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 encryption

1 INTRODUCTION

Cloud computing has significantly impacted the computing infrastructure and enabled a large pool of applications. For example, data outsourcing 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,
1.2 Our Contribution

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”.

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 \( O(L) \) 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 [4]. 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 3). 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 3). 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. $a \leftarrow X$ denotes that $a$ is randomly selected from distribution $X$. $H_i$ 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, while scalars are represented as non-bold letters i.e., $v$. $A^{O_1, \ldots, O_n}()$ denotes that algorithm $A$ is provided with access to oracles $O_1, \ldots, O_n$. The norm of a vector $v$ is denoted by $\|v\|$. $[x]$ rounds $x$ to the closest integer. $x \equiv y$ means $x$ is defined as $y$. The function $\text{gcd}(x,y)$ 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 $\text{IBE} = (\text{Setup}, \text{Extract}, \text{Enc}, \text{Dec})$ defined as follows.

- $(mpk,msk) \leftarrow \text{Setup}(^1\text{K})$: On the input of the security parameter(s), this algorithm publishes system-wide public parameters $\text{params}$, outputs the master public key $mpk$ and the master secret key $msk$.
- $sk \leftarrow \text{Extract}(id,msk,mpk)$: On the input of a user’s identity $id \in \{0,1\}^*$, $mpk$, and $msk$, this algorithm outputs the user’s private key $sk$.
- $c \leftarrow \text{Enc}(m,id,mpk)$: On the input of a message $m \in \{0,1\}^*$, identity $id$, and $mpk$, this algorithm outputs a ciphertext $c$.
- $m \leftarrow \text{Dec}(c,sk)$: 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 $\text{KeyQuery}($) oracle, as defined below, we associate a bit $b \in \{0,1\}$ to the adversary $A$ in the following experiment $\text{Exp}_{\text{IBE-ANO-RE-CPA}}(1^k)$:
Given a PEKS scheme, and the CKA) is defined in the following experiment. A PEKS scheme consists of the following algorithms.

2.2 Public Key Encryption with Keyword Search

A PEKS scheme is a tuple of four algorithms

\[ (pk, sk) \leftarrow \text{KeyGen}(1^k) \]

\[ s_w \leftarrow \text{PEKS}(pk, w) \]

\[ t_w \leftarrow \text{Trapdoor}(sk, w) \]

\[ d \leftarrow \text{Test}(t_w, s_w) \]

Key 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 \( \text{TdQuery}(\cdot) \) oracle as defined below, we associate a bit \( b \in \{0, 1\} \) to the adversary \( A \) in the following experiment \( \text{Exp}^{\text{PEKS-IND-CKA}}(1^k) \).

\[ wSet \leftarrow wSet \cup w \]

\[ t_w \leftarrow \text{Trapdoor}(w, sk, pk) \]

\[ \text{return } t_w \]

Experiment \( \text{Exp}^{\text{PEKS-IND-CKA}}(1^k) \)

After queries to \( \text{KeyQuery}(\cdot) \) and random oracle \( H \) in the \( \text{find} \) stage, \( A \) outputs a challenge \( (id_0, id_1, m) \leftarrow A^{\text{KeyQuery}(\cdot), H}(mpk) \): find stage

\[ c \leftarrow \text{Enc}_H(m, id_0, mpk) \]

\[ b' \leftarrow A^{\text{KeyQuery}(\cdot), H}(c) \]: guess stage

if \( \{id_0, id_1\} \cap wSet = \emptyset \) return \( b' \) else, return 0

\( A \)'s advantage in the above experiment is defined as:

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

2.2.1 Consistency of PEKS

Right-Keyward Consistency: 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. More specifically, right-keyword consistency refers to the decryption error in the underlying IBE scheme which leads to the inconsistency of the Test algorithm in the resulting PEKS scheme.

Adversary-Based Consistency: We define the adversary-based consistency \cite{12} in the following definition.

Definition 5. Adversary-based consistency of a PEKS scheme is defined in the following experiment.

\[ (pk, sk) \leftarrow \text{KeyGen}(1^k) \]

\[ (w_0, w_1) \leftarrow A^{\text{KeyQuery}(\cdot)}(pk) \]: find stage

\[ s_{w_0} \leftarrow \text{PEKS}(pk, w_0) \]

\[ b' \leftarrow A^{\text{KeyQuery}(\cdot)}(s_{w_0}) \]: guess stage

if \( \{w_0, w_1\} \cap wSet = \emptyset \) return \( b' \) else, return 0

\( A \)'s advantage in the above experiment is defined as:

\[ \text{Adv}_{\text{PEKS-Consist}}^A(1^k) = \text{Pr}[\text{Exp}^{\text{PEKS-Consist}}(1^k) = 1] - \text{Pr}[\text{Exp}^{\text{PEKS-Consist}}(1^k) = 0] \]

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

2.3 Tools and Definitions

Integer Lattices: Let \( B = [b_1 \ldots b_m] \in \mathbb{R}^{m \times m} \) be an \( m \times m \) matrix whose columns are linearly independent vectors.
b_1, \ldots, b_m \in \mathbb{R}^m$. The m-dimensional full-rank lattice \( \Lambda \) generated by \( B \) is the set,
\[
\Lambda = \mathcal{L}(B) = \left\{ y \in \mathbb{R}^m : \exists s \in \mathbb{Z}^m, y = Bs = \sum_{i=1}^{m} s_i b_i \right\}
\]

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

**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. Hofheinz et al. [23] proposed a very efficient public key encryption scheme based on NTRU lattices. Reggey [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 \( f = \sum_{i=0}^{N-1} x^i f_i \) and \( g = \sum_{i=0}^{N-1} g_i x^i \) as polynomials in \( \mathcal{R}[x] \), \( f g \) denotes polynomial multiplication in \( \mathcal{R}[x] \) while \( f \ast g \triangleq f g \mod (x^N + 1) \) is referred to as convolution product.

For an \( N \)-dimensional anti-circular matrix \( \mathcal{A}_N \) we have \( \mathcal{A}_N(f) + \mathcal{A}_N(g) = \mathcal{A}_N(f + g) \), and \( \mathcal{A}_N(f) \times \mathcal{A}_N(g) = (f \ast g) \).

Definition 7. For prime integer \( q \) and \( f, g \in \mathcal{R} \), \( h = g \ast f^{-1} \mod q \), the NTRU lattice with \( h \) and \( q \) is \( \Lambda_{h,q} = \{ (u, v) \in \mathbb{R}^2 : u + v \ast h = 0 \text{ mod } q \} \). \( \Lambda_{h,q} \) is a full-rank lattice generated by \( \mathcal{A}_{h,q} = \left( \begin{array}{cc} \mathcal{A}_h(h) & I_N \\ qI_N & 0_N \end{array} \right) \), where \( I \) 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 generated by selecting \( F, G \in \mathcal{R} \) and computing \( f \ast G - g \ast F = q \). The new base \( B_{f,g} = \left( \begin{array}{cc} \mathcal{A}(g) & -\mathcal{A}(f) \\ \mathcal{A}(G) & -\mathcal{A}(F) \end{array} \right) \)

generates the same lattice \( \Lambda_{h,q} \).

Definition 8. (Gram-Schmidt norm [25]) Given \( B = (b_i)_{i \in I} \)

as a finite basis and \( B = (b_i)_{i \in I} \) as its Gram-Schmidt orthogonalization, the Gram-Schmidt norm of \( B \) is
\[
\|B\| = \max_{i \in I} \|b_i\|.
\]

Definition 9. (Statistical Distance [14]) Given two random variables \( X \) and \( Y \) taking values in a finite set \( S \), the statistical distance is defined as:
\[
\Delta(X, Y) = \frac{1}{2} \sum_{s \in S} \Pr[X = s] - \Pr[Y = s]
\]

\( X \) is said to be \( \delta \)-uniform over \( S \) 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 \( \rho_{\sigma,c}(x) := \exp(-\frac{|x-c|^2}{2\sigma^2}) \). Given a lattice \( \Lambda \subset \mathbb{R}^n \), the discrete Gaussian distribution over \( \Lambda \) is
\[
D_{\Lambda, \rho_{\sigma,c}}(x) = \frac{\rho_{\sigma,c}(x)}{\rho_{\sigma,c}(\Lambda)} \text{ for all } x \in \Lambda.
\]

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 \( \epsilon \geq 0 \), the smoothing parameter \( \eta_{\epsilon}(\Lambda) \) is the smallest \( s \geq 0 \) such that \( \lambda_{1/\sqrt{s\pi \eta_{\epsilon}}}(\Lambda^*) \leq \epsilon \). 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 \( n, \lambda \), \( \epsilon \leq 2^{-\lambda/2}/(4\sqrt{2N}) \), any basis \( B \in \mathbb{Z}^{n \times N} \) and any target vector \( c \in \mathbb{Z}^{1 \times n} \), the algorithm \( (v_0 \leftarrow \text{Gaussian-Sampler}(B, \sigma, c)) \) as defined in [13], [25] is such that \( \Delta(D_{\Lambda}(B), \rho_{\sigma,c}, v_0) \leq 2^{-\lambda} \).

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

In [16], Reggey has shown that for a certain error distribution \( \chi \), denoted \( \Psi_{\alpha} \), the LWE problem is as hard as the worst-case SIVP and GapSVP under quantum reduction.

Definition 12. (Distribution of \( \Psi_{\alpha} [16], [29] \)) For an \( \alpha \in (0, 1) \) and a prime \( q \), \( \Psi_{\alpha} \) denotes the distribution over \( \mathcal{Z}_q \) of the random variable \( [q x] \mod q \) is a normal random variable with mean 0 and the standard deviation \( \sqrt{\frac{q}{2\pi}} \).

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

### 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.

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

1) Compute \( \sigma_f \leftarrow 1.17 \sqrt{(\frac{n}{2})} \) and select \( f, g \leftarrow D_{N, \sigma_f} \) to compute \( \|B_{f,g}\| \) and \( \text{Norm} \leftarrow \max(\|(g, f)\| \),
Given cryptographic hash functions

2) Given the receiver's private key and a keyword w ∈ {0, 1}∗, the receiver computes t ← H1(w) and returns the ciphertext t = H1(w) to the sender. Let E1 be the event that there exist two pairs (z1, z′ 1) and (z2, z′ 2) such that H2(z1, z′ 1) = H2(z2, z′ 2) for z1 ≠ z2 and z′ 1 ≠ z′ 2. Then Pr[ 1 ] represents the probability of this event. Therefore, the probability that B1 = B2 is negligible due to the decryption error. Hence, the probability of the last term is also negligible.

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 (h, B) ← KeyGen(q, N), a searchable ciphertext sw ← PEKS(pk, w), and a trapdoor generated by the receiver tsw ← Trapdoor(sk, w), our proposed scheme is complete.

Proof. To show the completeness of our scheme for sw = (c0, c1, H2(k, c1)), the Test algorithm should return 1 when 1/qf.c0+c1. Thus, we work as follows.

c1 − c0 * s = r * s + e2 − (1/qf) * k − (1/qf) * (c0 + c1) * s = r * s + e2 − (1/qf) * k − (1/qf) * 0 = r * s + e2 − (1/qf) * 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 z = r * s + e2 − e1 * tw are in the range (−1/qf, 1/qf). If Norm ≥ 1.17√q, this process is repeated by sampling new f and q. Finally, compute

h = g * f mod q and B = \( A(g) - A(f) \) with outputs (pk ← h, sk ← B).

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 v in an NTRU lattice, the associated root Hermite factor is computed as γn = \( \sqrt{N/(2\pi e)} \cdot \text{det}(\Lambda)^{1/2} ||v|| \). Based on [32], [33], γ ≈ 1.004 guarantees intractability and provides approximately 192-bit security.

Following Lemma 2 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 ring-LWE problem.

**Proof.** Since the output of the PEKS algorithm of our scheme corresponds to the encryption algorithm of [34, 35], for \( A \) 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 \( c_0 = r \ast h + e_1 \) (for a uniform \( r, e_1 \in \{-1, 0, 1\}^N \)) and uses \( H(w) \) to compute \( c_1 = r \ast H(w) + e_2 + \left\lfloor \frac{q}{2} \right\rfloor k \) that is also statistically close to uniform. Therefore, the adversary’s view of \( \langle c_0, c_1, H_2(k, c_1) \rangle \) 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 \( A \) that can break PEKS-IND-CKA of NTRU-PEKS scheme as in Definition 2, one can build an adversary \( F \) that uses \( A \) as subroutine and breaks the security of the IBE scheme in Definition 2.

**Proof.** The proof works by having adversaries \( F \) and \( A \) initiating the find phase as in Definition 8 and Definition 10 respectively.

Algorithm \( F_{\text{keyQuery}}(\cdot) : H(\text{find}, mpk) \)

- \((mpk, msk) \leftarrow \text{Setup}(q, N)\): \( F \) receives \( mpk \) and passes it to \( A \).

Algorithm \( A^{T_{\text{query}}(\cdot) : H(\text{find}, pk)} \)

- Queries on \( T_{\text{query}}(\cdot) \): Upon such queries, \( F \) queries \( \text{KeyQuery}(\cdot) \) which keeps a list \( id\text{Set} \) 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 \((s, t_w) \leftarrow \text{Gaussian-Sampler}(msk, \sigma, (H(w), 0)) \) and passes \( t_w \) to \( F \). \( F \) sends \( t_w \) to \( A \).

After the find phase, a hidden fair coin \( b \in \{0, 1\} \) is flipped. Execute \((w_0, w_1) \leftarrow A^{T_{\text{query}}(\cdot) : H(\text{guess}, pk)} \)

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

Finally, \( A \) outputs its decision bit \( b' \in \{0, 1\} \). \( F \) 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.

\[ \text{Adv}^{\text{PEKS-IND-CKA}}_A(q, N) \leq \text{Adv}^{\text{NTRU-IBE-IND-CPA}}_F(q, N) \]

**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 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 \( \Lambda_q^+(A) \). The following theorem, defines the properties of \( \text{TrapGen} \) algorithm [14, 41], which is used in the \( \text{KeyGen} \) algorithm of the LWE-PEKS scheme.

\((A, S) \leftarrow \text{TrapGen}(q, n): \) Given a prime \( q \), a positive \( n \) and \( \delta = \frac{1}{2} \), there is a polynomial time algorithm \( \text{TrapGen}(q, n) \) that outputs a pair \((A \in \mathbb{Z}_q^{n \times m}, S \in \mathbb{Z}_q^{m \times m})\) s.t., \( A \) is statistically close to uniform and \( S \) is a basis for \( \Lambda_q^+(A) \), where \( m > 6n \log q \) and \( ||S|| \leq O(\sqrt{n \log q}) \) hold with a high probability.

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

In the following we define the sampling algorithm which is used to generate trapdoors in our scheme (i.e., \( \text{SampleLeft} \)), the same algorithm, with identical properties has also been used in [29, 42].

\( e \leftarrow \text{SampleLeft}(A, M_1, T_A, u, \sigma) : \) Given an \( n \)-rank matrix \( A \in \mathbb{Z}_q^{n \times m} \), a matrix \( M_1 \in \mathbb{Z}_q^{n \times m_1} \), a short basis of \( \Lambda_q^+(A) \), a vector \( u \in \mathbb{Z}_q^n \), and a Gaussian parameter \( \sigma > \sqrt{m} \cdot w(\sqrt{\log (m + m_1)}) \), this algorithm outputs a vector \( e \in \mathbb{Z}_q^{m + m_1} \) sampled from the distribution statistically close to \( D_{\Lambda_q(A)}(F_1, \sigma) \) where \( F_1 := (A | M_1) \).

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

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

1. Use \( \text{TrapGen}(q, n) \) to pick a random matrix \( A_0 \in \mathbb{Z}_q^{n \times m_1} \) with basis \( T_{A_0} \) for \( \Lambda_q^+(A_0) \) s.t. \( ||T_{A_0}|| \leq O(\sqrt{n \log q}) \).
2) Select $l+1$ random matrices $A_1, \ldots, A_l, B \leftarrow \mathbb{Z}_q^{m \times m}$ and a random vector $u \leftarrow \mathbb{Z}_q^n$.
3) Output the public key $pk \leftarrow (A_0, A_1, \ldots, A_l, B, u)$ and secret key $sk \leftarrow T_{A_0}$.

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

1) Choose a uniformly random $s_j \leftarrow \mathbb{Z}_q^n$ and matrices $R_{ij} \leftarrow \{−1, 1\}^{m \times m}$ for $i = 1, \ldots, l$ and set $R_{bj} \leftarrow \sum_{i=1}^{l} b_i R_{ij} \in \{−1, 1\}^{m \times m}$.
2) Choose noise vectors $x_j \leftarrow \mathbb{Z}_q$ and $y_j \leftarrow \mathbb{Z}_q$, and set $z_j = R_{bj} y_j \in \mathbb{Z}_q^n$.
3) Set $c_{0j} \leftarrow u^T s_j + x_j + b'_j \frac{q}{2} \in \mathbb{Z}_q$ and $c_{1j} \leftarrow F_w^T s_j + y_j \in \mathbb{Z}_q^m$.
4) Output searchable ciphertext $s_{wj} = (c_{0j}, c_{1j}, b'_j)$ for $j = 1, \ldots, \kappa$.

t_w \leftarrow \text{Trapdoor}(pk, sk, w)$: On the input of the keys and a keyword $w = (b_1, \ldots, b_l) \in \{0, 1\}^l$, the receiver computes as follows.

1) Let $A_w \leftarrow B + \sum_{j=1}^{l} b_j A_j \in \mathbb{Z}_q^{m \times m}$ and sample $t_w \in \mathbb{Z}_q^{2m}$ as $t_w \leftarrow \text{SampleLeft}(A_0, A_w, T_{A_0}, u, \sigma \llbracket)$.
2) Output the trapdoors as $t_w$.

Given $F_w := (A_0 A_w)$, then $F_w \cdot t_w = u \in \mathbb{Z}_q$ and $t_w$ is distributed as $D_{A_w} F_w \cdot u \sigma$.

d \leftarrow \text{Test}(t_w, s_w)$: Given a trapdoor $td$ for a keyword $w = (b_1, \ldots, b_l) \in \{0, 1\}^l$, and $\kappa$ searchable ciphertexts $s_{wj} = (c_{0j}, c_{1j}, b'_j)$ for $j = 1, \ldots, \kappa$ on keyword $w$, it computes as follows.

i. Set $\nu_j \leftarrow c_{0j} - t_w \cdot c_{1j}$ and check if $|\nu_j - \frac{q}{2}| < \frac{q}{16}$, set $\nu_j \leftarrow 1$ and otherwise, else $\nu_j \leftarrow 0$.
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 $(pk, sk) \leftarrow \text{KeyGen}(\lambda)$, a searchable ciphertext $s_w \leftarrow \text{PEKS}(pk, w)$ and a trapdoor generated by the receiver $td \leftarrow \text{Trapdoor}(pk, sk, w)$, the proposed scheme is complete.

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

\[ \nu_j = c_{0j} - t_w \cdot c_{1j} \]
\[ = u^T s_j + x_j + b'_j \frac{q}{2} - t_w (F_{w}^T s_j + y_j) \]
\[ = u^T s_j + x_j + b'_j \frac{q}{2} - u^T s_j - t_w \cdot y_j \]
\[ = b'_j \frac{q}{2} + x - t_w \cdot y_j \]

\[ \nu_j = \frac{q}{2} + x - t_w \cdot y_j \]

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

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 $F_w \leftarrow (A_0 A_w)$. 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 $t_w$ that is generated by the SampleLeft on the input of $F_w$. Therefore, for the suggested parameters and based on [29], 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 \( \gamma = \frac{\text{det}(\Lambda_{\mathbf{v}})}{\parallel \mathbf{v} \parallel} \). 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.

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 [53]. 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 whereas 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 \( s \approx \sqrt{\frac{np}{2}} \) where \( n \) is the base of natural logarithm. As illustrated in Section [53], the error terms \( r, e_1, e_2 \) 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 \( n = 512 \), \( q = 8191 \) and \( \sigma = 17.41 \).
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 \( x_1 \times x_2 \). However, since one of the operands has very small coefficients (i.e., \( r \in \{-1, 0, 1\}^N \)), 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 \( N \cdot \log N \) 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 = \( 2N \log_2 (2\pi s) \) 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 \([3]\). 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 \( n \cdot \log_2 q((l+2) \cdot m + l) \) bits, where \( m \) is the bit length of the keyword. As mentioned in \([14]\), \( \delta \) should be set such that \( n^\delta > \log_2 q \). 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 \( \text{vec} \cdot ZZ \) 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 \( \kappa = 80\text{-bit security} \) (with embedding degree \( k = 6 \)) and KSS curve with \( \kappa = 192\text{-bit security} \) (with embedding degree \( k = 18 \)). We made both of the implementations open-sourced for further improvements and adoption \([14]\). 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 2.4GHz, 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.

https://github.com/Rbehnia/Full_PEKS
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 $15 \times$ and $18 \times$ faster than BCOP in the PEKS and Test algorithms, respectively. However, due to costly Gaussian Sampling, the Trapdoor algorithm is approximately $8 \times$ 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 $40 \times$ 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-
ware and ARM processors. We observed significant improvements on the ARM processor, wherein specifically, for \( k = 192 \), 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 \( O(L) \) 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 instance 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., [53], [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
