathbf{e_2} \overset{\$}{\leftarrow} \{-1, 0, 1\}^N$, $\mathbf{k} \overset{\$}{\leftarrow} \{0, 1\}^N$, and sends $\langle \mathbf{c_0}, \mathbf{c_1}, H_2(\mathbf{k}, \mathbf{c_1})\rangle$ to $\mathcal{A}$.
- $t_{w_1} \leftarrow \texttt{Trapdoor}^H(pk, w_b)$: $\mathcal{C}$ samples short vectors s, t_w such that $\mathbf{s} + \mathbf{t_w} * \mathbf{h} = H(w_1)$ and returns $\mathbf{t_w}$ to $\mathcal{A}$.

Following Definition 11, $\mathcal{A}$ wins when $w_0 \neq w_1$, and the $\texttt{Test}$ algorithm outputs 1 (i.e, $H_2(\mathbf{k}, \mathbf{c_1}) = H_2(\mathbf{y}, \mathbf{c_1})$).

Note that in the above game, $\mathcal{A}$ wins when $w \neq w'$ and $H_2(z_1, z_1') = H_2(z_2, z_2')$. Let's assume $\mathcal{A}$ makes q_1 queries to H_1 and q_2 queries to H_2 oracles. Let E_1 be the event that there exists (x_1, x_2) such that $H_1(x_1) = H_1(x_2)$ and $x_1 \neq x_2$ and let E_2 be the event that there exist two pairs (z_1, z_1') and (z_2, z_2') such that $H_2(z_1, z_1') = H_2(z_2, z_2')$ for $z_1 \neq z_2$ and $z_1' \neq z_2'$. Then if $\Pr[\cdot]$ represents the probability of consistency definition, $Adv_{PEKS,\mathcal{A}}^{PEKS\text{-}Consist}(1^k) \leq \Pr[E_1] + \Pr[E_2] + \Pr[Exp_{PEKS,\mathcal{A}}^{PEKS\text{-}Consist} = 1 \wedge \bar{E_1} \wedge \bar{E_2}]$

Given the domain of our hash functions, the first and second terms are upper bounded by $(q_1+2)^2/N2^{\log_2 q}$ and $(q_2+2)^2/N2^{\log_2 q}$, respectively. For the last term, if $H_1(x_1) \neq H_1(x_2)$, then in our scheme, the probability that $B_1 = B_2$ is negligible due to the decryption error. Therefore, $H_2(y_1, B_1) \neq H_2(y_2, B_2)$, hence, the probability of the last term is also negligible. □

3.1.2 Security Analysis

In this section, we focus on analyzing the security of the NTRU-PEKS scheme.

The security of lattice-based schemes can be determined by hardness of the underlying lattice problem (in case of NTRU-PEKS, ring-LWE). Therefore, similar to the LWE-PEKS scheme, we use the root Hermite factor to assess the security of the NTRU-PEKS scheme. According to [32], for a short planted vector $\mathbf{v}$ in an NTRU lattice, the associated root Hermite factor is computed as $\gamma^n = \frac{\sqrt{N/(2\pi \mathbf{e})} \times \det(\Lambda)^{1/n} \|\mathbf{v}\|}{0.4 \times \|\mathbf{v}\|}$. Based on [33, 34], $\gamma \approx 1.004$ guarantees intractability and provides approximately 192-bit security.

Following Lemma 3, to establish the security of our NTRU-PEKS scheme, we need to rely on the security of the underlying IBE scheme. Ducas et al. provided the proof of IBE-IND-CPA of their scheme in [13]. Therefore, we are left to prove the anonymity of their scheme via Theorem 2.

Theorem 3. *The IBE scheme of Ducas et al. is anonymous in the sense of Definition 8 under the decision ring-LWE problem.*

Proof. Since the output of the PEKS algorithm of our scheme corresponds to the encryption algorithm of [35, 36], for $\mathcal{A}$ to determine s_w corresponds to which keyword with any probability $\Pr \geq \frac{1}{2} + \epsilon$ - for any non-negligible ϵ, it has to solve the decision ring-LWE. Our scheme works over the polynomial ring $\mathbb{Z}[x]/(x^N + 1)$, for a power-of-two N and a prime $q \equiv 1 \mod 2N$. The ring-LWE based `PEKS` algorithm computes a pseudorandom ring-LWE vector $\mathbf{c_0} = \mathbf{r} * \mathbf{h} + \mathbf{e_1}$ (for a uniform $\mathbf{r}, \mathbf{e_1} \xleftarrow{\$} \{-1, 0, 1\}^N$) and uses $H(w)$ to compute $\mathbf{c_1} = \mathbf{r} * \mathbf{H(w)} + \mathbf{e_2} + \lfloor \frac{q}{2} \rfloor \mathbf{k}$ that is also statistically close to uniform. Therefore, the adversary's view of $\langle c_0, c_1, H_2(\mathbf{k}, \mathbf{c_1}) \rangle$ is indistinguishable from uniform distribution under the hardness of decision ring-LWE. The pseudorandomness is preserved when t_w is chosen from the error distribution (by adopting the transformation to Hermite's normal form) similar to the one in standard LWE [37]. □

Theorem 4. *If there exists an adversary $\mathcal{A}$ that can break IND-CKA of NTRU-PEKS scheme as in Definition 10, one can build an adversary $\mathcal{F}$ that uses $\mathcal{A}$ as subroutine and breaks the security of the IBE scheme in Definition 8.*

Proof. The proof works by having adversaries $\mathcal{F}$ and $\mathcal{A}$ initiating the *find* phase as in Definition 8 and Definition 10 respectively.

Algorithm $\mathcal{F}^{\texttt{KeyQuery}(.),H}(\mathit{find}, mpk)$

- $(mpk, msk) \xleftarrow{\$} \texttt{Setup}(q, N)$: $\mathcal{F}$ receives mpk and passes it to $\mathcal{A}$.

Algorithm $\mathcal{A}^{TdQuery(.),H}(\mathit{find}, pk)$

- Queries on `TdQuery`(.): Upon such queries, $\mathcal{F}$ queries `KeyQuery`(.) which keeps a list *idSet* maintaining all the previously requested queries and responses. If the submitted query exists, the same response is returned, otherwise, to sample short vectors s, t_w the oracle uses msk to run $(\mathbf{s}, \mathbf{t_w}) \overset{\$}{\leftarrow}$ `Gaussian-Sampler`$(msk, \sigma, (H(w), 0))$ and passes $\mathbf{t_w}$ to $\mathcal{F}$. $\mathcal{F}$ sends $\mathbf{t_w}$ to $\mathcal{A}$.

After the *find* phase, a hidden fair coin $b \in \{0, 1\}$ is flipped.
Execute $(w_0, w_1) \leftarrow \mathcal{A}^{\texttt{TdQuery}(.), H}(guess, pk)$

- Upon receiving (w_0, w_1), $\mathcal{F}$ selects a message $m \in \{0\}^N$ and calls `Enc`(m, w_0, w_1) that runs encryption on (w_b, m) which works as in Definition 7 and outputs $s_w = \langle \mathbf{c_0}, \mathbf{c_1}, H_2(\mathbf{k}, \mathbf{c_1}) \rangle$. $\mathcal{F}$ relays s_w to $\mathcal{A}$.

Finally, $\mathcal{A}$ outputs its decision bit $b' \in \{0, 1\}$. $\mathcal{F}$ also outputs b' as its response. Omitting the terms that are negligible in terms of q and N, the upper bound on *IND-CKA* of NTRU-PEKS is as follows.

$$Adv_{\mathcal{A}}^{\textit{PEKS-IND-CKA}}(q, N) \leq Adv_{\mathcal{F}}^{\textit{NTRU-IBE-ANO-CPA}}(q, N)$$

□

3.2 Lattice-Based PEKS Scheme in the Standard Model

Similar to [14], we treat keywords as a sequence of l bits $w = (b_1, \ldots, b_l) \in \{1, -1\}^l$. Before presenting the scheme in details, we review the tools that are needed for the correctness of our LWE-PEKS scheme. In [38], Ajtai illustrated how to sample a random uniform matrix (with a small Gram-Schmidt norm) $A \in \mathbb{Z}_q^{n \times m}$ with an associated basis S_A of $\Lambda_q^{\perp}(A)$. The following theorem, defines the properties of `TrapGen` algorithm [39, 14] which is used in the `KeyGen` algorithm of the LWE-PEKS scheme.

$(\mathbf{A}, \mathbf{S}) \leftarrow$ `TrapGen`(q, n) : Given a prime q, a positive n and $\delta = \frac{1}{3}$, there is a polynomial time algorithm `TrapGen`(q, n) that outputs a pair $(\mathbf{A} \in \mathbb{Z}_q^{n \times m}, \mathbf{S} \in \mathbb{Z}^{m \times m})$ s.t., $\mathbf{A}$ is statistically close to uniform and $\mathbf{S}$ is a basis for $\Lambda_q^{\perp}(\mathbf{A})$, where $m = 6n \log q$ and $\|\tilde{\mathbf{S}}\| \leq \mathcal{O}(\sqrt{n \log q})$ and $\|\mathbf{S}\| \leq \mathcal{O}(n \log n)$ hold with a high probability.

Following [14], we set $\sigma_{TG} = \mathcal{O}(\sqrt{n \log q})$ as the maximum Gram-Schmidt norm of the basis generated by `TrapGen`(q, n).

In the following we define the sampling algorithm which is used to generate trapdoors in our scheme (i.e., `SampleLeft`), the same algorithm, with identical properties has also been used in [28, 40].

$\mathbf{e} \leftarrow$ `SampleLeft`$(\mathbf{A}, \mathbf{M_1}, \mathbf{T_A}, \mathbf{u}, \sigma)$: Given an n-rank matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, a matrix $\mathbf{M_1} \in \mathbb{Z}^{n \times m_1}$, a short basis of $\Lambda_q^{\perp}(\mathbf{A})$, a vector $\mathbf{u} \in \mathbb{Z}_q^n$, and a Gaussian parameter $\sigma > \left\|\tilde{\mathbf{T}}_{\mathbf{A}}\right\| \cdot \omega(\sqrt{\log(m + m_1)})$, this algorithm outputs a vector $\mathbf{e} \in \mathbb{Z}^{m + m_1}$ sampled from the distribution statistically close to $\mathcal{D}_{\Lambda_q^u(\mathbf{F_1}), \sigma}$ where $\mathbf{F_1} := (\mathbf{A} | \mathbf{M_1})$.

In the following, we present the LWE-PEKS scheme in detail.

$(pk, sk) \leftarrow$ `KeyGen`(λ): On the input of the security parameter λ, and the parameters q, m, n, σ, α (set as instructed in the following section), the receiver works as follows to generate her key pair.

1. Use `TrapGen`(q, n) to pick a random matrix $\mathbf{A_0} \in \mathbb{Z}^{n \times m}$ with basis $\mathbf{T_{A_0}}$ for $\Lambda_q^{\top}(\mathbf{A_0})$ *s.t.* $\left\|\tilde{\mathbf{T}}_{\mathbf{A_0}}\right\| \leq \mathcal{O}(\sqrt{n \log q})$.

2. Select $l+1$ random matrices $\mathbf{A_1},\dots\mathbf{A_l},\mathbf{B} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$ and a random vector $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_q^n$.

3. Output the public key $pk \leftarrow (\mathbf{A_0},\mathbf{A_1},\dots,\mathbf{A_l},\mathbf{B},\mathbf{u})$ and secret key $sk \leftarrow \mathbf{T_{A_0}}$.

$\underline{s_w \leftarrow \texttt{PEKS}(pk,w)}$: On the input of the pk and an l-bit keyword $w=(b_1,\dots,b_l)\in\{1,-1\}^l$, the sender picks $b'_j \xleftarrow{\$} \{0,1\}$ for $j=1,\dots,\kappa$, sets $\mathbf{A_w} \leftarrow \mathbf{B}+\sum_{\mathbf{i=1}}^{\mathbf{l}} b_i\mathbf{A_i} \in \mathbb{Z}_q^{n\times m}$ and $\mathbf{F_w} \leftarrow (\mathbf{A_0}|\mathbf{A_w}) \in \mathbb{Z}_q^{n\times 2m}$. For each b'_j, it computes as follows.

1. Choose a uniformly random $\mathbf{s_j} \xleftarrow{\$} \mathbb{Z}_q^n$ and matrices $\mathbf{R_{i_j}} \xleftarrow{\$} \{-1,1\}^{m\times m}$ for $i=1,\dots,l$ and set $\mathbf{R_{b'_j}} \leftarrow \sum_{i=1}^{l} b_i\mathbf{R_{i_j}} \in \{-l,\dots,l\}^{m\times m}$.

2. Choose noise vectors $x_j \xleftarrow{\Psi_\alpha} \mathbb{Z}_q$ and $\mathbf{y_j} \xleftarrow{\Psi_\alpha^m} \mathbb{Z}_q^m$, and set $\mathbf{z_j} \leftarrow \mathbf{R}_{\mathbf{b'_j}}^\top\mathbf{y_j} \in \mathbb{Z}_q^m$.

3. Set $c_{0_j} \leftarrow \mathbf{u}^\top\mathbf{s_j}+x_j+b'_j\lfloor\frac{q}{2}\rfloor \in \mathbb{Z}_q$ and $\mathbf{c_{1_j}} \leftarrow \mathbf{F_w}^\top\mathbf{s_j}+\begin{bmatrix}\mathbf{y_j}\\ \mathbf{z_j}\end{bmatrix} \in \mathbb{Z}_q^{2m}$.

4. Output searchable ciphertext $s_{w_j}=(c_{0_j},\mathbf{c_{1_j}},b'_j)$ for $j=1,\dots,\kappa$.

$\underline{\mathbf{t_w} \leftarrow \texttt{Trapdoor}(pk,sk,w)}$: On the input of the keys and a keyword $w=(b_i,\dots,b_l)\in\{1,-1\}^l$, the sender computes as follows.

1. Let $\mathbf{A_w} \leftarrow \mathbf{B}+\sum_{i=1}^{l} b_i\mathbf{A_i} \in \mathbb{Z}_q^{n\times m}$ and sample $\mathbf{t_w} \in \mathbb{Z}_q^{2m}$ as $\mathbf{t_w} \leftarrow \texttt{SampleLeft}(\mathbf{A_0},\mathbf{A_w},\mathbf{T_{A_0}},\mathbf{u},\sigma)$[1] .

2. Output the trapdoor as $\mathbf{t_w}$.

Given $\mathbf{F_w} := (\mathbf{A_0}|\mathbf{A_w})$, then $\mathbf{F_w}\cdot\mathbf{t_w}=\mathbf{u}\in\mathbb{Z}_q$, and $\mathbf{t_w}$ is distributed as $D_{\Lambda_q^u F_w,\sigma}$.
$\underline{d \leftarrow \texttt{Test}(\mathbf{t_w},s_w)}$: Given a trapdoor td for a keyword $w=(b_i,\dots,b_l)\in\{1,-1\}^l$, and κ searchable ciphertexts $s_{w_j}=(c_{0_j},c_{1_j},b'_j)$ for $j=1,\dots,\kappa$ on keyword w, it computes as follows.

i. Set $\nu_j \leftarrow c_{0_j}-\mathbf{t_w}\mathbf{c_{1_j}} \in \mathbb{Z}_q$ and check if $|\delta_j-\lfloor\frac{q}{2}\rfloor| < \lfloor\frac{q}{4}\rfloor$, set $\nu_j \leftarrow 1$ and otherwise, $\nu_j \leftarrow 0$.

ii. If $\nu_j=b'_j$ holds for all $1\le j\le\kappa$, set $d\leftarrow 1$, else $d\leftarrow 0$.

3.2.1 Completeness and Consistency

In this section we show the completeness and consistency of our scheme.

Lemma 2. *Given a public-private key pair $(pk,sk)\leftarrow KeyGen(\lambda)$, a searchable ciphertext $s_w \leftarrow$ $\texttt{PEKS}(pk,w)$ and a trapdoor generated by the receiver $td \leftarrow \texttt{Trapdoor}(pk,sk,w)$, the proposed scheme is complete.*

Proof. To show the completeness of our scheme for $sw_j := (c_{0_j},c_{1_j},b'_j)$, the `Test` algorithm should return 1 when $\nu_j=b'_j$ where $\nu_j \leftarrow c_{0_j}-tdc_{1_j}$ for all $j=1,\dots,\kappa$. To affirm this, we work

[1]As shown in [14], A_0 is of rank n, with a high probability.

as follows.

$$\begin{aligned}\nu_j &= c_{0_j} - \mathbf{t_w}^\top \mathbf{c_{1_j}}\\ &= \mathbf{u}^\top \mathbf{s_j} + x_j + b'_j \lfloor \frac{q}{2} \rfloor - \mathbf{t_w}\big(\mathbf{F}_{\mathbf{id}}^\top \mathbf{s_j} + \begin{bmatrix}\mathbf{y_j}\\ \mathbf{z_j}\end{bmatrix}\big)\\ &= \mathbf{u}^\top \mathbf{s_j} + x_j + b'_j \lfloor \frac{q}{2} \rfloor - \mathbf{u}^\top \mathbf{s_j} - \mathbf{t_w}^\top \begin{bmatrix}\mathbf{y_j}\\ \mathbf{z_j}\end{bmatrix}\\ &= b'_j \lfloor \frac{q}{2} \rfloor + x - \mathbf{t_w}^\top \begin{bmatrix}\mathbf{y_j}\\ \mathbf{z_j}\end{bmatrix}\end{aligned}$$

Where $x - t_w^\top \begin{bmatrix} y_j \\ z_j \end{bmatrix}$ is the error term. Based on Lemma 21 in [14], the error term is bounded by $q \cdot \sigma \cdot l \cdot m \cdot \alpha \cdot \omega \cdot (\sqrt{\log m}) + \mathcal{O}(\sigma m^{3/2})$. For the system to work correctly, one needs to make sure that:

i. $\alpha < [\sigma \cdot l \cdot m \cdot \omega(\sqrt{\log m})]^{-1}$ and $q = \Omega(\sigma m^{3/2})$,

ii. $m > 6n \log q$ so `TrapGen` can operate,

iii. σ is large enough so that `SampleLeft` as defined above, and `SampleRight` (which is similar to `SampleLeft`, and is used in the proof of [14]) can operate, i.e., $\sigma > l \cdot m \cdot \omega(\sqrt{\log m})$

iv. For Regev's [17] reduction to work, set $q > 2\sqrt{n}/\alpha$.

□

To achieve these requirements, we set $q \geq m^{2.5} \cdot \omega(\sqrt{\log n}), m = 6n^{1+\delta}$, $\sigma = ml \cdot \omega(\sqrt{\log n})$, $\alpha = [l^2 m^2 \cdot \omega(\sqrt{\log n})]^{-1}$.

Following the results of Lemma 13 and Lemma 19 in [14], setting the parameters of the scheme as suggested above will ensure the right-keyword consistency of our PEKS scheme with a high probability.

Theorem 5. *The LWE-PEKS scheme is consistent in the sense of Definition 11.*

Proof. For the `Test` algorithm to return 1, all the κ bits of b'_j and ν_j for $1 \leq j \leq \kappa$ should match. This implies that given A_w (obtained from the bit string in the keyword w), the `SampleLeft` algorithm should sample short vectors statistically close (i.e., have negligible statistical distance) to $\mathbf{F_w} \leftarrow (\mathbf{A_0}|\mathbf{A_w})$. Therefore, our adversary based consistency comes from the Theorem 3.4 in [40] (and the signing algorithm in [28]) that proves the statistical closeness of $\mathbf{t_w}$ that is generated by the `SampleLeft` on the input of $\mathbf{F_w}$. Therefore, for the suggested parameters and based on [28, 40], our LWE-PEKS scheme is consistent. □

3.2.2 Security Analysis

Based on [33], the hardness of lattice problems is measured using the root Hermite factor.

Lemma 3. *If an IBE scheme is IBE-IND-CPA and IBE-ANO-RE-CPA-secure, then it is also IBE-ANO-CPA-secure.*

Proof. Please refer to [12]. □

Following Lemma 3, to establish the security of our LWE-PEKS scheme, we need to establish the anonymity property of the underlying IBE scheme. In [14], Agrawal et al. proved the security of their adaptive IBE scheme with a strong privacy property called *indistinguishable from random*, which is a stronger security notion as compared to the anonymity property defined in [41, 12]. In the initial proposal of [14], for the security proof of the adaptive variant of the scheme, there was a restriction where $q > Q$ where Q is the number of queries made by the adversary. This restriction was later lifted in by a more refined analysis in [16]. This implies that based on Lemma 3, the resulting LWE-PEKS scheme is secure in the standard model. Based on [33], the hardness of lattice problems is measured using the root Hermite factor. For a vector $\mathbf{v}$ in an N-dimension lattice that is larger than the n^{th} root of the determinant, the root Hermite factor is computed as $\gamma = \frac{\|\mathbf{v}\|}{\det(\Lambda_{h,q})^{1/n}}$. For our LWE-PEKS scheme, we follow the suggested parameters in [42, 34] to achieve $\approx$ 192-bit security for message and randomness recovery attack with $\gamma \approx 1.0042$.

Secure channel requirement. Baek et al. [43] highlighted the requirement of a secure channel for trapdoor transmission between the receiver and the server and proposed the notion of Secure-Channel Free (SCF) PEKS schemes where the keywords are encrypted by both the server's and receiver's public keys. Offline keyword-guessing attack, as introduced by Byun et al. [44], implies the ability of an adversary to find which keyword was used to generate the trapdoor. This inherent issue is due to low-entropy nature of the commonly selected keywords and public availability of the encryption key [19]. Since Byun et al.'s work [44], there have been a number of attempts in proposing schemes that address keyword guessing attacks [45, 46, 47]. However, in all the proposals, once the trapdoor is revealed to the server, the keyword guessing attacks remain a perpetual problem [47]. Jeong et al. [47] showed the trade-off between the security of a PEKS scheme against keyword-guessing attacks and its consistency - by mapping a trapdoor to multiple keywords. For our scheme, we can assume a conventional or even post quantum secure [48, 49] SSL/TLS connection between the receiver and the server. We believe such reliable protocols provide the best means for communicating trapdoors to the servers. Establishing a secure line through SSL/TLS could be much more efficient than using any public key encryption as in SFC-PEKS. Since in such protocols, after the hand shake protocol, all communications are encrypted using symmetric encryption.

3.3 Alternative NTRU-based Constructions

Bellare et al. [50] proposed a new variation of public key encryption with search capability called Efficiently Searchable Encryption (ESE). The idea behind ESE is to store a deterministically computed "tag" along with the ciphertext. To respond to search queries, the server only needs to lookup for a tag in a list of sorted tags. This significantly reduces the search time on the server. For ESE to provide privacy, the keywords need to be selected from a distribution with a high min-entropy. To compensate for privacy in absence of high min-entropy distribution for keywords, the authors suggested truncating the output of the hash function to increase the probability of collisions. However, this directly affects the consistency of the scheme and shifts the burden of decrypting unrelated responds to the receiver. As compared to PEKS schemes, in ESE schemes, the tag can be computed from both the plaintext and ciphertext. This highly differentiates the applications of these two searchable encryption schemes.

In this paper, we focused on PEKS scheme as it does not have consistency issues or min-

Table 1: Analytical performance analysis and comparison.

Schemes	**Computation**			**Storage**			
	`Test`	`PEKS`	`Trapdoor`	***PK Size***	***SK* Size**	***SC Size***	***TD Size***
BCO [5]	bp	$1bp + sm$	sm	$2\lvert q'\rvert$	$\lvert q'\rvert$	$3\lvert q'\rvert$	$2\lvert q'\rvert$
ZI [18]	$ex + bp$	$2sm + 2ex$ $+2bp$	$sm + 1pa$	$2\lvert q'\rvert$	$\lvert q'\rvert$	$38\lvert q'\rvert$	$2\lvert q'\rvert + \lvert q'\rvert$
NTRU-PEKS	$Conv$	$2Conv$‡	$GSamp$	$N\lvert q\rvert$	$2N$ $\log_2(2s\pi)$†	$3N\lvert q\rvert$	$N\lvert q\rvert$
LWE-PEKS	$\kappa\ mul_v$	$2\kappa\ (mul_v+$ $mul_{vm})$	$SampLeft$	$nm\lvert q\rvert$	$nm\sqrt{n\lvert q\rvert}$	$\kappa(2m+1)$ $\lvert q\rvert$	$2m$ $\log_2 \sigma$II

For 192-bit security, we set $N = 1024$ and $q \approx 2^{27}$ which gives us a root Hermite factor $\gamma = 1.0042$ for our scheme and for BCO and ZI schemes, we set $q' \approx 2^{192}$.
PK and SK denote public key and private key, respectively. SC and TD refer to the searchable ciphertext and trapdoor, respectively. mul_v and mul_{vm} represent vector-vector and vector-matrix multiplication in $\mathbb{Z}_q$, respectively. $SampLeft$ represents the cost of the $SampLeft$ sampling function as in [14]. $Conv$ denotes convolution product as defined in Section 2. $GSamp$ denotes a Gaussian Sampling function as in [13]. bp denotes a bilinear pairing operation [52], pa and sm denote point addition and scalar multiplication in $\mathbb{G}$, respectively, and ex denotes exponentiation in $\mathbb{G}_T$.
Public key, private key and SC are stored on the sender, receiver and server's machines, respectively. `PEKS`, `Trapdoor` and `Test` algorithms are run by the senders, receiver and server machines, respectively.
‡ With a slight storage sacrifice, sender can pre-compute one of the convolution products.
† The value of s defines the norm of the Gram-Schmidt coefficient. In [13], the authors set the norm $s \approx \sqrt{\frac{q\mathtt{e}}{2}}$, where $\mathtt{e}$ is the base of natural logarithm.
II σ is the standard deviation of the distribution that is used in the `SampleLeft` algorithm defined in Section 3.2.

Table 2: Parameter sizes of our scheme and its pairing-based counterparts for $\kappa = 192$.

Schemes	***Public Key Size (Kb)***	***Private Key Size (Kb)***	***SC Size (Kb)***	***TD Size (Kb)***
BCO [5]	0.38	0.19	0.57	0.38
NTRU-PEKS	27.2	32	52	27

entropy distribution requirement, and fits better for our target real-life applications (as discussed in Section 1). Nevertheless, for the sake of completeness, to extend the advantages of NTRU-based encryption [51] to ESE, we also instantiated an NTRU-based ESE scheme based on the encrypt-with-hash transformation proposed in [50]. We compared it with its counterpart which was instantiated based on El-Gamal encryption. Our implementations of NTRU-based ESE and El-Gamal ESE (developed on elliptic curves) were run on an Intel i7 6700HQ 2.6GHz CPU with 12GB of RAM . We observed that encryption for NTRU-based ESE takes $0.011ms$ where encryption in El-Gamal ESE takes $2.595ms$. As for decryption, NTRU-based ESE takes $0.013ms$ and El-Gamal ESE takes $0.782ms$. The differences are substantial, since the NTRU-base ESE is $236\times$ and $60\times$ faster in encryption and decryption, respectively.

4 Performance Analysis and Comparison

We first give the analytical performance comparison and then describe our experimental setup and evaluation metrics. We then provide a detailed performance analysis of our scheme in a real cloud setting. To the best of our knowledge, this is the first deployment of PEKS schemes in real-life cloud infrastructure with public data sets.

Table 3: Sender-side computation and communication comparison of our NTRU-PEKS scheme with state-of-the-art.

Schemes		PEKS (*ms*)	Sending PEKS (*ms*)
Commodity Hardware			
BCO [5]	$\kappa = 80$	6.78	80.34
	$\kappa = 192$	66.31	81.75
NTRU-PEKS	$\kappa = 80$	**1.97**	86.32
	$\kappa = 192$	**4.44**	93.78
IoT Device			
BCO [5]	$\kappa = 80$	57.09	83.65
	$\kappa = 192$	904.36	85.12
NTRU-PEKS	$\kappa = 80$	**6.90**	95.60
	$\kappa = 192$	**22.58**	99.54

4.1 Analytical Performance Comparison

As depicted in Table 1, we analyze the analytical performances of our schemes and their pairing-based counterparts in terms of computation, storage and communication. Based on [19], the selected pairing-based counterparts are the most efficient schemes proposed in random oracle and standard models.

Computation: The `Test` algorithm of our NTRU-PEKS scheme only requires one convolution product, which is more efficient than the bilinear pairing operation required in almost all of the existing pairing-based PEKS schemes. As it is shown in Table 2, the dominant operations of the `PEKS` algorithm in our NTRU-PEKS scheme are two convolution products of form $x_1 * x_2$. However, since one of the operands has very small coefficients (i.e., $r \overset{\$}{\leftarrow} \{-1,0,1\}^N$), the convolution products can be computed very efficiently. Specifically, in our case, since N has been selected as a power-of-two integer, the convolution product can be computed in $N \log N$ operations by the Fast Fourier Transform. The `PEKS` algorithm in pairing-based schemes again requires bilinear pairing operations (BCO requires one and ZI requires two) that is more costly than the convolution products. The `Trapdoor` algorithm in NTRU-PEKS requires a Gaussian Sampling, similar as in [24, 13]. This is the most costly operation in our scheme. This algorithm in Boneh et. al.'s scheme only requires one scalar multiplication and consequently, it is the fastest. Although LWE-PEKS offers a very high security promise, our simulation results show that it is not efficient to be deployed with the current state of computing power in commodity hardware.

Storage and Communication: In terms of storage and communication, our schemes require more space than their pairing-based counterparts. For instance the sender needs to store the receiver's public key of size $N|q|$ for our NTRU-PEKS. Referring to Table 2, for 192-bit security, it can be up to $27.2Kb$. As depicted in Table 2 the size of the searchable ciphertext in NTRU-PEKS is significantly larger than the ones in BCO and ZI schemes. However, in section 4.3, we show that in the actual real-life experiment, even though our NTRU-PEKS scheme incurs a higher communication overhead, this overhead is insignificant (as compared to the pairing-based counterparts) even with a moderate-speed home network.

Based on the analytical analysis and the simulation results depicted in Table 1, BCO [5] scheme outperforms ZI [18] in term of computation and storage. Moreover, although LWE-PEKS offers a very high-level of security (maximum security guarantee that lattice-based schemes can

Table 4: Search-Time comparison of our NTRU-PEKS scheme with state-of-the-art in real cloud setting.

Schemes		`Trapdoor` (*ms*)	**Sending Trapdoor** (*ms*)	`Test` (*ms*)
BCO [5]	$\kappa = 80$	1.26	128.08	4.55
	$\kappa = 192$	4.13	137.60	60.75
NTRU-PEKS	$\kappa = 80$	9.71	146.19	**1.05**
	$\kappa = 192$	31.59	151.08	**3.40**

offer), based on our analytical results, due to the storage requirements and the relatively costly computations, it is not feasible to be deployed in practice with current processing and storage capabilities of commodity hardware. Therefore, our focus in our real-life experiments is on BCO and NTRU-PEKS schemes.

4.2 Evaluation Metrics and Experimental Setup

Evaluation Metrics: We implemented our NTRU-PEKS scheme and BCO both on an IoT device (ARM Cortex A53) and a commodity hardware. As aforementioned (Figure 1), PEKS schemes have the potential applications in IoT settings. Since in most of these applications, the IoT devices are conceived to be in the role of a sender, in our experiments, their performance is evaluated in terms of PEKS generation and sending it to the server. In other applications (e.g., secure e-mail system), commodity hardware may also generate PEKS and send it to the e-mail server. Thus, their cost is also evaluated on commodity hardware. The receiver/auditor generates a trapdoor to search over the database and process the results. In most applications, the receiver is conceived to be equipped with a commodity hardware (e.g., Laptop). Therefore, we evaluated trapdoor generation and sending it to the server on commodity hardware only.
Software Libraries and Hardware Configurations:

We fully implemented our NTRU-PEKS scheme in C++, using NTL [53], ZeroMQ and b2 libraries. NTL library was used for low-level arithmetic and matrix operations whereas ZeroMQ was used for network communication. b2 library is a portable C implementation of high-speed Blake2 hash function [54]. Blake2 is used in the full implementation of NTRU-PEKS to map keywords to *vec_ ZZ* type. More specifically, we used Blake2 as a pseudorandom function (PRF), in our `Trapdoor` and `PEKS` algorithms. The implementation of the pairing-based counterpart [5] was obtained from MIRACL library, that was provided as a simulation. We extended this

Table 5: Simulated comparison of our schemes with state-of-the-art

Schemes [†]		`Test` (*ms*)	`PEKS` (*ms*)	`Trapdoor` (*ms*)
BCO [5]	$\kappa = 80$	3.38	2.53	0.36
	$\kappa = 192$	43.39	46.02	2.69
ZI [18]	$\kappa = 80$	8.12	16.37	1.05
	$\kappa = 192$	118.65	194.42	5.61
NTRU-PEKS	$\kappa = 80$	**0.34**	**0.69**	5.15
	$\kappa = 192$	**1.23**	**2.50**	17.35

[†] These results are based on simulations performed on commodity hardware described in Section 4.

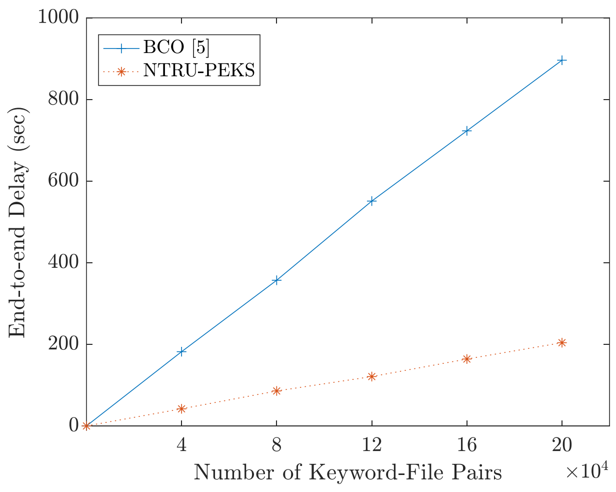


Figure 2: End-to-end search time in real cloud setting for $\kappa = 80$.

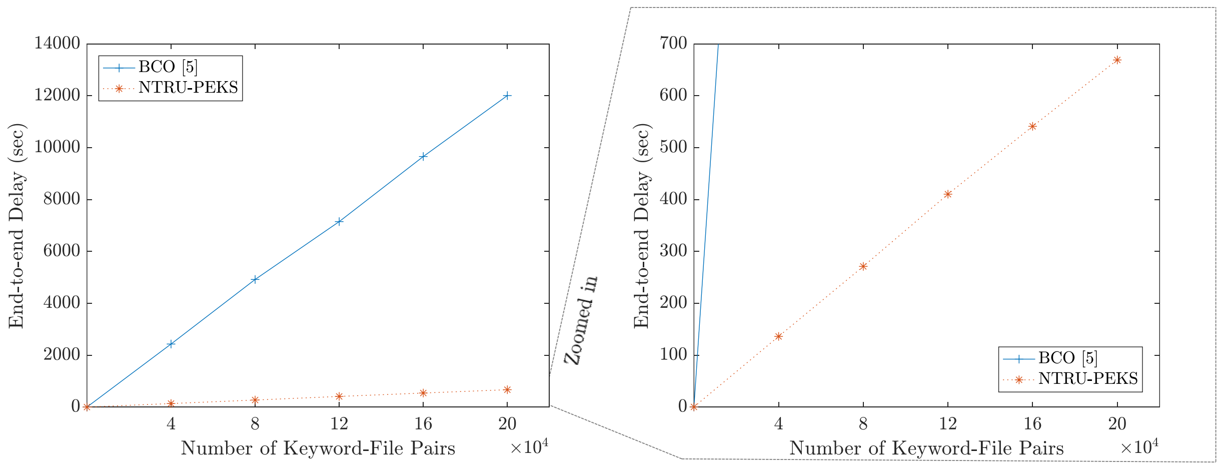


Figure 3: End-to-end search time in real cloud setting for $\kappa = 192$.

implementation for real cloud setting, therefore, in addition to MIRACL, ZeroMQ library is also used in this implementation. Elliptic curves for BCO scheme were selected based on MIRACL, specifically, we used MNT curve with $\kappa = 80$-bit security (with embedding degree $k = 6$) and KSS curve with $\kappa = 192$-bit security (with embedding degree $k = 18$). We made both of the implementations open-sourced for further improvements and adoption[2].

As the commodity hardware, we used an Intel Core i7-6700HQ laptop with a 2.6GHz CPU and 12GB RAM. We selected our IoT device as an ARM Cortex A53 processor, due to its flexibility and low-power consumption [55]. Although it is a low-power device (can run with a small 2200mAh battery), ARM Cortex A53 is equipped with a 64-bit 1GHz processor and 1GB SDRAM. It is extensively preferred in practice since it combines powerful processing power with low energy consumption [55]. At the server-side, we used an Amazon EC2 instance located in Oregon, with a single core Intel(R) Xeon(R) CPU E5-2676 operating at 2.4GH, 2GB RAM and

[2]`https://github.com/Rbehnia/Full_PEKS`

250 GB SSD. Since our parameter sizes are larger than pairing-based schemes, we preferred to be conservative and selected a home network with a 75Mbps connection for both commodity hardware and IoT device. The ping to server is measured as 25.23 ms and 26.78 ms from commodity hardware and IoT device, respectively. In our experiments, we used subsets of publicly available Enron e-mail dataset.

4.3 Performance Evaluation and Comparisons

Tables 3 and 4 depict the experimental results obtained from our real cloud experiments. Comparing the simulation results in Table 5 with the full-fledged implementation results in Table 3 and Table 4 the differences are apparent. We observed that these differences are mainly due to the memory accesses. For each `Test` operation, the server has to access and read the files containing the searchable ciphertext. Another reason for this difference for NTRU-PEKS is the PRF calls. However, since these are implemented with a very high-speed hash function (i.e., Blake2), their impact is rather insignificant. In our real-life experiments, our NTRU-PEKS scheme is $14.93\times$ and $17.87\times$ faster than BCO in the `PEKS` and `Test` algorithms, respectively. However, due to costly Gaussian Sampling, the `Trapdoor` algorithm is $7.65\times$ slower than the one in BCO scheme.

Figures 2 and 3 show the end-to-end delay when the receiver searches over the database, for $\kappa = 80$ and $\kappa = 192$, respectively. As depicted, especially for $\kappa = 192$, there is a significant difference due to the fact that `Trapdoor` algorithm (which is slower in our scheme) is only run once for one search, whereas the `Test` runs linear with the number keyword-file pairs. Therefore, we conducted experiments with varying sizes of database, up to 400,000 keyword-file pairs. For 400,000 keyword-file pairs, BCO algorithm takes 3.34 hours, whereas NTRU-PEKS takes 11.15 minutes. Both of these times are dominated by the `Test` computation on the server-side, since the receiver generates and sends the Trapdoor only once for a each search.

We implemented `PEKS` on both the commodity hardware and IoT devices. We observed significant improvements on the IoT device, wherein specifically, `PEKS` is $40.51\times$ faster than BCO in ARM Cortex A53. This difference is mainly due to the fact that our `PEKS` algorithm does not require any expensive operations (e.g., exponentiation or pairing computation) and matrix operations can be efficiently computed on a wide range of devices. Another advantage of our scheme is the energy efficiency (longer battery life) on IoT devices. The energy consumption of a device is linear with the computation time ($E = V \cdot I \cdot t$, where E = energy, V = voltage, I = current and t = time). Therefore, with our NTRU-PEKS scheme, battery replacement cost and cryptographic overhead on energy consumption highly decreases.

We observed that although parameter sizes for NTRU-PEKS are much larger than the pairing-based counterpart, they do not significantly affect the communication delay. More specifically, Tables 3 and 4 show that communication difference between NTRU-PEKS and BCO is only around 10-15ms. The reason behind this is the round-trip delay time (RTT) from our moderate-speed home network (which is located in the same state as the server, i.e., Oregon) to the server is 25.23 ms and 26.78 ms , for commodity hardware and IoT device, respectively. With a three way handshake in TCP, RTT dominates the total communication cost, resulting with in an insignificant difference between our NTRU-PEKS and BCO. This shows that, although NTRU-based schemes have larger parameters, their efficiency results in a lower end-to-end delay as compared to their communication efficient counter parts (e.g., pairing-based schemes).

4.4 Discussion

We present the first full-fledged implementations for PEKS schemes, and make our implementation open-sourced for further adoption and improvements. Our experiments showed that (i) Simulated results may not reflect the real-life experiments in PEKS schemes. (ii) `Test` algorithm dominates the total search time since it runs $\mathcal{O}(L)$ times (linear with number of keyword-file pairs, L). (iii) The efficiency of `PEKS` algorithm is also crucial since it is to be run on energy-constrained devices in IoT settings. (iv) Given that lattice-based schemes have larger parameters and require significantly larger ciphertexts/trapdoors to be transferred, in a real cloud setting, with a moderate speed network, the communication time difference with pairing-based schemes could be insignificant.

For real-world cases with large databases, our NTRU-PEKS scheme seems to be the only practical solution at this moment. We believe that this is one of the main aspects of our scheme that makes it an attractive candidate to be implemented for real-world applications.

Acknowledgments

The authors would like to thank Leo Ducas and Thomas Prest for their valuable discussions. This work is supported by the NSF CAREER Award CNS-1652389.

References

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica *et al.*, "Above the clouds: A berkeley view of cloud computing," 2009.

[2] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proceeding IEEE Symposium on Security and Privacy S&P 2000*, 2000, pp. 44–55.

[3] A. A. Yavuz and J. Guajardo, "Dynamic searchable symmetric encryption with minimal leakage and efficient updates on commodity hardware," in *Selected Areas in Cryptography – SAC 2015*, O. Dunkelman and L. Keliher, Eds. Springer Berlin Heidelberg, 2016, pp. 241–259.

[4] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, ser. CCS '12. New York, NY, USA: ACM, 2012, pp. 965–976.

[5] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, *Public Key Encryption with Keyword Search*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 506–522.

[6] D. J. Park, K. Kim, and P. J. Lee, "Public key encryption with conjunctive field keyword search," in *Information Security Applications*, C. H. Lim and M. Yung, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 73–86.

[7] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in *Theory of Cryptography Proceedings*, S. P. Vadhan, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 535–554.

[8] E. Shi, J. Bethencourt, T.-H. H. Chan, D. Song, and A. Perrig, "Multi-dimensional range query over encrypted data," in *Proceedings of the 2007 IEEE Symposium on Security and Privacy*, ser. SP '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 350–364.

[9] J. Bringer, H. Chabanne, and B. Kindarji, "Error-tolerant searchable encryption," in *2009 IEEE International Conference on Communications*, 2009, pp. 1–6.

[10] R. Bost, B. Minaud, and O. Ohrimenko, "Forward and backward private searchable encryption from constrained cryptographic primitives," Cryptology ePrint Archive, Report 2017/805, 2017, https://eprint.iacr.org/2017/805.

[11] R. Bost, "Sophos - forward secure searchable encryption," Cryptology ePrint Archive, Report 2016/728, 2016, https://eprint.iacr.org/2016/728.

[12] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi, "Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions," in *Advances in Cryptology – CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005. Proceedings*, V. Shoup, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 205–222.

[13] L. Ducas, V. Lyubashevsky, and T. Prest, *Efficient Identity-Based Encryption over NTRU Lattices*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 22–41.

[14] S. Agrawal, D. Boneh, and X. Boyen, "Efficient lattice (h)ibe in the standard model," in *Advances in Cryptology – EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30 – June 3, 2010. Proceedings*, H. Gilbert, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 553–572.

[15] B. Waters, "Efficient identity-based encryption without random oracles," in *Advances in Cryptology – EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005. Proceedings*, R. Cramer, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 114–127.

[16] X. Boyen, "Lattice mixing and vanishing trapdoors: A framework for fully secure short signatures and more," in *Public Key Cryptography – PKC 2010: 13th International Conference on Practice and Theory in Public Key Cryptography, Paris, France, May 26-28, 2010. Proceedings*, P. Q. Nguyen and D. Pointcheval, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 499–517.

[17] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," in *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, ser. STOC '05. New York, NY, USA: ACM, 2005, pp. 84–93.

[18] R. Zhang and H. Imai, "Generic combination of public key encryption with keyword search and public key encryption," in *Cryptology and Network Security Proceedings*, F. Bao, S. Ling, T. Okamoto, H. Wang, and C. Xing, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 159–174.

[19] C. Bösch, P. Hartel, W. Jonker, and A. Peter, "A survey of provably secure searchable encryption," *ACM Comput. Surv.*, vol. 47, no. 2, pp. 18:1–18:51, 2014.

[20] R. Behnia, A. A. Yavuz, and M. O. Ozmen, "High-speed high-security public key encryption with keyword search," in *Data and Applications Security and Privacy XXXI: 31st Annual IFIP WG 11.3 Conference, DBSec 2017, Philadelphia, PA, USA, July 19-21, 2017, Proceedings*, G. Livraga and S. Zhu, Eds. Cham: Springer International Publishing, 2017, pp. 365–385.

[21] M. Ajtai, "Generating hard instances of lattice problems," in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing.* ACM, 1996, pp. 99–108.

[22] J. Hoffstein, J. Pipher, and J. H. Silverman, *NTRU: A ring-based public key cryptosystem.* Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 267–288.

[23] J. Hoffstein, N. Howgrave-Graham, J. Pipher, J. H. Silverman, and W. Whyte, "NTRUSign: Digital signatures using the NTRU lattice," in *Topics in Cryptology — CT-RSA 2003 Proceedings*, M. Joye, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 122–140.

[24] C. Gentry, C. Peikert, and V. Vaikuntanathan, "Trapdoors for hard lattices and new cryptographic constructions," in *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, ser. STOC '08. New York, NY, USA: ACM, 2008, pp. 197–206.

[25] P. Q. Nguyen and O. Regev, "Learning a parallelepiped: Cryptanalysis of ggh and ntru signatures," *J. Cryptol.*, vol. 22, no. 2, pp. 139–160, Apr. 2009.

[26] D. Micciancio and O. Regev, "Worst-case to average-case reductions based on gaussian measures," *SIAM Journal on Computing*, vol. 37, no. 1, pp. 267–302, 2007.

[27] L. Ducas and P. Q. Nguyen, "Faster gaussian lattice sampling using lazy floating-point arithmetic," in *Advances in Cryptology – ASIACRYPT 2012 Proceedings*, X. Wang and K. Sako, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 415–432.

[28] C. Peikert, "Bonsai trees (or, arboriculture in lattice-based cryptography)," Cryptology ePrint Archive, Report 2009/359, 2009, http://eprint.iacr.org/2009/359.

[29] D. Micciancio and S. Goldwasser, *Complexity of Lattice Problems: a cryptographic perspective*, ser. The Kluwer International Series in Engineering and Computer Science. Boston, Massachusetts: Kluwer Academic Publishers, Mar. 2002, vol. 671.

[30] S. Halevi, "A sufficient condition for key-privacy." *IACR Cryptology ePrint Archive*, vol. 2005, p. 5, 2005.

[31] G. Di Crescenzo and V. Saraswat, *Public Key Encryption with Searchable Keywords Based on Jacobi Symbols.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 282–296.

[32] L. Ducas, A. Durmus, T. Lepoint, and V. Lyubashevsky, "Lattice signatures and bimodal gaussians," in *Advances in Cryptology – CRYPTO 2013 Proceedings, Part I*, R. Canetti and J. A. Garay, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 40–56.

[33] N. Gama and P. Q. Nguyen, "Predicting lattice reduction," in *Advances in Cryptology – EUROCRYPT 2008: 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, N. Smart, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 31–51.

[34] Y. Chen and P. Q. Nguyen, "Bkz 2.0: Better lattice security estimates," in *Advances in Cryptology – ASIACRYPT 2011 Proceedings*, D. H. Lee and X. Wang, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 1–20.

[35] V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," in *Advances in Cryptology – EUROCRYPT 2010 Proceedings*, H. Gilbert, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 1–23.

[36] ——, "A toolkit for ring-lwe cryptography," in *Advances in Cryptology – EUROCRYPT 2013 Proceedings*, T. Johansson and P. Q. Nguyen, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 35–54.

[37] D. Micciancio and O. Regev, "Lattice-based cryptography," in *Post-Quantum Cryptography*, D. J. Bernstein, J. Buchmann, and E. Dahmen, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 147–191.

[38] M. Ajtai, "Generating hard instances of the short basis problem," in *Automata, Languages and Programming: 26th International Colloquium, ICALP'99 Prague, Czech Republic, July 11–15, 1999 Proceedings*, J. Wiedermann, P. van Emde Boas, and M. Nielsen, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 1–9.

[39] J. Alwen and C. Peikert, "Generating shorter bases for hard random lattices," *Theory of Computing Systems*, vol. 48, no. 3, pp. 535–553, Apr 2011.

[40] D. Cash, D. Hofheinz, and E. Kiltz, "How to delegate a lattice basis," Cryptology ePrint Archive, Report 2009/351, 2009, http://eprint.iacr.org/2009/351.

[41] M. Bellare, A. Boldyreva, A. Desai, and D. Pointcheval, "Key-privacy in public-key encryption," in *Advances in Cryptology — ASIACRYPT 2001 Proceedings*, C. Boyd, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 566–582.

[42] R. Lindner and C. Peikert, "Better key sizes (and attacks) for lwe-based encryption," in *Topics in Cryptology – CT-RSA 2011: The Cryptographers' Track at the RSA Conference 2011, San Francisco, CA, USA, February 14-18, 2011. Proceedings*, A. Kiayias, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 319–339.

[43] J. Baek, R. Safavi-Naini, and W. Susilo, "Public key encryption with keyword search revisited," in *Computational Science and Its Applications – ICCSA 2008 Proceedings, Part I*, O. Gervasi, B. Murgante, A. Laganà, D. Taniar, Y. Mun, and M. L. Gavrilova, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 1249–1259.

[44] J. W. Byun, H. S. Rhee, H.-A. Park, and D. H. Lee, "Off-line keyword guessing attacks on recent keyword search schemes over encrypted data," in *Secure Data Management: Third VLDB Workshop Proceedings*, W. Jonker and M. Petković, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 75–83.

[45] C. Hu and P. Liu, "A secure searchable public key encryption scheme with a designated tester against keyword guessing attacks and its extension," in *Advances in Computer Science, Environment, Ecoinformatics, and Education Proceedings, Part II*, S. Lin and X. Huang, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 131–136.

[46] L. Fang, W. Susilo, C. Ge, and J. Wang, "Public key encryption with keyword search secure against keyword guessing attacks without random oracle," *Information Sciences*, vol. 238, pp. 221 – 241, 2013.

[47] I. R. Jeong, J. O. Kwon, D. Hong, and D. H. Lee, "Constructing PEKS schemes secure against keyword guessing attacks is possible?" *Computer Communications*, vol. 32, no. 2, pp. 394 – 396, 2009.

[48] J. W. Bos, C. Costello, M. Naehrig, and D. Stebila, "Post-quantum key exchange for the tls protocol from the ring learning with errors problem," in *2015 IEEE Symposium on Security and Privacy*, 2015, pp. 553–570.

[49] E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe, "Post-quantum key exchange—a new hope," in *25th USENIX Security Symposium (USENIX Security 16).* Austin, TX: USENIX Association, 2016, pp. 327–343.

[50] M. Bellare, A. Boldyreva, and A. O'Neill, "Deterministic and efficiently searchable encryption," in *Advances in Cryptology - CRYPTO 2007 Proceedings*, A. Menezes, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 535–552.

[51] W. Whyte, N. Howgrave-Graham, J. Hoffstein, J. Pipher, J. H. Silverman, and P. S. Hirschhorn, "IEEE P1363. 1 Draft 10: Draft Standard for Public Key Cryptographic Techniques Based on Hard Problems over Lattices." *IACR Cryptology EPrint Archive*, vol. 2008, p. 361, 2008.

[52] D. Boneh and M. Franklin, *Identity-Based Encryption from the Weil Pairing.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 213–229.

[53] V. Shoup, "NTL: A library for doing number theory," http://www.shoup.net/ntl, 2003.

[54] J.-P. Aumasson, L. Henzen, W. Meier, and R. C.-W. Phan, "Sha-3 proposal blake," Submission to NIST (Round 3), 2010. [Online]. Available: http://131002.net/blake/blake.pdf

[55] V. Vujović and M. Maksimović, "Raspberry pi as a sensor web node for home automation," *Comput. Electr. Eng.*, vol. 44, no. C, pp. 153–171, May 2015.