Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2016, Article ID 6549570, 7 pages
http://dx.doi.org/10.1155/2016/6549570

Hindawi

Research Article

Public Key Encryption with Keyword Search from Lattices in Multiuser Environments

Daini Wu, Xiaoming Wang, and Qingqing Gan

Department of Computer Science, Jinan University, Guangzhou 510632, China

Correspondence should be addressed to Xiaoming Wang; twxm@jnu.edu.cn

Received 30 May 2016; Accepted 17 October 2016

Academic Editor: Nazrul Islam

A public key encryption scheme with keyword search capabilities is proposed using lattices for applications in multiuser environments. The proposed scheme enables a cloud server to check if any given encrypted data contains certain keywords specified by multiple users, but the server would not have knowledge of the keywords specified by the users or the contents of the encrypted data, which provides data privacy as well as privacy for user queries in multiuser environments. It can be proven secure under the standard learning with errors assumption in the random oracle model.

1. Introduction

Cloud storage has established itself as a widely used service for businesses and individuals; however, it comes with inherent challenges including security risks, reliable data storage, and data accessibility. The fact that data owners lose full control over their data brings about concerns with regard to privacy, data integrity, and confidentiality. Data owners have no way to prevent unauthorized individuals or even the party hosting the cloud servers to access or tamper with their data. To ensure data confidentiality and privacy, any sensitive data would need to be encrypted before it is sent to a cloud sever. Unfortunately, data encryption greatly restricts the ability of cloud servers to handle user access requests. A typical example is that of an encrypted document; the encryption changes the contents of the document, making it very hard to search/sort documents using keywords.

To resolve this problem, the notion of the public key encryption with keyword search (PEKS) was proposed by Boneh et al. [1]. The construction of this scheme was inspired by identity-based encryption (IBE) and aims to allow a user to search for encrypted keywords without decryption. In the PEKS scheme, a sender uploads an encrypted email to a server along with an encrypted list of keywords. The receiver sends the desired keyword (denoted as a trapdoor) to the email server, which then tests the encrypted emails for the presence of this trapdoor. Soon afterwards, Waters et al. [2] demonstrated that a PEKS scheme can be used in a wide range of practical applications such as building encrypted and searchable audit logs. Subsequently, many intuitions have been proposed to improve upon this construction (e.g., [3–8]).

The security of most PEKS schemes and their variants are based on the hard problems of number theory such as large integer factorization, the discrete logarithm problem, and bilinear mapping with the Diffie-Hellman problems. However, Shor [9] has discovered a function which can effectively solve the discrete logarithm problem, as well as an algorithm for quantum factoring. This means that quantum computers are a threat to the security of these cryptosystems.

One possible solution that has undergone rapid development in recent years is lattice-based cryptographic schemes. So far, there has been no quantum algorithm which can effectively solve this class of problems, which makes them an attractive candidate for robust encryption. Ajtai [10] first presented the proof of the hardness of lattice problems, and following this work, many new constructions with lattices have been proposed (e.g., [11–14]). However, there have only been two PEKS schemes from lattices proposed in the literature so far, and these schemes were constructed in a single-user environment. Notably, there is currently no similar work available in the literature for PEKS scheme, which means that there is no a PEKS system constructed using lattices in multiuser environment. It is very clear that

the schemes proposed for use in a single-user environment cannot be directly and effectively used in multiuser environments because of the increased requirements of the latter. In PEKS schemes constructed in a single-user environment, data owner can only share his data with a single user and also only permit a single user to search for the encrypted keywords on the encrypted data. While in multiuser environments, such as cloud storage, data owners hope to share their data with multiple users and also permit the multiple users to search over the owner's shared data on the cloud server side. Therefore, PEKS schemes constructed in a single-user environment can not satisfy the characteristics of cloud environment.

In order to solve this problem, a public key encryption scheme with keyword search is proposed using lattices for applications in multiuser environments, which is called PEKSL scheme. In the proposed scheme, cloud server can translate a keyword encrypted under one public key into the same keyword encrypted under another public key, so that it checks if any given encrypted data contains certain keywords specified by multiple users but learns nothing else about the data. This provides data privacy as well as privacy for user queries in multiuser environments. Furthermore, it can be proven secure under the hardness of the standard learning with errors problem in the random oracle model.

1.1. Related Work. To enable users search over encrypted outsourced data through keywords without decrypting the data at first, the notion of public key encryption with keyword search (PEKS) was first put forth by Boneh et al. [1] and its construction makes use of the construction of identity-based encryption (IBE). This construction was later improved by several schemes using the bilinear pairing. Waters et al. [2] demonstrated that these PEKS schemes could be useful to build encrypted and searchable audit logs. Abdalla et al. [4] presented an improved universal transformation from anonymous IBE to PEKS, and Baek et al. [6] proposed a PEKS scheme with a designated server to remove a secure channel. Golle et al. [3] defined the first security model for conducting conjunctive keyword searches to achieve a combinable multikeyword search. Subsequently, Boneh and Waters [5] extended the PEKS scheme to support conjunctive, subset, and range comparisons over the supplied keywords. Camenisch et al. [7] proposed oblivious generation of the keyword search trapdoor to maintain the privacy of the keyword against a curious trapdoor generator, and Cao et al. [8] presented ranked searches using multikeywords over encrypted cloud data and established a variety of privacy requirements.

Nowadays, lattice-based cryptographic schemes have a more fast development, and many new constructions with lattices have been proposed. Ajtai [10] first presented the proof of the hardness of lattice problems, and following this work, many new constructions with lattices have been proposed. These include group signature schemes (e.g., [11, 12]), hierarchical identity-based encryption schemes (e.g., [13, 14]), broadcast encryption on lattice schemes (e.g., [15, 16]), attribute-based encryption (e.g., [17]), and fully homomorphic encryption schemes (e.g., [18, 19]). Notably, there were only two public key encryption with keyword search schemes from lattice [20, 21] so far in the literature, and their schemes were constructed in a single-user environment.

1.2. Paper Organization. The following sections are described briefly as follows. We review the basic concepts about integer lattices, discrete Gaussians, learning with errors problem, the formal models, and a security model of PEKSL in Section 2. We give the specific construction about the PEKSL in a multiuser environment and then prove our scheme under the security model in Section 3. Finally, we conclude this paper in Conclusion.

2. Preliminaries

We describe some preliminaries and other useful concepts that are used in our approach in this section.

2.1. Integer Lattices

Definition 1 (see [20]). Given n linearly independent vectors $b_1, b_2, \ldots, b_n \in \mathbb{R}^m$, the lattice Λ generated by them is denoted $L(b_1, b_2, \ldots, b_n)$ and define

$$L(b_1, b_2, \ldots, b_n) = \left\{ \sum_{i=1}^{n} x_i b_i \mid x_i \in \mathbb{Z} \right\}. \tag{1}$$

The vectors $(b_1, b_2, \ldots, b_n)$ are called the basis of lattice.

For prime q and $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and $u \in Z_q^n$, define

$$\begin{aligned} \Lambda_q^{\perp}(\mathbf{A}) &= \{e \in \mathbb{Z}^m \mid \mathbf{A} \cdot e = 0 \bmod q\}, \\ \Lambda_q^{u}(\mathbf{A}) &= \{e \in \mathbb{Z}^m \mid \mathbf{A} \cdot e = u \bmod q\}. \end{aligned} \tag{2}$$

2.2. The Gram-Schmidt Norm of Basis. Let $\widetilde{S} = \{\widetilde{s}_1, \widetilde{s}_2, \ldots, \widetilde{s}_k\} \subset \mathbb{R}^m$ denote the Gram-Schmidt orthogonalization of the set of linearly independent vectors $S = \{s_1, s_2, \ldots, s_k\}$ in $\mathbb{R}^m$. It is defined as follows: $\widetilde{s}_1 = s_1$ and $\widetilde{s}_1$ is the component of s_i orthogonal to span $(s_1, s_2, \ldots, s_i)$, where $2 \le i \le k$. We refer to $\|\widetilde{S}\|$ as the Gram-Schmidt norm of S.

Lemma 2 (see [14]). *Let Λ be an m-dimensional lattices. There is a deterministic polynomial-algorithm that, given an arbitrary basis of Λ and a full-rank set $S = \{s_1, s_2, \ldots, s_m\}$ in Λ, returns a basis $\mathbf{T}$ of Λ satisfying*

$$\begin{aligned} \|\mathbf{T}\| &\le \left\|\widetilde{S}\right\|, \\ \|\mathbf{T}\| &\le \|S\| \frac{\sqrt{m}}{2}. \end{aligned} \tag{3}$$

Theorem 3 (see [22]). *Let n and m be positive integers and let q be a prime, with $q \ge 3$ and $m = \lceil 6n \log_2 q \rceil$; there is a probabilistic polynomial time algorithm $TrapGen(q, n)$ that outputs a pair $(\mathbf{A} \in \mathbb{Z}_q^{n\times m}, \mathbf{T}_A \in \mathbb{Z}_q^{m\times m})$ such that $\mathbf{A}$ is*

statistically close to a uniform matrix in $\mathbb{Z}_q^{n\times m}$ *and* $\mathbf{T}_A$ *is a basis for* $\Lambda_q^{\perp}(\mathbf{A})$ *satisfying*

$$\begin{aligned}\left\|\widetilde{\mathbf{T}}_A\right\| &\le O\left(\sqrt{n\log_2 q}\right),\\ \left\|\mathbf{T}_A\right\| &\le O\left(n\log_2 q\right)\end{aligned} \tag{4}$$

with all but negligible probability in n.

2.3. Discrete Gaussians

Definition 4. Choose a subset $L \subseteq \mathbb{Z}^m$. For any positive parameter $\sigma \in \mathbb{R}_{>0}$ and any vector $c \in \mathbb{R}^m$ define

$$\begin{aligned}\rho_{\sigma,c}(x) &= \exp\left(-\pi\frac{\|x-c\|^2}{\sigma^2}\right),\\ \rho_{\sigma,c}(L) &= \sum_{x\in L}\rho_{\sigma,c}(x).\end{aligned} \tag{5}$$

The discrete Gaussians distribution over L with parameter σ and center c is

$$\mathcal{D}_{L,\sigma,c}(y) = \frac{\rho_{\sigma,c}(y)}{\rho_{\sigma,c}(L)}, \quad \forall y \in L. \tag{6}$$

Lemma 5 (see [23]). *Let* $q \ge 2$ *and* $\mathbf{A}$ *be a matrix in* $\mathbb{Z}_q^{n\times m}$ *with* $m > n$. *Let* $\mathbf{T}_A$ *be a basis for* $\Lambda_q^{\perp}(\mathbf{A})$ *and* $\sigma \ge \|\widetilde{T}_A\|\omega\sqrt{\log m}$. *Then for* $c \in \mathbb{R}^m$ *and* $u \in \mathbb{Z}_q^n$,

(i) *there is a PPT algorithm SampleGaussian*$(\mathbf{A}, \mathbf{T}_A, \sigma, c)$ *that returns* $x \in \Lambda_q^{\perp}(\mathbf{A})$ *drawn from a distribution statically close to* $\mathcal{D}_{\Lambda,\sigma,c}$,

(ii) *there is a PPT algorithm SamplePre*$(\mathbf{A}, \mathbf{T}_A, \sigma, c)$ *that returns* $x \in \Lambda_q^u(\mathbf{A})$ *drawn from a distribution statically close to* $\mathcal{D}_{\Lambda,\sigma,c}$, *whenever* $\Lambda_q^u(\mathbf{A})$ *is not empty.*

Theorem 6 (see [14]). *Define* $\sigma_R = \sqrt{n\log q}\omega\sqrt{\log m}$. $\mathcal{D}_{m\times m}$ *stands for the distribution on matrices in* $\mathbb{Z}^{m\times m}$ *defined as* $(\mathcal{D}_{\mathbb{Z}^m,\sigma_R})^m$ *conditioned on the resulting matrix being* $\mathbb{Z}_q$*-invertible. There are some important algorithms:*

(i) *Algorithm SampleR*(1^m). *The following simple algorithm samples matrices in* $\mathbb{Z}^{m\times m}$ *from a distribution that is statistically close to* $\mathcal{D}_{m\times m}$.

(a) *Let* $\mathbf{T}$ *be the canonical basis of the lattice* $\mathbb{Z}^m$.

(b) *For* $i = 1, 2, \ldots, m$, *do* $r_i \xleftarrow{R}$ *SampleGaussian*$(\mathbb{Z}^m, \mathbf{T}, \sigma_R, 0)$.

(c) *If* $\mathbf{R}$ *is* $\mathbb{Z}_q$*-invertible, output* $\mathbf{R}$; *otherwise repeat step 2.*

(ii) *Algorithm BasisDel*$(\mathbf{A}, \mathbf{R}, \mathbf{T}_A, \sigma)$. *Input a rank n matrix* $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, *a* $\mathbb{Z}_q$*-invertible matrix* $\mathbf{R}$ *in* $\mathbb{Z}_q^{n\times m}$ *sampled from* $\mathcal{D}_{m\times m}$ *(or a product of such), a basis* $\mathbf{T}_A$ *for* $\Lambda_q^{\perp}(\mathbf{A})$, *and a parameter* $\sigma > \|\widetilde{T}_A\|\sigma_R\sqrt{m}\omega(\log^{2/3} m)$.

(a) *Let* $\mathbf{T}_A = \{a_1, a_2, \ldots, a_m\} \subseteq \mathbb{Z}^{m\times m}$ *and calculate* $\mathbf{T}'_B = \{\mathbf{R}a_1, \mathbf{R}a_2, \ldots, \mathbf{R}a_m\} \subseteq \mathbb{Z}^{m\times m}$. *Observe that* $\mathbf{T}'_B$ *is a set of independent vectors in* $\Lambda_q^{\perp}(\mathbf{B})$, *where* $\mathbf{B} = \mathbf{A}\mathbf{R}^{-1}$ *in* $\mathbb{Z}_q^{n\times m}$.

(b) *Convert* $\mathbf{T}'_B$ *into a basis* $\mathbf{T}''_B$ *of* $\Lambda_q^{\perp}(\mathbf{B})$ *whose Gram-Schmidt norm is no more than that of* $\mathbf{T}'_B$.

(c) *Call RandBasis*$(\mathbf{T}''_B, \sigma)$ *and output the resulting basis* $\mathbf{T}_B$ *of* $\Lambda_q^{\perp}(\mathbf{B})$.

2.4. LWE Problem

Definition 7 (see [20]). Consider a prime q, a positive integer n, and a distribution χ over $\mathbb{Z}_q$, all public. $(\mathbb{Z}_q, n, \chi)$-LWE problem instance consists of access to an unspecified challenge oracle Ψ, being either a noisy pseudorandom sample Ψ_s carrying some constant random secret key $s \in \mathbb{Z}_q^n$ or a truly random sampler Ψ_ϕ, whose behaviors are respectively as follows:

(i) Ψ_s: output noisy pseudorandom sample of the form $(u_i, v_i) = (u_i, u_i^T \cdot s + x_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, where $s \in \mathbb{Z}_q^n$ is an uniformly distributed persistent value invariant across invocations, $x_i \in \mathbb{Z}_q$ is a fresh randomness from χ, and u_i is uniform in $\mathbb{Z}_q^n$.

(ii) Ψ_ϕ: output truly uniform random sample from $\mathbb{Z}_q^n \times \mathbb{Z}_q$.

The $(\mathbb{Z}_q, n, \chi)$-LWE problem allows repeat queries to the challenge oracle Ψ. We say that an algorithm A decides the $(\mathbb{Z}_q, n, \chi)$-LWE problem if $|\Pr[A^{\Psi_s} = 1] - \Pr[A^{\Psi_\phi} = 1]|$ is nonnegligible for a random $s \in \mathbb{Z}_q^n$.

3. PEKSL Scheme in a Multiuser Environment

3.1. Intuition behind the Construction. In PEKS scheme, a sender (Bob) sends an encrypted email to the email gateway using a receiver's (Alice's) public key appended with some encrypted keywords. When Alice gives the email gateway a trapdoor associated with searching keywords, it tests if the keywords are relevant to the email and learns nothing else. Since the ciphertext encrypted with receiver's public key can only be tested by a trapdoor given by Alice and cannot be tested by a trapdoor obtained from another user, such as Charles, PEKS scheme is constrained to be used in a single-user environment where the receiver can only be a single user. In order to meet the need that multiple users can execute encrypted keyword searches over the encrypted files, we add a *Re-KeyGen* algorithm in the definition of PEKS and construct PEKSL scheme, which is inspired by the concept of proxy reencryption. The *Re-KeyGen* algorithm is used to convert a keyword encrypted by a public key into the same keyword encrypted under another public key. More precisely, given a special information (i.e., a reencryption key), the *Re-KeyGen* algorithm enables cloud server to convert the encrypted keyword with Alice's public key in the ciphertext of the same keyword for Charles so that cloud server can check if the encrypted keywords are relevant to an email upon

a trapdoor generated by Charles. Based on the above idea, we construct our PEKSL scheme using lattices for a multiuser environment, where multiple users can execute encrypted keyword searches over the data uploaded by data owner to the cloud server.

3.2. Definition. A PEKSL scheme includes a trust center (TC), data owners, cloud servers, and data users. TC is trusted and is responsible for generating system parameters. The data owners refer to a special type of users who create the private/confidential data and then outsource them to cloud servers in an encrypted form so it can be shared with authorized users. The cloud servers are responsible for producing query results on the encrypted data and then sending these results to the users. Users generally refer to those who are authorized to search for encrypted keywords in the encrypted data. In this paper we consider the users to be from the public domain.

A PEKSL scheme for a multiuser environment involves the following six algorithms:

(i) $Setup(1^{\lambda})$. After input a secure parameter λ, the *Setup* algorithm outputs a global public/private key pair (mpk, msk) hold by TC.

(ii) $KeyGen(msk, mpk, i)$. The *KeyGen* algorithm takes as input (mpk, msk) and a user's identity label i and then generates a public key pk_i, a private key sk_i for user i, and a secret key k_i for TC.

(iii) $Re\text{-}KeyGen(k_i, k_j)$. After input a key pair (k_i, k_j), the *Re-KeyGen* algorithm produces a reencryption key $RK_{i\leftrightarrow j}$.

(iv) $PEKSL(w, pk_i)$. The *PEKS* algorithm produces a searchable encryption CT of keyword w with user's public key pk_i.

(v) $Trapdoor(w, sk_j, pk_j)$. The trapdoor algorithm generates a trapdoor T_w for keyword w with a public/private key pair (pk_j, sk_j).

(vi) $Test(CT, T_w, RK_{i\leftrightarrow j})$. The Test algorithm verifies whether a ciphertext matches a trapdoor.

3.3. Security Game. The security of PEKSL scheme for a multiuser environment is indistinguishable against chosen keyword attack. According to the rules of the security game, an adversary can get the most of trapdoors except those which are relevant to two specified keywords. Yet, it can not distinguish which keyword appended with to a given PEKS ciphertext.

In the following game, a PEKSL scheme for a multiuser environment is indistinguishable against chosen keyword attack if an adversary **F** has a negligible advantage against a challenger **C** in polynomial time.

Security Game

(i) *Setup*. **C** calls *Setup* algorithm to produce a key pair (mpk, msk) and sends mpk to **F**.

(ii) *Phase 1*. **F** makes the following queries:

(a) Uncorrupted key generation oracle: return a fresh key pair (pk_i, sk_i); give pk_i to **F**.

(b) Corrupted key generation oracle: return a fresh key pair (pk_i, sk_i); **F** is given (pk_i, sk_i).

(c) Trapdoor Oracle: after input (pk_i, w) by **F**, return the trapdoor $T_w = Trapdoor(sk_i, pk_i, w)$, where sk_i is the secret key that corresponds to pk_i.

(d) Reencryption key generation oracle: return **F** a reencryption key $RK_{i\leftrightarrow j}$. Noting that correspondent pk_i and pk_j are from uncorrupted key generation oracle or corrupted key generation oracle as in [24], that is, the reencryption key queries are not allowed between uncorrupted oracle and corrupted oracle.

(iii) *Challenge*. **F** gives **C** two keywords w_0, w_1 that it hopes to be challenged on. The only limitation is that **F** have not queried for the trapdoors T_{w_0} or T_{w_1}. **C** chooses a random $b \in \{0, 1\}$ and sends **F** a PEKSL ciphertext $CT^* = PEKS(w, pk_i)$ as the challenge ciphertext.

(iv) *Phase 2*. **F** adaptively queries **C** for the trapdoor T_w for any keyword w he wants again, and it is important to note that the w must satisfy $w \neq w_0, w_1$.

(v) *Guess*. Finally, **F** gives $b' \in \{0, 1\}$. If $b' = b$, **F** wins the game.

In this game, we define the advantage of **F** that is $|\Pr[b' = b] - 1/2|$.

3.4. Construction. Choose two positive integers n, m, a prime q, satisfying $m > 6n \log q$ and $q \geq 3$, and a hash function $H : \{0, 1\}^* \to \mathbb{Z}_q^n$. Construction of PEKSL scheme in a multiuser environment is described as follows:

(i) $Setup(1^n)$. Take a secure parameter n as input, TC invokes $TrapGen(q, n)$ to produce a randomly uniform matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ along with a short basis $\mathbf{T}_A \in \mathbb{Z}_q^{m\times m}$ for $\Lambda_q^{\perp}(\mathbf{A})$, and output the global public key $mpk = \mathbf{A}$ as well as global private key $msk = \mathbf{T}_A$.

(ii) $KeyGen(msk, mpk, i)$. After input (mpk, msk) and a user identity label i, TC invokes firstly *Algorithm SampleR*(1^m) to generate a $\mathbb{Z}_q$-invertible matrix $\mathbf{R}_i$ in $\mathbb{Z}_q^{m\times m}$ sampled from $\mathcal{D}_{m\times m}$, where m is a security parameter. Then TC invokes *Algorithm BasisDel*$(\mathbf{A}, \mathbf{R}_i, \mathbf{T}_A, \sigma)$ to generate a basis $\mathbf{T}_i$ of $\Lambda_q^{\perp}(\mathbf{A}\mathbf{R}_i^{-1})$, and output the user's public key $pk_i = \mathbf{A}\mathbf{R}_i^{-1}$ in $\mathbb{Z}_q^{n\times m}$ and private key $sk_i = \mathbf{T}_i$ and a secret key $k_i = \mathbf{R}_i$. TC sends sk_i to the user i by a security way and saves a list of (pk_i, k_i).

(iii) $Re\text{-}KeyGen(k_i, k_j)$. TC gets $(k_i = \mathbf{R}_i, k_j = \mathbf{R}_j)$ from the list of (pk_i, k_i) and computes $RK_{i\leftrightarrow j} = \mathbf{R}_i\mathbf{R}_j^{-1}$. TC sends $RK_{i\leftrightarrow j}$ to the cloud server by a security way.

(iv) $PEKS(w, pk_i)$. The data owner computes $u = H(w)$ and chooses $r \xleftarrow{R} \mathbb{Z}_q^n$ uniformly and computes

$c_1 = r^T pk_i$ and $c_2 = u^T r + x$, where $x \xleftarrow{R} \chi$ is a noise vector. Output the ciphertext $CT = (c_1, c_2)$.

(v) $Trapdoor(w, sk_j, pk_j)$. A user computes $u = H(w)$ and invokes *Algorithm SamplePre*(pk_j, sk_j, u, σ) to generate $z \in \mathbb{Z}_q^m$, where σ is a Gaussian parameter. Note that $pk_j z = u$ in $\mathbb{Z}_q^n$. Output z as a trapdoor; that is $T_w = z$.

(vi) $Test(CT, T_w, RK_{i\leftrightarrow j})$. The cloud server computes

$$\begin{aligned}\widehat{c_1} &= c_1 \times RK_{i\leftrightarrow j} = r^T \times pk_i \times RK_{i\leftrightarrow j} \\ &= r^T \times \mathbf{A}\mathbf{R}_i \times \mathbf{R}_i\mathbf{R}_j^{-1} = r^T \mathbf{A}\mathbf{R}_j^{-1} = r^T pk_j\end{aligned} \tag{7}$$

then let $b = c_2 - z^T \widehat{c_1}^T$. If $|b| < q/4$, return 1; otherwise, return 0.

3.5. Security Analysis. We show the security analysis of PEKSL scheme for a multiuser environment in this section. In the random oracle model, supposing that there is an adversary **F** that has a nonnegligible advantage to attack our mechanism in polynomial time, we are able to construct a polynomial time algorithm **S** having a probability $\varepsilon/2eQ_H$ to solve the LWE assumption, where Q_H denotes the query times of which **F** queries the random oracle $H(\cdot)$.

Proof. With the adversary **F**, we construct the algorithm **S** as follows.

S requests from Ψ (Ψ is a pseudorandom LWE oracle) and receives a fresh pair $(u_i, v_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ for each $i = 0, 1, 2, \ldots, m$ and selects a random integer $Q^* \in [1, Q_H]$.

(i) *Setup*. Given LWE samples, **S** assembles the random matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ from m, more exactly, making the ith column of $\mathbf{A}$ be the vector u_i ($i = 1, 2, 3, \ldots, m$), and gives the public key $\mathbf{A}$ to **F**.

(ii) *Phase 1*. **S** answers queries of **F** as follows:

(a) Uncorrupted key generation oracle: after input an index i, if this i is uncorrupted, **S** runs *Algorithm SampleR*(1^m) to obtain a $\mathbb{Z}_q$-invertible matrix $\mathbf{R}_i$ in $\mathbb{Z}_q^{n\times m}$ and computes $pk_i = \mathbf{A}\mathbf{R}_i$, $corrupted_i = 0$. Finally **S** saves the tuple $(i, pk_i, \mathbf{R}_i, corrupted_i = 0)$ in PK-List. **S** sends pk_i to **F**.

(b) Corrupted key generation oracle: after input an index i, if this i is corrupted, **S** runs firstly *Algorithm SampleR*(1^m) to obtain a $\mathbb{Z}_q$-invertible matrix $\mathbf{R}_i \in \mathbb{Z}_q^{n\times m}$ and then *Algorithm BasisDel*$(\mathbf{A}, \mathbf{R}_i, \mathbf{T}_A, \sigma)$ to get a basis $\mathbf{T}_i$ for pk_i. Finally **S** responds to **F** with $(pk_i, sk_i = \mathbf{T}_i)$ and records the tuple $(i, pk_i, \mathbf{R}_i, sk_i, corrupted = 1)$ in PK-List.

(c) H Oracle: to answer H queries, **S** saves the tuples (pk_i, w_i, u_i, z_i) in a H-list, which is initially empty. For the Qth query, **F** sends $w_i \in \{0, 1\}^*$ to H and **S** answers as follows:

(1) If this query is the query number Q^* (i.e., $Q = Q^*$), **S** defines $H(w_i) = u_0$, $pk^* = pk_i$ satisfying $corrupted^* = 0$, and returns $H(w_i)$.

(2) Otherwise, if this query (pk_i, w_i) has saved in the H-list, then **S** responds to $H(w_i) = u_i$. If the query (pk_i, w_i) does not appears on the H-list and the corresponding $corrupted_i = 0$ in PK-List, **S** chooses $z_i \xleftarrow{R} \mathscr{D}_{\mathbb{Z}^m, r}$ and computes $u_i = pk_i z_i$. Finally **S** saves the quadruple (pk_i, w_i, u_i, z_i) in H-list for future use and returns $H(w_i) = u_i$ to **F**.

(d) Trapdoor Oracle: for input $w_i \in \{0, 1\}^*$ and pk_i, if $H(w_i) = u_0$, **S** aborts and fails. Otherwise, **S** retrieves the saved quadruple (pk_i, w_i, u_i, z_i) from the H-list. Here we assume that a trapdoor query on w_i is preceded by a $H(\cdot)$ query with w_i. By construction, z_i is the trapdoor for the keyword w_i. **S** return z_i as the trapdoor to **F**.

(e) Reencryption key generation oracle: after input (pk_i, pk_j), **S** checks whether both pk_i and pk_j occur in PK-List, if not, **S** aborts. Otherwise, **S** does the following operations:

(1) If $corrupted_i = corrupted_j$, **S** responds to **F** with $\mathbf{R}_i\mathbf{R}_j^{-1}$.

(2) If $corrupted_i \neq corrupted_j$, **S** aborts.

(iii) *Challenge*. **F** sends two keywords w_0, w_1 with a pk it wants to challenge on, when it decides to finish Phase 1. **S** calls the algorithm mentioned above to answer $H(\cdot)$ queries twice with input $(pk, w_0), (pk, w_1)$. If $pk \neq pk^*$, then **S** aborts; if both $H(w_0) \neq u_0$ and $H(w_1) \neq u_0$, then **S** aborts either. Otherwise, supposing $H(w_b) = u_0$, **S** retrieves $(v_0, v_1, v_2, \ldots, v_m) \in \mathbb{Z}_q$ from the LWE instance and sets $c_1 = [v_1, v_2, \ldots, v_m]^T \in \mathbb{Z}_q^m$ and $c_2 = v_0 \in \mathbb{Z}_q$. **S** responds with challenge ciphertext (c_1, c_2). Note that when Ψ is a pseudorandom LWE oracle and for the random $r \xleftarrow{R} \mathbb{Z}_q^n$ and the noise value $x \in \chi^m$, then $c_1 = r^T pk_i$ and $c_2 = u_0^T r + x$ is valid encryption for w_b. While Ψ is a random oracle, (c_1, c_2) is uniformly sampled from $\mathbb{Z}_q^m \times \mathbb{Z}_q$.

(iv) *Phase 2*. **S** answers queries of **F** the same way it does in Phase 1 if $pk \neq pk^*$, $w \neq w_0 \neq w_1$, and the pk is from uncorrupted key generation oracle.

(v) *Guess*. At last **F** guesses $b' \in \{0, 1\}$. **S** returns 1 if $b = b'$; otherwise **S** returns 0.

if **S** does not halt, the distribution of both the challenge ciphertext and the public parameters is identical to its distribution in the real system. If **S** aborts the challenge ciphertext will be independently randomly sampled from $\mathbb{Z}_q^m \times \mathbb{Z}_q$. That is, if **S** does not abort, the advantage of **S** in solving LWE is identical to **F**'s advantage to attack this mechanism. □

Probability Analysis. We now analyze the situations that **S** did not come to a halt in the simulation. There are two independent situations:

(i) ε_1: **S** did not report failure during any **F**'s trapdoor queries.

(ii) ε_2: **S** did not report failure during any **F**'s challenge queries.

The trapdoor of the same keyword is supposed not to be queried twice by **F**. And because Q^* is chosen randomly in 1 and Q_H, **S** does not abort in the simulation of $Trapdoor(\cdot)$ with probability $(1 - 1/Q_H)^{Q_H}$; that is, $\Pr[\varepsilon_1] = 1 - 1/(Q_H + 1)^{Q_H} \geq 1/e$.

The only situation where **S** reports failure during challenge phase is that $H(w_0) \neq u_0$ and $H(w_1) \neq u_0$. The probability that $H(w_0) = u_0$ or $H(w_1) = u_0$ is $1-(1-1/(Q_H+1))^2$. Hence, we have $\Pr[\varepsilon_2] = 1 - (1 - 1/(Q_H + 1))^2 \geq 1/Q_H$.

So we have the probability of **S** in solving LWE which is at least $1/2 + \varepsilon/2eQ_H$.

4. Conclusion

A public key encryption with keyword search scheme is first presented using lattices for multiuser environment. Fruitful lattice-based cryptographic schemes have been proposed before, but as far as we know, it is the first lattice-based public key encryption with keyword search scheme targeted towards applications in multiuser environments. Its security can be proven based on the standard LWE assumption in the random oracle model.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This work was granted partially by National Natural Science Foundation of China (61272415 and 61070164), Science and Technology Planning Project of Guangdong Province, China (2013B010401015), and Natural Science Foundation of Guangdong Province, China (S2012010008767).

References

[1] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *International Conference on the Theory and Applications of Cryptographic Techniques EUROCRYPT 2004: Advances in Cryptology—EUROCRYPT 2004*, vol. 3027 of *Lecture Notes in Computer Science*, pp. 506–522, Springer, Berlin, Germany, 2004.

[2] B. R. Waters, D. Balfanz, G. Durfee, and D. K. Smetters, "Building an encrypted and searchable audit log," in *Proceedings of the NDSS Symposium*, vol. 4, pp. 5–6, February 2004.

[3] P. Golle, J. Staddon, and B. Waters, "Secure conjunctive keyword search over encrypted data," in *Applied Cryptography and Network Security: Second International Conference, ACNS 2004, Yellow Mountain, China, June 8–11, 2004. Proceedings*, vol. 3089 of *Lecture Notes in Computer Science*, pp. 31–45, Springer, Berlin, Germany, 2004.

[4] M. Abdalla, M. Bellare, D. Catalano et al., "Searchable encryption revisited: consistency properties, relation to anonymous IBE, and extensions," in *Advances in Cryptology—CRYPTO 2005*, V. Shoup, Ed., vol. 3621 of *Lecture Notes in Computer Science*, pp. 205–222, Springer, Berlin, Germany, 2005.

[5] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in *Theory of Cryptography Conference: TCC 2007: Theory of Cryptography*, vol. 4392 of *Lecture Notes in Computer Science*, pp. 535–554, Springer, Berlin, Germany, 2007.

[6] J. Baek, R. Safavi-Naini, and W. Susilo, "Public key encryption with keyword search revisited," in *Computational Science and Its Applications—ICCSA 2008: International Conference, Perugia, Italy, June 30– July 3, 2008, Proceedings, Part I*, vol. 5072 of *Lecture Notes in Computer Science*, pp. 1249–1259, Springer, Berlin, Germany, 2008.

[7] J. Camenisch, M. Kohlweiss, A. Rial, and C. Sheedy, "Blind and anonymous identity-based encryption and authorised private searches on public key encrypted data," in *Public Key Cryptography-PKC 2009*, pp. 196–214, Springer, Berlin, Germany, 2009.

[8] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 1, pp. 222–233, 2014.

[9] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proceedings of the 35th Annual Symposium on Foundations of Computer Science (SFCS '94)*, pp. 124–134, IEEE, 1994.

[10] M. Ajtai, "Generating hard instances of lattice problems," in *Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC '96)*, pp. 99–108, Philadelphia, Pa, USA, May 1996.

[11] S. D. Gordon, J. Katz, and V. Vaikuntanathan, "A group signature scheme from lattice assumptions," in *Advances in Cryptology—ASIACRYPT 2010*, vol. 6477, pp. 395–412, Springer, 2010.

[12] A. Langlois, S. Ling, K. Nguyen, and H. Wang, "Lattice-based group signature scheme with verifier-local revocation," in *Public-Key Cryptography—PKC 2014*, H. Krawczyk, Ed., vol. 8383 of *Lecture Notes in Computer Science*, pp. 345–361, Springer, Berlin, Germany, 2014.

[13] S. Agrawal, D. Boneh, and X. Boyen, "Efficient lattice (H)IBE in the standard model," in *Advances in Cryptology—EUROCRYPT 2010*, vol. 6110, pp. 553–572, Springer, 2010.

[14] S. Agrawal, D. Boneh, and X. Boyen, "Lattice basis delegation in fixed dimension and shorter-cipher text hierarchical ibe," in *Annual Cryptology Conference: CRYPTO 2010: Advances in Cryptology—CRYPTO 2010*, vol. 6223 of *Lecture Notes in Computer Science*, pp. 98–115, Springer, Berlin, Germany, 2010.

[15] X. Li, B. Yang, Y. Guo, and W. Sun, "Provably secure group based broadcast encryption on lattice," *Journal of Information and Computational Science*, vol. 8, no. 2, pp. 179–193, 2011.

[16] A. Georgescu, "Anonymous lattice-based broadcast encryption," in *Information and Communication Technology*, pp. 353–362, Springer, Berlin, Germany, 2013.

[17] Y. Wang, "Lattice ciphertext policy attribute-based encryption in the standard model," *International Journal of Network Security*, vol. 16, no. 6, pp. 444–451, 2014.

[18] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the 41th Annual ACM Symposium on Theory of Computing (STOC '09)*, pp. 169–178, ACM, 2009.

[19] C. Gentry, "Toward basing fully homomorphic encryption on worst-case hardness," in *Annual Cryptology Conference: CRYPTO 2010: Advances in Cryptology—CRYPTO 2010*, vol. 6223 of *Lecture Notes in Computer Science*, pp. 116–137, Springer, Berlin, Germany, 2010.

[20] C. Gu, Y. Guang, Y. Zhu, and Y. Zheng, "Public key encryption with keyword search from lattices," *International Journal of Information Technology*, vol. 19, no. 1, 2013.

[21] C. Hou, F. Liu, H. Bai, and L. Ren, "Public-key encryption with keyword search from lattice," in *Proceedings of the 8th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC '13)*, pp. 336–339, IEEE, Compiegne, France, October 2013.

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

[23] C. Gentry, C. Peikert, and V. Vaikuntanathan, "Trapdoors for hard lattices and new cryptographic constructions," in *Proceedings of the 14th Annual ACM Symposium on Theory of Computing (STOC '08)*, pp. 197–206, Victoria, Canada, May 2008.

[24] C.-K. Chu, J. Weng, S. S. M. Chow, J. Zhou, and R. H. Deng, "Conditional proxy broadcast re-encryption," in *Information Security and Privacy*, C. Boyd and J. G. Nieto, Eds., vol. 5594 of *Lecture Notes in Computer Science*, pp. 327–342, Springer, Berlin, Germany, 2009.