Lattice-based searchable public-key encryption scheme for secure cloud storage

Run Xie*
School of Computer Science and Engineering,
University of Electronic Science and Technology of China,
Chengdu 611731, China

and

School of Mathematical,
Yibin University,
Yibin 644000, China
Email: xrryun@126.com
*Corresponding author

Chunxiang Xu
School of Computer Science and Engineering,
University of Electronic Science and Technology of China,
Chengdu 611731, China
Email: chxxu@uestc.edu.cn

Chanlian He
School of Computer and Information Engineering,
Yibin University,
Yibin 644000, China
Email: hel7811@163.com

Xiaojun Zhang
School of Computer Science,
Southwest Petroleum University,
Chengdu 610500, China
Email: zhangxjdzkd2012@163.com

Abstract: With the improvement of awareness of data privacy, the user’s sensitive data are usually encrypted before uploading them to cloud. Searchable encryption is a critical technique on promoting secure and efficient cloud storage. In particular, public-key encryption with keyword search (PEKS) provides an elegant approach to achieve data retrieval in encrypted storage. All existing searchable public-key encryption schemes only provide the security based on classical cryptography hardness assumption. With the development of
quantum computers, these schemes will be insecure. Based on the lattice hardness assumptions, we propose a new searchable public-key encryption scheme with a designated tester (dPEKS). Our scheme has advantages: First, our scheme is the first searchable public-key encryption scheme that is considered to be secure even if quantum computers are ever developed. Second, our scheme achieves the trapdoor indistinguishability. The trapdoor indistinguishability implies the security against outside offline keyword guessing attacks (KGAs). Last, our scheme can achieve the trapdoor anonymity for server.

**Keywords:** cloud storage; keyword-guessing attack; lattice; searchable encryption; trapdoor indistinguishability.


**Biographical notes:**

Run Xie received MSc in Mathematics and Applied Mathematics at Southwest Jiaotong University in 2006, P.R. China. He is a PhD candidate in information security at University of Electronic Science Technology of China (UESTC). He is presently engaged in cryptography, network security and cloud computing security.

Chunxiang Xu received BSc, MSc and PhD at Xidian University, in 1985, 1988 and 2004 respectively, P.R. China. She is presently engaged in information security, cloud computing security and cryptography as a Professor at University of Electronic Science Technology of China (UESTC).

Changlian received MSc in the soft engineering special at University of Electronic Science Technology of China (UESTC) in 2009, P.R. China. She is presently engaged in cryptography, network security and cloud computing security.

Xiaojun Zhang received BSc in Mathematics and Applied Mathematics at Hebei Normal University in 2009, P.R. China and received MSc in Pure Mathematics at Guangxi University in 2012. He received PhD in Information Security at University of Electronic Science Technology of China (UESTC) in 2015. He is a Lecturer at School of Computer Science in Southwest Petroleum University. He is presently engaged in cryptography, network security and cloud computing security.

1 Introduction

Cloud storage service offers great convenience to users. Users remotely outsource their data to the cloud so as to enjoy the on-demand high-quality services from a shared pool of configurable computing resources with low cost. To protect data privacy, users have to encrypt the sensitive data before uploading to clouds. However, the simple encryption of the data makes the data utilisation (such as data retrieval) a very challenging task in the cloud.
Public-key encryption scheme for secure cloud storage

If users access the data they stored by downloading all the data set first then searching over the decrypted, this requires high consumption of bandwidth and deviates original intention. If they submit the keys to the cloud, the data confidentiality will be lost.

The searchable encryption (Song, Wagner and Perrig, 2000) allows the cloud server to search over encrypted data without decryption, which was first introduced by Song et al. (2000). It does not leak any information about the data and query. Therefore, searchable encryption is a critical technique promoting efficient and secure cloud storage.

The searchable encryption has been developed into two different types. The first type is the symmetric searchable encryption (SSE in short) scheme. The SSE schemes require that a sender is securely granted a secret key from the intended receiver. They suffer from risks of key leakage in management and distribution (Song, Wagner and Perrig, 2000). The second type is the searchable public-key encryption scheme with keyword search (PEKS in short), which allows any one seeing the receiver’s public key to encrypt documents.

The PEKS provides an efficient mechanism to achieve data retrieval in encrypted storage. In a PEKS scheme, the sender generates the searchable ciphertext of a keyword with receiver’s public key and stores it to the server. To retrieve the encrypted data associated with a given keyword, the receiver creates a search request (trapdoor) using the keyword and his private key. After receiving a trapdoor, the cloud server performs a test to see whether there are any encrypted data matching the trapdoor and return corresponding encrypted data to the receiver.

The first searchable public-key encryption scheme with keyword search (Boneh and Boyen, 2004) was proposed by Boneh et al. (2004). In Boneh et al.’s scheme, the secure channel is required to protect the trapdoors in the transport channel. Since building a secure channel is usually expensive, this requirement limits applications of the searchable public-key encryption scheme. To overcome this obstacle, in 2008, Baek et al. proposed secure channel-free public-key encryption scheme with keyword search (Baek, Safavi-Naini and Susilo, 2008) (SCF-PEKS in short), which removes the secure channel requirement.

Nevertheless, Yau et al. (2008) showed that this scheme is insecure for the following reason. With outside keyword-guessing attacks (outside KGAs), an outside adversary can reveal encrypted keywords, if he obtains a trapdoor in channel. To resist this threat, in (Rhee et al., 2010), a searchable public-key encryption scheme with a designated tester (dPEKS in short) is proposed. In their scheme, only a designated server can test whether given trapdoor matches the ciphertexts.

Afterwards, a variety of PEKS schemes (Xu et al., 2013; Fang et al., 2009; Gu and Zhu, 2010; Jeong et al., 2009; Liu, Wang and Wu, 2009; Rhee, Susilo and Kim, 2009; Chen, Chen and Li, 2014; Liu et al., 2015; Zhu and Yang, 2015; Hu, Yang and Liu, 2015; Yu et al., 2014; Hu and Liu, 2011; Zhao et al., 2012) were proposed with different functionalities and levels of efficiency. Many of the searchable public-key encryption schemes pay more attention to improving the security against outside KGA (Xu et al., 2013; Fang et al., 2009; Gu and Zhu, 2010; Jeong et al., 2009; Liu, Wang and Wu, 2009; Rhee, Susilo and Kim, 2009). Only a few schemes (Hu and Liu, 2011; Zhao et al., 2012; Fang et al., 2013; Rhee et al., 2010) can effectively resist outside KGA. Furthermore, all the existing dPEKS schemes only provide the security based on classical cryptography hardness assumption. With the development of quantum computers and enhance cloud-computing power, these schemes will suffer serious security threats. A natural goal is to find the new solutions that are secure even if quantum computers are ever developed.
Fortunately, the situation has changed with lattice-based cryptography appearance. Currently, lattice-based cryptography is considered as the most promising options for post-quantum cryptography. In addition, it enjoys the benefits of provable security under worst-case hardness assumptions, asymptotic efficiency. Therefore, it is a significant research work to design strong secure (stand against outside KGA) and lattice-based dPEKS.

1.1 Our contributions

Motivated by the issue discussion above, we construct a new searchable public-key encryption scheme with a designated tester (dPEKS). Our contributions are summarised as follows:

- First, our scheme is first lattice-based searchable public-key encryption scheme with a designated tester. It enjoys provable security under LWE hardness assumption, asymptotic efficiency, especially security against quantum computers. Since our scheme is constructed over lattice, it is usually required only linear operations on small integers, while the existing dPEKS are required pairing operations and exponentiation. Therefore, our scheme is the promising candidate for the traditional schemes.

- Second, our scheme achieves the trapdoor indistinguishability. The trapdoors indistinguishability implies the security against off-line KGA from outside attacks. Until now, very few schemes can resist outside off-line KGA. According to (Fang et al., 2013), in Boneh et al.’s original framework, it is not possible to construct a dPEKS scheme secure against inside (server) KGA. In this sense, our scheme provides the strongest security level.

- Last, our scheme can achieve the trapdoor anonymity for server.

2 Preliminaries

In this section, we first review the definition of dPEKS and its security model which is defined in Rhee et al. (2010). Meanwhile, we also review the related notions of lattice, the sampling technique from lattice and the LWE assumption. At the end, we review a lattice-based IBE scheme (Gentry, Peikert and Vaikuntanathan, 2008) which is the main building block to construct our scheme.

2.1 Definition of dPEKS and security model

2.1.1 Definition of dPEKS

As stated in Rhee et al. (2010), a dPEKS scheme is defined as follows.

**Definition 2.1** A dPEKS scheme consists of the following four PPT (probability polynomial-time) algorithms \((\text{GlobalSetup}, \text{KeyGen}, \text{dPEKS}, \text{dTrapdoor}, \text{dTest})\). In the sequel, \(CP\) denotes a set of common parameters.
• **GlobalSetup** \( (n) \): Taking as input security parameter \( n \), this algorithm outputs a set of common parameters \( CP \).

• **KeyGen** \( (CP) \): Taking as input \( CP \), this algorithm outputs the receivers a public/private key pair \( (PK_r, SK_r) \) and the servers a public/private key pair \( (PK_s, SK_s) \).

• **dPEKS** \( (CP, PK_r, PK_s, w) \): Taking as input \( CP \), the receiver’s public key \( PK_r \), the server’s public key \( PK_s \) and a keyword \( w \), this algorithm outputs a dPEKS ciphertext \( CT_w \) corresponding to \( w \).

• **dTrapdoor** \( (CP, PK_s, SK_r, PK_r, w) \): Taking as input a receiver’s public/private key \( (PK_s, SK_r) \), the server’s public key \( PK_r \) and a keyword \( w \), this algorithm outputs the trapdoor \( T_w \) of \( w \).

• **dTest** \( (CP, CT_w, SK_s, T_w) \): Taking as input a dPEKS ciphertext \( CT_w \) of keyword \( w \), the trapdoor \( T_w \) and the server’s private key \( SK_s \), this algorithm outputs ‘yes’, if \( w' = w \), and otherwise it outputs ‘no’.

2.1.2 Security model of dPEKS

2.1.2.1 Security of dPEKS ciphertext

In this security model, the dPEKS ciphertexts (an encrypted list of keywords) should satisfy indistinguishability against a chosen plaintext attack (C-IND-CPA in short). According to Rhee et al. (2010), the C-IND-CPA guarantees that

1. A server cannot distinguish between the dPEKS ciphertexts of two challenge keywords \( w_0 \) and \( w_1 \) it chooses, if he has not obtained their trapdoor.
2. An outside adversary (including a receiver) who can generate the trapdoors of any keyword (excluding challenge keywords) cannot distinguish between the dPEKS ciphertexts of \( w_0 \) and \( w_1 \) it chooses, if he has not obtained the server’s private key.

Let \( \mathcal{F}_1 \) and \( \mathcal{F}_2 \) be a malicious server and an outside adversary, respectively. The C-IND-CPA can be defined with the following two games.

**Game1** Here, \( \mathcal{G} \) is a challenger and \( \mathcal{F}_1 \) is a malicious server.

- **Setup**: \( \mathcal{F}_1 \) generates \( (PK_s, SK_s) \) as his public/private key pair. \( \mathcal{G} \) generates \( (PK_r, SK_r) \) as receiver’s public/private key pair. Next, the tuples \( (PK_r, SK_r, PK_s) \) are given to \( \mathcal{F}_1 \), and the tuples \( (PK_s, SK_s, PK_r) \) are given to \( \mathcal{G} \).

- **Phase 1 Trapdoor queries**: \( \mathcal{F}_1 \) queries many keywords \( w \in \{0,1\}^* \) to obtain their trapdoors \( T_w \). \( \mathcal{G} \) can adaptively asks \( T_w = d\text{Trapdoor}(PK_s, SK_r, w) \) for any keyword \( w \in \{0,1\}^* \) and returns \( T_w \) to \( \mathcal{F}_1 \), where dTrapdoor is the trapdoor generation oracle.

- **Challenge**: \( \mathcal{F}_1 \) chooses the keyword pair \( (w_0, w_1) \) as his challenge. Here, the restriction is that \( w_0 \) and \( w_1 \) have not been queried to obtain the trapdoors \( T_w \) and
Phase 2 Trapdoor queries: $\mathcal{F}_1$ can still query $w$ to obtain its trapdoor as phase 1. If the $w \neq w_0, w_1$, $\mathcal{G}$ adaptively responses $\mathcal{F}_1$ with $T_w$ as phase 1.

Outputs: $\mathcal{F}_1$ outputs its guess $b' \in \{0, 1\}$. If $b' = b$, then $\mathcal{F}_1$ wins Game1.

Let $\text{adv}_{\mathcal{F}_1}^{\text{dPEKS-ind-CPA}} = |Pr(b' = b) - \frac{1}{2}|$ denote the advantage probability that $\mathcal{F}_1$ wins the Game1.

Game2 Here, $\mathcal{G}$ is a challenger and $\mathcal{F}_2$ an outside adversary (including a malicious receiver).

Setup: $\mathcal{F}_2$ generates $(PK_r, SK_r)$ as his public/private key pair. $\mathcal{G}$ generates $(PK_s, SK_s)$ as his public/private key pair. The tuples $(PK_r, SK_r, PK_s)$ are given to $\mathcal{F}_2$, the tuples $(PK_s, SK_s, PK_s)$ are given to $\mathcal{G}$. Here, $\mathcal{F}_2$ can generate the trapdoor of any keyword since he holds $SK_r$.

Challenge: $\mathcal{F}_2$ chooses the keyword pair $(w_0, w_1)$ as his challenge. Here, the restrictions is that $\mathcal{F}_2$ did not previously ask the dTest oracle for the trapdoors of $w_0$ and $w_1$. Receiving $w_0$ and $w_1$, $\mathcal{G}$ chooses $b \in \{0, 1\}$ and generates the ciphertext $CT_n$ of $wb$, and returns it to $\mathcal{F}_2$.

Output: $\mathcal{F}_2$ outputs its guess $b' \in \{0, 1\}$. If $b' = b$, then $\mathcal{F}_2$ wins Game2.

Let $\text{adv}_{\mathcal{F}_2}^{\text{dPEKS-ind-CPA}} = |Pr(b' = b) - \frac{1}{2}|$ denote the advantage probability that $\mathcal{F}_2$ wins the Game2.

Definition 2.2 For any polynomial-time attackers $\mathcal{F}_1$ and $\mathcal{F}_2$, we say that a dPEKS scheme satisfies dPEKS indistinguishability against an adaptive chosen plaintext attack, if $\text{adv}_{\mathcal{F}_1}^{\text{dPEKS-ind-CPA}} = |Pr(b' = b) - \frac{1}{2}|$ is negligible.

2.1.2 Security of trapdoor

According to Rhee et al. (2010), the trapdoor indistinguishability (T-IND-CPA) requires that an adversary (excluding the receiver and the server) should not be able to distinguish between the trapdoors of two challenge keywords, $w_0$ and $w_1$, where $w_0$ and $w_1$ are chosen by the adversary.

Let $\mathcal{F}_3$ denote an outside adversary, the security of trapdoor is defined with the following Game3.

Game3 $\mathcal{G}$ is a challenger and $\mathcal{F}_3$ is an outside adversary.

Setup: Running GlobalSetup and KeyGen, the common parameter $CP$, the receiver’s key pair $(PK_r, SK_r)$ and the server’s key pair $(PK_s, SK_s)$ are generated. $CP$, $PK_r$ and $PK_s$ are given to $\mathcal{F}_3$, while $SK_s$ and $SK_r$ are kept secret from $\mathcal{F}_3$. 
• **Phase 1 Trapdoor queries:** \( \mathcal{F}_3 \) queries many keywords \( w \in \{0,1\}^* \) to obtain trapdoors \( T_w \). \( \mathcal{G} \) adaptively asks \( T_w = \text{dTrapdoor}(PK_s, SK_s, w) \) for any \( w \in \{0,1\}^* \) and returns \( T_w \) to \( \mathcal{F}_3 \).

• **Challenge:** \( \mathcal{F}_3 \) chooses \((w_0, w_1)\) as his challenge and sends them to \( \mathcal{G} \). Here, the restriction is that \( w_0 \) and \( w_1 \) have not been queried to obtain the trapdoors \( T_{w_0} \) and \( T_{w_1} \), and that \( \mathcal{F}_3 \) did not previously ask for \( T_{w_0} \) and \( T_{w_1} \) in phase 1. Receiving \((w_0, w_1)\), \( \mathcal{G} \) chooses a random \( b \in \{0,1\} \) and computes \( b \cdot T_w \leftarrow \text{dTrapdoor}(PK_s, PK_s, w_0) \). Then he returns \( T_{w^b} \) to \( \mathcal{F}_3 \).

• **Phase 2 Trapdoor queries:** In this phase, \( \mathcal{F}_3 \) can still query the trapdoor of \( w \) as phase 1, where \( w \neq w_0, w_1 \). \( \mathcal{G} \) adaptively answers \( \mathcal{F}_3 \) as Phase 1.

• **Outputs:** \( \mathcal{F}_3 \) outputs its guess \( b' \in \{0,1\} \). If \( b' = b \), then \( \mathcal{F}_3 \) wins **Game3**.

Let \( \text{adv}_{\mathcal{F}_3}^{T_{\text{ind}} \hspace{1pt} \text{-spa}} = | Pr(b' = b) - \frac{1}{2} | \) denote the advantage probability of \( \mathcal{F}_3 \) wins the **Game3**.

**Definition 2.3** For any polynomial-time \( \mathcal{F}_3 \), we say that a dPEKS scheme satisfies trapdoor indistinguishability, if \( \text{adv}_{\mathcal{F}_3}^{T_{\text{ind}} \hspace{1pt} \text{-spa}} = | Pr(b' = b) - \frac{1}{2} | \) is negligible.

### 2.2 Lattice and lattice-based IBE scheme

In this section, we first review the related notions and the complexity assumptions, and then provide the definition of the lattice-based IBE scheme.

#### 2.2.1 Statistical distance

Given two random variables \( \mathcal{D}_1 \) and \( \mathcal{D}_2 \) over the finite set \( S \), the statistical distance between \( \mathcal{D}_1 \) and \( \mathcal{D}_2 \) is defined as

\[
\Delta(\mathcal{D}_1; \mathcal{D}_2) := \frac{1}{2} \sum_{s \in S} | Pr[\mathcal{D}_1 = s] - Pr[\mathcal{D}_2 = s] |.
\]

Let \( \mathcal{U} \) be a uniform random variable over \( S \). The distribution \( \mathcal{D} \) is called \( \epsilon \)-uniform over \( S \), if \( \Delta(\mathcal{D}; \mathcal{U}) \leq \epsilon \).

It is said that \( \mathcal{D}_1 \) and \( \mathcal{D}_2 \) are statistically close, if \( d(n) := \Delta(\mathcal{D}_1; \mathcal{D}_2) \) is a negligible function of \( n \).

#### 2.2.2 Lattices

Here, we overview the notions of lattices and the well-known LWE assumption.

A lattice is the set of vectors which are all integer combinations of \( n \) linearly independent vectors. Let \( \mathbb{A} = \{ a_1, \ldots, a_n \} \subset \mathbb{R}^n \) be \( n \) linearly independent vectors, then \( L(\mathbb{A}) \)
denotes the set of all integral linear combinations of \( \{a_1, \ldots, a_t\} \). The \( L(A) \) is called a \( t \)-dimensional lattice in \( \mathbb{R}^n \) (Micciancio and Goldwasser, 2012).

For \( t = n \), \( L(A) \) is called full-rank \( n \)-dimensional lattice. In addition, let \( L^t(A) \) denote the set of \( \{e \in \mathbb{Z}^n : Ae = 0\} \), then \( L^t(A) \) is called the orthogonal lattice of \( L(A) \). Obviously, \( L^t(A) \) is full-rank lattice. On hard random lattices, the well-known lattice problem is the shortest vector problem \( \text{SVP}_p \), where the meaning of \( p \) is in the \( l_p \) norm. The \( \text{SVP}_p \) is defined as follows:

**Definition 2.4** (Micciancio and Goldwasser, 2012) Let \( L(A) \) be a lattice, where \( A \) is a basis of \( L(A) \). An input to \( \text{GapSVP}_p \) is a pair \((A, d)\), where \( d \) a rational number. In **YES** inputs \( \exists v_0 \subset L(A) \) s.t. \( \|v_0\|_p \leq d \) and in **NO** inputs \( \forall v \not\in L(A) \) s.t. \( \|v\|_p > \eta d \).

According to the result of Ajtai (1999), there is the trapdoor basis generate algorithm on lattice (\( \text{TrapGen} \) in short). Namely, there is the following lemma (Gentry, Peikert and Vaikuntanathan, 2008).

**Lemma 2.1** For any prime \( q = \text{poly}(n) \) and \( m \geq 5n \lg q \), there exists an efficient PPT (probabilistic polynomial-time) algorithm that, on input \( 1^n \), outputs a \( A \in Z_q^{m \times n} \) and \( T_i \in Z_q^{n \times n} \), where \( A \)'s distribution is statistically close to uniform over \( Z_q^{m \times n} \) and \( T_i \in Z_q^{n \times n} \) is a invertible matrix and \( \|T_i\| \leq m^{1/2} \).

The \( \text{TrapGen} \) also denotes \((A, T_i) \leftarrow \text{TrapGen}(n, m, q)\). Furthermore, Gentry et al. introduced the \( \text{SamplePre} \) algorithm in Gentry, Peikert and Vaikuntanathan (2008). Specially, there is the following lemma.

**Lemma 2.2** Given \( q \geq 2 \) and a matrix \( A \in Z_q^{m \times n} \) with \( m \geq n \), let \( T_i \) be a basis of \( L^t(A) \) and \( \alpha \geq \|T_i\|_1 \alpha \frac{\sqrt{\log m}}{m} \). Then for \( u \in Z_q^n \), there is a PPT algorithm \( \text{SamplePre} \) such that \( x \leftarrow \text{SamplePre}(A, T_i, u, \alpha) \), where \( x \in L^t_q(A) \) sampled from a distribution statistically close to \( D_{Z_q^n(A), x} \), whenever \( L^t_q(A) \) is not empty.

### 2.2.3 Learning with errors

The learning with errors (LWE) problem was introduced in Micciancio and Goldwasser (2012). Given security parameters \( n \), a prime \( q(n) \) and a distribution \( \chi \) over \( Z_q \), then for a uniform vector \( s \leftarrow Z_q^n \) (secret), the \( M_{s,x} \) is the distribution of the variable \((a.s + x)\) on \( Z_q^n \times Z_q \), where \( x \) is a fresh sample from \( \chi \) and \( a_i \) is uniform in \( Z_q^n \). Let \( \mathcal{U} \) be the uniform distribution on \( Z_q^n \times Z_q \). The LWE problem is defined as follows:

**Definition 2.5** Micciancio and Goldwasser (2012) The decision \((n, \chi, Z_q)\)-LWE problem is to distinguish between \( M_{s,x} \) and \( \mathcal{U} \) with a non-negligible probability, where \( M_{s,x} \) and \( \mathcal{U} \) are obtained via oracle repeated access to the given distribution.

The (LWE) assumption is that the LWE problem is hard.
To show the relationship between LWE problem and $\text{GapSVP}_\alpha$, we need the following definition.

**Definition 2.6** (Agrawal, Boneh and Boyen, 2010) Let $\alpha \in (0, 1)$ and $\mathcal{D}(0, \alpha / \sqrt{2\pi})$ be the normal distribution, where 0 and $\alpha / \sqrt{2\pi}$ are standard deviation and mean, respectively. Let $\mathcal{N}$ be sampled from $\mathcal{D}$. The $\Psi_\alpha$ is defined as the distribution over $\mathbb{Z}_q$ of the random variable $\lceil q\mathcal{N} \rceil$, where $q$ is a prime.

In $(n, \chi, \mathbb{Z}_q)$-LWE problem, let $\chi = \Psi_\alpha$, then the relation between LWE problem and $\text{GapSVP}$ is showed in the following theorem.

**Theorem 2.1** (Regev, 2005) For $(n, \chi, \mathbb{Z}_q)$ such that $q > \frac{2n}{\alpha}$, assume there is an efficient algorithm (possibly quantum) that solves $(n, \chi, \mathbb{Z}_q)$-LWE problem, then there exists an efficient algorithm (quantum) that approximates the $\text{GapSVP}$ and $\text{SIVP}$ problems, to within $O(n/\alpha)$ in the worst case.

Putting the above together, if LWE is hard, then the distributions $s\chi$ is statistically close to the uniform distribution $\mathcal{U}$ on $\mathbb{Z}_q^* \times \mathbb{Z}_q$. Namely, the collection of distributions $s\chi$ is pseudorandom.

**Remark:** Usually, a LWE instance simple refers to $a\tilde{s} + x$ since the vector $a$ is uniform random.

In addition, to show that our scheme works correctly, the following lemma (Agrawal, Boneh and Boyen, 2010) will be required.

**Lemma 2.3** Let $x \leftarrow \Psi_\alpha^m$, where $\Psi_\alpha^m$ is the descartes product of $\Psi_\alpha$. Given a vector $s \in \mathbb{Z}_q^*$, then the $|x\cdot s|$ treated as an integer in $[0, q-1]$ satisfies

$$|x\cdot s| \leq \|q\alpha \cdot (\log m + \sqrt{m}/2)\|
$$

with all but negligible probability in $m$.

Identity-based encryption (IBE) is a useful tool to construct a searchable public-key encryption scheme. Recently, some IBE schemes (Li and Zhang, 2013; Luo and Chen, 2014; Gentry, Peikert and Vaikuntanathan, 2008) were proposed. Based on the lattice-based IBE (Gentry, Peikert and Vaikuntanathan, 2008), we give a searchable public-key encryption scheme. Now, we first review this lattice-based IBE scheme.

### 2.2.4 Identity-based encryption

Let $u \in \mathbb{Z}_q^*$ denote the user’s identity. Let $D_{\mathcal{E}_r}$ be a discrete Gaussian distribution on $\mathbb{Z}_q^*$ with $r \geq \text{Ran}(\log m)$, where $\text{R}$ is some value. With the $\text{TrapGen}$ and $\text{SamplePre}$ technology, a lattice-based IBE scheme was introduced by Gentry et al. It is defined as follows.

**IBE.Setup** ($\Gamma$): Given security parameters $n, m$ and $q$, then $(A, T_r) \leftarrow \text{TrapGen}(n, m, q)$. Here, $A \in \mathbb{Z}_q^{\omega_m}$ is statistically close to uniform over $\mathbb{Z}_q^{\omega_m}$ and $T_r \in L^*(A)$ is a good basis. $A$ and $T_r$ are the master public key and the master secret key, respectively.
In this section, we construct a lattice-based searchable public-key encryption scheme with SamplePre technology, then $e \leftarrow \text{SamplePre}(A, T_\epsilon, u)$. Store $(u; e)$ locally and return $e$.

**IBEExtract** $(A, T_\epsilon, u)$: Given the identity of user $u$, with SamplePre technology, then $e \leftarrow \text{SamplePre}(A, T_\epsilon, u)$. Store $(u; e)$ locally and return $e$.

**IBEEnc**$(A, u, b)$: Choose $s \leftarrow Z_q^*$ uniformly, then a bit $b \in \{0,1\}$ encrypted with the identity $u$ as follows. $p = A \cdot s + x \in Z_q^*$, $c = u \cdot s + x + b \lfloor q/2 \rfloor \in Z_q$, where $x \leftarrow \chi^m$ and $x \leftarrow \chi$. At the end, it outputs the ciphertext $(p, c)$ of $b$.

**IBEDec**$(e, (p, c))$: To decrypt $b$, let $b = c - e \cdot p \in Z_q$. Output 0 if $b$ is closer to 0 than to $\lfloor q/2 \rfloor \mod q$, otherwise output 1.

### 3 Lattice-based dPEKS

In this section, we construct a lattice-based searchable public-key encryption scheme with a designated tester (dPEKS). According to the model of Section 2, our scheme is described as follows. Here, all operations are performed over $Z_q$ and let $\chi = \Psi_a$.

- **Setup** $(n)$: Let $n$ be the natural security parameter. Let $m = O(n \log q)$ and $q = O(n^2)$ be some positive integers. Let $D_{\chi^m}$ be a discrete Gaussian distribution with $r \simeq \text{Ran}(\sqrt{\log m})$. Let a hash function $H_i : \{0,1\}^* \rightarrow w \in Z_q^*$ that maps keywords to a vectors in $Z_q^*$, and $H_i : \{0,1\}^* \rightarrow u \in Z_q$. Then the common parameters are $CP = (n, m, q, k, H_1, H_2)$, where $k$ is some positive integers with $k<n$.

- **KeyGen** $(CP)$: 1. Generates server’s key: Let $PK_s$ and $SK_s$ denote the server’s public key and private key, respectively. Takes $CP$ as input and runs TrapGen algorithm, then $(M_s, T_s) \leftarrow \text{TrapGen}(n, m, q)$, where $M_s \in Z_q^{m \times m}$ and $T_s \in Z_q^{m \times m}$ is a basis of $L^x(M_s)$ and is invertible matrix. In the end, this algorithm outputs $PK_s = M_s$, $SK_s = T_s$. 2. Generates receiver’s key: Let $PK_r$ and $SK_r$ denote the receiver’s public key and private key, respectively. Takes $CP$ as input and runs TrapGen algorithm, then $(M_r, T_r) \leftarrow \text{TrapGen}(n, m, q)$ and chooses an random matrix $U \in Z_q^{m \times m}$, where $M_r \in Z_q^{m \times m}$ and $T_r \in Z_q^{m \times m}$ is a basis of $L^x(M_r)$ and is invertible matrix. In the end, this algorithm outputs $PK_r = (M_r, U_r)$, $SK_r = T_r$.

- **dPEKS** $(PK_r, PK_s, w)$: Takes $CP$, a keyword $w$, the receiver’s public key $(M_r, U_r)$ and the server’s public key $M_s$ as input, then a sender chooses a random string $l \in \{0,1\}^*$, $x_1, x_2, x_3 \in \chi^m$, two random vectors $s_1, s_2 \in Z_q^*$ and an random matrix $V \in Z_q^{m \times m}$. Let $H_i(w) = \omega = (\beta_1, \ldots, \beta_m) \in Z_q^*$ and computes $U_w = M_s \cdot U'_w$, where $U'_w \in Z_q^{m \times m}$ is defined as follows.

$$U'_w = \begin{pmatrix}
H_i(u_1) \beta_1 & H_i(u_2) \beta_2 & \cdots & H_i(u_m) \beta_m \\
\vdots & \ddots & \ddots & \vdots \\
\vdots & \ddots & \ddots & \vdots
\end{pmatrix}$$
Here, each $u_i$ is the elements of $U_i$. At the end, the sender computes a ciphertext of keyword $w$ as follows.

$$CT_w = (CT_1, CT_2),$$

$$CT_1 = (CT_{11}, CT_{12}, CT_{13}) = (V, M'x_1 + x_1, V'x_1 + x_1 + L_1 \lfloor q/2 \rfloor);$$

$$CT_2 = (CT_{21}, CT_{22}) = (M'x_2 + x_1, U_1x_2 + x_2 + L_2 \lfloor q/2 \rfloor);$$

- **dTrapdoor** ($PK_r, SK_r, PK_w, w'$): Takes the receiver’s public $PK_r = (M_r, U_r)$ key and private key $SK_r = T_r$, the server’s public key $M_s$, a keyword $w'$ as input, and the receiver chooses two uniform random vectors $x \in Z^r_x, s_x \in Z^r_x$. Let $h(w') = w' = (\beta_1', \ldots, \beta_k')$ and $U_v = M_U E_v$, where $U_v \in Z^{ex}_{nx}$ is defined as follows.

$$U_v' = \begin{pmatrix}
\beta_6 u_{i_1} \beta_6 u_{i_2} \\
\beta_6 u_{i_3} \beta_6 u_{i_4} \\
\vdots \\
\beta_6 u_{i_k} \beta_6 u_{i_k}
\end{pmatrix}$$

Here, each $u_i$ is the elements of $U_i$. Furthermore, runs SamplePre algorithm, $e_{i_v} \leftarrow \text{SamplePre}(M_r, T_r, u_{i_v})$, $i = 1, \ldots, m$, where $u_{i_v}$ is the $i$-th column of $U_v$, such that $M_i e_{i_v} = u_{i_v}$. Let $(e_{1_v}, \ldots, e_{k_v}) = E_v \in Z^{ex}_{nx}$, then the receiver generates the trapdoor of $w'$ as follows.

$$T_v = (T_{v1}, T_{v2}) = (M'x_v + x_v, f_s(E_v)).$$

The $f_s(E_v)$ is defined as follows.

Let $x = (x_1, \ldots, x_n)$, with periodically take the components of $x$ such that $e_1 + x_i$ is 1-th elements of $f_s(E_v), e_2 + x_i$ is 2-th elements of $f_s(E_v)$, and so on.

$$f_s(E_v) = \begin{pmatrix}
e_1 + x_1 & e_2 + x_2 \\
\vdots & \vdots \\
e_k + x_k & e_{k+1} + x_{k+1}
\end{pmatrix}$$

- **dTest** ($T_v, CT_w, SK_r, PK_r$): Takes $CP$, $CT_w$ and the server’s private key and public key $SK_r, PK_r$, as input.

1. Runs SamplePre, then $e_{i_v} \leftarrow \text{SamplePre}(M_r, T_r, u_{i_v})$, $i = 1, \ldots, k$, where $v_i$ is the $i$-th column of $CT_1 = V$ such that $M_i e_{i_v} = v_i$. Let $(e_{1_v}, \ldots, e_{k_v}) = E_v$, then the server computes $L_{v1} = CT_{12} - E_v CT_{12}$, for $L_{vi} (i$-th component of $L_{v1}) i = (1, \ldots, k)$, if $L_{vi}$ is closer to 0 than to $\lfloor q/2 \rfloor \bmod q$ output 0, otherwise output 1.

2. With his private key $SK_r$, the server computes $(T_{v1}^i)^T(T_{v1}^i) = x$ and recovers $E_v$ with $x$. Next, he computes $L_{v1}^x = CT_{12} - E_v CT_{12} = L_1 \lfloor q/2 \rfloor + x_1 - E_v x_1$. Let $L_{vi}^x (i$-th component of $L_{v1}^x) i = (1, \ldots, k)$, if $L_{vi}^x$ is closer to 0 than to $\lfloor q/2 \rfloor \bmod q$ output $L_{vi}^x = 0$, otherwise output $L_{vi}^x = 1$.

3. Return ‘Correct’, if $L_{v1}^x = L_{v1}$, otherwise output ‘Incorrect’. 

Public-key encryption scheme for secure cloud storage
4 Correctness and security

4.1 Correctness

In this section, we prove that our scheme is correct.

Theorem 4.1 Suppose the lattice-based IBE scheme is correct, then our lattice-based dPEKS scheme is correct.

Proof: First, the server can correctly decrypt $L_i = L_i$ with his private key, if the lattice-based IBE scheme is correct. Because $CT_{21} = M_s x_s + x_s + L_i \{ q/2 \}$, $CT_{22} = E_s CT_{21} + x_s - E_s x_s$. Obviously, $L_i = L_i'$ implies $\|x_i' - (e_i')^\top x_i\| < \lceil q/2 \rceil$ and $w = w'$.

Now, we only need to show $\|x_i' - (e_i')^\top x_i\| < \lceil q/2 \rceil$.

Here $(x_i', x_s) \leftarrow \Psi_{\alpha}^{m+1}$ and $e_i' \in D_{a(m+1)}$. Let $q \geq 6r(m+1)$ and $\alpha \leq 1 / (r \sqrt{m+1} \log(m+1))$, then $\|1 - e_i'\| \leq r \sqrt{m+1}$ (except with exponentially small probability, see Regev, 2005).

Let $x_i' - (e_i')^\top x_i = (1 - e_i')^\top (x_i', x_s)$. By Lemma 2.3, we have

$$\|1 - e_i'\| (x_i', x_s) \leq \|1 - e_i\| (q \alpha \sqrt{\log(m+1)} + \sqrt{m+1} / 2).$$

Therefore it suffices to show that $\|x_i' - (e_i')^\top x_i\| < \lceil q/2 \rceil$ with overwhelming probability.

4.2 Security

4.2.1 Security of dPEKS Ciphertext

Theorem 4.2 For any polynomial-time adversary $\mathcal{A}$, if the LWE assumption hold, then our scheme can achieve dPEKS ciphertext indistinguishability (C-IND-CPA).

Proof: According to the security model of Section 2, our proof is divided into two parts. In the first, let the polynomial-time adversary $\mathcal{A}$ be the malicious server. In the second, let the polynomial-time adversary $\mathcal{B}$ be an outside adversary (including receiver). Let $\mathcal{G}$ denote a simulator.

The First Part If $\mathcal{A}$ can break our scheme with an advantage probability $\varepsilon_1$, then we construct a simulator $\mathcal{G}$ which can solve the decision LWE problem with the advantage probability $\varepsilon_1$.

• Setup: Taking $CP$, $\mathcal{A}$ generates $(PK_s = M_s, SK_s = T_s)$ as his public/private key pair, where the invertible matrix $T_s$ is a basis of $L_i(M_i)$. Taking $CP$, $\mathcal{G}$ generates $(PK_r = (M_r, U_r), SK_r = T_r)$ as his public/private key pair. The $PK_s$ and $PK_r$ are shared with all users. Meanwhile, let $E$ be an LWE oracle which can generate two random
variables $A_{x,z} = A^\top z + x_z \in \mathbb{Z}_q^k$ and $u$, where $A$ is an uniform random matrix over $\mathbb{Z}_q^{n \times k}$, $x_z \in \mathbb{X}^k$, $z$ is uniform (secret) vector over $\mathbb{Z}_q^n$, and $u \in \mathbb{Z}_q^k$ is uniform random vector. $(A_{x,z}, u)$ are given to $G$ as his challenge.

- **Hash queries**: $F_i$ inquires a hash value of $w$, the $H_i$ replies with an element $H_i(w) = w \in \mathbb{Z}_q^k$, ensuring to reply identically when the same query is occurred. Similar, $H_i$ replies $F_i$ with an element $H_i(u) \in \mathbb{Z}_q^k$.

- **Phase 1 Trapdoor queries**: $F_i$ inquires a trapdoor $T_w$ corresponding to $w$, then $G$ responds as follows. According to security model, $G$ can adaptively ask $T_w = d\text{Trapdoor}(PK_w, SK_w, w)$. Because $U_v$ and $E_v, (m, T_v, U_w, T_w) = (T_1^w, T_2^w, (M, z, \chi x E_v))$. Then $G$ sends $T_w$ to $F_i$, where $s_z \in \mathbb{Z}_q^k$ and $s \in \mathbb{X}^k$.

- **Challenge**: $F_i$ chooses $w_0$ and $w_1$ as his challenge keywords. Let $H_i(w_0) = w_0, H_i(w_1) = w_1 \in \mathbb{Z}_q^k$, then $F_i$ sends them to $G$. Here, the restriction is that $F_i$ did not previously inquired the trapdoors of $w_0$ and $w_1$. Receiving $w_0, w_1$, then $G$ selects a random matrix $V \in Z_{q^{13}}$, an uniform random string $L_s \in \{0,1\}^l$, the random vectors $s_1, s_2 \in Z_{q^{13}}^k$, and $x_1, x_2 \in \mathbb{X}^k$. Meanwhile, $G$ computes $U_{w_0}$ and $U_{w_1}$ with $H_i(w_0), w_0, w_1$. At the end, using the challenge $(A_{x,z}, u)$, $G$ computes the dPEKS ciphertexts as follows.

1. If $G$ chooses $A_{x,z}$, then the dPEKS ciphertexts $CT_{w_0} = (CT_{w_0}^1, CT_{w_0}^2)$ are $CT_{w_0}^1 = (V, M, s_1 + x_1, V^\top s_2 + x_2 + L_s q^{13}, CT_{w_0}^2 = (M, z, s_2 + x_2).$

2. If $G$ chooses $u$, then the dPEKS ciphertexts $CT_{w_1} = (CT_{w_1}^1, CT_{w_1}^2)$ are $CT_{w_1}^1 = (V, M, s_1 + x_1, V^\top s_2 + x_2 + L_s q^{13}, CT_{w_1}^2 = (M, s_2 + x_2, U_{w_1} s_2 + u + L_s q^{13}).$

Then $G$ picks an random $b \in \{0,1\}$ and sends $CT_{w_0}$ to $F_i$.

- **Phase 2 Trapdoor queries**: After receiving $CT_{w_0}$, $F_i$ can still inquiry the trapdoor of $w$, as long as $w \neq w_0, w_1$. $G$ answers $F_i$ as Phase 1 by asking the dTrapdoor oracle.

- **Guess**: Finally, $F_i$ output guesses $b^*$. If $b^* = b$, $G$ outputs 1 (indicating that $A_{x,z}$ is selected from challenge); otherwise, it outputs 0.
Probability Analysis If $A_{x,z}$ is selected, the simulation is perfect, and $\mathcal{F}_1$ will guess the bits $b$ correctly with probability $\frac{1}{2} + \varepsilon_1$. Else, $U_{u,s} + u$ is a uniformly random vector. $\mathcal{F}_1$ will guess the bits $b$ correctly with probability $\frac{1}{2}$. Therefore, $\mathcal{G}$ can distinguish between $A_{x,z}$ and $u$ with advantage probability $\left|\frac{1}{2} + \varepsilon_1 - \frac{1}{2}\right| = \varepsilon_1$.

The Second Part If $\mathcal{F}_2$ can break our scheme with an advantage probability $\varepsilon_2$, then we construct a simulator $\mathcal{G}$ which can solve the decision LWE problem with the advantage probability $\varepsilon_2$.

- **Setup**: $\mathcal{F}_2$ generates $(PK_r = (M_r, U_r), SK_r = T_r)$ as his public and private key pair. $\mathcal{G}$ generates $(PK_s = M_s, SK_s = T_s)$ as the server’s public and private key pair. Here, $(PK_r, SK_r)$ are given to $\mathcal{F}_2$. $(PK_s, SK_s)$ and $PK_r$ are given to $\mathcal{G}$. Let $\mathcal{E}$ generate two random variables $A_{x,z} = A'z + x$, and $u$. Then $(A_{x,z}, u)$ are given to $\mathcal{G}$ as his challenge.

- **Hash queries**: $\mathcal{F}_2$ inquires a hash value of $w$, the $\mathcal{H}_r$ replies with an element $\mathcal{H}_r(w) = w \in Z_q^*$, ensuring to reply identically when the same query is occurred. Similar, $\mathcal{H}_s$ replies $\mathcal{F}_2$ with an element $\mathcal{H}_s(u) \in Z_q^*$.

- **Challenge**: Suppose $\mathcal{F}_2$ chooses $w_0$ and $w_1$ as challenge keywords. Let $\mathcal{H}_r(w_0) = w_0$, $\mathcal{H}_s(w_1) = w_1 \in Z_q^*$, then $\mathcal{F}_2$ sends them to $\mathcal{G}$. The restriction is that $\mathcal{F}_2$ did not previously inquire the dTest oracle for the trapdoors, $T_{w_0}$ and $T_{w_1}$. Receiving $w_0$ and $w_1$, $\mathcal{G}$ picks the random matrix $V \in Z_q^{k \times n}$, $s_1, s_2 \in Z_q^*$ and $x_1, x_2 \in \mathcal{X}^n$, $x_2 \in \mathcal{X}^k$, an uniform random string $L \in \{0,1\}^k$. Similar to the first part, $\mathcal{G}$ computes $U_{w_0}$ and $U_{w_1}$ with $\mathcal{H}_r, w_0$ and $w_1$. Using the challenge LWE instances $A_{x,z}$ and $u$, $\mathcal{G}$ computes the dPEKS ciphertext as follows.

1. If $\mathcal{G}$ chooses $A_{x,z}$, then the dPEKS ciphertexts $CT_{w_0} = (CT_{w_0}^1, CT_{w_0}^2)$ ( $b \in \{0,1\}$ ) are $CT_{w_0}^1 = (V, M_r s_1 + x, V^T s_1 + x_1 + L \cdot \lfloor q/2 \rfloor)$, $CT_{w_0}^2 = (M_s s_1 + x_1$, $U_{w_0}s_2 + A_{x,z} + L \cdot \lfloor q/2 \rfloor)$. Since $k<n$, there exists $s'$ such that $U_{w_0} s' = U_{w_0} s_2 + A'z$. Thus, for some unknown $s'$, $U_{w_0} s_2 + A_{x,z} = U_{w_0} s' + x_2$ is a LWE instance. Its distribution is identical to that in the actual system.

2. If $\mathcal{G}$ chooses $u$, then the dPEKS ciphertexts $CT_{w_1} = (CT_{w_1}^1, CT_{w_1}^2)$ ( $b \in \{0,1\}$ ) are $CT_{w_1}^1 = (V, M_r s_1 + x_1, V^T s_1 + x_2 + L \cdot \lfloor q/2 \rfloor)$, $CT_{w_1}^2 = (M_s s_1 + x_1$, $U_{w_1}s_2 + u + L \cdot \lfloor q/2 \rfloor)$. Then $\mathcal{G}$ picks a random $b \in \{0,1\}$, and send $CT_{w_0}$ to $\mathcal{F}_2$. 


**Guess:** Finally, $\mathcal{F}_2$ outputs guesses $b'$. If $b' = b$, $\mathcal{G}$ outputs 1 (indicating that $A_{s,x}$ is selected from challenge); otherwise, it outputs 0.

**Probability Analysis:** Based on the same reasons as the first part, $\mathcal{G}$ can distinguish between $A_{s,x}$ and $u$ with advantage probability $\left(\frac{1}{2} + \epsilon_2\right) - \frac{1}{2} = \epsilon_2$.

This completes the proof of dPEKS ciphertexts indistinguishability.

### 4.2.2 Security of trapdoor

In this section, we show that our scheme achieves the trapdoor indistinguishability for any outside adversary (excluding server and receiver).

**Theorem 4.3** For any polynomial-time outside adversary $\mathcal{F}_3$, if the LWE assumption holds, our scheme achieves the trapdoor indistinguishability against a chosen keywords attack.

**Proof:** Suppose there exists an adversary $\mathcal{F}_3$ breaking the trapdoor indistinguishability of our scheme with advantage probability $\epsilon_3$, then we construct a simulator $\mathcal{G}$ which can solve decision LWE problem with the advantage probability $\epsilon_3$. Here, $\mathcal{F}_3$ is an outside adversary (excluding receiver and server).

- **Setup:** By running GlobalSetup and KeyGen, then the common parameter $CP$, the receiver’s key pair $(PK_r = (M_r, U_r), SK_r = T_r)$ and the server’s key pair $(PK_s = M_s, SK_s = T_s)$ are generated. Here, $CP$, $PK_r$ and $PK_s$ are given to $\mathcal{F}_3$ while $SK_r$ and $SK_s$ are kept secret from him. Meanwhile, let LWE oracle $\mathcal{E}$ generate two random variables $A_{s,x} = A \cdot z + x_s$ and $u$. Then $(A_{s,x}, u)$ are given to $\mathcal{G}$ as his challenge, where $A_{s,x}$ is a LWE instance and $u$ is a uniform random vector.

- **Hash queries:** $\mathcal{F}_3$ inquires a hash value of $w$, the $\mathcal{H}_t$ replies with an element $\mathcal{H}_t(w) = w \in Z_q$, ensuring to reply identically when the same query is occurred. Similar, $\mathcal{H}_2$ replies $\mathcal{F}_3$ with an element $\mathcal{H}_2(u) \in Z_q$.

- **Phase 1 Trapdoor queries:** $\mathcal{F}_3$ inquires a trapdoor $T_w$ corresponding to $w$, then $\mathcal{G}$ asks dTrapdoor oracle and returns $T_w = (T'_w, T''_w) = (M'_s s + x, f_s(E'_u))$ to $\mathcal{F}_3$. Here $s \in Z_q$ and $x \in Z_q$ are two uniform random vectors.

- **Challenge:** $\mathcal{F}_3$ chooses $w_0, w_1$ as his challenge and sends the hash values $w_0, w_1$ to $\mathcal{G}$. Here, none of $w_0$ and $w_1$ has been inquired for obtaining the corresponding trapdoors, $T_{w_0}$ and $T_{w_1}$, and that $\mathcal{F}_3$ did not previously inquired the trapdoors, $T_{w_0}$ and $T_{w_1}$, in Phase 1. $\mathcal{G}$ computes the trapdoor as follows with the challenge $(A_{s,x}, u)$. 

1 If $G$ chooses $A_{z^T}$, then the trapdoors of $w(b \in \{0,1\})$ are $T_w = (T^1_w, T^2_w) = (M_s s_w + A_{z^T}, f_{A_{z^T}}(E_{w})).$

2 If $G$ chooses $u$, then the trapdoors of $w(b \in \{0,1\})$ are $T_w = (T^1_w, T^2_w) = (M_s s_w + u, f_{A_{z^T}}(E_{w})).$

The columns of $E_w$ are sampling from $D_{k^T}$ (using $SampleD$ algorithm), where $r > \eta(g(L^1(M_s))).$

Then $G$ chooses a random $b \in \{0,1\}$ and sends $T_w$ to $F_s$.

- **Phase 2 Trapdoor queries:** $F_s$ can still inquire the trapdoor of $w$, and $G$ can adaptively reply as phase 1, as long as $w \neq w_0, w_1$.

- **Guess:** $F_s$ outputs guesses $b'$. If $b' = b$, $G$ outputs 1; otherwise, it outputs 0.

**Probability Analysis:** If $u$ is selected, the simulation is perfect, namely the distribution of $T_w$ is identical to that in the actual system. $F_s$ will guess the bits $b$ correctly with probability $\frac{1}{2} + \epsilon$. Otherwise, the LWE instance $A_{z^T}$ is independent from $F_s$’s view.

$F_s$ will guess the bits $b$ correctly with probability $\frac{1}{2}$. As a result, $G$ can distinguish between $A_{z^T}$ and $u$ with identical advantage probability $\epsilon$. This completes the proof of trapdoor indistinguishability.

### 4.2.3 Trapdoor anonymity

In this section, we show that our scheme satisfies the trapdoor anonymity for server.

Let $T_w = (T^1_w, T^2_w) = (M_s s_w + x, f_{A_{z^T}}(E_{w})).$ In our scheme, the server can obtain $E_w$ from $T_w$, where $M_s E_w = U_w$. Therefore the server is unable to identify the receiver’s identity from $E_w$, if the receivers do not show. Obviously, the server can compute $U_w$ with all of keywords. However, if there exist a lot of users in the system, the server has to compute all users of dPEKS ciphertext to test.

With the small keyword space, the inside KGAs are considered inevitable under the original framework. Very recently, in Chen et al. (2015), this issue is discussed in a new framework.

### 5 Conclusion

In this paper, we proposed a lattice-based searchable public-key encryption scheme for a designated tester. Our scheme is the first searchable public-key encryption scheme based on lattice hardness assumption (the LWE assumption). Moreover, our scheme achieves the dPEKS ciphertexts indistinguishability and trapdoor indistinguishability. Therefore, excluding inside KGAs, our scheme provides the strongest security level. Meanwhile, our
scheme enjoys provable security under LWE assumption. How to design the lattice-based dPEKS scheme that can stand against inside KGAs is an important work in the future.

Acknowledgements

This work is supported by the Research Foundation of Education Bureau of Sichuan Province (12ZB348), China. The authors would like to thank the reviewers for their detailed reviews and constructive comments, which have helped to improve the quality of this paper.

References


