Quantum-resistant Designated-ciphertext Searchable Encryption

Zi-Yuan Liu, Yi-Fan Tseng, and Raylin Tso

Department of Computer Science, National Chengchi University, Taipei, Taiwan
{yad50968, popcornking54387, tsoraylin}@gmail.com

Abstract. Public key encryption with keyword search (PEKS) was proposed by Boneh *et al.* in 2004; it allows users to search encrypted keywords without losing data privacy. Although extensive studies have been conducted on this topic, only a few focus on the insider keyword guessing attack that will cause users to leak sensitive information. More specifically, after receiving the trapdoor from the user, the malicious insider (e.g. server) can randomly encrypt possible keywords using the user's public key. Then, the insider can test whether the trapdoor corresponds to the selected keyword. To solve the above issue, we introduce the notion of *designated-ciphertext searchable encryption* (DCSE) in this work. Then, we propose a generic construction that employs an anonymous identity-based encryption and key encapsulation mechanism. Additionally, we demonstrated that our work satisfies the indistinguishability under chosen-keyword attack (IND-CKA) and indistinguishability under insider keyword guessing attack (IND-IKGA) in the standard model. Moreover, we provide an instantiation from the NTRU lattices. Compared with other state-of-the-art schemes, our scheme is not only more efficient and practical, it also provides more robust security.

Keywords: quantum-resistant · searchable encryption · insider keyword guessing attack

1 Introduction

With the development of the 5G and Internet of Things (IoT), the importance of cloud storage is increasing. However, because the cloud providers cannot be easily trusted, to avoid data leakage or abuse, data owners need to ensure the privacy of sensitive data. One straightforward method is encrypting data before uploading to cloud servers. Unfortunately, encrypted data cannot be used for some useful operations, such as sorting or searching. Specifically, searching functionality is important for cloud storage. If a data owner wants to search for some specific files among the encrypted data, it becomes necessary to download and decrypt all the data to search. To solve this issue, in 2000, Song *et al.* proposed the first searchable encryption (SE) that allows the ciphertext to be searched using the corresponding trapdoor [27]. Because their construction used symmetric key primitive, only the owner of a secret key can generate the

ciphertext and trapdoor. Therefore, similar to the symmetric encryption, their work suffers from the key distribution problem when being deployed in public cloud environments. To circumvent this issue and allow multiple data owners to easily generate different ciphertexts for a single data receiver, Boneh *et al.* proposed the first public key encryption with keyword search (PEKS) in 2004 [4]. The scheme, unlike [27], is built on a public key cryptosystem. Consider the following scenario. A data owner, Alice, wants to store files that can be accessed and searched for by a data receiver, Bob, to a cloud server without leaking data information. Therefore, before uploading these files, she encrypted them with the keywords, using Bob's public keys. If Bob would like to request the cloud server to search for the encrypted files that contain keyword "important", he can generate the trapdoor for "important" using his private key. Then, the cloud server could use the trapdoor to search for encrypted files tagged with the keyword "important".

Because PEKS is more suitable for purposes such as cloud service, IoT, and email service, many schemes have been proposed over these two decades. However, as most of the schemes assume the insider (e.g. cloud server, mail server, and IoT gateway) is trustworthy, they do not take into account the attack from insiders. In actual fact, due to the small number of commonly used keywords, the insider can guess the keyword from a trapdoor to obtain some useful information, called *insider keyword guessing attack* (IKGA). More concretely, after receiving a trapdoor from a data receiver, the malicious insider can randomly encrypt possible keywords using the receiver's public key. Then, the insider can test whether the trapdoor corresponds to the selected keywords. Unfortunately, scant research on how to construct a PEKS that can resist IKGA [20, 16, 12].

On the other hand, because Shor has demonstrated the existence of quantum algorithms that can solve the discrete logarithm assumption and integer factor assumption [26, 25], the potential threat of quantum computers to classical cryptography is predictable. Specifically, Google recently proposed a 53-qubit quantum computer [2]. There is no doubt that quantum computer will mature over the next few decades. To resist quantum computer attacks, many studies on quantum-resistant PEKS have been proposed [3, 29, 21, 30]. However, only two works are IKGA secure [21, 30]. However, these schemes are not practical enough, because of size constraints; public keys and private keys can be as large as hundreds of megabytes (MB). Therefore, how to construct an efficient, practical, and IKGA secure quantum-resistant scheme is a significant and emerging issue.

1.1 Contributions

In this paper, we introduce a new cryptographic primitive called "designated-ciphertext searchable encryption" (DCSE). This notion aims to provide a searchable encryption scheme that is secure against IKGA. Our strategy is to prevent an insider from producing useful ciphertexts that may be tested successfully on the corresponding trapdoor. In a DCSE scheme, a data sender can generate a searchable ciphertext and its corresponding tag by encrypting a keyword using

a data receiver's public key. While only the data receiver can generate a useful trapdoor using a (keyword, tag) pair. Using the valid trapdoor, the insider (cloud server) can search the corresponding ciphertext. Because the trapdoor is only useful for the corresponding tag that is assigned to a ciphertext. Therefore, it is difficult for the insider to generate useful ciphertext that can be tested with the trapdoor.

The generic construction for DCSE is presented in our work by employing an anonymous identity-based encryption $\mathcal{IBE} = \{Setup, Extract, Enc, Dec\}$ and an IND-CCA2 secure key encapsulation mechanism $\mathcal{KEM} = \{KeyGen, Encaps, Decaps\}$. A high-level overview of our generic construction is provided below. The data receiver's public and private keys are generated using $\mathcal{IBE}.Setup$ and $\mathcal{KEM}.KeyGen$. To generate a searchable ciphertext for a keyword w and its corresponding tag, the data sender first generates an encapsulation e and a key k using $\mathcal{KEM}.Encaps$ with the data receiver's public key, and sets e as the tag. Then, he/she encrypts the concatenation of the keyword w and the key k to a ciphertext c using $\mathcal{IBE}.Enc$. To search for the ciphertext encrypted the keyword w, the data receiver needs to obtain a key k' from the encapsulation e using $\mathcal{KEM}.Decaps$. Then, he/she extracts a trapdoor t from the concatenation of the keyword w and the key k', and sends t to the insider. Finally, using the trapdoor t, the insider can test the ciphertexts, and find the one that matches the keyword.

Additionally, we provide rigorous proofs to demonstrate that this generic construction satisfies the criteria of indistinguishability under chosen-keyword attack (IND-CKA) and indistinguishability under insider keyword guessing attack (IND-IKGA).

Furthermore, we provide an instantiation utilizing two efficient and secure lattice-based constructions: the identity-based encryption from NTRU proposed by Ducas *et al.* [11] and the NTRU-based key encapsulation mechanism proposed by Húlsing *et al.* [17]. The security of these constructions is based on the ring-LWE and NTRU assumption that, in turn, makes our instantiation quantum-resistant. We experimentally evaluate the performance of the instantiation. Each encryption, trapdoor, test algorithm only takes approximately 1, 0.3, 0.01 (ms) respectively on a modern laptop. In comparison with other state-of-the-art schemes, our scheme is not only more efficient and practical, it also provides more robust security.

1.2 Manuscript Organization

The remainder of this manuscript is organized as follows. In Section 2, we introduce some notations and preliminaries used in the work. In Section 3, we introduce two cryptographic building blocks: identity-based encryption (IBE) and key encapsulation mechanism (KEM). In Section 4, we introduce a new notion, "designated-ciphertext searchable encryption," and define its system model and security requirements. In Section 5, we construct the DCSE from the IBE and KEM, and provide its security proofs. In Section 6, an efficient NTRU-based DCSE is proposed. Finally, we conclude this work in Section 7.

2 Preliminary

2.1 Notations

For simplicity and readability, we use the following notations throughout the paper. Let λ be the natural security parameter. We use standard notations, O and o, to classify the growth of functions. The notation $\mathsf{negl}(n)$ is denoted as an arbitrary function f is *negligible* in n, where $f(n) = o(n^{-c})$ for every fixed constant c. The notation $\mathsf{poly}(n)$ is denoted as an arbitrary function $f(n) = O(n^c)$ for some constant c. By $\mathbb{N}$ (resp. $\mathbb{Z}$ and $\mathbb{R}$) we denote the set of positive integers (resp. integers and reals). In addition, for a prime q, $\mathbb{Z}_q$ denotes a finite field (or Galois field) with order q. For a power-of-two n, $\mathcal{R} = \mathbb{Z}[X]/(X^n + 1)$ and $\mathcal{R}_q = \mathbb{Z}[x]/(x^n + 1)$. The PPT is short for probabilistic polynomial-time. For two string a, b, the concatenation of a and b is denoted as $a\|b$. Matrices are denoted by bold capital letters (e.g., $\mathbf{X}$). For a vector x and a matrix $\mathbf{X}$, the Euclidean norm of x and $\mathbf{X}$ is denoted by $\|x\|$ and $\|\mathbf{X}\|$ respectively. For a finite set Q, $a \leftarrow Q$ denotes that a is sampled from Q with uniform distribution. For two vectors a, b, the inner product of a and b is denoted as $\langle a, b\rangle$.

2.2 Pseudorandom Generator

In our generic construction, we use a pseudorandom generator to generate a "pseudorandom". Informally, we say that a distribution $\mathcal{D}$ is pseudorandom if there does not exist any polynomial-time distinguisher that can distinguish a string $s \leftarrow \mathcal{D}$ from a string s chosen randomly and uniformly. We recall the definition of the pseudorandom generator in [18] Definition 3.15 below.

Definition 1 (Pseudorandom generator). *Let $F : \{0,1\}^n \to \{0,1\}^{n'}$ be a deterministic polynomial-time algorithm, where $n' = \mathsf{poly}(n)$ and $n' > n$. We say that F is a pseudorandom generator if it satisfies the following two conditions:*

- *Expansion: For every n, it holds that $n' > n$.*
- *Pseudorandomness: For all PPT distinguishers, $\mathcal{D}$,*

$$|Pr[\mathcal{D}(r) = 1] - Pr[\mathcal{D}(F(s)) = 1]| \leq \mathsf{negl}(n),$$

where r is chosen randomly and uniformly from $\{0,1\}^{n'}$, the seed s is chosen randomly and uniformly from $\{0,1\}^n$, and the probabilities depend on the random coins used by $\mathcal{D}$ and the choice of r and s.

2.3 Lattices

The construction of our instantiation is based on the NTRU lattices. In this section, we first briefly introduce the lattices theory, and then review some lattices hardness assumptions.

Lattices A m-dimension lattice Λ is an additive discrete subgroup of $\mathbb{R}^m$. Basically, a lattice is the set of all the integer combinations of some linearly independent vectors, called the basis of the lattice. The formal definition of the lattice is as follows.

Definition 2 (Lattice). *Let* $\mathbf{B} = [b_1|\cdots|b_n] \in \mathbb{R}^{m\times n}$ *be an* $m\times n$ *matrix, where* $b_1, \cdots, b_n \in \mathbb{R}^m$ *are* n *linear independent vectors. The* m*-dimensional lattice* Λ *generated by* $\mathbf{B}$ *is the set,*

$$\Lambda(\mathbf{B}) = \Lambda(b_1, \cdots, b_n) = \left\{ \sum_{i=1}^{n} b_i a_i | a_i \in \mathbb{Z} \right\}.$$

In addition, we call a lattice full-rank when $n = m$.

Hardness Assumptions In 2005, Regev introduced a new lattice hardness assumption, called learning with errors (LWE), and he demonstrated that several worst-case lattice problems can be reduced to the LWE problem [23]. In addition, he proposed the first public key cryptosystem based on the hardness of the LWE assumption.

Definition 3 (LWE Assumption). *Given* $n, m \in \mathbb{N}$*,* q *as a prime, a probability distribution* χ *over* $\mathbb{Z}_q$*. Suppose there exists an oracle* $\mathcal{O}_s^n$ *that outputs* m *samples of the form* $(a, \langle a, s\rangle + e)$ *where* $a \in \mathbb{Z}_q^n$ *and* $e \in \chi$ *are chosen freshly at random for each sample, and* $s \leftarrow \mathbb{Z}_q^n$ *is the same for every sample. The search-LWE assumption is to find the* s*. In addition, let* $\mathcal{O}_r$ *be an oracle that outputs samples* $(a, b) \leftarrow (\mathbb{Z}_q^n \times \mathbb{Z}_q^n)$ *uniformly at random. The decision-LWE assumption is to guess whether you are interacting with* $\mathcal{O}_s^n$ *or* $\mathcal{O}_r$*.*

With Regev's seminal work, many LWE-based cryptosystems were subsequently proposed [13, 14, 1, 7, 8]. Unfortunately, these cryptosystems encountered practical problems, because of the overly large key sizes and inefficiency. To solve the issue, in 2009, Lyubashevsky *et al.* introduced an algebraic variant of the LWE assumption [28] called ring-LWE [19]. The ring-LWE assumption is the LWE assumption specifically for polynomial rings over finite fields that can also be stated in "search" version and "decision" version that are defined as follows.

Definition 4 (Ring-LWE Assumption). *Given* $n, m \in \mathbb{N}$*, let* q *be a prime, a probability distribution* χ *over* $\mathcal{R}_q$*. Suppose there exists an oracle* $\mathcal{O}_s$ *that outputs* m *samples of the form* $(a, \langle a, s\rangle + e)$ *where* $a \in \mathcal{R}_q$ *and* $e \in \chi$ *is chosen freshly at random for each sample, and* $s \leftarrow \mathcal{R}_q$ *is the same for every sample. The search-Ring-LWE assumption is to find the* s*. In addition, let* $\mathcal{O}_r$ *be an oracle that outputs samples* $(a, b) \leftarrow (\mathcal{R}_q \times \mathcal{R}_q)$ *uniformly at random. The decision-Ring-LWE assumption is to guess whether the user is interacting with* $\mathcal{O}_s$ *or* $\mathcal{O}_r$*.*

Another lattice hardness assumption is the NTRU assumption, defined by Hoffstein *et al.* in 1998 [15].

Definition 5 (NTRU Assumption). *Let χ be a probability distribution over $\mathcal{R}_q$. The NTRU assumption is to distinguish the following two distributions. The first distribution sample is a polynomial $h = g/f$, where $f, g \leftarrow \chi$ and f is invertible, and the second distribution uniformly samples a polynomial h over $\mathcal{R}_q$.*

3 Cryptographic Building Blocks

In this section, we recall two important cryptographic primitives that are used as building blocks in our generic construction. They are the identity-based encryption and key encapsulation mechanism.

3.1 Identity-based Encryption

The identity-based encryption (IBE) is an essential primitive of public key encryption, in which the public key of a user is information that can identify the user (such as e-mail address, name, and social security number). Its concept was first proposed by Shamir as early as 1984 [24]. However, the first construction was realized by Cocks based on the quadratic residuosity problem in 2001 [9]. Later, Boneh and Franklin proposed a more practical and secure IBE using the pairing technique [5, 6].

An IBE scheme is a four-tuple of PPT algorithms $\mathcal{IBE} = (Setup, Extract, Enc, Dec)$, described as follows:

- $Setup(1^\lambda)$: Taking the security parameter λ as input, this algorithm outputs a master private key msk and master public key mpk.
- $Extract(\mathsf{msk}, \mathsf{id})$: Taking the master private key msk and an identity id as input, this algorithm outputs the corresponding private key $\mathsf{sk}_{\mathsf{id}}$ for the identity.
- $Enc(\mathsf{mpk}, \mathsf{id}, m)$: Taking the master public key, mpk, an identity id, and a message m as input, this algorithm outputs a ciphertext ct encrypted by id.
- $Dec(\mathsf{sk}_{\mathsf{id}}, \mathsf{ct})$: Taking a private key $\mathsf{sk}_{\mathsf{id}}$, and a ciphertext ct as input, this algorithm outputs a decrypted message m'.

Definition 6 (Correctness of IBE).
We say that an IBE scheme, $\mathcal{IBE}$, is correct if

$$\mathsf{Pr}[Dec(\mathsf{sk}_{\mathsf{id}}, Enc(\mathsf{mpk}, \mathsf{id}, m)) = m] = 1 - \mathsf{negl}(\lambda),$$

where $(\mathsf{mpk}, \mathsf{msk}) \leftarrow Setup(1^\lambda)$ *and* $\mathsf{sk}_{\mathsf{id}} \leftarrow Extract(\mathsf{msk}, \mathsf{id})$.

Besides, an IBE must satisfy the indistinguishability under adaptive chosen-identity chosen-plaintext attack (IND-ID-CPA), defined using the following game between a challenger $\mathcal{B}$ and an adversary $\mathcal{A}$.

Game IND-ID-CPA:

- **Setup.** In this stage, $\mathcal{B}$ runs the $Setup(1^\lambda)$ algorithm to generate the master public key mpk and master private key msk. Then, $\mathcal{B}$ keeps msk secret, and sends mpk to $\mathcal{A}$.
- **Phase 1.** $\mathcal{A}$ makes a polynomially bounded number of queries to the $Extract$ oracle on any identity id, and $\mathcal{B}$ returns a private key $\mathsf{sk}_{\mathsf{id}}$ to $\mathcal{A}$.
- **Challenge.** In this stage, $\mathcal{A}$ sends two challenge messages, m_0, m_1, and a challenge identity, id^*, to $\mathcal{B}$, where id^* has never been queried to $Extract$ oracle. After receiving the messages and identity, $\mathcal{B}$ chooses a random bit, $b \leftarrow \{0,1\}$, and generates the challenge ciphertext, $\mathsf{ct}^* \leftarrow Enc(\mathsf{mpk}, \mathsf{id}^*, m_b)$. Finally, $\mathcal{B}$ returns ct^* to $\mathcal{A}$.
- **Phase 2.** $\mathcal{A}$ can continue to ask for the $Extract$ oracle the same as in **Phase 1**. The only restriction is that $\mathcal{A}$ cannot issue an $Extract$ query on the challenge identity id^*.
- **Guess.** $\mathcal{A}$ outputs its guess b'. The adversary is said to win the game if $b' = b$. The advantage of $\mathcal{A}$ is as follows:

$$Adv^{IND-ID-CPA}_{\mathcal{IBE},\mathcal{A}}(\lambda) = |\mathsf{Pr}[b' = b] - \tfrac{1}{2}|.$$

Definition 7 (IND-ID-CPA of IBE).
We say that an IBE scheme $\mathcal{IBE}$ is IND-ID-CPA secure, if no PPT adversary $\mathcal{A}$ can win the above game with an advantage exceeding $\mathsf{negl}(\lambda)$.

Moreover, we say that an IBE scheme $\mathcal{IBE}$ is anonymous if it satisfies the following stronger notion of security:

Game IND-ANON-ID-CPA:

- **Setup.** In this stage, $\mathcal{B}$ runs the $Setup(1^\lambda)$ algorithm to generate the master public key mpk and master private key msk. Then $\mathcal{B}$ keeps msk secret, and sends mpk to $\mathcal{A}$.
- **Phase 1.** $\mathcal{A}$ makes a polynomially bounded number of queries to the $Extract$ oracle on any identity id, and $\mathcal{B}$ returns a private key $\mathsf{sk}_{\mathsf{id}}$ to $\mathcal{A}$.
- **Challenge.** In this stage, $\mathcal{A}$ sends a challenge message m, and two challenge identities $\mathsf{id}_0, \mathsf{id}_1$ to $\mathcal{B}$, where id_0 and id_1 have never been queried to $Extract$ oracle. After receiving the messages and identities, $\mathcal{B}$ chooses a random bit, $b \leftarrow \{0,1\}$, and generates the challenge ciphertext, $\mathsf{ct}^* \leftarrow Enc(\mathsf{mpk}, \mathsf{id}_b, m)$. Finally, $\mathcal{B}$ returns ct^* to $\mathcal{A}$.
- **Phase 2.** $\mathcal{A}$ can continue to ask for the $Extract$ oracle, the same as in **Phase 1**. The only restriction is that $\mathcal{A}$ cannot issue an $Extract$ query on the challenge identities id_0 and id_1.
- **Guess.** $\mathcal{A}$ outputs its guess b'. The adversary is said to have won the game, if $b' = b$. The advantage of $\mathcal{A}$ is defined as follows:

$$Adv^{IND-ANON-ID-CPA}_{\mathcal{IBE},\mathcal{A}}(\lambda) = |\mathsf{Pr}[b' = b] - \tfrac{1}{2}|.$$

Definition 8 (Anonymous IBE).
We say that an IBE scheme $\mathcal{IBE}$ is anonymous if is no PPT adversary $\mathcal{A}$ can win the above game with an advantage exceeding $\mathsf{negl}(\lambda)$.

3.2 Key Encapsulation Mechanism

The key encapsulation mechanism (KEM), first proposed by Cramer and Shoup, is a variant of the public key encryption [10]. Rather than encrypting a message, KEM "encaps" a random value using public key, and outputs an encapsulation. With the corresponding private key, anyone can "decaps" the encapsulation to obtain the same random value.

A KEM is a three-tuple of PPT algorithms, $\mathcal{KEM} = (KeyGen, Encaps, Decaps)$, described as follows.

- $KeyGen(1^\lambda)$: Taking the security parameter λ as input, this algorithm outputs a public key pk and a private key sk.
- $Encaps(\mathsf{pk})$: Taking the public key pk as input, this algorithm outputs a key k and an encapsulation e.
- $Decaps(\mathsf{sk}, e)$: Taking the private key sk and an encapsulation e as input, this algorithm outputs the corresponding key k, or an invalid symbol $\perp$.

Definition 9 (Correctness of KEM).
We say that a KEM scheme, $\mathcal{KEM}$, is correct, if

$$\mathsf{Pr}[Decaps(\mathsf{sk}, e) = k : (k, e) \leftarrow Encaps(\mathsf{pk})] = 1 - \mathsf{negl}(\lambda),$$

where $(\mathsf{pk}, \mathsf{sk}) \leftarrow KeyGen(1^\lambda)$.

Indistinguishability under the adaptive chosen-ciphertext-attack (IND-CCA2) security of a KEM is defined using the following game between a challenger $\mathcal{B}$ and an adversary $\mathcal{A}$.

Game IND-CCA2:

- **KeyGen.** In this stage, $\mathcal{B}$ runs the $KeyGen(1^\lambda)$ algorithm to generate the public/private key pair $(\mathsf{pk}, \mathsf{sk})$. Then, $\mathcal{B}$ sends pk to $\mathcal{A}$.
- **Phase 1.** $\mathcal{A}$ makes a polynomially bounded number of queries to the *Decaps* oracle on any encapsulation e; $\mathcal{B}$ returns a key k or invalid symbol $\perp$ to $\mathcal{A}$.
- **Challenge.** In this stage, $\mathcal{B}$ chooses a random bit $b \leftarrow \{0, 1\}$. Then, $\mathcal{B}$ generates $(e^*, k_0^*) \leftarrow Encaps(\mathsf{pk})$, and randomly chooses k_1^* from the key space $\mathcal{K}$. Finally, $\mathcal{B}$ returns the challenge ciphertext (e^*, k_b^*) to $\mathcal{A}$.
- **Phase 2.** $\mathcal{A}$ can continue to ask for the *Decaps* oracle, same as in **Phase 1**. The only restriction is that $\mathcal{A}$ cannot issue a *Decaps* query on e^*.
- **Guess.** $\mathcal{A}$ outputs its guess b'. The adversary is said to have won the game if $b' = b$. The advantage of $\mathcal{A}$ is defined as

$$Adv_{\mathcal{KEM},\mathcal{A}}^{IND-CCA2}(\lambda) = |\mathsf{Pr}[b' = b] - \tfrac{1}{2}|.$$

Definition 10 (IND-CCA2 of KEM).
We say that a KEM scheme $\mathcal{KEM}$ is IND-CCA2 secure, if there is no PPT adversary $\mathcal{A}$ that can win the above game with an advantage exceeding $\mathsf{negl}(\lambda)$.

4 Designated-ciphertext Searchable Encryption

In this section, we formalize the system model of a designated-ciphertext searchable encryption (DCSE) scheme and its security models.

4.1 System Model

We extend the system model of the PEKS in [4]. In more detail, the trapdoor in the DCSE is not only assigned to a keyword, it is assigned to a ciphertext. Let λ be a security parameter and $\mathcal{W}$ be a keyword space.

A DCSE is a four-tuple of PPT algorithms $\mathcal{DCSE} = (KeyGen,\ DCSE,\ Trapdoor,\ Test)$, described as follows.

- $KeyGen(1^\lambda)$: Taking the security parameter λ as input, this algorithm outputs a public key pk and a private key sk.
- $DCSE(\mathsf{pk}, w)$: Taking a data receiver's public key pk, and a keyword $w \in \mathcal{W}$, this algorithm outputs a searchable ciphertext c and a tag of ciphertext v.
- $Trapdoor(\mathsf{sk}, w', v')$: Taking a data receiver's private key sk, a keyword $w' \in \mathcal{W}$, and a tag of ciphertext v', this algorithm outputs a trapdoor t.
- $Test(c, t)$: Taking a searchable ciphertext c, and a trapdoor t as input, this algorithm outputs 1 if t matches c and 0 otherwise.

Definition 11 (Correctness of DCSE).
Let λ be a security parameter, $\mathcal{W}$ be a keyword space, $(\mathsf{pk}, \mathsf{sk}) \leftarrow KeyGen(1^\lambda)$, and (c, v) be a searchable ciphertext and a tag of the ciphertext be generated from $DCSE(\mathsf{pk}, w)$, where $w \in \mathcal{W}$. We say that a DCSE scheme is correct if:

$$\mathsf{Pr}[Test(c, Trapdoor(\mathsf{sk}, w, v)) = 1] = 1 - \mathsf{negl}(\lambda).$$

4.2 Security Models

We require that a DCSE scheme satisfies the following two security requirements: indistinguishability under chosen-keyword attack (IND-CKA) and indistinguishability under insider keyword guessing attack (IND-IKGA). The following two games are executed by an adversary $\mathcal{A}$ and a challenger $\mathcal{B}$. Note that because the ability of a malicious insider exceeds that of a malicious outsider, we only consider the insider keyword guessing attack here.

Indistinguishability under Chosen-keyword Attack The IND-CKA security ensures that the adversary cannot obtain any information on the keyword from a ciphertext and its corresponding tag.

Game IND-CKA:

- **KeyGen.** In this stage, $\mathcal{B}$ runs the $KeyGen(1^\lambda)$ algorithm to generate the user's public key pk and private key sk. Then, $\mathcal{B}$ sends sk to $\mathcal{A}$.

- **Phase 1.** $\mathcal{A}$ makes a polynomially bounded number of queries to the $Trapdoor$ oracle. When $\mathcal{A}$ issues such a query on (w, v), $\mathcal{B}$ returns a trapdoor t to $\mathcal{A}$ using $Trapdoor$ algorithm with the private key sk.
- **Challenge.** $\mathcal{A}$ sends two challenge keywords $w_0, w_1 \in \mathcal{W}$, where w_0, w_1 have not been queried in **Phase 1**. $\mathcal{B}$ chooses a random bit $b \leftarrow \{0, 1\}$, generates the challenge ciphertext $(c^*, v^*) \leftarrow DCSE(\mathsf{pk}, w_b)$, and returns it to $\mathcal{A}$.
- **Phase 2.** $\mathcal{A}$ can continue to ask for the $Trapdoor$ oracle, same as in **Phase 1**. The only restriction is that $\mathcal{A}$ cannot issue a $Trapdoor$ query on w_0 or w_1.
- **Guess.** $\mathcal{A}$ outputs its guess b'. The adversary is said to have won the game if $b' = b$.

The advantage of $\mathcal{A}$ wins this game is defined as

$$Adv^{IND-CKA}_{\mathcal{DCSE},\mathcal{A}}(\lambda) = |\mathsf{Pr}[b' = b] - \tfrac{1}{2}|.$$

Definition 12 (IND-CKA of DCSE).
We say that a DCSE scheme $\mathcal{DCSE}$ is IND-CKA secure if there is no PPT adversary $\mathcal{A}$ that can win the above game with an advantage exceeding $\mathsf{negl}(\lambda)$.

Indistinguishability under Insider Keyword Guessing Attack The IND-IKGA security ensures that the adversary cannot obtain any information about the keyword from a trapdoor.

Game IND-IKGA:

- **KeyGen.** In this stage, $\mathcal{B}$ runs the $KeyGen(1^\lambda)$ algorithm to generate the user's public key pk and private key sk. Then, $\mathcal{B}$ sends pk to $\mathcal{A}$.
- **Phase 1.** $\mathcal{A}$ makes a polynomially bounded number of queries to the $\mathsf{Trapdoor}$ oracle. When $\mathcal{A}$ issues such a query on (w, v), $\mathcal{B}$ returns a trapdoor t to $\mathcal{A}$ using $Trapdoor$ algorithm with private key sk.
- **Challenge.** $\mathcal{A}$ sends two challenge keywords $w_0, w_1 \in \mathcal{W}$, where w_0, w_1 have not been queried in **Phase 1**. $\mathcal{B}$ chooses a random bit $b \leftarrow \{0, 1\}$, a valid tag v^*, and generates the challenge trapdoor $t^* \leftarrow Trapdoor(\mathsf{sk}, w_b, v^*)$. Finally, $\mathcal{B}$ returns t^* to $\mathcal{A}$.
- **Phase 2.** $\mathcal{A}$ can continue to ask for the $Trapdoor$ oracle, same as in **Phase 1**. The only restriction is that $\mathcal{A}$ cannot issue a $Trapdoor$ query on w_0 or w_1.
- **Guess.** $\mathcal{A}$ outputs its guess b'. The adversary is said to have won the game if $b' = b$.

The advantage of $\mathcal{A}$ wins this game is defined as

$$Adv^{IND-IKGA}_{\mathcal{DCSE},\mathcal{A}}(\lambda) = |\mathsf{Pr}[b' = b] - \tfrac{1}{2}|.$$

Definition 13 (IND-IKGA of DCSE).
We say that a DCSE scheme $\mathcal{DCSE}$ is IND-IKGA secure, if there is no PPT adversary $\mathcal{A}$, that can win the above game with an advantage exceeding $\mathsf{negl}(\lambda)$.

5 Efficient Generic Construction of DCSE

In this section, we first propose a generic construction of the DCSE from an anonymous IBE and an IND-CCA2 KEM. Then, we present rigorous proofs to demonstrate that this construction satisfies the correctness and security requirements defined in Section 4.

5.1 Generic Construction

To construct a DCSE scheme $\mathcal{DCSE}$, we first set the following parameters. Let $\mathcal{IBE} = (Setup, Extract, Enc, Dec)$ be an anonymous IBE scheme, and $\mathcal{KEM} = (KeyGen, Encap, Decaps)$ be an IND-CCA2 secure KEM. Let $\mathcal{W}$ be the keyword space of $\mathcal{DCSE}$, and let $\mathcal{K}$ be the key space of $\mathcal{KEM}$. Let $F : \mathcal{X} \to \mathcal{Y}$ be a pseudorandom generator with appropriate domain $\mathcal{X}$ and range $\mathcal{Y}$. Here, the domain $\mathcal{X}$ includes the set of any keyword $w \in \mathcal{W}$ concatenating any key $k \in \mathcal{K}$. That is, $\mathcal{X} = \{w\|k \mid w \in \mathcal{W} \wedge k \in \mathcal{K}\}$. Furthermore, let the range $\mathcal{Y}$ include an appropriate length of randomness used by the algorithm $\mathcal{IBE}.Extract$. In addition, let H be a collision-resistant hash function defined on $\{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$.

We present a generic construction of DCSE from Algorithm 1 to Algorithm 4.

Algorithm 1 $KeyGen(1^\lambda)$

Input: a security parameter 1^λ
Output: user's key pair $(\mathsf{pk}, \mathsf{sk})$
1: $(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathcal{KEM}.KeyGen(1^\lambda)$
2: $(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow \mathcal{IBE}.Setup(1^\lambda)$
3: Set public key $\mathsf{pk} = (\mathsf{pk}_1, \mathsf{pk}_2)$, private key $\mathsf{sk} = (\mathsf{sk}_1, \mathsf{sk}_2)$
4: Output a key pair $(\mathsf{pk}, \mathsf{sk})$

Algorithm 2 $DCSE(\mathsf{pk}, w)$

Input: user's public key $\mathsf{pk} = (\mathsf{pk}_1, \mathsf{pk}_2)$ and a keyword $w \in \mathcal{W}$
Output: a ciphertext c and its corresponding tag v
1: $(e, k) \leftarrow \mathcal{KEM}.Encaps(\mathsf{pk}_1)$
2: Randomly choose $r \leftarrow \{0,1\}^*$
3: $f \leftarrow F(w\|k)$
4: $\mathsf{ct} \leftarrow \mathcal{IBE}.Enc(\mathsf{pk}_2, f, r)$
5: Compute $h = H(\mathsf{ct}, r)$
6: Output a ciphertext $c = (\mathsf{ct}, h)$ and tag $v = e$

Algorithm 3 $Trapdoor(\mathsf{sk}, w, v)$

Input: user's private key $\mathsf{sk} = (\mathsf{sk}_1, \mathsf{sk}_2)$, a keyword $w \in \mathcal{W}$, and a tag of the ciphertext $v = e$
Output: a trapdoor t for keyword w and tag v
1: $k \leftarrow \mathcal{KEM}.Decaps(v, \mathsf{sk}_1)$ $k = \perp$
2: **if** $k = \perp$ **then**
3: Set trapdoor t to be an invalid symbol $\perp$
4: **else**
5: $f \leftarrow F(w\|k)$
6: Set trapdoor $t \leftarrow \mathcal{IBE}.Extract(\mathsf{sk}_2, f)$
7: **end if**
8: Output a trapdoor t

Algorithm 4 $Test(c, t)$

Input: a ciphertext $c = (\mathsf{ct}, h)$, and a trapdoor t
Output: 1 if t matches c or 0 otherwise
1: **if** $t = \perp$ **then**
2: Output 0
3: **else**
4: $r \leftarrow \mathcal{IBE}.Dec(t, \mathsf{ct})$
5: Output 1 if $H(\mathsf{ct}, \mathsf{r}) = h$ and 0, otherwise
6: **end if**

5.2 Correctness and Security Proofs

Theorem 1. *The proposed construction is correct, according to Definition 11.*

Proof (Proof of Theorem 1). Let $(c = (\mathsf{ct}, h), v) \leftarrow DCSE(\mathsf{pk}, w)$ be a valid ciphertext and its corresponding tag, and let $t \leftarrow Trapdoor(\mathsf{sk}, w, v)$ be a valid trapdoor, where $(\mathsf{pk}, \mathsf{sk}) \leftarrow KeyGen(1^\lambda)$. Because t is actually the private key of identity $F(w\|k)$ in the IBE scheme, and ct is a ciphertext that encryptes a random value r using identity $F(w\|k)$. With the correctness of the IBE scheme (Definition 6), one can obtain $r \leftarrow \mathcal{IBE}.Dec(t, \mathsf{ct})$ with overwhelming probability. Therefore, $H(\mathsf{ct}, r) = h$; thus, we have $Test(c, t) = 1$.

In the following, we prove that the proposed scheme is IND-CKE secure and IND-IKGA secure.

Theorem 2. *The proposed scheme $\mathcal{DCSE}$ is IND-CKA secure if the underlying KEM scheme $\mathcal{KEM}$ is IND-CCA2 secure, the IBE scheme $\mathcal{IBE}$ is anonymous, and F is a secure pseudorandom generator.*

Proof (Proof of Theorem 2). We prove Theorem 2 using a sequence of games, defined as follows.

- **Game**$_0$: This is the original IND-CKA game, as shown in Section 4.2.

- **Game**$_1$: We now make a minor change to the above game. Rather than obtain k from $\mathcal{KEM}.Encaps(\mathsf{pk}_1)$, we choose k' from the range of the output of $\mathcal{KEM}.Encaps(\mathsf{pk}_1)$ randomly.
- **Game**$_2$: We now transform **Game**$_1$ into **Game**$_2$. In this game, let $f = F(w\|k')$; we substitute the value $\mathsf{ct} \leftarrow \mathcal{IBE}.Enc(f, r)$ with $\mathsf{ct} \leftarrow \mathcal{IBE}.Enc(f', r)$, where f' is chosen randomly from $\mathcal{Y}$, and $\mathcal{Y}$ is the output range of F.

Let Adv_i denote the adversary's advantage of winning in **Game**$_i$. We have the following claims.

Claim. For all the PPT algorithms $\mathcal{A}_{01}$, $|Adv_0 - Adv_1|$ is negligible, if the underlying KEM scheme $\mathcal{KEM}$ is IND-CCA2 secure.

Proof. Suppose that there exists an adversary $\mathcal{A}_{01}$ such that $|Adv_0 - Adv_1|$ is non-negligible, then, there exists another challenger $\mathcal{B}_{01}$ that can win the IND-CCA2 game in the underlying KEM scheme $\mathcal{KEM}$ with non-negligible advantage.

- **KeyGen.** $\mathcal{B}_{01}$ first invokes the IND-CCA2 game of $\mathcal{KEM}$ to obtain pk_1. Next, $\mathcal{B}_{01}$ computes $(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow \mathcal{IBE}.Setup(1^\lambda)$. Finally, $\mathcal{B}_{01}$ sets the public key $\mathsf{pk} = (\mathsf{pk}_1, \mathsf{pk}_2)$, and sends pk to $\mathcal{A}_{01}$.
- **Phase 1.** In this phase, $\mathcal{A}_{01}$ can make polynomially many $Trapdoor$ queries with (pk, w, v), and $\mathcal{B}_{01}$ responds as follows. $\mathcal{B}_{01}$ first invokes $\mathcal{KEM}.Decaps$ oracle on v. The oracle returns an invalid symbol $\perp$ or a valid key k. If the oracle returns $\perp$, $\mathcal{B}_{01}$ also responds with $\perp$ to $\mathcal{A}_{01}$. Otherwise, $\mathcal{B}_{01}$ computes $f \leftarrow F(w\|k)$ and $t \leftarrow \mathcal{IBE}.Extract(\mathsf{sk}_2, f)$. Finally, t is returned to $\mathcal{A}_{01}$.
- **Challenge.** $\mathcal{A}_{01}$ sends two challenge keywords $w_0, w_1 \in \mathcal{W}$, where w_0, w_1 have not been queried in **Phase 1**. After receiving these challenge keywords, $\mathcal{B}_{01}$ chooses a random bit $b \leftarrow \{0, 1\}$, and runs the following steps:
  - Invoke the Challenge phase of the IND-CCA2 game to obtain the challenge ciphertext (e^*, k^*)
  - Pick $r^* \leftarrow \{0, 1\}^*$
  - Compute $f^* \leftarrow F(w_b\|k^*)$
  - Compute $\mathsf{ct}^* \leftarrow \mathcal{IBE}.Enc(f^*, r^*)$
  - Compute $h^* = H(\mathsf{ct}^*, r^*)$
  - Set $v^* = e^*$

  Then, $\mathcal{B}_{01}$ returns $(c^* = (\mathsf{ct}^*, h^*), v^*)$ to $\mathcal{A}_{01}$.
- **Phase 2.** $\mathcal{A}_{01}$ can continue to make $Trapdoor$ queries, same as in **Phase 1**. The only restriction is that $\mathcal{A}_{01}$ cannot make a $Trapdoor$ query on w_0 or w_1.
- **Guess.** $\mathcal{A}_{01}$ outputs its guess b'. Then $\mathcal{B}_{01}$ outputs b'.

Note that, if k^* is a valid key, $\mathcal{B}_{01}$ gives the view of **Game$_0$** to $\mathcal{A}_{01}$; if k^* is a random element, then $\mathcal{B}_{01}$ gives the view of **Game**$_1$ to $\mathcal{A}_{01}$. Therefore, if $|Adv_0 - Adv_1|$ is non-negligible, $\mathcal{B}_{01}$ must also have non-negligible advantage against the IND-CCA2 game of the underlying KEM scheme.

Claim. For all the PPT algorithms, $\mathcal{A}_{12}$, $|Adv_1 - Adv_2|$ is negligible, if the underlying IBE scheme $\mathcal{IBE}$ is anonymous.

Proof. Supposing that there is an adversary $\mathcal{A}_{12}$ such that $|Adv_1 - Adv_2|$ is non-negligible, then, there exists another challenger $\mathcal{B}_{12}$ that can win the IND-ANON-ID-CPA game of the underlying IBE scheme $\mathcal{IBE}$ with non-negligible advantage. $\mathcal{B}_{12}$ constructs a hybrid game interacting with an adversary $\mathcal{A}_{12}$ as follows:

- **KeyGen.** $\mathcal{B}_{12}$ first invokes the IND-ANON-ID-CPA game of $\mathcal{IBE}$ to obtain pk_2; then, $\mathcal{B}_{12}$ computes $(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathcal{KEM}.KeyGen(1^\lambda)$. Finally, $\mathcal{B}_{12}$ sets the public key $\mathsf{pk} = (\mathsf{pk}_1, \mathsf{pk}_2)$, and sends pk to $\mathcal{A}_{12}$.
- **Phase 1.** In this phase, $\mathcal{A}_{12}$ is able to make polynomially many $Trapdoor$ queries with the (pk, w, v), and $\mathcal{B}_{12}$ responses as follows. $\mathcal{B}_{12}$ first obtains $k \leftarrow \mathsf{KEM.Decaps}(v, \mathsf{sk}_1)$. If k is an invalid symbol $\perp$, $\mathcal{B}_{12}$ returns $\perp$ to $\mathcal{A}_{12}$. Otherwise, $\mathcal{B}_{12}$ invokes $\mathcal{IBE}.Extract$ oracle on $F(k\|w)$ to obtain a trapdoor t. Finally, $\mathcal{B}_{12}$ sends t to $\mathcal{A}_{12}$.
- **Challenge.** $\mathcal{A}_{12}$ sends two challenge keywords w_0, w_1, where w_0, w_1 have not been queried in **Phase 1**. $\mathcal{B}_{12}$ chooses a random bit $b \leftarrow \{0,1\}$, and performs the following steps:
  - Compute $(e^*, k^*) \leftarrow \mathcal{KEM}.Encaps(\mathsf{pk}_1)$
  - Randomly choose k'^* from the range of the output of $\mathcal{KEM}.Encaps(\mathsf{pk}_1)$
  - Randomly choose $f' \leftarrow \mathcal{Y}$
  - Pick $r^* \leftarrow \{0,1\}^*$
  - Invoke the Challenge phase of the IND-ANON-ID-CPA game using $F(w_b\|k'^*, r^*)$ and (f', r^*) to obtain the challenge ciphertext ct^*
  - Compute $h^* = H(\mathsf{ct}^*, r^*)$
  - Set $v^* = e^*$

  Then, $\mathcal{B}_{12}$ returns $(c^* = (\mathsf{ct}^*, h^*), v^*)$ to $\mathcal{A}_{12}$.
- **Phase 2.** $\mathcal{A}_{12}$ can continue to make $Trapdoor$ queries, similar to **Phase 1**. The only restriction is that $\mathcal{A}_{12}$ cannot make a $\mathsf{Trapdoor}$ query on w_0 or w_1.
- **Guess.** $\mathcal{A}_{12}$ outputs its guess b'. Then, $\mathcal{B}_{12}$ outputs b'.

Note that if ct^* is generated from $(F(w_b\|k'^*), r^*)$, $\mathcal{B}_{12}$ gives the view of $\mathbf{Game}_1$ to $\mathcal{A}_{12}$; if ct^* is generated from (f', r^*), then $\mathcal{B}_{12}$ gives the view of $\mathbf{Game}_2$ to $\mathcal{A}_{12}$. Therefore, if $|Adv_1 - Adv_2|$ is non-negligible, $\mathcal{B}_{12}$ must also have non-negligible advantage in the IND-ANON-ID-CPA game of the underlying IBE scheme.

Claim. $Adv_2 = 0$

Proof. The proof is intuitive. Because the ciphertext c^* is irrelevant to the keywords w_0, w_1, the ciphertext reveals nothing about the information of the keywords. The adversary $\mathcal{A}_2$ can only return b' by guessing. Therefore, $Adv_2 = 0$.

Combining Claim 5.2, Claim 5.2, and Claim 5.2, we can conclude that the advantages of the adversary of the three adjacent games are negligibly close, and thus $|Adv_0 - Adv_2|$ is negligibly close to 0. This completes the proof of Theorem 2.

Theorem 3. *The proposed scheme is IND-IKGA secure, if the underlying KEM scheme $\mathcal{KEM}$ is IND-CCA2 secure, and F is a secure pseudorandom generator.*

Proof (Proof of Theorem 3).
We prove Theorem 3 through a sequence of games, defined as follows.

- **Game$_0$**: This is the original IND-IKGA game, as shown in Section 4.2.
- **Game$_1$**: This game is identical to **Game$_0$**, except that k is randomly chosen from the output range of $\mathcal{KEM}.Encaps(\mathsf{pk}_1)$, rather than being computed from $\mathcal{KEM}.Encaps(\mathsf{pk}_1)$.
- **Game$_2$** This game is the same as **Game$_1$**, except that f is chosen randomly from $\mathcal{Y}$, instead of being computed from $F(w_b\|k)$.

Let Adv_i denote the adversary's advantage in **Game$_i$**. We have the following claims.

Claim. For all the PPT algorithms, $\mathcal{A}_{01}$, $|Adv_0 - Adv_1|$ is negligible, if the underlying KEM scheme $\mathcal{KEM}$ is IND-CCA2 secure.

Proof. Supposing that there exists an adversary $\mathcal{A}_{01}$ such that $|Adv_0 - Adv_1|$ is non-negligible, then, there exists another challenger $\mathcal{B}_{01}$ that can win the IND-CCA2 game of the underlying KEM scheme $\mathcal{KEM}$ with non-negligible advantage.

- **KeyGen.** $\mathcal{B}_{01}$ first invokes the IND-CCA2 game of $\mathcal{KEM}$ to obtain pk_1. Next, $\mathcal{B}_{01}$ computes $(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow \mathcal{IBE}.Setup(1^\lambda)$. Finally, $\mathcal{B}_{01}$ sets the public key $\mathsf{pk} = (\mathsf{pk}_1, \mathsf{pk}_2)$, and sends pk to $\mathcal{A}_{01}$.
- **Phase 1.** In this phase, $\mathcal{A}_{01}$ can make polynomially many $Trapdoor$ queries with (pk, w, v), and $\mathcal{B}_{01}$ responses as follows. $\mathcal{B}_{01}$ first invokes $\mathcal{KEM}.Decaps$ oracle on v. The oracle returns an invalid symbol $\perp$ or a valid key k. If the oracle returns $\perp$, $\mathcal{B}_{01}$ also responses $\perp$ to $\mathcal{A}_{01}$. Otherwise, $\mathcal{B}_{01}$ computes $f = F(w\|k)$ and $t \leftarrow \mathcal{IBE}.Extract(\mathsf{sk}_2, f)$. Finally, t is returned to $\mathcal{A}_{01}$.
- **Challenge.** $\mathcal{A}_{01}$ sends two challenge keywords $w_0, w_1 \in \mathcal{W}$, where w_0, w_1 have not been queried in **Phase 1**. $\mathcal{B}_{01}$ chooses a random bit $b \leftarrow \{0, 1\}$, and runs the following steps:
  - Invoke the Challenge phase of the IND-CCA2 game to obtain the challenge (e^*, k^*)
  - Compute $f^* \leftarrow F(w_b\|k^*)$
  - Compute $t^* \leftarrow \mathcal{IBE}.Extract(\mathsf{sk}_2, f)$

  Then, $\mathcal{B}_{01}$ returns t^* to $\mathcal{A}_{01}$.
- **Phase 2.** $\mathcal{A}_{01}$ can continue to make $\mathsf{Trapdoor}$ queries, same as in **Phase 1**. The only restriction is that $\mathcal{A}_{01}$ cannot make a $Trapdoor$ query on w_0 or w_1.
- **Guess.** $\mathcal{A}_{01}$ outputs its guess b'. Then, $\mathcal{B}_{01}$ outputs b'.

Note that if k^* is a valid key, $\mathcal{B}_{01}$ gives the view of **Game$_0$** to $\mathcal{A}_{01}$; if k^* is a random element, then, $\mathcal{B}_{01}$ gives the view of **Game$_1$** to $\mathcal{A}_{01}$. Therefore, if $|Adv_0 - Adv_1|$ is non-negligible, $\mathcal{B}_{01}$ must also have non-negligible advantage in the IND-CCA2 game.

Claim. For all the PPT algorithms, $\mathcal{A}_{12}$, $|Adv_1 - Adv_2|$ is negligible, if F is a secure pseudorandom generator.

Proof. We prove the claim by describing a PPT reduction algorithm $\mathcal{B}_{12}$ that plays a pseudorandom generator security game. Given a challenge string $T \in \mathcal{Y}$ and the description of a pseudorandom generator F, $\mathcal{B}_{12}$ constructs a hybrid game, interacting with an adversary $\mathcal{A}_{12}$ as follows.

- **KeyGen.** $\mathcal{B}_{12}$ chooses the public parameters, as described in Section 5.1, except that, instead of choosing a proper pseudorandom generator from the pseudorandom generator family, $\mathcal{B}_{12}$ sets F as the public parameter. Then, $\mathcal{B}_{12}$ generates the key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow KeyGen(1^\lambda)$, and sends pk to $\mathcal{A}_{12}$. Note that $\mathcal{B}_{12}$ has full control of the private key sk.
- **Phase 1.** In this phase, $\mathcal{A}_{12}$ can make polynomially many $Trapdoor$ queries using (pk, w, v). Due to the knowledge of sk, $\mathcal{B}_{12}$ answers the queries by simply running the $Trapdoor$ algorithm.
- **Challenge.** $\mathcal{A}_{12}$ sends two challenge keywords $w_0, w_1 \in \mathcal{W}$, where w_0, w_1 have not been queried in **Phase 1**. $\mathcal{B}_{12}$ chooses a random bit $b \leftarrow \{0, 1\}$, and runs the following steps:
  - Set $f^* = T$
  - Compute $t^* \leftarrow \mathcal{IBE}.Extract(\mathsf{sk}_2, f^*)$

  Then, $\mathcal{B}_{12}$ returns t^* to $\mathcal{A}_{12}$.

Note that, if T is generated from F, $\mathcal{B}_{12}$ provides the view of $\mathbf{Game}_1$ to $\mathcal{A}_{12}$; if T is a random string sampled from $\mathcal{Y}$, then $\mathcal{B}_{12}$ provides the view of $\mathbf{Game}_2$ to $\mathcal{A}_{12}$. Therefore, if $|Adv_1 - Adv_2|$ is non-negligible, $\mathcal{B}_{12}$ must also have non-negligible advantage against the pseudorandom generator security game.

Claim. $Adv_2 = 0$.

Proof. The proof is intuitive. Because the trapdoor t^* is irrelevant to the keywords, w_0 and w_1, the trapdoor reveals nothing about the information of the keywords. The adversary $\mathcal{A}_2$ can only return b' by guessing. Therefore, $Adv_2 = 0$.

Combining Claim 5.2, Claim 5.2, and Claim 5.2, we can conclude that the advantages of the adversary of four adjacent games are negligibly close, and, thus, $|Adv_2 = Adv_0|$ is negligibly close to 0. This completes the proof of Theorem 3.

6 Efficient Instantiation and Comparison

In this section, we first propose an NTRU-based instantiation. Then, we compare different aspects in our instantiation with other state-of-the-art schemes.

6.1 Efficient Instantiation

Our instantiation utilizes Ducas *et al.*'s IBE [11] and Hülsing *et al.*'s KEM [17], hereafter referred to as DLP-IBE and HRSS-KEM, respectively, which are introduced below.

The DLP-IBE is the first lattice-based IBE scheme with practical parameters. Its security is based on the NTRU and Ring-LWE assumptions. In addition, the DLP-IBE has also been proven to be an anonymous IBE [3]. The first implementation of the DLP-IBE was provided by Ducas[1], written in C++, and based on the NTL library[2]. Although this implementation is very efficient, it is merely a Proof of Concept (PoC) without any optimization. To improve efficiency, McCarthy *et al.* propose a practical implementation of the DLP-IBE [22], written in ANSI C, using the number theoretic transform (NTT) optimizations.

The HRSS-KEM is a candidate cryptographic KEM in the Round 2 of the National Institute for Standards and Technology (NIST) Post-Quantum Project.[3]. In the work, Hülsing *et al.* first provide a OW-CPA secure NTRU-based encryption scheme with optimized parameters; then, they transform the scheme into a IND-CCA2 secure NTRU-based KEM under quantum-accessible random oracle model (QROM).

For concrete instantiation, we use SHA256 as a secure hash function, and symmetric encryption AES-256 as a pseudorandom generator. For the DLP-IBE, we select parameters $n = 1024, q \approx 2^{27}$ for 192-bit security level, and for HRSS-KEM, we select parameters $n = 701, p = 3, q = 8192$ for 128-bit security level. Based on the projects[4][5] of [11] and [17], we implement our NTRU-based DCSE written in Language C on Intel core i7-8700 3.2GHz CPU with 10G RAM.

6.2 Comparison

To compare with other state-of-the-art schemes, we set the parameters as follows. For the pairing-based PEKS scheme [4], we choose the 160-bit group order and 2048-bit group elements $\mathcal{G}, \mathcal{G}_T$. For the NTRU-based PEKS scheme [3], we choose $n = 1024, q = 2^{27}$ for 192-bits security level. While for the LWE-based PEKS schemes [29, 21, 30], we adopt the secure parameter the same as [29], that is $n = 256$, dimension $m = 9753$, and prime $q = 4093$. In addition, we set the number of distinct keywords $k = 1$ and unusual keywords $k' = 1$ for [21], and the security level $l = 10$ for [30].

Table 1 shows the comparison of our scheme with other schemes on the basis of its security properties. Only [21, 30], and our scheme possess both quantum-resistance IKGA security. Unfortunately, although the LWE-based schemes [29,

[1] https://github.com/tprest/Lattice-IBE
[2] https://www.shoup.net/ntl/
[3] https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions
[4] https://github.com/safecrypto/libsafecrypto
[5] https://github.com/ntru-hrss/ntru-hrss

21, 30] are quantum-resistant and supportive of other useful functions, as illustrated in Table 2, their overly large key sizes makes them impractical. Furthermore, compared with two IKGA secure quantum-resistant PEKS schemes [21, 30], our public key size and private key sizes are 1/115 and 1/2322 times smaller, respectively.

Table 1. Comparison with related schemes on the basis of security properties

Schemes	Quantum-resistance	IKGA security
[4]	✗	✗
[3]	✓	✗
[29]	✓	✗
[21]	✓	✓
[30]	✓	✓
Ours	✓	✓

Table 2. Comparison with related schemes on the basis of Key size, Trapdoor size, and Ciphertext size (in Bytes). Note that $|ID|$ refers to the length of user identity.

Schemes	PK	SK	Trapdoor	Ciphertext
[4]	0.38	0.19	0.38	0.57
[3]	27.2	35	27	52
[29]	$\|ID\|$	560128	113	113
[21]	3657.42	139325.1	71.42	57.14
[30]	3657.05	139325.1	142.86	14.28
Ours	31.88	59.98	38.98	23

In Table 3, we compare our instantiation with other two practical PEKS schemes [4, 3] on the basis of efficiency. Compared with [4], although our instantiation is 0.31x and 0.62x slower than that of the KeyGen and Test algorithms, respectively, our instantiation is 17x and 42x faster than that of the the Encrypt and Extract algorithms, respectively. As for [3], our instantiation is 245x, 9x, 11x, and 363x faster than that of the KeyGen, Encrypt, Extract, and Test algorithms, respectively. Additionally, we carefully experiment with the time required for the algorithms under different execution times (100, 500, 1000, 2000, 5000, 10000), the results are shown in Figure 1 to Figure 4.

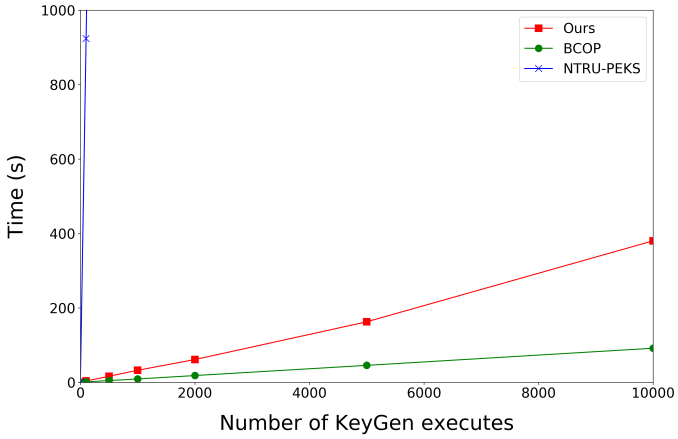
7 Conclusions and Open Problems

This work mainly proposed a new cryptographic primitive, designated-ciphertext searchable encryption (DCSE), to counter the insider keyword guessing attack in public key searchable encryption. We provided a generic construction of the

Table 3. Time taken (operations per second) by different operations of KeyGen (key generation), Encryption (PEKS in [4, 3] and DCSE in our scheme), Extract, and Test.

Scheme	KeyGen	Encryption	Extract	Test
[4]	84.88	186.48	17.41	100908.17
[3]	0.10	349.28	67.42	174.64
Ours	26.56	3224.35	739.06	63451.77

Fig. 1. Time taken by the key generation algorithm



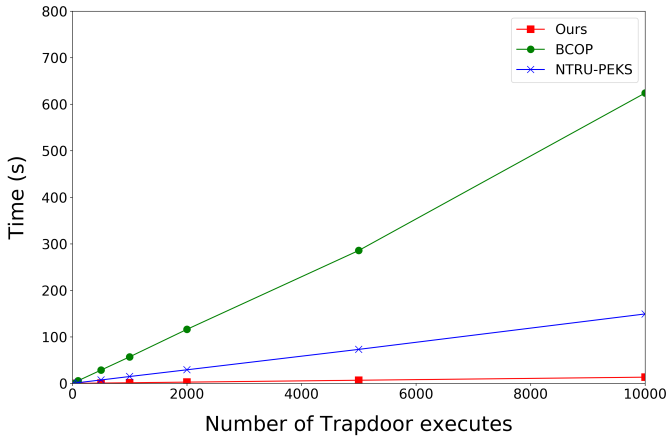
DCSE using an anonymous IBE and a IND-CCA2 KEM, and proved its security in the standard model. Furthermore, we provided a quantum-resistant instantiation from NTRU lattices utilizing [11] and [17]. In conclusion, this work provides a novel solution to the insider keyword guessing attack in a searchable encryption. In addition to yielding interesting theoretical results, the proposed scheme is very efficient and safe compared with other state-of-the-art schemes.

However, the proposed scheme has two challenges. The first is the communication overhead of the trapdoor that can be solved by designating multiple ciphertexts instead of only one. The second challenge is the complexity of the DCSE's construction; therefore, it is necessary to provide a simpler generic construction of the DCSE.

References

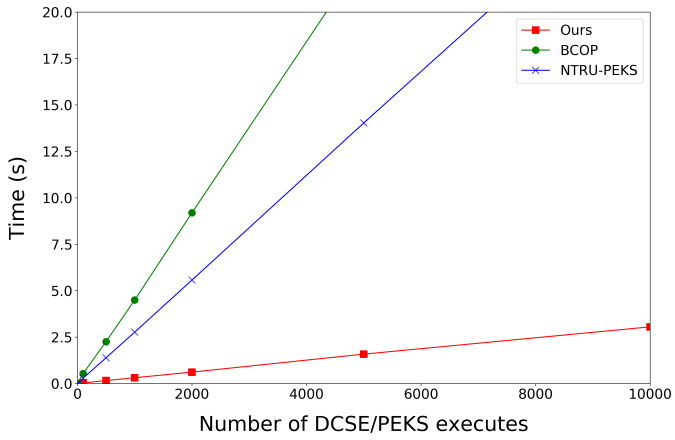
1. Agrawal, S., Boneh, D., Boyen, X.: Efficient lattice (h)ibe in the standard model. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 553–572. Springer (2010)
2. Arute, F., Arya, K., Babbush, R., Bacon, D., Bardin, J.C., Barends, R., Biswas, R., Boixo, S., Brandao, F.G., Buell, D.A., et al.: Quantum supremacy using a programmable superconducting processor. Nature **574**(7779), 505–510 (2019)

Fig. 2. Time taken by the extract algorithm



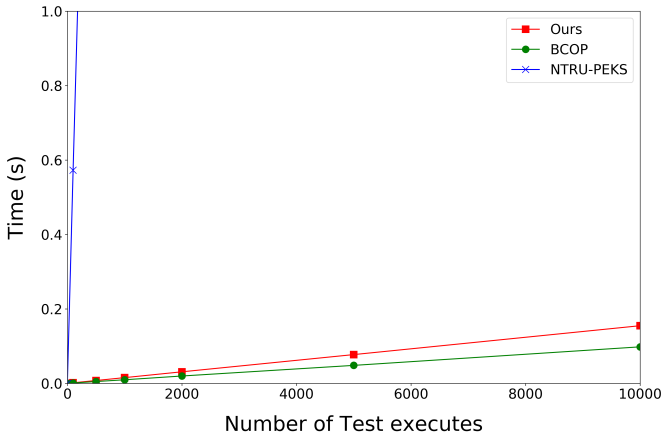
3. Behnia, R., Ozmen, M.O., Yavuz, A.A.: Lattice-based public key searchable encryption from experimental perspectives. IEEE Transactions on Dependable and Secure Computing (2018)
4. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: International conference on the theory and applications of cryptographic techniques. pp. 506–522. Springer (2004)
5. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Annual international cryptology conference. pp. 213–229. Springer (2001)
6. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. SIAM journal on computing **32**(3), 586–615 (2003)
7. Boyen, X.: Attribute-based functional encryption on lattices. In: Theory of Cryptography Conference. pp. 122–142. Springer (2013)
8. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) lwe. SIAM Journal on Computing **43**(2), 831–871 (2014)
9. Cocks, C.: An identity based encryption scheme based on quadratic residues. In: IMA International Conference on Cryptography and Coding. pp. 360–363. Springer (2001)
10. Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. SIAM Journal on Computing **33**(1), 167–226 (2003)
11. Ducas, L., Lyubashevsky, V., Prest, T.: Efficient identity-based encryption over ntru lattices. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 22–41. Springer (2014)
12. Fang, L., Susilo, W., Ge, C., Wang, J.: Public key encryption with keyword search secure against keyword guessing attacks without random oracle. Information Sciences **238**, 221–241 (2013)
13. Gentry, C., Boneh, D.: A Fully Homomorphic Encryption Scheme, vol. 20. Stanford University Stanford (2009)

Fig. 3. Time taken by the DCSE / PEKS algorithm



14. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Proceedings of the fortieth annual ACM symposium on Theory of computing. pp. 197–206. ACM (2008)
15. Hoffstein, J., Pipher, J., Silverman, J.H.: Ntru: A ring-based public key cryptosystem. In: International Algorithmic Number Theory Symposium. pp. 267–288. Springer (1998)
16. Huang, Q., Li, H.: An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks. Information Sciences **403**, 1–14 (2017)
17. Hülsing, A., Rijneveld, J., Schanck, J., Schwabe, P.: High-speed key encapsulation from ntru. In: International Conference on Cryptographic Hardware and Embedded Systems. pp. 232–252. Springer (2017)
18. Katz, J., Lindell, Y.: Introduction to Modern Cryptography. Chapman and Hall/CRC (2014)
19. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 1–23. Springer (2010)
20. Ma, S., Mu, Y., Susilo, W., Yang, B.: Witness-based searchable encryption. Information Sciences **453**, 364–378 (2018)
21. Mao, Y., Fu, X., Guo, C., Wu, G.: Public key encryption with conjunctive keyword search secure against keyword guessing attack from lattices. Transactions on Emerging Telecommunications Technologies p. e3531 (2018)
22. McCarthy, S., Smyth, N., O'Sullivan, E.: A practical implementation of identity-based encryption over ntru lattices. In: IMA International Conference on Cryptography and Coding. pp. 227–246. Springer (2017)
23. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing. pp. 84–93. STOC '05, ACM, New York, NY, USA (2005). https://doi.org/10.1145/1060590.1060603, http://doi.acm.org/10.1145/1060590.1060603

Fig. 4. Time taken by the test algorithm



24. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Workshop on the theory and application of cryptographic techniques. pp. 47–53. Springer (1984)
25. Shor, P.W.: Algorithms for quantum computation: Discrete logarithms and factoring. In: Proceedings 35th annual symposium on foundations of computer science. pp. 124–134. Ieee (1994)
26. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM review **41**(2), 303–332 (1999)
27. Song, D.X., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: Proceeding 2000 IEEE Symposium on Security and Privacy. S&P 2000. pp. 44–55. IEEE (2000)
28. Stehlé, D., Steinfeld, R., Tanaka, K., Xagawa, K.: Efficient public key encryption based on ideal lattices. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 617–635. Springer (2009)
29. Xu, L., Yuan, X., Steinfeld, R., Wang, C., Xu, C.: Multi-writer searchable encryption: An lwe-based realization and implementation. In: Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security. pp. 122–133. ACM (2019)
30. Zhang, X., Xu, C., Wang, H., Zhang, Y., Wang, S.: Fs-peks: Lattice-based forward secure public-key encryption with keyword search for cloud-assisted industrial internet of things. IEEE Transactions on Dependable and Secure Computing (2019)