Notes on a lattice-based proxy-oriented identity-based encryption with keyword search

Zi-Yuan Liu, Yi-Fan Tseng*, Raylin Tso, and Cheng-Yi Lee

Department of Computer Science, National Chengchi University, Taipei, Taiwan
{zyliu, yftseng, raylin}@cs.nccu.edu.tw, 109753120@g.nccu.edu.tw

Abstract. Zhang et al. recently proposed a lattice-based proxy-oriented identity-based encryption with keyword search (PO-IBEKS) at Information Sciences in 2019. They claimed that their scheme can resist insider keyword guessing attacks by preventing cloud server from generating ciphertext. In this note, we provide a cryptanalysis of their PO-IBEKS and demonstrate that their scheme cannot resist outsider/insider keyword guessing attacks, even though they satisfy unforgeability requirement. Furthermore, we uncover the root cause of the attack and provide a possible solution for Zhang et al.'s scheme to aid future designs of secure PO-IBEKS schemes.

Keywords: Insider Keyword Guessing Attack · Outsider Keyword Guessing Attack · Lattices · Identity-based Encryption · Keyword Search

1 Introduction

Public-key encryption with keyword search (PEKS), which was first proposed by Boneh et al. [3], lets us use ciphertext with more flexibility. In PEKS, a data sender can generate a ciphertext for a specific keyword, while a data receiver can produce a valid trapdoor. Then, a cloud server can perform tests to check whether the ciphertext and the trapdoor are associate with the same keyword. A basic security requirement of PEKS is to ensure that adversaries cannot obtain any information of the keyword from ciphertext (i.e., chosen keyword attacks) and trapdoor (i.e., keyword guessing attacks (KGA)). In the beginning, scholars only considered the adversaries who perform KGA were outsiders (i.e., the adversaries can only eavesdrop the trapdoor from the communicate channel). In 2006, Byun [4] further considered that a malicious insider (e.g., cloud server) might offline guess the keyword from the trapdoor, which was referred to as insider keyword guessing attacks (IKGA). Because the malicious server can adaptively generate ciphertext for any keywords he/she chooses and receive the trapdoor sent from the data receiver, he/she can test the ciphertext and the trapdoor. If the test passes, the malicious insider can obtain the keyword selected by the data receiver. As described by Byun [4], since the keywords are low-entropy and their space is usually small, the probability to obtain the information of keyword by performing IKGA is high.

Recently, Zhang et al. proposed a lattice-based proxy-oriented identity-based encryption with keyword search (PO-IBEKS) [9], to realize the advantages of identity-based cryptosystem (i.e., eliminate the need of public-key infrastructure) and resist IKGA at the same time. Unfortunately, we found that there are some flaws which make their scheme unable to withstand IKGA. More preciously, the cloud server cannot forge any ciphertext since the unforgeability is held, but the adversary can directly obtain the information about keyword from the trapdoor. In the note, we demonstrate the attack steps and discuss the root cause of the attack. Furthermore, we provide a possible solution to this scheme.

The remainder of this note is organized as follows. Section 2 provides some preliminaries. Section 3 introduces the definition and security model of PO-IBEKS. Section 4 presents a summary of PO-IBEKS proposed by Zhang et al. Section 5 demonstrates that Zhang et al.'s scheme is susceptible to IKGA. Section 6 discusses the root cause of the attack and provides a possible solution. Finally, Section 7 concludes this note.

2 Preliminaries

2.1 Notations

Let $\mathbb{Z}$ denotes a set of integer. For prime q, $\mathbb{Z}_q$ denotes a finite field (or Galois field) with order q. For an element e and finite set S, $e \leftarrow S$ indicates that e is selected uniformly at random from S. Moreover, for $a \in \mathbb{R}$, $\lfloor a \rceil$ is rounded down to the closest integer of a. Finally, $\|A\|$ represents the l_2 norm of A.

* Corresponding author.

2.2 Lattice

Here, we briefly summarize the lattice concept.

Given $n, m, q \in \mathbb{Z}$ and $A \in \mathbb{Z}_q^{n\times m}$, we can define two lattices as follows:

- $\Lambda_q(A) = \{y \in \mathbb{Z}_q^m | \exists z \in \mathbb{Z}_q^n, y = A^\top z \bmod q\}$;
- $\Lambda_q^\perp(A) = \{e \in \mathbb{Z}_q^m | Ae = 0 \bmod q\}$.

In addition, let $u \in \mathbb{Z}_q^n$, we can further define a coset $\Lambda_q^u(A) = \{e \in \mathbb{Z}^m | Ae = u \bmod q\}$.

Discrete Gaussian Let σ be a positive real number and $x \in \mathbb{Z}^m$. Here, we define the Gaussian distribution of $\mathcal{D}_\sigma$ with parameter σ by the probability distribution function $\rho_\sigma(x) = \exp(-\pi \cdot \|x\|^2/\sigma^2)$. Furthermore, for any set $\mathcal{L} \subset \mathbb{Z}^m$, we define $\rho_\sigma(\mathcal{L}) = \sum_{x\in\mathcal{L}} \rho_\sigma$. Then, the discrete Gaussian distribution over $\mathcal{L}$ with parameter σ is defined as:

$$\text{For all } x \in \mathcal{L},\ \mathcal{D}_{\mathcal{L},\sigma} = \rho_\sigma(x)/\rho_\sigma(\mathcal{L}).$$

Lattice with Trapdoors Here, we introduce the algorithms related to the lattice trapdoor [1,2,7] used in Zhang et al.'s scheme [9].

1. $\mathsf{TrapGen}(1^n, 1^m, q) \to (A, T_A)$: For input $n, m, q \in \mathbb{Z}$, this algorithm outputs matrix $A \in \mathbb{Z}_q^{n\times m}$ with its corresponding trapdoor $T_A \in \mathbb{Z}_q^{m\times m}$, and the following property holds:
$$\{A : (A, T_A) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)\} \approx \{A : A \leftarrow \mathbb{Z}_q^{n\times m}\}.$$
2. $\mathsf{SamplePre}(A, T_A, \mu, \sigma) \to t$: For an input matrix $A \in \mathbb{Z}_q^{n\times m}$ and its trapdoor $T_A \in \mathbb{Z}_q^{m\times m}$, a vector $\mu \in \mathbb{Z}_q^n$, and parameter $\sigma \in \mathbb{R}$, this algorithm outputs sample $t \in \mathbb{Z}_q^m$ such that $At = \mu$ and t is distributed in $\mathcal{D}_{\mathbb{Z}^m,\sigma}$.
3. $\mathsf{NewBasisDel}(A, R, T_A, \delta) \to T_B$: For an input matrix $A \in \mathbb{Z}_q^{n\times m}$, $\mathbb{Z}_q$-invertible matrix $R \leftarrow \mathcal{D}_{m\times m}$, trapdoor T_A, and parameter $\delta \in \mathbb{R}$, this algorithm outputs random matrix $T_B \in \Lambda_q^\perp(B)$, where $B = AR^{-1}$.

3 Definition and security model of PO-IBEKS

This section provides a summary of PO-IBEKS modeled in [9]. A PO-IBEKS consists of five entities (key generation center (KGC), cloud server, proxy, original data owner, and data receiver) and six polynomial-time algorithms (Setup, KeyExtract, Proxy-oriented key generation, IBEKS, Trapdoor, Test), described as follows:

- $\mathsf{Setup}(\kappa) \to (PP, MSK)$: This probabilistic polynomial time (PPT) algorithm takes a security parameter κ as its input and outputs system public parameters PP, and the master secret key MSK of KGC.
- $\mathsf{KeyExtract}(PP, MSK, id) \to SK_{id}$: This PPT algorithm is performed by KGC that takes the system public parameters PP, the master secret key MSK of KGC, and an identity id as its inputs and outputs a secret key SK_{id} of identity id.
- $\mathsf{Proxy\text{-}oriented\ key\ generation}\langle id_0(PP, SK_{id_0}, W), id_p(PP, SK_{id_p}, sig, W)\rangle \to (PK_{pro}, SK_{pro})$: This PPT algorithm is an interacted algorithm performed between original data owner id_0 and proxy id_p. The original data owner first generates a signature sig of the warrant W by using his/her private key SK_{id_0}, and sends it to the proxy. If the signature passes the validation, then proxy generates a proxy-oriented public/private key pair (PK_{pro}, SK_{pro}) using his/her private key SK_{id_p}, the signature, and the warrant W.
- $\mathsf{IBKES}(PP, w, id_r, PK_{pro}, SK_{pro}) \to C$: This PPT algorithm is performed by the proxy that takes the system public parameters PP, a keyword w, a data receiver's identity id_r, and the proxy-oriented public/private key pair (PK_{pro}, SK_{pro}) as its inputs and outputs a searchable ciphertext C associated with the keyword w.
- $\mathsf{Trapdoor}(PP, SK_{id_r}, w) \to d_w$: This PPT algorithm is performed by the data receiver that takes the system public parameters PP, the private key SK_{id_r} of the data receiver id_r, and a keyword w as its inputs and outputs a trapdoor d_w associated with the keyword w.
- $\mathsf{Test}(PP, A_{pro}, d_w, C) \to 1/0$: This deterministic polynomial-time algorithm is performed by the cloud sever that takes the system public parameters PP, the proxy-oriented public key PK_{pro}, a trapdoor d_w, and a searchable ciphertext C as its inputs and outputs 1 if C and d_w contain the same keyword w, and 0 otherwise.

Definition 1 (Correctness consistency of PO-IBEKS). *A PO-IBEKS meets correctness consistency requirement if for any honestly generated system public parameters PP, master secret key MSK, proxy-oriented public/private key pair (PK_{pro}, SK_{pro}) of the proxy id_p, the private key SK_{id_r} of id_r, and for any keywords w,* $\mathsf{Test}(PP, A_{pro}, d_w, C) = 1$ *holds, where* $C \leftarrow \mathsf{IBEKS}(PP, w, id_r, PK_{pro}, SK_{pro})$ *and* $d_w \leftarrow \mathsf{Trapdoor}(PP, SK_{id_r}, w)$.

In [9], the authors define three threat models: ciphertext indistinguishability, existential unforgeability, and delegation security. They also state that since their PO-IBEKS achieves existential unforgeability, so that a misbehaved cloud server cannot forge a valid searchable ciphertext to perform the IKGA. Therefore, the following we only recall the security model of existential unforgeability. The existential unforgeability is defined by the following interactive game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

- Setup: By inputting a security parameter κ, the challenger $\mathcal{C}$ first generates system public parameters PP and the master secret key MSK of KGC, and then returns PP to $\mathcal{A}$. $\mathcal{A}$ submits a target identity id_p^* to $\mathcal{C}$.
- Query: $\mathcal{A}$ is allowed to adaptively query the following oracles:
  - KeyExtract oracle: Once receiving a query on an identity $id \neq id_p^*$, $\mathcal{C}$ generates corresponding private key SK_{id} and returns it to $\mathcal{A}$.
  - Trapdoor oracle: Once receiving a query on a keyword w, $\mathcal{C}$ generates corresponding trapdoor d_w and returns it to $\mathcal{A}$.
  - Authenticated searchable ciphertext oracle: Once receiving a query on a pair of a keyword w under the identities (id_p, id_r), $\mathcal{C}$ generates corresponding authenticated searchable ciphertext C with the restriction that $id_p \neq id_p^*$, and returns it to $\mathcal{A}$.
- Forgery: Finally, $\mathcal{A}$ outputs a forged authenticated searchable ciphertext of C^* associated with (w^*, id_p^*, id_r). If the ciphertext can pass the test process, we say that $\mathcal{A}$ wins the game.

4 Zhang et al.'s PO-IBEKS

In this section, we revisit the PO-IBEKS proposed by Zhang et al. [9].

- Setup. In this algorithm, given a security parameter κ, the KGC executes the following steps to generate the system public parameters:
  1. chooses a discrete Gaussian distribution χ and security Gaussian parameters σ, δ.
  2. generates a matrix $A \in \mathbb{Z}_q^{n\times m}$ together with the master secret key $MSK = T_A \in \mathbb{Z}_q^{m\times m}$ by using $\mathsf{TrapGen}(q, n)$.
  3. selects a uniform random vector $v \leftarrow \mathbb{Z}_q^n$.
  4. selects five secure cryptographic hash functions: $H_1 : \{0,1\}^{\ell_1} \to \mathbb{Z}_q^{m\times m}$, $H_2 : \{0,1\}^{\ell_1} \times \{0,1\}^{\ell_1} \times \{0,1\}^{\ell_2} \times \mathbb{Z}_q^n \to \mathbb{Z}_q^n$, $H_3 : \{0,1\}^{\ell_1} \times \{0,1\}^{\ell_1} \times \{0,1\}^{\ell_2} \times \mathbb{Z}_q^m \to \mathbb{Z}_q^{m\times m}$, $H_4 : \{0,1\}^{\ell_1} \times \{0,1\}^{\ell_1} \times \{0,1\}^{\ell_3} \to \mathbb{Z}_q^{m\times m}$, and $H_5 : \{0,1\}^{\ell} \times \mathbb{Z}_q^{m\times \ell} \to \mathbb{Z}_q^n$, where the outputs of H_1, H_3, and H_4 are distribution in $\mathcal{D}_{m\times m}$.
  5. outputs the system public parameters $PP = (A, v, H_1, H_2, H_3, H_4, H_5, \sigma, \delta)$, the KGC secretly keeps the master secret key MSK.
- KeyExtract. In this algorithm, given the system public parameters $PP = (A, v, H_1, H_2, H_3, H_4, H_5, \sigma, \delta)$, the master secret key $MSK = T_A$, and an identity $id \in \{0,1\}^{\ell_1}$, the KGC executes the following steps to generate a secret key SK_{id} for id:
  1. computes $R_{id} = H_1(id)$ and $A_{id} = A(R_{id})^{-1} \in \mathbb{Z}_q^{n\times m}$.
  2. generates a random short lattice basis $T_{id} \in \mathbb{Z}_q^{m\times m}$ of $\Lambda_q^{\perp}(A_{id})$ by running $\mathsf{NewBasisDel}(A, R_{id}, T_A, \sigma)$.
  3. sets the secret key $SK_{id} = T_{id}$, and sends it to identity id.
- Proxy-oriented key generation. In this interactive PPT algorithm, an data original data owner id_0 and proxy id_p cooperatively generate the proxy-oriented public/private key pair as follows:
  1. id_0 first generates a warrant $W \in \{0,1\}^{\ell}$ according to its requirements. Here, the explicit description of the relative rights and information of an original data owner and a proxy are included in the warrant W. Additionally, the warrant W also includes the information of the intended data receiver.
  2. id_0 then picks a uniform random vector $r \leftarrow \mathbb{Z}_q^n$, and computes $\mu = H_2(id_0 \| id_p \| W \| r)$. id_0 also runs $\beta_W \leftarrow \mathsf{SamplePre}(A_{id_0}, T_{id_0}, \mu, \delta) \in \mathbb{Z}_q^m$. Finally, id_0 sends (W, r, β_W) directly to id_p.
  3. After receiving (W, r, β_W) from id_0, id_p first verifies W by computing $A_{id_0}\beta_W = H_2(id_0 \| id_p \| W \| r)$, where β_W is distributed in $\mathcal{D}_{\Lambda_q^{\mu}(A_{id_0}),\delta}$. If the equation is not satisfied, id_p rejects it. Otherwise, id_p computes $R_W = H_3(id_0 \| id_p \| W \| \beta_W)$ and $T_{pro} \leftarrow \mathsf{NewBasisDel}(A_{id_p}, R_W, T_{id_p}, \sigma)$, and set (A_{pro}, T_{pro}) as the proxy-oriented public/private key pair, where $A_{pro} = A_{id_p}(R_W)^{-1} \in \mathbb{Z}_q^{n\times m}$.
- IBEKS. In this algorithm, given the system public parameters $PP = (A, v, H_1, H_2, H_3, H_4, H_5, \sigma, \delta)$, a keyword $w \in \{0,1\}^{\ell_3}$, a data receiver's identity id_r, the proxy-oriented public/private key pair (A_{pro}, T_{pro}), id_p executes the following steps to generate a ciphertext C:
  1. randomly chooses a uniform matrix $F \in \mathbb{Z}_q^{n\times \ell}$, and a random binary string $\tau = (\tau_1, \tau_2, \cdots, \tau_\ell) \in \{0,1\}^{\ell}$.
  2. samples a random noise vector $\eta = (\eta_1, \eta_2, \cdots, \eta_\ell) \leftarrow \chi$.
  3. samples ℓ random noise vectors $s_1, s_2, \cdots, s_\ell \leftarrow \chi^m$, and sets the noise matrix $S = (s_1, s_2, \cdots, s_\ell) \in \mathbb{Z}_q^{m\times \ell}$.
  4. computes $\gamma = H_4(id_p \| id_r \| w) \in \mathbb{Z}_q^{m\times m}$
  5. computes $\xi = (A_{id_r}\gamma^{-1})^{\top} F + S$ and $\zeta = v^{\top} F + \eta + (\tau_1, \tau_2, \cdots, \tau_\ell)\lfloor q/2 \rfloor$.

   6. computes $h = H_5(\tau \| \xi)$, and $\theta \leftarrow \mathsf{SamplePre}(A_{pro}, T_{pro}, h, \delta) \in \mathbb{Z}_q^m$.
   7. outputs a searchable ciphertext $C = (\xi, \zeta, \theta)$ to the cloud server.
- Trapdoor. In this algorithm, given the system public parameters $PP = (A, v, H_1, H_2, H_3, H_4, H_5, \sigma, \delta)$, a private key $SK_{id_r} = T_{id_r}$ of the data receiver id_r, and a keyword $w \in \{0,1\}^{\ell_3}$, id_r executes the following steps to generate a trapdoor d_w:
   1. computes $\gamma = H_4(id_p \| id_r \| w) \in \mathbb{Z}_q^{m\times m}$, and $D_d \leftarrow \mathsf{NewBasisDel}(A_{id_r}, \gamma, T_{id_r}, \sigma) \in \mathbb{Z}_q^{m\times m}$, where D_w is a short lattice basis of $\Lambda_q^{\perp}(A_{id_r}\gamma^{-1})$.
   2. generates $d_w \leftarrow \mathsf{SamplePre}(A_{id}\gamma^{-1}, D_w, v, \delta) \in \mathbb{Z}_q^m$, where $A_{id_r}\gamma^{-1}d_w = v$ is satisfied and d_w is distributed in $\mathcal{D}_{\Lambda_q^v(A_{id_r}\gamma^{-1}),\delta}$.
   3. outputs a trapdoor d_w.
- Test. In this algorithm, given system public parameters $PP = (A, v, H_1, H_2, H_3, H_4, H_5, \sigma, \delta)$, a proxy-oriented public key $PK_{pro} = A_{pro}$, a trapdoor d_w, and a searchable ciphertext $C = (\xi, \zeta, \theta)$, the cloud server executes the following steps:
   1. computes $\tau \leftarrow \zeta - d_w^{\top}\xi \in \mathbb{Z}_q^{\ell}$. For $j = 1, \cdots, \ell$, if $|\tau_j - \lfloor q/2 \rfloor| < \lfloor q/4 \rfloor$, sets $\tau_j = 1$; otherwise, sets $\tau_j = 0$, then outputs $\tau = (\tau_1, \cdots, \tau_\ell) \in \{0,1\}^{\ell}$.
   2. computes $h = H_5(\tau \| \xi)$, and checks whether the equation $A_{pro}\theta \stackrel{?}{=} h$ holds. If the equation holds, the cloud server returns 1; otherwise, it returns 0.

5 Cryptanalysis of Zhang et al.'s PO-IBEKS

At a high level, the current solution to the IKGA attack (such as public-key authenticated encryption with keyword search (PAEKS) [8] and dual-server PEKS [6,5]) is to prevent malicious cloud server from performing tests and generating ciphertext simultaneously. Zhang et al. [9] also follows this idea; more precisely, they use proxy to prevent the cloud server from adaptively performing encryption for any keywords. Unfortunately, any adversaries can adaptively choose keyword to guess the information of keyword hiding in the trapdoor. In other words, Zhang et al.'s scheme cannot resist outsider keyword guessing attack if there is no secure channel between the cloud server and the data receiver.

Although Zhang et al. claim that in their scheme, IKGA cannot be executed because the malicious cloud server cannot forge valid searchable ciphertext, we show below that any adversaries (whether outsider or insider) can execute IKGA, even if they do not forge any ciphertext.

Let id_p and id_r be the identity of the proxy and data receiver, respectively. After receiving a trapdoor d_w by normal process (insider adversary) or by eavesdropping (outsider adversary), the adversary performs the following steps:

1. randomly chooses a command keyword $w \in 0, 1^{\ell_3}$.
2. computes $\gamma = H_4(id_p \| id_r \| w) \in \mathbb{Z}_q^{m\times m}$.
3. computes $A_{id_r} = A(R_{id_r})^{-1} = A(H_1(id_r))^{-1}$, where A is a system public parameter.
4. checks whether $A_{id_r}\gamma^{-1}d_w \stackrel{?}{=} v$, where v is a system public parameter.
5. if the equation is satisfied, the adversary outputs a guessed keyword w. Otherwise, he/she goes to step 1.

As described in Section 1, since the keywords are low-entropy and their space is usually small, there is a high possibility that the adversary can obtain the correct keyword associated with the trapdoor.

6 Discussion and Possible Solution

In this section, we first discuss the root cause of the attack and then provide possible solution.

In Zhang et al.'s scheme [9], they follow the concept of Huang et al.'s PAEKS [8] (i.e., ciphertext is not only encrypted, but also authenticated) to prevent IKGA. The main difference lies in that in Zhang et al.'s scheme [9], it is the proxy, instead of original data sender, that performs encryption and authentication. Specifically, only proxy can generate an authenticated ciphertext that is valid for the trapdoor, thus, the cloud server cannot generate ciphertext adaptively for any keywords and perform IKGA.

While Zhang et al.'s scheme seems reasonable, the way they generate and use trapdoor is flawed. More accurately, in their scheme, a cloud server use a valid trapdoor d_w to decrypt a ciphertext to obtain τ, and further check whether the ciphertext is authenticated by proxy. However, as cryptanalysis in Section 5, anyone can retrieve information of keyword from trapdoor d_w directly without the need to generate an authenticated ciphertext, but only need to re-produce $\gamma = H_4(id_p \| id_r \| w)$ for different keyword w.

A trivial solution is to let the data receiver and the proxy share a session key k, and set $\gamma = H_4(id_p \| id_r \| w \| k)$. Since k is secret, any adversaries cannot perform IKGA. However, if future research follows this direction, further consideration should be given to attack from proxy as the system model gradually shifts to a dual-server model.

7 Conclusion

In this note, we present cryptanalysis of the PO-IBEKS proposed by Zhang et al. [9]. We show that whether the adversary is outsider or insider, their scheme cannot withstand KGA.

Acknowledgment

This research was supported by the Ministry of Science and Technology, Taiwan (ROC), under Project Numbers MOST 108-2218-E-004-001-, MOST 108-2218-E-004-002-MY2, MOST 109-2218-E-011-007-.

References

1. Agrawal, S., Boneh, D., Boyen, X.: Lattice Basis Delegation in Fixed Dimension and Shorter-ciphertext Hierarchical IBE. In: Rabin, T. (ed.) Annual Cryptology Conference. pp. 98–115. Springer, Berlin, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14623-7_6
2. Alwen, J., Peikert, C.: Generating Shorter Bases for Hard Random Lattices. Theory Comput. Syst. **48**(3), 535–553 (2011). https://doi.org/10.1007/s00224-010-9278-3
3. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public Key Encryption with Keyword Search. In: Cachin, C., Camenisch, J.L. (eds.) International Conference on the Theory and Applications of Cryptographic Techniques. pp. 506–522. Springer Berlin Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_30
4. Byun, J.W., Rhee, H.S., Park, H.A., Lee, D.H.: Off-line Keyword Guessing Attacks on Recent Keyword Search Schemes over Encrypted Data. In: Jonker, W., Petković, M. (eds.) Proceedings of the Third VLDB International Conference on Secure Data Management - SDM'06. pp. 75–83. Springer-Verlag, Berlin, Heidelberg (2006). https://doi.org/10.1007/11844662_6
5. Chen, R., Mu, Y., Yang, G., Guo, F., Wang, X.: A New Ggeneral Framework for Secure Public Key Encryption with Keyword Search. In: Foo, E., Stebila, D. (eds.) Australasian Conference on Information Security and Privacy. pp. 59–76. Springer (2015). https://doi.org/10.1007/978-3-319-19962-7_4
6. Chen, R., Mu, Y., Yang, G., Guo, F., Wang, X.: Dual-server Public-key Encryption with Keyword Search for Secure Cloud Storage. IEEE Trans. Inf. Forensics Secur. **11**(4), 789–798 (2015). https://doi.org/10.1109/TIFS.2015.2510822
7. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for Hard Lattices and New Cryptographic Constructions. In: Dwork, C. (ed.) Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing - STOC'08. pp. 197–206. ACM, New York, NY, United States (2008). https://doi.org/10.1145/1374376.1374407
8. Huang, Q., Li, H.: An Efficient Public-key Searchable Encryption Scheme Secure against Inside Keyword Guessing Attacks. Inf. Sci. **403-404**, 1 – 14 (2017). https://doi.org/10.1016/j.ins.2017.03.038
9. Zhang, X., Tang, Y., Wang, H., Xu, C., Miao, Y., Cheng, H.: Lattice-based Proxy-oriented Identity-based Encryption with Keyword Search for Cloud Storage. Inf. Sci. **494**, 193–207 (2019). https://doi.org/10.1016/j.ins.2019.04.051