# Lattice-Based Public Key Searchable Encryption from Experimental Perspectives

Rouzbeh Behnia, Muslum Ozgur Ozmen, Attila A. Yavuz†, *Member, IEEE,*

**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 promises. In this paper, we propose two novel lattice-based PEKS schemes that offer a high computational efficiency along with better security assurances than that of the 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 scheme and benchmarked its performance as deployed on Amazon Web Services cloud infrastructures.

**Index Terms**—applied cryptography, Public Key Encryption with Keyword Search (PEKS), lattice-based cryptography, searchable ecnryption

## 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-sized 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 applications.

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 *designated receiver*. The designated 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 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*.

Figure 1 illustrates a potential application 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., cell phone, desktop, etc.), wants her e-mails to be routed to her devices based on the keywords associated with 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, 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 is in storing and searching on private log files. PEKS schemes can enable a heterogeneous set of devices to send 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,

† *Work done in part while Attila A. Yavuz was at Oregon State University, Corvallis, OR, USA.*

- *Rouzbeh Behnia and Muslum Ozgur Ozmen are with the Department of Electrical Engineering and Computer Science, Oregon State University, Corvallis, OR, USA.*
  *E-mail: {rbehnia, ozmenmu}@oregonstate.edu*
- *Attila A. Yavuz is with the Department of Computer Science and Engineering, University of South Florida, Tampa, FL, USA.*
  *E-mail: attilaayavuz@usf.edu*

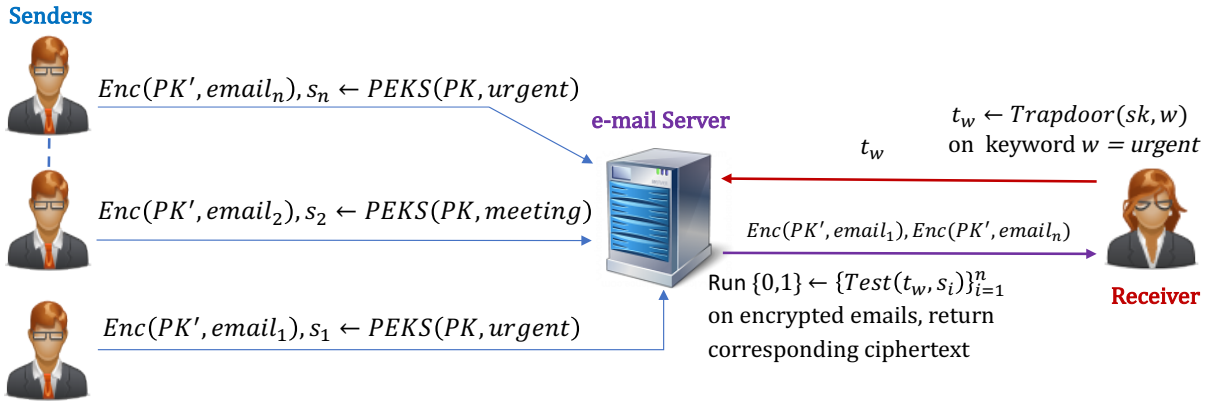


Fig. 1. PEKS scheme in secure email system

which requires the execution of a `Test` algorithm for each keyword-file pair in the database. The existing PEKS schemes (e.g., [5], [7]) introduce a significant end-to-end computational delay due to at least one pairing operation that is required in the `Test` algorithm which is to be called linear to the total 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 a continuous increment of 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 post-quantum security promises.
- 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 for providing a full-fledged implementation of PEKS schemes and benchmarking their deployment 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

The main goal of this paper is developing a significantly more efficient PEKS scheme and proposing a PEKS scheme with worst-case reduction by harnessing the existing lattice-based IBE schemes using Abdalla et. al. generic anonymous-IBE to PEKS transformation. We also provide an extensive analysis to highlight the advantages and disadvantages of such schemes.

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:

- First 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 consistency 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 provable security in Random Oracle Model (ROM). (ii) Our second scheme is referred to as LWE-PEKS, which leverages IBE constructions in [14], [15] 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 [16] and fast arithmetic operations (e.g., Fast Fourier Transform) over polynomial rings $\mathbb{Z}[x]/(x^N + 1)$. As it is shown in Table 2, the NTRU-PEKS scheme has significantly more efficient `Test` and `PEKS` algorithms than those in [5], [17], which are currently considered as the most efficient PEKS alternatives [18]. The efficiency of the `Test` algorithm is of vital importance, since it is executed by the server once per every keyword-file pair in the database which results in $\mathcal{O}(L)$ computation overhead, where $L$ is the total number of keyword-file pairs. The efficiency of the `PEKS` algorithm facilitates the implementation of PEKS schemes on battery-limited devices. Due to its computational efficiency, despite having larger parameters, we showed the deployment of NTRU-PEKS on actual cloud achieves a superior end-to-end response time as compared to its counterparts (see Section 4).
- Long-term Security: Both of our constructions are based on lattice-based tools that provide long-term secu-

rity 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: We provide a full-fledged implementation of our NTRU-PEKS scheme and its most efficient pairing-based counterpart with deployment in actual cloud setting 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 [19] and the benchmark results of the 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 (see Section 4).

### 1.3 Limitations

We present two lattice-based schemes, one with security in the random oracle model and one in the standard model. In this section, we point out their limitations as compared to their conventional pairing-based counterparts.

The `Trapdoor` algorithm of NTRU-PEKS (secure in the random oracle model) is slower than that of BCOP. One should note that the `Trapdoor` algorithm is to be *executed once per each query* on the receiver's side and has negligible effect on the end-to-end delay (e.g., see Figure 2). Another downside of NTRU-PEKS is the parameter sizes, for instance, the searchable ciphertext size in NTRU-PEKS is significantly larger than that of BCOP [5]. We should note that while the storage blow-up remains a concern, the increased communication overhead and disk access time have negligible effects on the end-to-end delay (e.g., see Figure 2). NTRU-PEKS provides very efficient `PEKS` and `Test` algorithms. The efficient `PEKS` algorithm provides significant efficiency for the sender as it is called to generate searchable ciphertext for each keyword to be attached to the file. The `Test` algorithm is the main factor in reducing end-to-end delay in PEKS schemes since *it is executed once for every keyword-file pairs in the database* (e.g., see Figure 2). Given all the computation efficiency gains, we believe the storage blow-up might be a favorable trade-off for certain applications where end-to-end delay and long-term security are more critical than the storage.

In the line of proposing PEKS schemes in the standard model, to the best of our knowledge, the proposed LWE-PEKS scheme provides the highest security promise with reduction to worst-case problems. However, as shown in Section 4, it is significantly less efficient (in both computation and storage) than Zhang and Imai's pairing-based PEKS scheme which is also secure in the standard model [17].

### 1.4 Differences Between this Article and its Preliminary Version in [19]

We highlight the main differences of this article and its preliminary version in [19] as follows: (i) We provide a full-fledged implementation of NTRU-PEKS and its most efficient counterpart (i.e., BCOP [5]) on commodity hardware, ARM processor and cloud server. (ii) We introduce the first LWE-based PEKS scheme in the standard model (LWE-PEKS) with a security reduction to the worst-case lattice-based problems. (iii) We provide a detailed cost dissection analysis to measure the computation cost, communication overhead, and disk access time of the proposed schemes. (iv) We expanded our security discussion and provided a set of recommended parameter lists for our scheme.

## 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. We represent vectors as bold letters $\mathbf{v}$, while scalars are represented as non-bold letters i.e., $v$. $\mathcal{A}^{\mathcal{O}_1,\ldots,\mathcal{O}_n}(.)$ denotes that algorithm $\mathcal{A}$ 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 Identity-Based Encryption

**Definition 1.** *An IBE scheme is a tuple of four algorithms* $IBE =$ `(Setup,Extract,Enc,Dec)` *defined as follows.*

- $(mpk, msk) \leftarrow$ `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 $msk$.
- $sk \leftarrow$ `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 $sk$.
- $c \leftarrow$ `Enc`$(m, id, mpk)$: On the input of a message $m \in \{0,1\}^*$, identity $id$, and $mpk$, this algorithm outputs a ciphertext $c$.
- $m \leftarrow$ `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 $c$.

Halevi in [20] introduced a condition for an IND-CPA encryption scheme to offer the notion of anonymity (ANO-CPA). This condition requires that given two public keys $PK_0$ and $PK_1$ and a ciphertext $c$, encrypted under $PK_b$, for $b \in \{0,1\}$, a computationally unbounded adversary should have a negligible advantage in determining $b$. Later, Abdalla et. al. [12] extended this condition to identity-based encryption schemes by including the handling of random oracles and weakening the statistical requirement to a computational one. The following definition defines the new anonymity under chosen plaintext attack for an IBE scheme (IBE-ANO-RE-CPA).

**Definition 2.** *Given an IBE scheme, the* `KeyQuery`$(\cdot)$ *oracle, as defined below, we associate a bit* $b \in \{0,1\}$ *to the adversary* $\mathcal{A}$ *in the following experiment* $Exp^{\text{IBE-ANO-RE-CPA-}b}_{IBE,\mathcal{A}}(1^k)$*:*

$\underline{\texttt{KeyQuery}(id)}$
$idSet \leftarrow idSet \cup id$
$return\ sk$

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

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

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

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

After queries to $\texttt{KeyQuery}(\cdot)$ and random oracle $H$ in the $find$ stage, $\mathcal{A}$ outputs a challenge $(id_0, id_1, m)$. $\mathcal{A}$ will then receive $c$, which is an encryption of the message $m$ under $id_b$ where $b \in \{0, 1\}$ is the output of a fair coin flip. In the $guess$ stage, given $c$, $\mathcal{A}$ will output its decision bit $b'$ and wins if $b' = b$ where $id_0$ and $id_1$ were never queried to the $\texttt{KeyQuery}(\cdot)$ oracle.

**Lemma 1.** *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]. □

**Abdalla et. al. IBE to PEKS transformation [12]**: The authors prove that if an IBE scheme is IBE-ANO-CPA (in the sense of [20]) and IBE-ANO-CPA, then one can obtain a secure PEKS scheme. We use this transformation to obtain NTRU-PEKS and LWE-PEKS

### 2.2 Public Key Encryption with Keyword Search

A PEKS scheme consists of the following algorithms.

**Definition 3.** *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.

Keyword indistinguishability against an adaptive chosen-keyword attack for a PEKS scheme (PEKS-IND-CKA) is defined in the following experiment.

**Definition 4.** *Given a PEKS scheme, and the* $\texttt{TdQuery}(\cdot)$ *oracle as defined below, we associate a bit* $b \in \{0, 1\}$ *to the adversary* $\mathcal{A}$ *in the following experiment* $Exp^{\text{PEKS-IND-CKA-b}}_{PEKS,\mathcal{A}}(1^k)$.

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

*Experiment* $Exp^{\text{PEKS-IND-CKA-b}}_{PEKS,\mathcal{A}}(1^k)$

$wSet \leftarrow \emptyset$, $(pk, sk) \leftarrow \texttt{KeyGen}(1^k)$
for a random oracle $H$
$(w_0, w_1) \leftarrow \mathcal{A}^{\texttt{TdQuery}(.),H}(pk)$ :*find* stage
$s_w \leftarrow \texttt{PEKS}^H(pk, w_b)$
$b' \leftarrow \mathcal{A}^{\texttt{TdQuery}(.),H}(s_w)$ :*guess* stage
*if* $\{w_0, w_1\} \cap wSet = \emptyset$ *return* $b'$ *else, return* 0

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

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

After queries to $\texttt{TdQuery}(\cdot)$ and hash oracle $H$, in the $find$ stage, $\mathcal{A}$ outputs a challenge $(w_0, w_1)$. $\mathcal{A}$will then receive $s_w$, which is a searchable encryption of $w_b$ where $b \in \{0, 1\}$ is the output of a fair coin flip, under $pk$ . In the $guess$ stage, given $s_w$, $\mathcal{A}$ will output its decision bit $b'$ and wins if $b' = b$ where $w_0$ and $w_1$ were never queried to the $\texttt{TdQuery}(\cdot)$ oracle.

#### 2.2.1 Consistency of PEKS

Due to the properties of NTRU-based encryption scheme, and following the work of [21], we investigate the consistency of our scheme from two aspects, namely, right-keyword consistency and adversary-based consistency [12].

**Right-Keyword Consistency:** Right-keyword consistency implies the success of a search query to retrieve records associated with keyword $w$ for which the $\texttt{PEKS}$ algorithm had computed a searchable ciphertext. More specifically, right-keyword consistency refers to the decryption error in the underlying IBE scheme which leads to the inconsistency of the $\texttt{Test}$ algorithm in the resulting PEKS scheme.

**Adversary-Based Consistency:** We define the adversary-based consistency [12] in the following definition.

**Definition 5.** *Adversary-based consistency of a PEKS scheme is defined in the following experiment.*
*Experiment* $Exp^{\text{PEKS-Consist}}_{PEKS,\mathcal{A}}(1^k)$

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

We note that for schemes with security in the standard model, the random oracle $H$ is eliminated in Definitions 2, 4 and 5 .

### 2.3 Tools and Definitions

**Integer Lattices:** Let $\mathbf{B} = [\mathbf{b_1}|\dots|\mathbf{b_m}] \in \mathbb{R}^{m \times m}$ 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 $\Lambda$ generated by $B$ is the set,

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

**Definition 6.** *For a prime $q$, $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{u} \in \mathbb{Z}_q^n$, define:*
$\Lambda_q(\mathbf{A}) := \{\mathbf{e} \in \mathbb{Z}^m : \exists \mathbf{s} \in \mathbb{Z}_q^n \text{ where } \mathbf{A}^\top \mathbf{s} = \mathbf{e} \mod q\}$
$\Lambda_q^\perp(\mathbf{A}) := \{\mathbf{e} \in \mathbb{Z}^m : \mathbf{Ae} = 0 \mod q\}$
$\Lambda_q^u(\mathbf{A}) := \{\mathbf{e} \in \mathbb{Z}^m : \mathbf{Ae} = \mathbf{u} \mod q\}$

**NTRU Lattices:** Ajtai [22] 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. [23] proposed a very efficient public key encryption scheme based on NTRU lattices. Regev [16] 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 7.** *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 the inefficiency of standard lattice operations. As proposed by [24], 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 $\Lambda_{h,q}$.

**Definition 8.** *(Gram-Schmidt norm [25]) 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 9.** *(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. [25] proposed a technique to use a short basis as a trapdoor without disclosing any information about the short basis and prevent attacks similar as in [26].

**Definition 10.** *An N-dimensional Gaussian function $\rho_{\sigma,c} : \mathbb{R} \to (0, 1]$ 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 $\Lambda$ 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* [27], 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 11.** *(Smoothing Parameter [27]) For any N-dimensional lattice $\Lambda$, its dual $\Lambda^*$ 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. [25] defined a requirement on the size of $\sigma$ related to the smoothing parameter. In [13], Ducas et al. showed that using Kullback-Leibler divergence, the required width of $\sigma$ can be reduced by a factor of $\sqrt{2}$. Based on [13], [25], [28], for positive integers $n, \lambda, \epsilon \leqslant 2^{-\lambda/2}/(4\sqrt{2N})$, any basis $\mathbf{B} \in \mathbb{Z}^{N \times N}$ and any target vector $\mathbf{c} \in \mathbb{Z}^{1 \times n}$, the algorithm $(\mathbf{v}_0 \leftarrow \texttt{Gaussian-Sampler}(\mathbf{B}, \sigma, \mathbf{c}))$ as defined in [13], [25] 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.

In [16], Regev has shown that for a certain error distribution $\chi$, denoted $\bar{\Psi}_\alpha$, the LWE problem is as hard as the worst-case SIVP and GapSVP under quantum reduction.

**Definition 12.** *(Distribution of $\bar{\Psi}_\alpha$ [16], [29]) 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 $\frac{\alpha}{\sqrt{2\pi}}$.*

**Definition 13.** *(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 $\bar{f}(x) = f_0 - \sum_{i=1}^{N-1} f_{N-i} x^i$.*

## 3 Proposed Schemes

In this section, we first propose our scheme based on NTRU lattices (i.e., 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 using Abdalla et. al. transformation [12] to transform Ducas et. al. IBE scheme [13]. For this transformation to work, aside being IND-CPA, the underlying IBE scheme should be anonymous in the sense of Definition 2. We show that Ducas et. al.'s IBE scheme is indeed anonymous via Theorem 2. Our NTRU-PEKS scheme consists of the following algorithms.

$(pk, sk) \leftarrow \texttt{KeyGen}(q, N)$: 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* $\leftarrow \max(\|(g, -f)\|,$

$\left\|(\frac{q\bar{f}}{f*\bar{f}+g*\bar{g}}, \frac{q\bar{g}}{f*\bar{f}+g*\bar{g}})\right\|$). 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 $(pk \leftarrow h, sk \leftarrow \mathbf{B})$.

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

$\mathbf{t_w} \leftarrow \texttt{Trapdoor}(sk, w)$: Given the receiver's private key $sk$, and a keyword $w \in \{0,1\}^*$, the receiver computes $\mathbf{t} \leftarrow H_1(w)$ and using the sampling algorithm $\texttt{Gaussian-Sampler}(\mathbf{B}, \sigma, (\mathbf{t}, 0))$, samples $s$ and $\mathbf{t_w}$ such that $\mathbf{s} + \mathbf{t_w} * \mathbf{h} = \mathbf{t}$.

$d \leftarrow \texttt{Test}(pk, t_w, s_w)$: On the input of a receiver's public key $pk$, a trapdoor $t_w$ and a searchable ciphertext $s_w = \langle \mathbf{c_0}, \mathbf{c_1}, H_2(\mathbf{k}, \mathbf{c_1}) \rangle$, 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 2.** *Given a public-private key pair* $(\mathbf{h}, \mathbf{B}) \leftarrow \texttt{KeyGen}(q, N)$, *a searchable ciphertext* $s_w \leftarrow \texttt{PEKS}(pk, w)$, *and a trapdoor generated by the receiver* $\mathbf{t_w} \leftarrow \texttt{Trapdoor}(sk, w)$ *our proposed scheme is complete.*

*Proof.* To show the completeness of our scheme for $s_w = \langle \mathbf{c_0}, \mathbf{c_1}, H_2(\mathbf{k}, \mathbf{c_1}) \rangle$, 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 $\kappa = 192$. As highlighted in [30], for such large values of $q$, the probability of decryption error is negligible with respect to the security parameter.

**Theorem 1.** *$\mathcal{A}$'s advantage in breaking the consistency of NTRU-PEKS scheme in the sense of Definition 5, after making $q_1$ and $q_2$ queries to $H_1(\cdot)$ and $H_2(\cdot)$, respectively, is:*

$$\mathcal{A}_{PEKS,\mathcal{A}}^{PEKS\text{-}Consist}(1^k) \leq \frac{(q_1+2)^2}{N2^{\log_2 q}} + \frac{(q_2+2)^2}{N2^{\log_2 q}} + \epsilon$$

*where the term $\epsilon$ is negligible in term of the security parameter $\kappa$.*

*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.

- $(w_0, w_1) \leftarrow \mathcal{A}^H(pk)$: $\mathcal{A}$ 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 $\mathbf{c_1} = \mathbf{r} * \mathbf{H}(\mathbf{w_0}) + \mathbf{e_2} + \lfloor \frac{q}{2} \rfloor \mathbf{k}$ for a random selection of $\mathbf{r}, \mathbf{e_1}, \mathbf{e_2} \xleftarrow{\$} \{-1, 0, 1\}^N$, $\mathbf{k} \xleftarrow{\$} \{0,1\}^N$, 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 $\frac{(q_1+2)^2}{N2^{\log_2 q}}$ and $\frac{(q_2+2)^2}{N2^{\log_2 q}}$, 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 [31], 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 [32], [33], $\gamma \approx 1.004$ guarantees intractability and provides approximately 192-bit security.

Following Lemma 1, 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 2.** *The IBE scheme of Ducas et al. is anonymous in the sense of Definition 2 under the decision ring-LWE problem.*

*Proof.* Since the output of the PEKS algorithm of our scheme corresponds to the encryption algorithm of [34], [35], for $\mathcal{A}$ to determine $s_w$ corresponds to which keyword with any probability $\Pr \geq \frac{1}{2} + \epsilon$, for any non-negligible $\epsilon$, 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 $\mathbf{c}_1 = \mathbf{r} * H(w) + \mathbf{e}_2 + \lfloor \frac{q}{2} \rfloor \mathbf{k}$ that is also statistically close to uniform. Therefore, the adversary's view of $\langle \mathbf{c_0}, \mathbf{c}_1, H_2(\mathbf{k}, \mathbf{c}_1) \rangle$ 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 [36]. □

**Theorem 3.** *If there exists an adversary $\mathcal{A}$ that can break PEKS-IND-CKA of NTRU-PEKS scheme as in Definition 4, one can build an adversary $\mathcal{F}$ that uses $\mathcal{A}$ as subroutine and breaks the security of the IBE scheme in Definition 2.*

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

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

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

- 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 $(\mathbf{s}, \mathbf{t_w}) \xleftarrow{\$} \texttt{Gaussian-Sampler}(msk, \sigma, (H(w), 0))$ 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 $\texttt{Enc}(m, w_0, w_1)$ 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.

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

□

**Discussion on Subfield Attacks:** Subfield attacks target the presence of a subfield in ideal lattices to solve the over-stretched version of the NTRU problem. Recently, there have been noticeable advancements to extend subfield attacks on ideal lattices [37], [38], [39]. However, as also shown in [38], these attacks do not affect the parameters that are used in our scheme.

### 3.2 Lattice-Based PEKS Scheme in the Standard Model

In this section, we present our highly secure LWE-PEKS scheme using Abdalla et. al. transformation [12] to transform Agrawal et. al. IBE scheme [14] . For this transformation to work, aside from being IND-CPA, the underlying IBE scheme should be anonymous in the sense of Definition 2. While the scheme proposed in [14] does provide the anonymity required by Abdalla et. al. transformation [12], in the adaptive security of the scheme, the number of adversarial queries was bounded by $q$. In other words, only the bounded form of the security of the scheme was proven in [14]. This restriction was later lifted in a more refined analysis in [15].

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 [40], 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 [14], [41] which is used in the `KeyGen` algorithm of the LWE-PEKS scheme.

$(\mathbf{A}, \mathbf{S}) \leftarrow \texttt{TrapGen}(q, n)$ : Given a prime $q$, a positive $n$ and $\delta = \frac{1}{3}$, there is a polynomial time algorithm $\texttt{TrapGen}(q, n)$ 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 $\texttt{TrapGen}(q, n)$.

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 [29], [42].

$\mathbf{e} \leftarrow \texttt{SampleLeft}(\mathbf{A}, \mathbf{M_1}, \mathbf{T_A}, \mathbf{u}, \sigma)$ : Given an $n$-rank matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, a matrix $\mathbf{M_1} \in \mathbb{Z}_q^{n \times m_1}$, 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.

$(pk, sk) \leftarrow \texttt{KeyGen}(\lambda)$: On the input of the security parameter $\lambda$, and the parameters $q, m, n, \sigma, \alpha$ (set as instructed in the following section), the receiver works as follows to generate her key pair.

1) Use $\texttt{TrapGen}(q, n)$ to pick a random matrix $\mathbf{A_0} \in \mathbb{Z}_q^{n \times m}$ 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 $\mathbf{A_1}, \ldots \mathbf{A_l}, \mathbf{B} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$ and a random vector $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_q^n$.
3) Output the public key $pk \leftarrow (\mathbf{A_0}, \mathbf{A_1}, \ldots, \mathbf{A_l}, \mathbf{B}, \mathbf{u})$ and secret key $sk \leftarrow \mathbf{T_{A_0}}$.

$s_w \leftarrow \texttt{PEKS}(pk, w)$: On the input of the $pk$ and an $l$-bit keyword $w = (b_1, \ldots, b_l) \in \{1, -1\}^l$, the sender picks $b'_j \xleftarrow{\$} \{0, 1\}$ for $j = 1, \ldots, \kappa$, 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 $i = 1, \ldots, l$ and set $\mathbf{R_{b'_j}} \leftarrow \sum_{i=1}^{l} b_i \mathbf{R_{i_j}} \in \{-l, \ldots, l\}^{m\times m}$.
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 $\mathbf{c_{1_j}} \leftarrow \mathbf{F}_\mathbf{w}^\top \mathbf{s_j} + \begin{bmatrix} \mathbf{y_j} \\ \mathbf{z_j} \end{bmatrix} \in \mathbb{Z}_q^{2m}$.
4) Output searchable ciphertext $s_{w_j} = (c_{0_j}, \mathbf{c_{1_j}}, b'_j)$ for $j = 1, \ldots, \kappa$.

$\mathbf{t_w} \leftarrow \texttt{Trapdoor}(pk, sk, w)$: On the input of the keys and a keyword $w = (b_i, \ldots, b_l) \in \{1, -1\}^l$, the receiver 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 $D_{\Lambda_q^u F_w, \sigma}$.

$d \leftarrow \texttt{Test}(\mathbf{t_w}, s_w)$: Given a trapdoor $td$ for a keyword $w = (b_i, \ldots, b_l) \in \{1, -1\}^l$, and $\kappa$ searchable ciphertexts $s_{w_j} = (c_{0_j}, c_{1_j}, b'_j)$ for $j = 1, \ldots, \kappa$ 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 $|\nu_j - \lfloor \frac{q}{2} \rfloor| < \lfloor \frac{q}{4} \rfloor$, set $\nu_j \leftarrow 1$ and otherwise, $\nu_j \leftarrow 0$.
ii. If $\nu_j = b'_j$ holds for all $1 \leq j \leq \kappa$, 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 3.** *Given a public-private key pair $(pk, sk) \leftarrow KeyGen(\lambda)$, a searchable ciphertext $s_w \leftarrow PEKS(pk, w)$ 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 $\nu_j \leftarrow c_{0_j} - td c_{1_j}$ for all $j = 1, \ldots, \kappa$. To affirm this, we work as follows.

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

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

TABLE 1
Parameter sizes of our schemes and their pairing-based counterparts for $\kappa = 192$.

| Schemes | *Public Key (Kb)* | *Private Key (Kb)* | *SC*† *(Kb)* | *TD*‡ *(Kb)* |
|---|---|---|---|---|
| BCOP [5] | 0.38 | 0.19 | 0.57 | 0.38 |
| ZI [17] | 0.76 | 0.19 | 0.89 | 0.57 |
| **NTRU-PEKS** | 27.2 | 32 | 52 | 27 |
| **LWE-PEKS** | 57,216 | 14,647,390 | 186,867 | 112 |

† SC refers to searchable ciphertext (i.e., the output of the `PEKS` algorithm).
‡ TD refers to trapdoor (i.e., the output of the `Trapdoor` algorithm).

Where $x - t_w^\top \begin{bmatrix} y_j \\ z_j \end{bmatrix}$ is the error term. Based on Lemma 22 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. $\sigma$ 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 [16] reduction to work, set $q > 2\sqrt{n}/\alpha$ .

□

To achieve these requirements, we set $q \geq m^{2.5} \cdot \omega(\sqrt{\log n}), m = 6n^{1+\delta}$, $\sigma = ml \cdot \omega(\sqrt{\log n})$, $\alpha = [l^2 m^2 \cdot \omega(\sqrt{\log n})]^{-1}$. This is based on the assumption that $\delta$ such that $n^\delta > \lceil \log_2 q \rceil$.

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

*Proof.* For the `Test` algorithm to return 1, all the $\kappa$ bits of $b'_j$ and $\nu_j$ 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 [42] (and the signing algorithm in [29]) 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 [29], [42], our LWE-PEKS scheme is consistent. □

### 3.2.2 Security Analysis

Based on [32], the hardness of lattice problems is measured using the root Hermite factor.

Following Lemma 1, 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 [12], [43]. 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 [15]. This implies that based on Lemma 1, the resulting LWE-PEKS scheme is secure in the standard model. Based on [32], 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 [33], [44] to achieve $\approx$ 192-bit security for message and randomness recovery attack with $\gamma \approx 1.0042$.

### 3.3 Secure Channel Requirement

Baek et al. [45] 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. SCF-PEKS schemes require the server to be initialized with a key pair through which the receiver encrypts the trapdoor before sending it over a public (insecure) channel. Upon receiving the encrypted trapdoor, the server will first decrypt the trapdoor before initiating the `Test` algorithm. Offline keyword-guessing attack, as introduced by Byun et al. [46], 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 [18]. Since Byun et al.'s work [46], there have been a number of attempts in proposing schemes that address keyword guessing attacks [47], [48], [49]. However, in all of the proposals, once the trapdoor is revealed to the server, keyword guessing attacks remain a perpetual problem [49]. Jeong et al. [49] 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 [50], [51] 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 SCF-PEKS. Since in such protocols, after the handshake protocol, all communications are encrypted using symmetric encryption.

### 3.4 Alternative NTRU-based Constructions

Bellare et al. [52] 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 a 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 responses 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.

TABLE 2
Micro Benchmark (One Execution) of our schemes and their pairing-based counterparts

| Schemes † | | `PEKS` (*ms*) | `Trapdoor` (*ms*) | `Test` (*ms*) |
|---|---|---|---|---|
| BCOP [5] | $\kappa = 80$ | 6.78 | 1.26 | 4.55 |
| | $\kappa = 192$ | 66.31 | 4.13 | 60.75 |
| ZI [17] | $\kappa = 80$ | 48.87 | 4.12 | 12.93 |
| | $\kappa = 192$ | 301.14 | 10.61 | 166.12 |
| **NTRU-PEKS** | $\kappa = 80$ | **1.97** | 9.71 | **1.05** |
| | $\kappa = 192$ | **4.44** | 31.59 | **3.40** |
| **LWE-PEKS** | $\kappa = 80$ | 2115.12 | 814.9 | 643 |
| | $\kappa = 192$ | 3727 | 1509.19 | 1214 |

In this paper, we focused on PEKS scheme as it does not have consistency issues or a min-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 [53] to ESE, we also instantiated an NTRU-based ESE scheme based on the encrypt-with-hash transformation proposed in [52]. 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 and cost dissection of our scheme in a 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.

### 4.1 Analytical Performance Comparison

In this section, we first present a set of recommended parameters for our scheme which were used in our instantiations and then discuss the analytical performance from the computation

**Parameters:** For NTRU-PEKS, with 80-bit security, we set $N = 512$ and $q = 2^{23}$. For 192-bit security, we set $N = 1024$ and $q = 2^{27}$. Following [13], the Gram-Schmidt norm of coefficients of the private key is set as $s \approx \sqrt{\frac{q\mathbf{e}}{2}}$, where $\mathbf{e}$ is the base of natural logarithm. As illustrated in Section 3.1, the error terms $\mathbf{r}, \mathbf{e_1}, \mathbf{e_2}$ are N-dimension vectors with coefficients in $\{-1, 0, 1\}^N$. For LWE-PEKS, with 80-bit security, we set $n = 256$, $q = 4093$ and $\sigma = 8.35$. For 192-bit

security, we set $n = 320$, $q = 4093$ and $\sigma = 8$. Error terms are sampled from the distribution as defined in Definition 12.

**Computation:** The main performance advantage of NTRU-PEKS stems from the fact that the `Test` algorithm is executed once on every keyword-file pair for each search query. Therefore, since the `Test` algorithm of our NTRU-PEKS scheme only requires one convolution product, which is much more efficient than the bilinear pairing operation required in almost all of the existing pairing-based PEKS schemes, we achieve noticeable performance gains in terms of the end-to-end delay. 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., $r \xleftarrow{\$} \{-1, 0, 1\}^N$), 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 BCOP scheme requires one bilinear pairing operation that is more costly than the convolution products. The `Trapdoor` algorithm in NTRU-PEKS requires a Gaussian Sampling, similar as in [13], [25]. This is the most costly operation in our scheme. This algorithm in BCOP only requires one scalar multiplication and consequently, it is the fastest. For LWE-PEKS, it requires the same type of operations as in NTRU-PEKS, however, mostly due to the very large dimension of the parameters, it suffers from significant performance loss as compared to its pairing-based counterpart.

**Storage and Communication:** In terms of storage and communication, our schemes are more storage intensive than their pairing-based counterparts. For instance, in our most efficient scheme (i.e., NTRU-PEKS), the sender needs to store the receiver's public key of size $N \cdot \log_2 q$. Referring to Table 1, for 192-bit security, it can be up to 27.2 Kb. The searchable ciphertext in NTRU-PEKS scheme $3N \log_2(q)$ bits, which corresponds to 52 Kb. Trapdoor in NTRU-PEKS scheme $2N \log_2(2s\pi)$ bits, where $s$ defines the norm of the Gram-Schmidt coefficient. As depicted in Table 1 the size of the searchable ciphertext in NTRU-PEKS is significantly larger than the one in BCOP [5]. However, in Section 4.3, we show that in the deployment of the schemes, even though our NTRU-PEKS scheme incurs a higher communication overhead, this overhead is insignificant when the end-to-end search delay is considered (as compared to the pairing-based counterparts), even with a moderate-speed network.

For LWE-PEKS, the public key is $n \cdot \log_2 q((l+2) \cdot m + 1)$ bits, where as alluded to in Section 3.2, $m = 6n^{1+\delta}$ and $l$ is the bit length of the keyword. As suggested in [14], $\delta$ should be set such that $n^\delta > \lceil \log_2 q \rceil$. The searchable ciphertext size is $\kappa(2m + 1) \log_2 q$. As depicted, in Table 1 the parameter sizes of LWE-PEKS is significantly larger than those of NTRU-PEKS scheme.

### 4.2 Evaluation Metrics and Experimental Setup

**Evaluation Metrics:** We implemented our NTRU-PEKS scheme based on the preliminary implementations in [54], [55]). As aforementioned, PEKS schemes have potential applications in settings with heterogeneous devices. Since in most of these applications, the smaller devices are conceived to be in the role of a sender, in our experiments, their performance is evaluated in terms of searchable ciphertext generation and sending it to the server. In other applications (e.g., secure e-mail system), commodity hardware may also generate searchable ciphertext 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 [56], 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 [57]. 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 implementation for real cloud setting, therefore, in addition to MIRACL, ZeroMQ library is also used in this implementation. Elliptic curves for BCOP 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]. For the schemes in the standard model, we only simulated the implementation of our LWE-PEKS and its pairing-based counterpart [17]. Therefore, due to their poor computational performance and parameter sizes (for LWE-PEKS), we decided not to go ahead with the full-fledged implementation and merely provide an overview of how costly these schemes could be as compared to the schemes in the random oracle model.

As the commodity hardware, we used an Intel Core i7-6700HQ laptop with a 2.6 GHz CPU and 12GB RAM. As a low-power device, we selected ARM Cortex A53 processor, due to its flexibility and low-power consumption [58]. Although it is a low-power device (can run with a small 2200 mAh battery), ARM Cortex A53 is equipped with a 64-bit 1 GHz processor and 1 GB SDRAM. It is extensively preferred in practice since it combines powerful processing power with low energy consumption [58]. 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 250 GB SSD. *Since our parameter sizes are larger than pairing-based schemes, we preferred to be conservative and selected a moderate-speed network with a 75 Mbps connection for both commodity hardware and ARM processor.* The ping to the server is measured as 25.23 ms and 26.78 ms from commodity hardware and ARM processor,

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

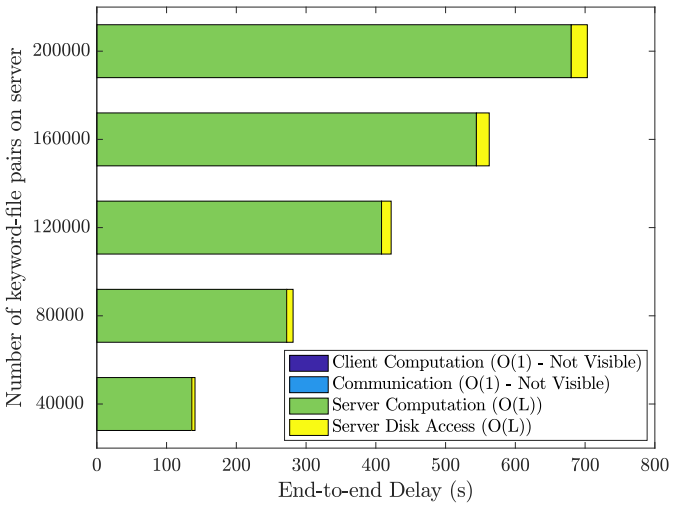


Fig. 2. End-to-end search time of NTRU-PEKS for $\kappa = 192$.

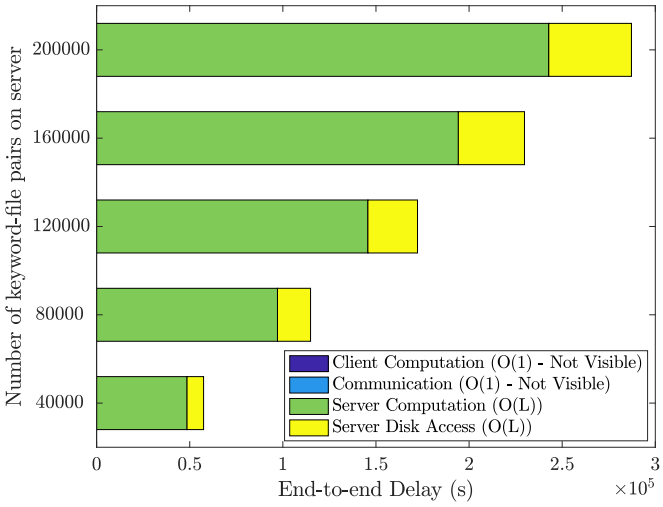


Fig. 3. End-to-end search time of LWE-PEKS for $\kappa = 192$.

respectively. In our experiments, we used subsets of publicly available Enron e-mail dataset.

One can notice that despite the strong security assurances that schemes in the standard model offer, the computation and storage overhead (especially for LWE-PEKS) marks their impracticality, especially with the current state computation capabilities of commodity hardware.

### 4.3 Performance Evaluation

Table 2 shows the experimental results obtained from the full-fledged implementation of our lattice-based PEKS schemes and the one proposed in [5], [17]. As illustrated, our NTRU-PEKS scheme is 15× and 18× faster than BCOP in the `PEKS` and `Test` algorithms, respectively. However, due to costly Gaussian Sampling, the `Trapdoor` algorithm is approximately 8× slower than the one in BCOP scheme. We note that due to the latest advancements in Gaussian sampling techniques, one can instantiate the `Test` algorithm with more recent algorithms, for instance, using the method proposed in [59]. As for communication, sending one searchable ciphertext (for $\kappa = 192$ bits, equivalent to 52 Kb of data), with our moderate-speed network setting takes only 94 ms which is only 12 ms higher than that of BCOP. This communication cost becomes much costlier in LWE-PEKS. Sending a Trapdoor, which needs to be done once per receiver query on the server, takes 92 ms in our scheme which is only 12 ms more than that in the BCOP scheme.

As for our implementation on ARM processor, the searchable ciphertext generation takes 22.58 ms in our scheme which is 40× faster than that of BCOP. As aforementioned, we did not benchmark the `Trapdoor` algorithm since in most applications, these devices operate as senders. The communication of one searchable ciphertext in our scheme takes 100 ms on ARM processor with the aforementioned network setting which is only 14.5 ms higher than that of BCOP scheme.

### 4.4 Detailed Analysis and Cost Dissection

Figure 2 and Figure 3 show the cost dissection of the end-to-end delays for NTRU-PEKS and LWE-PEKS when the receiver searches over the database for $\kappa = 192$ based on our full-fledged implementation and simulation results, respectively. As depicted, server computation which is running the `Test` algorithm, dominates the total cost for both of the schemes since it is executed once per each keyword-file pair (linear) in the database. Disk access time on the server is also linear to the number of keyword-file pairs in the database, however, it is much faster as compared to the `Test` algorithm . Therefore, it contributes to 3% of the total end-to-end delay in NTRU-PEKS. Due to the larger parameter sizes in LWE-PEKS this increases to 15% of the total end-to-end delay.

We conducted experiments with varying sizes of database, up to 200,000 keyword-file pairs. As depicted in Figure 4, for 200,000 keyword-file pairs, BCOP algorithm takes 3.34 hours, whereas NTRU-PEKS takes 11.71 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 each search.

We implemented `PEKS` on both the commodity hard-

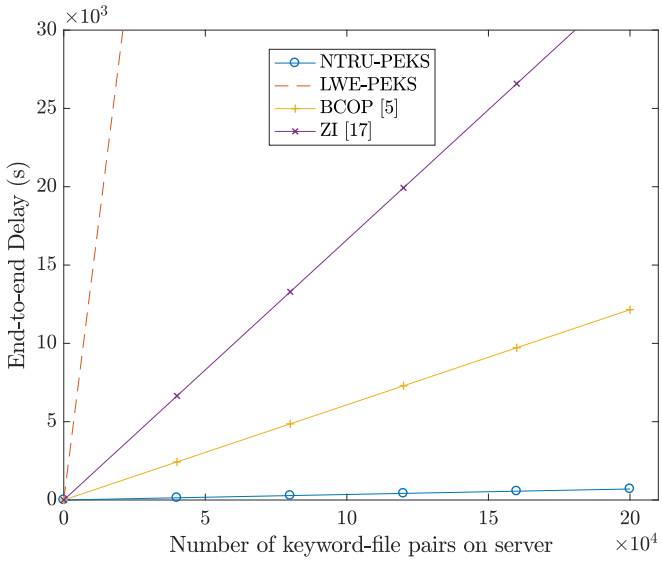


Fig. 4. End-to-end search time comparison for $\kappa = 192$.

ware and ARM processors. We observed significant improvements on the ARM processor, wherein specifically, for $\kappa = 192$, the PEKS algorithm only takes $22.58$ ms which is $40\times$ faster than BCOP 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 these 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 would potentially decrease significantly.

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, as aforementioned communication difference between NTRU-PEKS and BCOP is only around $10 - 15$ ms. 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 ARM processor, respectively. With a three-way handshake in TCP, RTT dominates the total communication cost, resulting in an insignificant difference between our NTRU-PEKS and BCOP. This shows that, although NTRU-based schemes have larger parameters, their computational results in a lower end-to-end delay as compared to their communication efficient counterparts (e.g., pairing-based schemes). When the schemes in the standard model are considered, based on our simulated results, as depicted in Figure 4, the gap between the BCOP scheme and ZI scheme (which is secure in the standard model) is much less than the gap between NTRU-PEKS and the LWE-PEKS which is secure in the standard model.

### 4.5 Discussion

We presented the first full-fledged implementations for PEKS schemes, and make our implementation open-sourced for further adoption and improvements. Our experiments showed that (i) Test algorithm dominates the total search time since it runs $\mathcal{O}(L)$ times (linear with the number of keyword-file pairs, $L$). (ii) The efficiency of PEKS algorithm is also crucial since it is to be run on energy-constrained devices in heterogeneous settings. (iii) 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.

## 5 Related Work

Searchable encryption can be instantiated from both symmetric or asymmetric key settings. Song et al. [2] proposed the first SE scheme that relies on symmetric key cryptography. Kamara et al. [60] proposed the first DSSE scheme to address the limitation of its static ancestors. While being highly efficient, symmetric SE schemes are more suitable for applications that involve a single client who outsources her own data to the cloud relying on her private key.

In this paper, given the target applications that need multiple heterogeneous entities to create searchable encrypted data, our focus is on SE schemes instanced in asymmetric settings. In particular, we concentrate on PEKS, as it requires neither specific probability distributions on keywords nor performance/consistency trade-offs as dictated by some other asymmetric alternatives (e.g., ESE as discussed in Section 3.4). In PEKS schemes, decryption and trapdoor generation take place using the private key of the receiver, while any user can use the corresponding public key to generate searchable ciphertext. With a few exceptions, all of the proposed PEKS schemes are developed using costly bilinear pairing operations. The first instance of pairing-free PEKS schemes are constructed by Crescenzo and Saraswat [21] based on the IBE scheme in [61], which is constructed using quadratic residue for a composite modulus. Khader [62] proposed the first instance of such schemes in the standard model based on a k-resilient IBE, she also put forth a scheme which supports multiple-keyword search.

In the Trapdoor algorithm of NTRU-PEKS and LWE-PEKS, following the works in [13] and [14], the receiver uses its secret, which is a short basis of the lattice, as a trapdoor to sample short vectors. Using the short basis of the lattice as a trapdoor to efficiently sample private keys (trapdoors, in the case of PEKS schemes) has been extensively studied in the literature (e.g., [63], [64], [65], [66], [67]).

## Acknowledgments

The authors would like to thank Leo Ducas and Thomas Prest for their valuable discussions. The authors would also like to thank the anonymous reviewers of IEEE TDSC whose comments significantly helped to improve the quality of this paper. 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, 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] M. O. Ozmen, T. Hoang, and A. A. Yavuz, "Forward-private dynamic searchable symmetric encryption with efficient search," in *2018 IEEE International Conference on Communications (ICC)*, May 2018, pp. 1–6.

[5] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public Key Encryption with Keyword Search," in *Advances in Cryptology - EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques*, C. Cachin and J. L. Camenisch, Eds. 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. 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. 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 S&P*. IEEE Computer Society, 2007, pp. 350–364.

[9] J. Bringer, H. Chabanne, and B. Kindarji, "Error-tolerant searchable encryption," in *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*, V. Shoup, Ed. Springer Berlin Heidelberg, 2005, pp. 205–222.

[13] L. Ducas, V. Lyubashevsky, and T. Prest, "Efficient identity-based encryption over ntru lattices," in *Advances in Cryptology – ASIACRYPT 2014: 20th International Conference on the Theory and Application of Cryptology and Information Security*, P. Sarkar and T. Iwata, Eds. 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*, H. Gilbert, Ed. Springer Berlin Heidelberg, 2010, pp. 553–572.

[15] 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*, P. Q. Nguyen and D. Pointcheval, Eds. Springer Berlin Heidelberg, 2010, pp. 499–517.

[16] 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*. ACM, 2005, pp. 84–93.

[17] 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. Springer Berlin Heidelberg, 2007, pp. 159–174.

[18] 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.

[19] 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*, G. Livraga and S. Zhu, Eds. Cham: Springer International Publishing, 2017, pp. 365–385.

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

[21] G. Di Crescenzo and V. Saraswat, "Public key encryption with searchable keywords based on jacobi symbols," in *Progress in Cryptology – INDOCRYPT 2007: 8th International Conference on Cryptology in India*, K. Srinathan, C. P. Rangan, and M. Yung, Eds. Springer Berlin Heidelberg, 2007, pp. 282–296.

[22] 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.

[23] J. Hoffstein, J. Pipher, and J. H. Silverman, "NTRU: A ring-based public key cryptosystem," pp. 267–288, 1998.

[24] 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. Springer Berlin Heidelberg, 2003, pp. 122–140.

[25] 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*. New York, NY, USA: ACM, 2008, pp. 197–206.

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

[27] 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.

[28] L. Ducas and P. Q. Nguyen, "Faster gaussian lattice sampling using lazy floating-point arithmetic," in *ASIACRYPT 2012: 18th International Conference on the Theory and Application of Cryptology and Information Security*, X. Wang and K. Sako, Eds. Springer Berlin Heidelberg, 2012, pp. 415–432.

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

[30] T. Prest, "Gaussian sampling in lattice-based cryptography," PhD Thesis, École Normale Supérieure, 2015.

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

[32] 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*, N. Smart, Ed. Springer Berlin Heidelberg, 2008, pp. 31–51.

[33] Y. Chen and P. Q. Nguyen, "BKZ 2.0: Better lattice security estimates," in *ASIACRYPT 2011: 17th International Conference on the Theory and Application of Cryptology and Information Security*, D. H. Lee and X. Wang, Eds. Springer Berlin Heidelberg, 2011, pp. 1–20.

[34] V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," in *Advances in Cryptology – EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, H. Gilbert, Ed. Springer Berlin Heidelberg, 2010, pp. 1–23.

[35] ——, "A toolkit for ring-lwe cryptography," in *Advances in Cryptology – EUROCRYPT 2013: 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques*, T. Johansson and P. Q. Nguyen, Eds. Springer Berlin Heidelberg, 2013, pp. 35–54.

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

[37] D. J. Bernstein, "A subfield-logarithm attack against ideal lattices," http://blog.cr.yp.to/20140213-ideal.html, 2014.

[38] M. Albrecht, S. Bai, and L. Ducas, "A subfield lattice attack on overstretched NTRU assumptions," in *Advances in Cryptology – CRYPTO 2016*, M. Robshaw and J. Katz, Eds. Springer Berlin Heidelberg, 2016, pp. 153–178.

[39] P. Kirchner and P.-A. Fouque, "Revisiting lattice attacks on overstretched ntru parameters," in *Advances in Cryptology – EUROCRYPT 2017: 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, J.-S. Coron and J. B. Nielsen, Eds. Cham: Springer International Publishing, 2017, pp. 3–26.

[40] M. Ajtai, "Generating hard instances of the short basis problem," in *Automata, Languages and Programming: 26th International Colloquium*, J. Wiedermann, P. van Emde Boas, and M. Nielsen, Eds. Springer Berlin Heidelberg, 1999, pp. 1–9.

[41] 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.

[42] 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.

[43] 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. Springer Berlin Heidelberg, 2001, pp. 566–582.

[44] 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*, A. Kiayias, Ed. Springer Berlin Heidelberg, 2011, pp. 319–339.

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

[46] 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. Springer Berlin Heidelberg, 2006, pp. 75–83.

[47] 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*, S. Lin and X. Huang, Eds. Springer Berlin Heidelberg, 2011, pp. 131–136.

[48] 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.

[49] 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.

[50] 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.

[51] 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.

[52] M. Bellare, A. Boldyreva, and A. O'Neill, "Deterministic and efficiently searchable encryption," in *Advances in Cryptology - CRYPTO 2007: 27th Annual International Cryptology Conference*, A. Menezes, Ed. Springer Berlin Heidelberg, 2007, pp. 535–552.

[53] 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.

[54] T. Prest, "Lattice-IBE," https://github.com/tprest/Lattice-IBE, 2013.

[55] R. Behnia, "NTRUPEKS," https://github.com/Rbehnia/NTRUPEKS, 2016.

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

[57] 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

[58] 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.

[59] L. Ducas and T. Prest, "Fast fourier orthogonalization," in *Proceedings of the ACM on International Symposium on Symbolic and Algebraic Computation*, ser. ISSAC '16. New York, NY, USA: ACM, 2016, pp. 191–198.

[60] S. Kamara and K. Lauter, "Cryptographic cloud storage," in *Financial Cryptography and Data Security*, R. Sion, R. Curtmola, S. Dietrich, A. Kiayias, J. M. Miret, K. Sako, and F. Sebé, Eds. Springer Berlin Heidelberg, 2010, pp. 136–149.

[61] C. Cocks, "An identity based encryption scheme based on quadratic residues," in *Cryptography and Coding: 8th IMA International Conference Cirencester*, B. Honary, Ed. Springer Berlin Heidelberg, 2001, pp. 360–363.

[62] D. Khader, "Public key encryption with keyword search based on K-resilient IBE," in *Proceedings of the 2007 International Conference on Computational Science and Its Applications*. Springer-Verlag, 2007, pp. 1086–1095.

[63] D. Micciancio and C. Peikert, "Trapdoors for lattices: Simpler, tighter, faster, smaller," in *Proceedings of the 31st Annual International Conference on Theory and Applications of Cryptographic Techniques*, ser. EUROCRYPT'12. Springer-Verlag, 2012, pp. 700–718.

[64] 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.

[65] R. W. F. Lai, H. K. F. Cheung, and S. S. M. Chow, "Trapdoors for ideal lattices with applications," in *Information Security and Cryptology*, D. Lin, M. Yung, and J. Zhou, Eds. Cham: Springer International Publishing, 2015, pp. 239–256.

[66] N. Genise and D. Micciancio, "Faster gaussian sampling for trapdoor lattices with arbitrary modulus," Cryptology ePrint Archive, Report 2017/308, 2017, https://eprint.iacr.org/2017/308.

[67] R. Bendlin, S. Krehbiel, and C. Peikert, "How to share a lattice trapdoor: Threshold protocols for signatures and (h)ibe," in *Applied Cryptography and Network Security*, M. Jacobson, M. Locasto, P. Mohassel, and R. Safavi-Naini, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 218–236.