FS-PEKS : lattice-based forward secure public-key encryption with keyword search for cloud-assisted Industrial Internet of Things

Zhang, Xiaojun; Xu, Chunxiang; Wang, Huaxiong; Zhang, Yuan; Wang, Shixiong

2021

Zhang, X., Xu, C., Wang, H., Zhang, Y. & Wang, S. (2021). FS-PEKS : lattice-based forward secure public-key encryption with keyword search for cloud-assisted Industrial Internet of Things. IEEE Transactions On Dependable and Secure Computing, 1019(1032), 18-3. https://dx.doi.org/10.1109/TDSC.2019.2914117

https://hdl.handle.net/10356/151970

https://doi.org/10.1109/TDSC.2019.2914117

FS-PEKS: Lattice-based Forward Secure Public-key Encryption with Keyword Search for Cloud-assisted Industrial Internet of Things

Xiaojun Zhang, Chunxiang Xu, *Member, IEEE*, Huaxiong Wang, Yuan Zhang, *Student Member, IEEE*, and Shixiong Wang

Abstract—Cloud-assisted Industrial Internet of Things (IIoT) relies on cloud computing to provide massive data storage services. To ensure the confidentiality, sensitive industrial data need to be encrypted before being outsourced to cloud storage server. Public-key encryption with keyword search (PEKS) enables users to search target encrypted data by keywords. However, most existing PEKS schemes are based on conventional hardness assumptions, which are vulnerable to adversaries equipped with quantum computers in the near future. Moreover, they suffer from key exposure, and thus the security would be broken once the keys are compromised. In this paper, we propose a forward secure PEKS scheme (FS-PEKS) based on lattice assumptions for cloud-assisted IIoT, which is post-quantum secure. We integrate a lattice-based delegation mechanism into FS-PEKS to achieve forward security, such that the security of the system is still guaranteed even the keys are compromised by the adversaries. We define the first formal security model on forward security of PEKS, and prove the security of FS-PEKS under the model. As the keywords of industrial data are with inherently low entropy, we further extend FS-PEKS to resist insider keyword guessing attacks (IKGA). The comprehensive performance evaluation demonstrates that FS-PEKS is practical for cloud-assisted IIoT.

Index Terms—Cloud-assisted Industrial Internet of Things, public-key encryption with keyword search, lattice assumptions, forward security, insider keyword guessing attacks.

1 Introduction

Internet of Things (IoT) deployment is composed of various types of sensors, actuators, and other intelligent terminal equipments connected to the Internet [1], [2]. It provides identification, computation, and mutual information exchange among the connected devices. As a special type of IoT, Industrial Internet of Things (IIoT), which relies on various kinds of mobile intelligent terminal devices and wireless communication technologies to realize remote monitoring and management in modern industrial sectors [3], has been increasingly prevalent. With the sharp increase of the massive industrial data, cloud computing technologies have been integrated into IIoT to store and process these information [4], [5]. Thus, apart from upgrading to intelligent industries, the cloud-assisted IIoT also brings a vast improvement in industrial manufacturing efficiency and lowers the production cost.

Despite the great benefits on managing industrial data brought by cloud-assisted IIoT, security and privacy concerns in data outsourcing have been raised, of which data confidentiality is one of the most important aspects. From the perspective of cloud users, the contents of the outsourced industrial critical data are very sensitive and should be kept confidential for privacy preservation [6], [7], [8]. Therefore, sensitive industrial data should be encrypted before being outsourced to cloud-assisted IIoT.

Along with data confidentiality, data sharing is also indispensable. In a cloud-assisted IIoT, a data administrator (i.e., a data sender) collects different industrial data from various kinds of sensors, and shares them with an intended recipient (e.g., a senior skilled worker). The data administrator encrypts the data with the public key of the intended recipient, and further uploads the encrypted data (i.e., ciphertext) to the cloud storage server. To retrieve the encrypted data, a trivial approach for the recipient is to download all corresponding ciphertexts stored in the cloud storage server, decrypt the entire encrypted industrial data set, and further retrieve the target data locally. Nonetheless, it is considerably impractical due to heavy communication

- *This work is supported by National Key R&D Program of China (No.2017YFB0802000), National Natural Science Foundation of China (NO.61872060), China Postdoctoral Science Foundation Funded Project (No.2017M623008), Singapore Ministry of Education under Research Grant MOE2016-T2-2-014(S) and by the National Research Foundation, Prime Minister's Office, Singapore under its Strategic Capability Research Centres Funding Initiative. We are also grateful to Dr. Su Le who provided us valuable suggestions.*

- *X. Zhang is with the School of Computer Science, Research Center for Cyber Security, Southwest Petroleum University, China; State Key Laboratory of Cryptology, P.O.Box 5159, Beijing 100878, China; the Center for Cyber Security, School of Computer Science and Engineering, University of Electronic Science and Technology of China, China; the School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore (e-mail: zhangxjdzkd2012@163.com).*
- *C. Xu, and Y. Zhang are with the Center for Cyber Security, School of Computer Science and Engineering, University of Electronic Science and Technology of China, China (e-mail: chxxu@uestc.edu.cn; ZY_LoYe@126.com).*
- *H. Wang and S. Wang are with the School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore (e-mail: HXWang@ntu.edu.sg; wsx09@foxmail.com).*
- *Y. Zhang is also with the Department of Electrical and Computer Engineering, University of Waterloo, Canada.*
- *S. Wang is also with the College of Computer, National University of Defense Technology, China.*

overhead and computational costs. This makes efficient retrieval of target data from the cloud storage a formidable task.

Public-key encryption with keyword search (PEKS) [9] could be a good candidate in cloud-assisted IIoT to achieve data retrieval without leaking privacy. PEKS is a cryptographic primitive that supports searching over encrypted data by keywords. In PEKS, a data sender firstly encrypts data as well as its keyword under the data receiver's public key, and uploads corresponding ciphertexts to the storage server (i.e., cloud server). The data receiver could generate a trapdoor of a specific keyword by using his/her private key, and submits it to the storage server. The storage server then could test whether the ciphertext of the keyword matches the trapdoor for data retrieval.

While existing PEKS schemes [10], [11], [12], [13], [14] bring significant benefits to cloud-assisted IIoT, there are two main hindrances in widely adopting PEKS into the system in the near future. On the one hand, most existing PEKS schemes are constructed on conventional cryptographic hardness assumptions. However, as pointed out in [15], with the emergence of the quantum computers, PEKS schemes will be threatened. Recent breakthrough results [16], [17] indicate that adopting quantum computers in industrial sectors would be possible in the near future, and thus poses the post-quantum secure PEKS schemes more demanding than ever. On the other hand, the most computationally expensive part of PEKS for the cloud server is to retrieve target data from the entire outsourced database, as the cloud server is required to perform a test algorithm for each keyword over the database during the search process. Existing PEKS schemes introduce a significant end-to-end computation delay due to costly public-key cryptographic operations, such as bilinear pairing and modular exponentiation operations. In a cloud-assisted IIoT, the cloud server may simultaneously process industrial data from various kinds of sensors or mobile terminal devices, to satisfy the data retrieval requirements of multiple data receivers. As such, existing PEKS schemes are confronted with performance bottleneck on the cloud server side.

In addition to the aforementioned hindrances, deployment of PEKS in cloud-assisted IIoT would also face two security challenges. With the explosive use of mobile intelligent terminal devices with limited key protection, most existing PEKS schemes suffer from key-exposure problems [18]. Once a data receiver's private key is compromised, adversaries could reveal the contents of trapdoors that the data receiver previously submitted, and further violate the confidentiality of outsourced data. Recent reports show that such attacks have become commonplace [19], [20]. We also analyze the inherent characteristic of industrial data in IIoT and point out that the keywords in industrial data are with inherently low entropy, making it practical for the misbehaved cloud server to launch insider keyword guessing attacks (IKGA) [21].

In this paper, we propose a forward secure PEKS scheme for cloud-assisted IIoT, called FS-PEKS. FS-PEKS is based on lattice-based cryptography [22], which enjoys very strong security level based on worst-case hardness, and enables FS-PEKS to be post-quantum secure. We propose a key update mechanism and integrate it into FS-PEKS to resist key exposure. In FS-PEKS, every private key is associated with a time period, and required to be updated at the end of each time period. The key idea behind FS-PEKS to achieve forward security is to exploit the lattice basis delegation mechanism [23], which flexibly updates the compromised or expired private key of each entity even after multiple time periods, termed lazy update. To enable the cloud server to perform the testing process with a high efficiency, we set a fixed binary string in generating the PEKS ciphertext. Once the first decrypted bit is different from that of corresponding location in the fixed binary string, the cloud server could abort immediately, without decrypting all bits of the fixed binary string. We further extend FS-PEKS to thwart IKGA, where the preimage sample function [24] and learning with errors (LWE) encryption [25] are employed to generate authenticated PEKS ciphertexts, such that the misbehaved cloud server cannot break the security of FS-PEKS by performing IKGA.

Specifically, the contributions of this work are elaborated as follows.

- We propose a lattice-based forward secure PEKS scheme, called FS-PEKS. FS-PEKS is constructed on a hierarchy identity-based encryption based on the hardness assumption of deciding LWE problem, and therefore is secure against quantum attacks. We define the first formal security model of forward secure PEKS scheme, and present formal security proof of FS-PEKS under the proposed model.
- FS-PEKS achieves forward security, and keeps the updated private key size constant at the end of each time period, and independent of the time periods. We extend FS-PEKS to prevent IKGA from the misbehaved cloud server, without the need of establishing a secure channel between the cloud server and data receivers.
- We conduct a comprehensive performance evaluation. Compared with existing schemes, FS-PEKS is efficient in terms of computational costs and communication overhead. Specifically, in the keyword testing process, the cloud server only needs to perform simple addition and multiplication operations over a moderate module, without more time-consuming cryptographic operations, such as bilinear pairing and modular exponentiation operations. Thus, FS-PEKS greatly decreases the end-to-end delay from the cloud server to the data receiver, making it quite practical for post-quantum secure cloud-assisted I-IoT.

The paper is organized as follows. In Section 2, we present the problem statement, including PEKS with new requirements for cloud-assisted IIoT, forward security of PEKS and keyword guessing attacks (KGA). In Section 3, we present preliminaries, including system model, security model, design goals, and lattice-based background. In Section 4, we propose FS-PEKS and its extension to resist IKGA. In Section 5, we give the correctness and security proofs. In Section 6, we conduct a comprehensive performance evaluation. In Section 7, we review the related work. Finally, we draw the conclusions and future work in Section 8.

2 Problem Statement

2.1 Public-key encryption with keyword search

PEKS scheme enables privacy-preserving encrypted data retrieval and sharing. In a PEKS scheme, a data sender firstly generates the encrypted data $E_{pk}(M)$ under a public-key encryption system, and produces the corresponding ciphertext CT_w of keyword w (called PEKS ciphertext) with the PEKS scheme, where the keyword w is extracted from the primitive data M. Finally the data sender uploads $E_{pk}(M) \| CT_w$ to the cloud server. When the intended data receiver wishes to retrieve the encrypted data $E_{pk}(M)$ associated with some specific keyword w, the data receiver generates the trapdoor t_w of the specific keyword, and sends it to the cloud server via a secure channel. The cloud server performs the search on a collection of encrypted data with the trapdoor t_w, and checks whether there exists a keyword in encrypted data matches the one selected by the data receiver. If the cloud server succeeds in such a matching, it returns the corresponding encrypted data $E_{pk}(M)$ associated with the keyword w to the data receiver. Finally, the data receiver retrieves the intended encrypted data, and decrypts it with the private key sk of the public-key encryption system.

In order to evaluate whether a PEKS scheme is practical for cloud-assisted IIoT, there are two main factors to be considered. On the one hand, a data sender, e.g., a data administrator of an industrial sector, takes charge of encrypting industrial data and corresponding keywords contained in the data, and further uploads them to the cloud storage server. Compared with efficiency, the data sender is more concerned with data privacy, e.g., whether the encrypted industrial data are shared correctly by the intended data receiver without leaking any information to others. In particular, with the rapid development of advanced quantum computers, the application of PEKS in cloud-assisted IIoT that could achieve quantum-resistance will be a critical consideration. On the other hand, the major challenge from the data receiver's perspective is the delay from the cloud server to the data receiver. As the most computationally expensive part of the PEKS is generally the search and testing phase, especially in mobile cloud-assisted IIoT settings, the cloud server may perform the search and test algorithm to serve multiple data receivers simultaneously, thus, the data receiver, e.g, a senior skilled worker, may need to bear a heavy delay, and cannot timely use the shared industrial data in practice.

2.2 Forward security of PEKS

As far as we are concerned, the security of modern cryptographic systems applied in industrial sectors wholly depends on the assumption that the individual's private key is absolutely secure. Actually, with the explosive use of mobile intelligent terminal devices in IIoT, due to limited key protection, the private key of the individual may be compromised, or even known by the cloud server. To complete various security tasks, the individual needs to maintain a complex procedure for key management. Compared with the enterprises, the individual has a relatively weaker sense of security protection: even some careless mistakes or faults in managing the private key may lead to key exposure [18]. In PEKS without supporting forward security, once the private key of the data receiver is compromised, the adversary might use the exposed private key to generate the trapdoor, and submit it to the cloud server as a legitimate request. The cloud server tests it successfully, and returns the previous encrypted data to the adversary. Eventually, the previous encrypted data can be decrypted by the adversary under the exposed private key. More recently, Zhang et al. [26] have presented a devastating file-injection attack process. It is possible to reveal the contents of past search queries of dynamic searchable symmetric encryption schemes with a few injection of files. This fact highlights the importance of forward security in any real-world deployment, especially in the mobile cloud storage application settings.

2.3 Keyword guessing attacks

In most of the existing PEKS schemes, there exists an inherent security limitation: vulnerability against off-line keyword guessing attacks (KGA). More specifically, an outside adversary can encrypt any selected keyword by using the data receiver's public key. Once the trapdoor is intercepted by the adversary, it could run the test algorithm to identify the ciphertext of the keyword which matches the targeted trapdoor. This enables the adversary to learn the keyword hidden in the trapdoor, and thus violates the data privacy. To deal with such KGA, trivial approaches have been explored. Firstly, set up a secure channel between the data receiver and the cloud server such that the adversary cannot intercept the trapdoor [27]. Secondly, designate a particular cloud server to search and test the results in PEKS, such that only the unique cloud server could conduct the testing process. This is is called designated-tester PEKS [28]. Apart from the existing KGA by the outside adversary, the misbehaved cloud server could also perform insider keyword guessing attacks (IKGA). It executes the exhaustive search as what an outside adversary does, even if the trapdoor is transmitted via a secure channel, and finally find the keyword which is indeed generated by the data receiver. Public-key encryption with fuzzy keyword search [29] could resist IKGA to some extent. In such a scheme, each keyword corresponds to an exact trapdoor. With the fuzzy trapdoor, the misbehaved cloud server may know about the exact keyword as two or more keywords may share the same trapdoor. Nevertheless, the misbehaved cloud server could still know which small set the underlying keyword belongs to. In addition, the data receiver has to locally filter out the non-matching ones from the small set returned from the cloud server, which will introduce heavy communication overhead and computational costs to the data receiver. In recent, a public-key authenticated searchable encryption scheme has been proposed [10], in which a data sender not only encrypts a keyword, but also authenticates it, such that the data receiver could be convinced that the encrypted keyword could only be generated by the data sender.

3 Preliminaries

3.1 System model

In this section, we introduce the system model of PEKS for cloud-assisted IIoT in Fig. 1, which has three entities: a data sender, a data receiver, and a cloud server.

Data Sender: As a data administrator of an industrial sector, the data sender collects industrial data (such as production information, operation status of the equipment, and other information collected by the sensors), encrypts the industrial data as well as keywords contained in the data under the public key of an intended data receiver, and further uploads them to the cloud server associated with the IIoT.

Data Receiver: As a cloud user, the data receiver uses the private key to generate the trapdoor associated with the specific keyword, and sends it to the cloud server to retrieve the intended encrypted industrial data.

Cloud Server: It is associated with IIoT, and it is responsible for the computation and storage of industrial data in the IIoT system. Once receiving a trapdoor from the data receiver, it performs the testing process and returns corresponding encrypted industrial data.

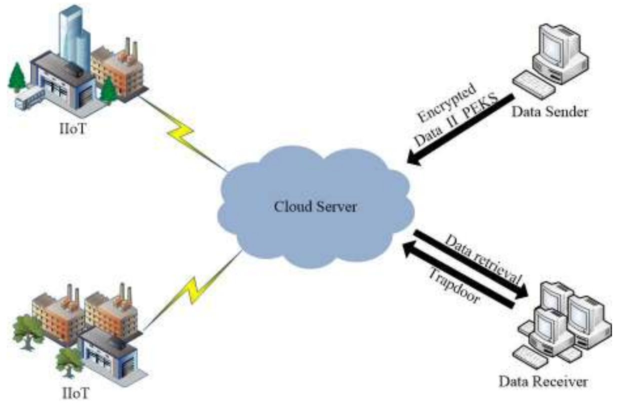

Fig. 1. System model of PEKS for cloud-assisted IIoT

A formal definition of FS-PEKS is given in the following.

Definition 1. *FS-PEKS consists of five polynomial-time algorithms, **Setup**, **KeyUpdate**, **PEKS**, **Trapdoor**, **Test**.*

Setup: The probabilistic polynomial-time (PPT) algorithm takes as inputs a secure parameters κ, outputs the system public parameter Σ, and the initial public-private key pairs of the data sender and the data receiver respectively.

KeyUpdate: This PPT algorithm takes as inputs key pair $(\mathsf{PK}_{r\|i}, \mathsf{SK}_{r\|i})$ of the data receiver, time period i, outputs key pair $(\mathsf{PK}_{r\|j}, \mathsf{SK}_{r\|j})$ in time period j, where $i < j$. This PPT algorithm also takes as inputs key pair $(\mathsf{PK}_{s\|i}, \mathsf{SK}_{s\|i})$ of the data sender, time period i, outputs key pair $(\mathsf{PK}_{s\|j}, \mathsf{SK}_{s\|j})$ in time period j for $i < j$.

PEKS: This PPT algorithm takes as inputs Σ, the public key $\mathsf{PK}_{r\|j}$ of the data receiver, the keyword w and the current time period j with $j = 1, \cdots, \eta$, where η denotes the total time periods. It outputs a forward secure PEKS ciphertext CT_j associated with the keyword w.

Trapdoor: This PPT algorithm takes as inputs Σ, the public-private key pair $(\mathsf{PK}_{r\|j}, \mathsf{SK}_{r\|j})$ of the data receiver in current time period j and a keyword w, outputs a trapdoor $t_{w\|j}$ associated with the keyword w.

Test: This deterministic polynomial-time algorithm takes as inputs a trapdoor $t_{w\|j}$ in time period j, a forward secure PEKS ciphertext CT_j, outputs 1 if CT_j and $t_{w\|j}$ contain the same keyword w, and 0 otherwise.

Correctness consistence: FS-PEKS requires that for any honestly generated key pair $(\mathsf{PK}_{s\|j}, \mathsf{SK}_{s\|j})$ of the data sender, key pair $(\mathsf{PK}_{r\|j}, \mathsf{SK}_{r\|j})$ of the data receiver in any time period j, and for any keyword w, $\mathsf{Test}(t_{w\|j}, CT_j, j) = 1$ holds with probability 1, where $CT_j \leftarrow \mathsf{PEKS}(w, \mathsf{PK}_{r\|j})$ and $t_w \leftarrow \mathsf{Trapdoor}(w, \mathsf{SK}_{r\|j}, \mathsf{PK}_{r\|j}, j)$.

3.2 Security model

Now we define ciphertext indistinguishability of FS-PEKS under the adaptively chosen keyword attacks. The challenger $\mathcal{C}$ generates the system public parameters, prepares the initial public keys of the data sender and the data receiver, and returns them to the adversary $\mathcal{A}$. The adversary $\mathcal{A}$ is allowed to perform queries as follows.

Hash oracle: $\mathcal{A}$ is allowed to issue all hash oracles in time period j, $j = 1, \cdots, \eta$, where η is the total number of time periods, and $\mathcal{A}$ can obtain corresponding hash value.

Trapdoor oracle: $\mathcal{A}$ can adaptively query to $\mathcal{C}$ on the trapdoor t_w for any keyword w of his choice in time period j. To achieve forward security, the restriction is that the time period $j > j^*$, j^* is the break-in time.

Break-in phase: This phase models the possibility of key exposure. Once receiving this query for private key $\mathsf{SK}_{r\|j}$ of the data receiver in time period j from $\mathcal{A}$, $\mathcal{C}$ returns corresponding private key $\mathsf{SK}_{r\|j}$ in time period j to $\mathcal{A}$. The restriction is that the time period $j > j^*$, where j^* is the break-in time period.

Challenge phase: $\mathcal{A}$ adaptively chooses two keywords (w_0^*, w_1^*) in time period j^* which have not been queried for trapdoor oracle, and submits them to $\mathcal{C}$ as the challenged keywords. $\mathcal{C}$ randomly chooses a bit $b \in \{0, 1\}$, computes $CT_{j^*}^* \leftarrow \mathsf{PEKS}(w_b^*, \mathsf{PK}_{r\|j^*})$ and returns it to $\mathcal{A}$.

$\mathcal{A}$ continues to issue queries for trapdoor oracle as above, with the restriction that neither w_0^*, or w_1^* could be submitted to the oracle.

Guess: Finally, $\mathcal{A}$ outputs a bit $b' \in \{0, 1\}$. It wins the game if and only if $b' = b$.

We define $\mathcal{A}$'s advantage of successfully distinguishing the ciphertexts of PEKS in the break-in time period j^* under the adaptively chosen keyword attacks as $\mathsf{Adv}_{\mathcal{A}}^{\mathcal{C}}(\kappa) = |\Pr[b' = b] - 1/2|$.

Furthermore, as the keywords in industrial data are with inherently low entropy, IKGA has been a threat to the deployment of PEKS in cloud-assisted IIoT. In essence, an adversary, including the misbehaved cloud server, may guess the target keyword, and try to generate a forged searchable ciphertext, such that it could pass the testing process. Thus, to resist IKGA from the adversary, we propose an extension of FS-PEKS in Section 4, and provide the security proof of the resistance to IKGA in Section 5. Here, we define the security of the IKGA-resistance scheme in the extension of FS-PEKS as follows.

The challenger $\mathcal{C}$ generates the system public parameters, prepares the initial public keys of the data sender and the data receiver, and returns them to the adversary $\mathcal{F}$. The adversary $\mathcal{F}$ is allowed to perform queries as follows.

Hash oracle: $\mathcal{F}$ is allowed to issue all hash oracles in time period j, $j = 1, \cdots, \eta$, where η is the total number of time periods, and $\mathcal{F}$ could obtain the corresponding hash value.

Trapdoor oracle: $\mathcal{F}$ can adaptively query to $\mathcal{C}$ on the trapdoor t_w for any keyword w of his choice in time period j.

Searchable ciphertext oracle: $\mathcal{F}$ queries to $\mathcal{C}$ on any keyword w in time period j, $\mathcal{C}$ can respond with corresponding searchable ciphertext. To achieve forward security, the restriction is that the time period $j > j^*$, j^* is the break-in time.

Break-in phase: This phase models the possibility of key exposure. Once receiving this query for private key $\mathsf{SK}_{s\|j}$ of the data sender in time period j from $\mathcal{F}$, $\mathcal{C}$ returns corresponding private key $\mathsf{SK}_{s\|j}$ in time period j to $\mathcal{F}$. The restriction is that the time period $j > j^*$, where j^* is the break-in time period.

Forgery phase: $\mathcal{F}$ outputs a forged searchable ciphertext associated with w^* in time period j^*, which could pass the testing process.

We denote $\mathcal{F}$'s advantage of successfully performing IKGA in the break-in time period j^* as $\mathsf{Adv}_{\mathcal{F}}^{\mathcal{C}}(\kappa)$.

3.3 Design goals

In this paper, we target FS-PEKS for cloud-assisted IIoT, there exist two types of challenges.

1) How to achieve provable security of FS-PEKS. As we discussed before, existing PEKS schemes suffer from key exposure, once the private key is compromised, an adversary can reveal the contents of past search queries. Thus, the previous encrypted data can be decrypted by the adversary. Existing LWE-based encryption schemes cannot be directly applied to our construction, we need to modify and integrate other lattice-based techniques into the construction of FS-PEKS. In addition, how to provide security proof of FS-PEKS under the formal definition with reduction to the hardness assumption of deciding LWE problem, is also a challenging and tricky issue.
2) How to enable FS-PEKS to resist IKGA. We have analyzed the inherent characteristic of industrial data and point out that the keywords in industrial data are with inherently low entropy. Thus, KGA has been a threat to the deployment of PEKS in cloud-assisted IIoT. Particularly, in a much more stronger attack model, IKGA could be launched by a misbehaved cloud server.

To make FS-PEKS in cloud-assisted IIoT practical under the aforementioned model, the following properties should be achieved.

1) Security: FS-PEKS should achieve quantum resistance, forward security, and IKGA resistance.
2) Efficiency: FS-PEKS should maintain high computational efficiency and low communication overhead. In particular, the reduced testing time and smaller trapdoor size will contribute to minimizing the end-to-end delay from the cloud server to the data receiver.

3.4 Lattice-based background

Lattice-based cryptography is secure against quantum attacks, and enjoys strong security guarantee based on worst-case hardness. In addition, lattice-based cryptography inherently has efficient implementations, since it only needs simple addition and multiplication operations over a moderate module.

We firstly provide some definitions about lattice-based cryptography as follows.

Definition 2. *Let $B = \{b_1, \cdots, b_m\} \in \mathbb{R}^{m \times m}$ be an $(m \times m)$-dimension matrix, where the columns are linearly independent vectors $b_1, \cdots, b_m \in \mathbb{R}^m$. The m-dimensional full-rank lattice Λ generated by B is $\mathcal{L}(B) = \{y \in \mathbb{R}^m : \exists z = (z_1, z_2, \cdots, z_m)^\top \in \mathbb{Z}^m, y = Bz = \sum_{i \in [m]} z_i b_i\}$.*

The basis of the lattice $\Lambda = \mathcal{L}(B)$ is $B = \{b_1, \cdots, b_m\}$. Let $\widetilde{B} = \{\widetilde{b_1}, \cdots, \widetilde{b_m}\}$ denote the Gram-Schmidt orthogonalization of the vectors $b_1, \cdots, b_m$ taken in that order.

Definition 3. *With a matrix $A \in \mathbb{Z}_q^{n \times m}$, a vector $\mu \in \mathbb{Z}_q^n$, we define the q-module integer lattices in [30] as follows.*

1) $\Lambda_q(A) = \{y \in \mathbb{Z}_q^m : \exists z \in \mathbb{Z}_q^n, y = A^\top z \text{ mod } q\}$.
2) $\Lambda_q^\perp(A) = \{e \in \mathbb{Z}_q^m : Ae = 0 \text{ mod } q\}$.
3) $\Lambda_q^\mu(A) = \{e \in \mathbb{Z}_q^m : Ae = \mu \text{ mod } q\}$.

Definition 4. *For any $\sigma > 0$, the Gaussian function on $\mathbb{R}^m$ centered at c is $\forall x \in \mathbb{Z}^m$, $\rho_{\sigma,c}(x) = exp(-\pi\|x - c\|^2/\sigma^2)$, and $\rho_{\sigma,c}(L) = \Sigma_{x \in L}\rho_{\sigma,c}(x)$, where L is a subset of $\mathbb{Z}^m$. The discrete Gaussian distribution over L with center c and parameter σ is $\forall y \in L$, $\mathcal{D}_{L,\sigma,c}(y) = \rho_{\sigma,c}(y)/\rho_{\sigma,c}(L)$.*

Definition 5. *Given a prime q, a positive integer n, and Gaussian noise distribution χ. A (Z_q, n, χ)-LWE problem [25] consists of access to an unspecified challenge LWE oracle $\mathcal{O}_L$, which is either a truly random sampler $\mathcal{O}$, or, a noisy pseudo-random sampler $\mathcal{O}'$. They are described respectively as follows.*

$\mathcal{O}$: This oracle outputs truly uniform samples from $\mathbb{Z}_q^n \times \mathbb{Z}_q^\ell$.

$\mathcal{O}'$: This oracle outputs samples of the form $(u_k, v_{k1}, \cdots, v_{k\ell}) = (u_k, u_k^\top b_1 + z_1, \cdots, u_k^\top b_\ell + z_\ell) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^\ell$, where each $b_l \in \mathbb{Z}_q^n$ $(l = 1, \cdots, \ell)$ is a uniformly distributed persistent value which is invariant across invocations, each $z_l \in \mathbb{Z}_q$ is a fresh sample from χ, and u_k is uniform vector in $\mathbb{Z}_q^n$.

The $(\mathbb{Z}_q, n, \chi)$-LWE problem allows to repeatedly query to the challenge oracle $\mathcal{O}_L$. Here we say that an adversary $\mathcal{A}$ decides the $(\mathbb{Z}_q, n, \chi)$-LWE problem if $\mathsf{LWE}_{\mathsf{adv}[\mathcal{A}]} := |\Pr[\mathcal{A}^{\mathcal{O}'} = 1] - \Pr[\mathcal{A}^{\mathcal{O}} = 1]|$ is non-negligible. As proved in [25], [31], the $(\mathbb{Z}_q, n, \chi)$-LWE problem is as hard as the worst-case SIVP and GapSVP under a quantum reduction.

Definition 6. *The Inhomogeneous Small Integer Solution (ISIS) Problem: with a matrix $U \in \mathbb{Z}_q^{n \times m}$, a uniform vector $\vartheta \in \mathbb{Z}_q^n$, and a real number $\varpi > 0$, the goal is to solve a nonzero integer vector $\xi \in \mathbb{Z}^m$ such that $U\xi = \vartheta$ mod q and $\|\xi\| \leq \varpi$.*

As proved in [24], for any prime $q > \varpi \cdot \omega(\sqrt{n \log n})$ and any poly-bounded $\varpi = poly(n)$, the average-case hardness assumption of ISIS problem is as hard as approximating the problem SIVP in the worst case to within certain factor $\varpi \cdot \tilde{O}(\sqrt{n})$.

In our scheme, we take advantage of the algorithm $\mathsf{TrapGen}$ in [32] to generate $A \in \mathbb{Z}_q^{m \times n}, T_A \in \mathbb{Z}_q^{m \times m}$, such that A is statistically close to a uniform matrix in $\mathbb{Z}_q^{n \times m}$, and T_A is a random short lattice basis of $\Lambda_q^{\perp}(A)$, and each Euclidean norm of all the rows is bounded by $O(n \log n)$.

Now we describe the algorithm $\mathsf{SamplePre}$ [24] as follows.

Lemma 1. *Taking as inputs $A \in \mathbb{Z}_q^{n\times m}$, $T_A \in \mathbb{Z}_q^{m\times m}$, a vector $\mu \in \mathbb{Z}_q^n$, $m \geq 2n\lceil \log q \rceil$, and a parameter $\sigma \geq \|\widetilde{T_A}\| \cdot \omega(\sqrt{\log m})$, where $\|\widetilde{T_A}\|$ is the Euclidean norm of $\widetilde{T_A}$, the* PPT *algorithm* $\mathsf{SamplePre}(A, T_A, \mu, \sigma)$ *outputs a sample $t \in \mathbb{Z}_q^m$ distributed in $\mathcal{D}_{\Lambda_q^{\mu}(A),\sigma}$, where $\mathcal{D}_{\Lambda_q^{\mu}(A),\sigma}$ is a Gaussian noise distribution over $\Lambda_q^{\mu}(A)$ with parameter σ, such that $At = \mu$ mod q.*

Next we introduce the lattice basis delegation $\mathsf{NewBasisDel}$ [23], which is a key technique to update the private key. $\mathsf{NewBasisDel}$ refers to the distribution $\mathcal{D}_{m\times m}$ on matrices in $\mathbb{Z}_q^{m\times m}$, which denotes $(\mathcal{D}_{\mathbb{Z}^m,\delta_R})^m$ conditioned on the resulting matrix being $\mathbb{Z}_q$-invertible, where $\delta_R = \sqrt{n \log q} \cdot \omega(\sqrt{\log m})$.

Lemma 2. *Taking as inputs $A \in \mathbb{Z}_q^{n\times m}$, a $\mathbb{Z}_q$-invertible matrix R sampled from $\mathcal{D}_{m\times m}$, a short lattice basis T_A, and parameter $\delta \geq \|\widetilde{T_A}\| \cdot \delta_R \sqrt{m}\omega(\log^{3/2} m)$, the* PPT *algorithm* $\mathsf{NewBasisDel}$ *outputs a random short lattice basis T_B of $\Lambda_q^{\perp}(B)$, where $B = AR^{-1}$.*

Finally, we introduce $\mathsf{SampleR}$ and $\mathsf{SampleRwithBasis}$ [23], which are important to realize the security proof of ciphertext indistinguishability and the resistance to IKGA.

Lemma 3. *The* PPT *algorithm* $\mathsf{SampleR}$ *performs as follows.*

1) *Set T to be a canonical basis of the lattice $\mathbb{Z}^m$.*
2) *For $i = 1, 2, \cdots, m$, sample each r_i from the algorithm* $\mathsf{SampleGaussian}(\mathbb{Z}^m, T, \delta_R, 0)$ *described in [23].*
3) *Output R, if $R = \{r_1, r_2, \cdots, r_m\}$ is $\mathbb{Z}_q$-invertible, otherwise repeat the step 2.*

Lemma 4. *Taking as inputs $m \geq 2n\lceil \log q \rceil$, $q \geq 3$, $A \in \mathbb{Z}_q^{n\times m}$, the* PPT *algorithm* $\mathsf{SampleRwithBasis}$ *outputs a low-norm matrix R which is statistically close to $\mathcal{D}_{m\times m}$, and a random short lattice basis T_B for $\Lambda_q^{\perp}(B)$ with $B = AR^{-1}$, such that $\|\tilde{T_B}\| \leq \delta_R/\omega(\sqrt{\log m})$ with an overwhelming probability.*

4 THE PROPOSED FS-PEKS

4.1 Overview

FS-PEKS consists of **Setup**, **KeyUpdate**, **PEKS**, **Trapdoor**, and **Test** algorithms. In order to construct secure lattice-based FS-PEKS, in **Setup**, we need to set secure parameters for q-module lattices, and set the initial public-private key pairs of the data sender and data receiver, respectively. The key idea in **KeyUpdate** is to modify the lattice basis delegation $\mathsf{NewBasisDel}$, which enables the cloud user (i.e., a data sender or a data receiver) to flexibly achieve private key update for each time period $j = 1, 2, \cdots, \eta$. More specifically, we modify $\mathsf{NewBasisDel}$ as updating private key in each time period, which could generate a series of private keys $\mathsf{SK}_{user\|1}, \mathsf{SK}_{user\|2}, \cdots, \mathsf{SK}_{user\|\eta}$ of an entity. As corresponding public keys $\mathsf{PK}_{user\|1}, \mathsf{PK}_{user\|2}, \cdots, \mathsf{PK}_{user\|\eta}$ could be computed by any entity based on the initial public key and the corresponding hash function, the proposed FS-PEKS does not need the cloud user to update corresponding public key in each time period essentially. From the perspective of the data receiver, when transit changes from time period i to time period j, $i < j$, the data receiver revokes $\mathsf{SK}_{r\|i}, \mathsf{SK}_{r\|i+1}, \cdots, \mathsf{SK}_{r\|j-1}$ from the local storage, the new private keys remain $\mathsf{SK}_{r\|j}, \mathsf{SK}_{r\|j+1}, \cdots, \mathsf{SK}_{r\|\eta}$. From this period onwards, any outside adversary cannot succeed in searching over the PEKS ciphertext for previous time periods, even if the private keys $\mathsf{SK}_{r\|j}, \mathsf{SK}_{r\|j+1}, \cdots, \mathsf{SK}_{r\|\eta}$ are exposed.

In **PEKS**, the data sender sets a fixed binary string $\gamma_j = (1, 1, \cdots, 1) \in \{1\}^{\ell}$. With the pubic key $\mathsf{PK}_{r\|j}$ of the data receiver, the data sender runs the modified LWE-based encryption mechanism to generate the PEKS ciphertext of the fixed binary string, which is associated with the keyword. In **Trapdoor**, with the private key $\mathsf{SK}_{r\|j}$, the data receiver runs the modified $\mathsf{NewBasisDel}$ to generate a random lattice basis $T_{w\|j}$, which is associated with the selected keyword, and further employs the preimage sample function $\mathsf{SamplePre}$ to generate the trapdoor $t_{w\|j}$ of the selected keyword, and submits it to the cloud server for search on the PEKS ciphertext. In **Test**, with the trapdoor $t_{w\|j}$, the cloud server could decrypt the PEKS cipheretxt under the decryption mode of LWE. In particular, once a bit is decrypted as 0 for the first time, the cloud server aborts, without decrypting each bit of the fixed binary string further. Up to the recovery of the entire fixed binary string $\gamma_j = (1, 1, \cdots, 1) \in \{1\}^{\ell}$, the cloud server could make sure the trapdoor $t_{w\|j}$ and the PEKS ciphertext contain the same keyword.

We further extend FS-PEKS to resist IKGA. Without using the fixed binary string $\gamma_j = (1, 1, \cdots, 1) \in \{1\}^{\ell}$ in **PEKS**, the data sender chooses a random binary string $\gamma_j = (\gamma_{j1}, \gamma_{j2}, \cdots, \gamma_{j\ell}) \in \{0, 1\}^{\ell}$ to generate the corresponding ciphertext as before. In addition, with the private key $\mathsf{SK}_{s\|j}$ of the data sender in time period j, the data sender employs $\mathsf{SamplePre}$ to generate a signature associated with the random binary string, and integrates the signature into the authenticated PEKS ciphertext. In **Test**, the cloud server needs to recover the entire random binary string, and then verifies the validity of the signature. Consequently, without the private key $\mathsf{SK}_{s\|j}$ of the data sender, the misbehaved cloud server cannot generate a valid signature, or even cannot perform such IKGA even if it masters the trapdoor $t_{w\|j}$ from the data receiver.

4.2 Construction of FS-PEKS

Now we describe the lattice-based FS-PEKS scheme for cloud-assisted IIoT. It consists of the following five polynomial-time algorithms.

Setup: Taking as input a security parameter κ, the system initialization sets the discrete Gaussian distribution χ and security Gaussian parameters $\delta = (\delta_1, \cdots, \delta_\eta)$, $\sigma = (\sigma_1, \cdots, \sigma_\eta)$ for each time period $j = 1, \cdots, \eta$, and the system initialization performs in the following steps.

1) Randomly select a uniformly random vector $\mu \leftarrow \mathbb{Z}_q^n$.
2) Set two secure hash functions $H_1 : \mathbb{Z}_q^{n\times m} \times N \rightarrow \mathbb{Z}_q^{m\times m}$, and $H_2 : \{0, 1\}^{\ell_1} \times N \rightarrow \mathbb{Z}_q^{m\times m}$, where the set $N = \{0, 1, \cdots, \eta\}$, the outputs of H_1 and H_2 are both distributed in $\mathcal{D}_{m\times m}$.
3) Run $\mathsf{TrapGen}(q, n)$ to generate the data receiver's initial public key $A_r \in \mathbb{Z}_q^{n\times m}$ together with the private key $T_r \in \mathbb{Z}_q^{m\times m}$ for $\Lambda_q^{\perp}(A_r)$. Run $\mathsf{TrapGen}(q, n)$

to generate the data sender's initial public key $A_s \in \mathbb{Z}_q^{n\times m}$ together with the private key $T_s \in \mathbb{Z}_q^{m\times m}$ for $\Lambda_q^{\perp}(A_s)$.

Finally, the system initialization outputs the public parameter $\Sigma = (A_s, A_r, \mu, H_1, H_2, \chi, \delta, \sigma)$.

KeyUpdate: Taking as inputs Σ, the current time period j and the private key $T_{r\|i}$ in previous time period i, the data receiver performs as follows.

1) Compute $R_{r\|i} = H_1(A_r\|i)\cdots H_1(A_r\|1) \in \mathbb{Z}_q^{m\times m}$ and $A_{r\|i} = A_r(R_{r\|i})^{-1} \in \mathbb{Z}_q^{n\times m}$.
2) Compute $R_{r\|i\to j} = H_1(A_r\|j)\cdots H_1(A_r\|i+1) \in \mathbb{Z}_q^{m\times m}$ and run $\mathsf{NewBasisDel}(A_{r\|i}, R_{r\|i\to j}, T_{r\|i}, \delta_j)$ to generate $\mathsf{SK}_{r\|j} = T_{r\|j}$ for $\Lambda_q^{\perp}(A_{r\|j})$ as the private key in current time period j, where $A_{r\|j} = A_{r\|i}(R_{r\|i\to j})^{-1} = A_r(R_{r\|j})^{-1} \in \mathbb{Z}_q^{n\times m}$.

PEKS: This PPT algorithm is performed by the data sender. Taking as inputs Σ, the current time period j, the public key $A_{r\|j}$ of the data receiver and keyword $w \in \{0,1\}^{\ell_1}$, the data sender performs as follows.

1) Set a fixed binary string $\gamma_j = (1, 1, \cdots, 1) \in \{1\}^{\ell}$, where ℓ is the security-level of testing in cloud storage, and randomly select a uniform $(n \times \ell)$-dimension matrix $B_j \leftarrow \mathbb{Z}_q^{n\times \ell}$.
2) Select each noise $e_{j1}, e_{j2}, \cdots, e_{j\ell} \leftarrow \mathbb{Z}_q$ according to χ, set $e_j = (e_{j1}, e_{j2}, \cdots, e_{j\ell})$. Select each noise vector $\nu_{j1}, \nu_{j2}, \cdots, \nu_{j\ell} \leftarrow \mathbb{Z}_q^m$ according to χ^m, and set the noise matrix $V_j = (\nu_{j1}, \nu_{j2}, \cdots, \nu_{j\ell}) \in \mathbb{Z}_q^{m\times \ell}$.
3) Compute $\beta_j = H_2(w\|j)$, and generate the PEKS ciphertext $CT_{j1} = \mu^{\top} B_j + e_j + \gamma_j\lfloor q/2\rfloor$, $CT_{j2} = (A_{r\|j}\beta_j^{-1})^{\top} B_j + V_j$.

Finally, the data sender sends $CT_j = (CT_{j1}, CT_{j2})$ to the data receiver as the PEKS ciphertext.

Trapdoor: This PPT algorithm is performed by the data receiver. Taking as inputs Σ, the public-private key pair $(A_{r\|j}, T_{r\|j})$ of the data receiver in current time period j, and a keyword w, the data receiver performs as follows.

1) Compute $\beta_j = H_2(w\|j)$, and run $\mathsf{NewBasisDel}(A_{r\|j}, \beta_j, T_{r\|j}, \delta_j)$ to generate a random short lattice basis $T_{w\|j} \in \mathbb{Z}_q^{m\times m}$ for $\Lambda_q^{\perp}(A_{r\|j}\beta_j^{-1})$.
2) Run $\mathsf{SamplePre}(A_{r\|j}\beta_j^{-1}, T_{w\|j}, \mu, \sigma_j)$ to generate the trapdoor $t_{w\|j} \in \mathbb{Z}_q^m$.

Note that $A_{r\|j}\beta_j^{-1} t_{w\|j} = \mu \in \mathbb{Z}_q^n$, and the trapdoor $t_{w\|j}$ is distributed in $\mathcal{D}_{\Lambda_q^{\mu}(A_{r\|j}\beta_j^{-1}),\sigma_j}$.

Finally, the data receiver sends the trapdoor $t_{w\|j}$ to the cloud server via a secure channel.

Test: This deterministic polynomial-time algorithm is performed by the cloud server. Taking as inputs Σ, the PEKS ciphertext CT_j, a trapdoor $t_{w\|j}$ from the data receiver in current time period j, the cloud server performs as follows.

Compute $\gamma_j = (\gamma_{j1}, \gamma_{j2}, \cdots, \gamma_{j\ell}) \leftarrow CT_{j1} - t_{w\|j}^{\top} CT_{j2}$. For $l = 1, 2, \cdots, \ell$, compare each γ_{jl} and $\lfloor q/2\rfloor$ treating them as integers in $\{1, 2, \cdots, q\} \subset \mathbb{Z}$. Whenever a γ_{jl} satisfies $|\gamma_{jl} - \lfloor q/2\rfloor| \geq \lfloor q/4\rfloor$ in $\mathbb{Z}$, the cloud server aborts. Otherwise $|\gamma_{jl} - \lfloor q/2\rfloor| < \lfloor q/4\rfloor$ in $\mathbb{Z}$, set $\gamma_{jl} \leftarrow 1$, up to $\gamma_{j\ell} \leftarrow 1$. Finally, once the cloud server recovers $\gamma_j = (1, 1, \cdots, 1) \in \{1\}^{\ell}$, it returns 1, which means that the trapdoor $t_{w\|j}$ and the PEKS ciphertext CT_j contain the same keyword w.

4.3 Extension to resist IKGA

The aforementioned FS-PEKS could achieve ciphertext indistinguishability, which will be proved in Section 5 under the assumption that the cloud server is honest-and-trusted, and the trapdoor is transmitted to the cloud server via a secure channel. In order to satisfy the open and flexible requirements of the next generation networks, we consider two more challenging attack situations, described as follows. Firstly, once an outside adversary intercepts the trapdoor, it could execute the exhaustive search. It firstly chooses any keyword w and regenerates the PEKS ciphertext associated with the keyword, then uses the PEKS ciphertext and the trapdoor to run the test algorithm, to identify the keyword which is selected by the data receiver. Secondly, as a misbehaved cloud server, it could also perform IKGA. Particularly, even if the trapdoor is transmitted via a secure channel, the misbehaved cloud server could also execute the exhaustive search as what an outside adversary does, to identify the keyword which is selected by the data receiver.

With the above security challenges, we further extend FS-PEKS to address the above security issues, such that the solution does not rely on a secure channel, but could also resist IKGA. The critical parts of the extension of FS-PEKS are described as follows.

Firstly, in **Setup**, the scheme adds another secure hash function $H_3 : \mathbb{Z}_q^{m\times \ell} \times \{0,1\}^{\ell} \to \mathbb{Z}_q^n$. In **KeyUpdate**, in a similar manner, the scheme flexibly realizes the **KeyUpdate** process from the private key $T_{s\|i}$ of the data sender in time period i to the private key $T_{s\|j}$ in time period j.

Secondly, in **PEKS**, without setting a fixed binary string $\gamma_j = (1, 1, \cdots, 1) \in \{1\}^{\ell}$, the data sender chooses a random binary string $\gamma_j = (\gamma_{j1}, \gamma_{j2}, \cdots, \gamma_{j\ell}) \in \{0,1\}^{\ell}$, thus $CT_{j1} = \mu^{\top} B_j + e_j + (\gamma_{j1}, \gamma_{j2}, \cdots, \gamma_{j\ell})\lfloor q/2\rfloor$, $CT_{j2} = (A_{r\|j}\beta_j^{-1})^{\top} B_j + V_j$ could also be computed as before. Furthermore, the data sender computes $h_j = H_3(CT_{j2}\|\gamma_j) \in \mathbb{Z}_q^n$, and runs $\mathsf{SamplePre}(A_{s\|j}, T_{s\|j}, h_j, \sigma_j)$ to generate $\xi_j \in \mathbb{Z}_q^m$. Eventually, the data sender returns $CT_j = (CT_{j1}, CT_{j2}, \xi_j)$ to the data receiver as the PEKS ciphertext associated with the keyword w.

Finally, in **Test**, the cloud server performs as follows.

1) Compute $\gamma_j = (\gamma_{j1}, \gamma_{j2}, \cdots, \gamma_{j\ell}) \leftarrow CT_{j1} - t_{w\|j}^{\top} CT_{j2}$. For $l = 1, 2, \cdots, \ell$, compare each γ_{jl} and $\lfloor q/2\rfloor$ treating them as integers in $\{1, 2, \cdots, q\} \subset \mathbb{Z}$. If they are close, i.e., if $|\gamma_{jl} - \lfloor q/2\rfloor| < \lfloor q/4\rfloor$ in $\mathbb{Z}$, set $\gamma_{jl} \leftarrow 1$, and otherwise set $\gamma_{jl} \leftarrow 0$. Then output $\gamma_j = (\gamma_{j1}, \gamma_{j2}, \cdots, \gamma_{j\ell}) \in \{0,1\}^{\ell}$.
2) Compute $h_j = H_3(CT_{j2}\|\gamma_j) \in \mathbb{Z}_q^n$ and check whether the equation $A_{s\|j}\xi_j = h_j$ holds, and whether ξ_j is distributed in $\mathcal{D}_{\Lambda_q^{h_j}(A_{s\|j}),\sigma_j}$. If they hold, the cloud server returns 1, otherwise, it returns 0.

5 Correctness and Security

5.1 Correctness of FS-PEKS

Let the data receiver's key pair be $(A_{r\|j}, T_{r\|j})$, the data sender's key pair be $(A_{s\|j}, T_{s\|j})$ in the current time period j. Let w be the keyword contained in CT_j and w' be that in $t_{w'\|j}$.

In **Test**, with the trapdoor $t_{w'\|j}$ of the current time j, the cloud server could easily recover $\gamma'_j = (\gamma'_{j1}, \gamma'_{j2}, \cdots, \gamma'_{j\ell}) \leftarrow CT_{j1} - t^\top_{w'\|j} CT_{j2}$ under the decryption mode of LWE. Now we take into account the following two cases.

1) If $w = w'$, then $\gamma'_j \leftarrow CT_{j1} - t^\top_{w'\|j} CT_{j2} = CT_{j1} - t^\top_{w\|j} CT_{j2} = (\gamma_{j1}, \gamma_{j2}, \cdots, \gamma_{j\ell})\lfloor q/2 \rfloor + e_j - t^\top_{w\|j} V_j$. Here $e_j - t^\top_{w'_j\|j} V_j$ is actually an ℓ dimensional noise row vector. To decrypt correctly, the scheme needs to guarantee that each component of the error vector is less than $q/5$ as discussed in [24]. Thus $\gamma'_j = \gamma_j = (1, 1, \cdots, 1) \in \{1\}^\ell$.
2) If $w \neq w'$, as $\gamma'_j \leftarrow CT_{j1} - t^\top_{w'\|j} CT_{j2} \neq (\gamma_{j1}, \gamma_{j2}, \cdots, \gamma_{j\ell})\lfloor q/2 \rfloor + e_j - t^\top_{w\|j} V_j$, thus the PEKS ciphertext CT_j can be decrypted as $\gamma_j = (1, 1, \cdots, 1) \in \{1\}^\ell$ with a negligible probability.

Therefore, FS-PEKS satisfies correctness consistence, the cloud server could be assured that the PEKS ciphertext $CT_j = (CT_{j1}, CT_{j2})$ and the trapdoor $t_{w'\|j}$ contain the same keyword w. Finally the cloud server responds with the corresponding encrypted industrial data associated with the keyword to the data receiver, such that the data receiver could further decrypt it and get the primitive industrial data shared by the data sender.

5.2 Provable security of FS-PEKS

Now we prove that in FS-PEKS, even if the key exposure occurs, the adversary cannot take advantage of the exposed private key of the data receiver to break the ciphertext indistinguishability in previous time periods.

Theorem 1. *Lattice-based FS-PEKS achieves ciphertext indistinguishability under the adaptively chosen keyword attacks in the random oracle model, provided that the hardness assumption of deciding (Z_q, n, χ)-LWE problem holds.*

Proof: Assume there exists an adversary $\mathcal{A}$ with a non-negligible probability ε breaking ciphertext indistinguishability under the adaptively chosen keyword attacks in the random oracle model, we build a challenger $\mathcal{C}$ with a non-negligible probability solving the hardness assumption of deciding the (Z_q, n, χ)-LWE problem by running the adversary $\mathcal{A}$ as a subroutine.

Setup: $\mathcal{C}$ requests from LWE oracle $\mathcal{O}_L$ for each fresh pair $(u_k, v_{k1}, \cdots, v_{k\ell}) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^\ell$, where $k = 0, 1, \ldots, m$. $\mathcal{C}$ randomly guesses $\jmath = j^*$ as the time period when $\mathcal{A}$ breaks the ciphertext indistinguishability. Simultaneously, to maintain consistency, $\mathcal{C}$ sets two lists L_1, L_2. Let Q be the maximum number of $\mathcal{A}$'s queries to H_2. $\mathcal{C}$ selects a random number $J^* \in [Q]$. Finally, $\mathcal{C}$ prepares a simulated attack environment for $\mathcal{A}$ as follows.

1) $\mathcal{C}$ samples $\jmath + 1$ random matrices $R^*, R^*_1, \cdots, R^*_\jmath \leftarrow \mathcal{D}_{m\times m}$ by running $\mathsf{SampleR}$.
2) $\mathcal{C}$ assembles the random matrix $F^* \leftarrow \mathbb{Z}_q^{n\times m}$ from m of the given LWE samples, by letting the k-th column of F^* be the vector $u_k \in \mathbb{Z}_q^n$ for all $k = 1, \cdots, m$.
3) $\mathcal{C}$ sets $A_r = F^* R^* R^*_\jmath \cdots R^*_1$, the matrix A_r is uniform in $\mathbb{Z}_q^{n\times m}$ since all the $R^*_1, \cdots, R^*_\jmath \in \mathbb{Z}_q^{m\times m}$ are invertible modulo q and F^* is uniform in $\mathbb{Z}_q^{n\times m}$. $\mathcal{C}$ also sets $\mu = u_0 \in \mathbb{Z}_q^n$, chooses a random matrix $A_s \leftarrow \mathbb{Z}_q^{n\times m}$, and finally sets $\Sigma = (A_s, A_r = F^* R^* R^*_\jmath \cdots R^*_1, \mu, H_1, H_2)$.

Finally, $\mathcal{C}$ returns Σ to the adversary $\mathcal{A}$.

We firstly assume that:

- Regardless when $\mathcal{A}$ makes the H_1-query on $A_r\|j$, we assume that it has queried corresponding hash value in time period $i < j$.
- Regardless when $\mathcal{A}$ makes the private key query in certain time period, we assume that it has made all relevant H_1 queries before.

Now, $\mathcal{A}$ performs the following queries.

H_1 query: For each query on $A_r\|j$, where $j = 1, 2, \cdots, \jmath$, $\mathcal{C}$ sets $H_1(A_r\|j) = R^*_j$, and returns R^*_j to $\mathcal{A}$.

If $j = \jmath + 1$, $\mathcal{C}$ computes $A_{r\|j-1} = A_r(R^*_\jmath \cdots R^*_1)^{-1}$, runs $\mathsf{SampleRwithBasis}(A_{r\|j-1})$ to obtain a random $R_j \leftarrow \mathcal{D}_{m\times m}$ and a short random lattice basis $T_{r\|j}$ for $\Lambda_q^\perp(A_{r\|j})$, where $A_{r\|j} = A_{r\|j-1}(R_j)^{-1} \in \mathbb{Z}_q^{n\times m}$, and adds $(A_r\|j, A_{r\|j}, R_j, T_{r\|j})$ into L_1 list, and returns R_j to $\mathcal{A}$.

If $j > \jmath + 1$, $\mathcal{C}$ performs as follows. $\mathcal{C}$ finds $(A_r\|j-1, A_{r\|j-1}, R_{j-1}, T_{r\|j-1})$ from L_1 list, chooses a matrix $R_j \leftarrow \mathcal{D}_{m\times m}$, runs the algorithm $\mathsf{NewBasisDel}(A_{r\|j-1}, R_j, T_{r\|j-1}, \delta_j)$ to generate a random short lattice basis $T_{r\|j}$ for $\Lambda_q^\perp(A_{r\|j})$, where $A_{r\|j} = A_{r\|j-1}(R_j)^{-1}$, adds $(A_r\|j, A_{r\|j}, R_j, T_{r\|j})$ into L_1 list, and finally returns R_j to $\mathcal{A}$.

H_2 query: For the Q_{H_2}-th query, here $Q_{H_2} = 1, 2, \cdots, Q$, $\mathcal{A}$ queries on a distinct $w\|j$ in time period j, $\mathcal{C}$ performs as follows.

If $Q_{H_2} = J^*$ such that $w = w^*$ and $j = j^*$, $\mathcal{C}$ sets $H_2(w\|j) = R^*$, and returns it to $\mathcal{A}$. Otherwise, $\mathcal{C}$ looks into L_1 list to find $(A_r\|j, A_{r\|j}, R_j, T_{r\|j})$ and chooses a matrix $R_{w\|j} \leftarrow \mathcal{D}_{m\times m}$, then runs $\mathsf{NewBasisDel}(A_{r\|j}, R_{w\|j}, T_{r\|j}, \delta_j)$ to generate a random short lattice basis $T_{w\|j}$ for $\Lambda_q^\perp(A_{r\|j}(R_{w\|j})^{-1})$. Finally, $\mathcal{C}$ saves the tuple $(w\|j, A_{r\|j}(R_{w\|j})^{-1}, R_{w\|j}, T_{w\|j})$ in L_2 list, and returns $R_{w\|j}$ to $\mathcal{A}$.

Trapdoor oracle: Upon receiving the query on a keyword w in time period j from $\mathcal{A}$. $\mathcal{C}$ first looks into L_2 list, if $(w\|j, A_{r\|j}(R_{w\|j})^{-1}, R_{w\|j}, T_{w\|j})$ is in L_2 list, $\mathcal{C}$ gets $T_{w\|j}$, then runs $\mathsf{SamplePre}(A_{r\|j}(R_{w\|j})^{-1}, T_{w\|j}, \mu, \sigma_j)$ to generate the trapdoor $t_{w\|j}$, and returns to $\mathcal{A}$.

Break-in phase: This phase models the possibility of key exposure. $\mathcal{A}$ queries on the private key for the data receiver in time period j, with the restriction that the time period $j > j^*$, where $j^* = \jmath$ is the break-in time period. As we assume $\mathcal{A}$ may have made relative queries to H_1 on $A_r\|j$, $\mathcal{C}$ could provide $\mathcal{A}$ with corresponding private key $T_{r\|j}$ as follows.

For the previous time period i query, we note that $i = \jmath + 1$ is the smallest time period when $H_1(A_r\|i) \neq R^*_i$. Since we have assumed that $\mathcal{A}$ would have made H_1 query on $A_r\|i$, we could retrieve the saved tuple $(A_r\|i, A_{r\|i}, R_i, T_{r\|i})$ from L_1 list. We could construct $A_{r\|i} = A_{r\|\jmath+1} = A_r(R^*_1)^{-1} \cdots (R^*_\jmath)^{-1}(H_1(A_r\|\jmath + 1))^{-1}$ and $T_{r\|i}$ is a short random basis for $\Lambda_q^\perp(A_{r\|i})$. $\mathcal{C}$ could compute the low norm matrix $R_{r\|i\to j} = H_1(A_r\|j) \cdots H_1(A_r\|i+1)$ as before. Then $\mathcal{C}$ runs $\mathsf{NewBasisDel}(A_{r\|i}, R_{r\|i\to j}, T_{r\|i}, \delta_j)$ to generate the public key $A_{r\|j} = A_{r\|i}(R_{r\|i\to j})^{-1}$ and a short random lattice basis $T_{r\|j}$ as the private key of the data receiver in time period j, and returns $T_{r\|j}$ to the adversary $\mathcal{A}$.

Challenge phase: $\mathcal{A}$ sends (w_0^*, w_1^*) in time period $\jmath = j^*$ to $\mathcal{C}$, where w_0^* and w_1^* are two challenged keywords, then $\mathcal{C}$ randomly chooses a bit $b \leftarrow \{0,1\}$. If $b = 0$, $\mathcal{C}$ returns a random PEKS ciphertext $CT_\jmath^* = (CT_{\jmath 1}^*, CT_{\jmath 2}^*)$ associated with the keyword w_0^* to the adversary $\mathcal{A}$. Otherwise, $\mathcal{C}$ performs as follows.

1) For each $k = 0, 1, \ldots, m$, $l = 1, \ldots, \ell$, retrieve $v_{kl} \in \mathbb{Z}_q$ from the LWE instance, set $v_0 = (v_{01}, \cdots, v_{0\ell})$, $v^* = (v_1, \cdots, v_m)^\top \in \mathbb{Z}_q^{m\times\ell}$, where each $v_k = (v_{k1}, \cdots, v_{k\ell})^\top \in \mathbb{Z}_q^\ell$, and set $\gamma_\jmath^* = (1, 1, \cdots, 1) \in \{1\}^\ell$.
2) Compute $CT_{\jmath 1}^* = v_0 + \gamma_\jmath^* \lfloor q/2 \rfloor$, and set $CT_{\jmath 2}^* = v^*$.

Finally, $\mathcal{C}$ sends $CT_\jmath^* = (CT_{\jmath 1}^*, CT_{\jmath 2}^*)$ associated with the keyword w_1^* to the adversary $\mathcal{A}$.

Guess. Eventually, $\mathcal{A}$ outputs $b' \leftarrow \{0,1\}$.

The goal of $\mathcal{A}$ is to decide which keyword is used in the challenged $CT_\jmath^*$, as $\mathcal{C}$ could return the PEKS ciphertext which is associated with the keyword w_1^* with probability $1/2$. In fact, it is easy to see that the above challenged forward secure PEKS ciphertext associated with the keyword w_1^* has the correct distribution. According to the setting of the public parameter Σ, recall that $A_r = F^* R^* R_\jmath^* \cdots R_1^*$, thus we have $A_{r\|\jmath}(H_2(w_1^*\|\jmath))^{-1} = A_r(R_1^*)^{-1} \cdots (R_\jmath^*)^{-1}(R^*)^{-1} = F^*$ and $CT_{\jmath 2}^* = v^* = (F^*)^\top B_\jmath^* + V_\jmath^*$ for some random matrix $B_\jmath^* \in \mathbb{Z}_q^{n\times\ell}$ and $V_\jmath^* \in \mathbb{Z}_q^{m\times\ell}$ with Gaussian distribution. Therefore $CT_{\jmath 1}^*, CT_{\jmath 2}^*$ have the correct forms. We consider the case that $\mathcal{A}$ could succeed in guessing that the keyword w_1^* is used in generation $CT_\jmath^*$ with a non-negligible probability ε, where the keyword w_1^* is just the J^*-th H_2 query. It means that $w_1^* = w^*$ and $j = j^* = \jmath$, and this case occurs with probability $1/Q$. Moreover, the challenger $\mathcal{C}$ could successfully guess the break-in time $j^* = \jmath$ with probability $1/\eta$. Therefore, if $\mathcal{A}$ with a non-negligible probability ε could break ciphertext indistinguishability under the adaptively chosen keyword attacks in the random oracle model, then $\mathcal{C}$ has a non-negligible probability at least $\mathsf{Adv}_{\mathcal{A}}^{\mathcal{C}}(\kappa) = \varepsilon/2\eta Q$ in deciding the (Z_q, n, χ)-LWE instance by running $\mathcal{A}$ as a subroutine. □

5.3 Security proof of the resistance to IKGA

Now we prove that in the extension of FS-PEKS, even if the key exposure of the data sender occurs, any adversary cannot forge the searchable ciphertext in previous time periods. Thus, the extension of FS-PEKS could resist IKGA from the misbehaved cloud server.

Theorem 2. *The extension of FS-PEKS achieves IKGA resistance, provided that the hardness assumption of ISIS problem holds.*

Proof: Assume that the misbehaved cloud server, as an adversary $\mathcal{F}$, could perform IKGA by forging the searchable ciphertext in the random oracle model with a non-negligible probability ϵ, we could then build a challenger $\mathcal{C}$ also with a non-negligible probability solving the hardness assumption of ISIS problem by running the adversary $\mathcal{F}$ as a subroutine.

Setup: $\mathcal{C}$ randomly guesses $\jmath = j^*$ as the time period when $\mathcal{F}$ forges the searchable ciphertext, thus $\mathcal{F}$ could further perform IKGA successfully. $\mathcal{C}$ requests an instance of ISIS problem $(U, \vartheta) \in \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^n$, and manages to solve the vector $\xi_\jmath^* \in \mathbb{Z}_q^m$, such that $U\xi_\jmath^* = \vartheta$ and $0 < \|\xi_\jmath^*\| \leq \sigma_\jmath \sqrt{m}$. To maintain consistency, $\mathcal{C}$ sets three lists L_1, L_2, L_3. Let M be the maximum number of $\mathcal{F}$'s queries to H_3. $\mathcal{C}$ selects a random number $I^* \in [M]$. Finally, $\mathcal{C}$ prepares a simulated attack environment for $\mathcal{F}$ as follows.

1) $\mathcal{C}$ samples $\jmath$ random matrices $S_1^*, \cdots, S_\jmath^* \leftarrow \mathcal{D}_{m\times m}$ by running $\mathsf{SampleR}$.
2) $\mathcal{C}$ sets $A_s = US_\jmath^* \cdots S_1^*$, the matrix A_s is uniform in $\mathbb{Z}_q^{n\times m}$ since all the $S_1^*, \cdots, S_\jmath^* \in \mathbb{Z}_q^{m\times m}$ are invertible modulo q, and U is uniform in $\mathbb{Z}_q^{m\times m}$. $\mathcal{C}$ also randomly chooses $\mu \leftarrow \mathbb{Z}_q^n$, chooses a random matrix $A_r \leftarrow \mathbb{Z}_q^{n\times m}$, and finally sets $\Sigma = (A_s = US_\jmath^* \cdots S_1^*, A_r, \mu, H_1, H_2, H_3)$.

Finally, $\mathcal{C}$ returns Σ to the adversary $\mathcal{F}$.

We also assume that:

- Regardless when $\mathcal{F}$ makes the H_1-query on $A_s\|j$, we assume that it has queried corresponding hash value in time period $i < j$.
- Regardless when $\mathcal{F}$ makes the private key query in certain time period, we assume that it has made all relevant H_1 queries before.

Now, $\mathcal{F}$ performs the following queries.

H_1 query: For each query on $A_s\|j$, where $j = 1, 2, \cdots, \jmath$, $\mathcal{C}$ sets $H_1(A_s\|j) = S_j^*$, and returns S_j^* to $\mathcal{F}$.

If $j = \jmath + 1$, $\mathcal{C}$ computes $A_{s\|j-1} = A_s(S_\jmath^* \cdots S_1^*)^{-1}$, runs $\mathsf{SampleRwithBasis}(A_{s\|j-1})$ to obtain a random $S_j \leftarrow \mathcal{D}_{m\times m}$ and a short random lattice basis $T_{s\|j}$ for $\Lambda_q^\perp(A_{s\|j})$, where $A_{s\|j} = A_{s\|j-1}(S_j)^{-1} \in \mathbb{Z}_q^{n\times m}$, and adds $(A_s\|j, A_{s\|j}, S_j, T_{s\|j})$ into L_1 list, and returns S_j to $\mathcal{F}$.

If $j > \jmath + 1$, $\mathcal{C}$ performs as follows. $\mathcal{C}$ finds $(A_s\|j-1, A_{s\|j-1}, S_{j-1}, T_{s\|j-1})$ from L_1 list, chooses a matrix $S_j \leftarrow \mathcal{D}_{m\times m}$, runs the algorithm $\mathsf{NewBasisDel}(A_{s\|j-1}, S_j, T_{s\|j-1}, \delta_j)$ to generate a random short lattice basis $T_{s\|j}$ for $\Lambda_q^\perp(A_{s\|j})$, where $A_{s\|j} = A_{s\|j-1}(S_j)^{-1}$, adds $(A_s\|j, A_{s\|j}, S_j, T_{s\|j})$ into L_1 list, and finally returns S_j to $\mathcal{F}$.

Similarly, $\mathcal{C}$ could also answer the query on $A_r\|j$ from $\mathcal{F}$ in the same approach.

H_2 query: $\mathcal{F}$ queries on a distinct $w\|j$ in time period j, $\mathcal{C}$ also looks into L_1 list to find $(A_r\|j, A_{r\|j}, R_j, T_{r\|j})$ and chooses a matrix $R_{w\|j} \leftarrow \mathcal{D}_{m\times m}$, then runs $\mathsf{NewBasisDel}(A_{r\|j}, R_{w\|j}, T_{r\|j}, \delta_j)$ to generate a random short lattice basis $T_{w\|j}$ for $\Lambda_q^\perp(A_{r\|j}(R_{w\|j})^{-1})$. Finally, $\mathcal{C}$ saves the tuple $(w\|j, A_{r\|j}(R_{w\|j})^{-1}, R_{w\|j}, T_{w\|j})$ in L_2 list, and returns $R_{w\|j}$ to $\mathcal{F}$.

H_3 query: For the M_{H_3}-th query, here $M_{H_3} = 1, 2, \cdots, M$, $\mathcal{F}$ queries on a distinct $CT_{j2}\|\gamma_j$, $\mathcal{C}$ returns h_j to $\mathcal{F}$, if it exists in list L_3. If $M_{H_3} = I^*$, such that $\gamma_j = \gamma_\jmath^*$, and $CT_{j2} = CT_{\jmath 2}^*$ is just the second component of the searchable ciphertext associated with the keyword w^*, under the uniform matrix $B_\jmath^* \leftarrow \mathbb{Z}_q^{n\times m}$, the noise matrix $V_\jmath^* \in \mathbb{Z}_q^{m\times\ell}$, $\mathcal{C}$ adds $(CT_{\jmath 2}^*, \gamma_\jmath^*, \perp, \vartheta)$ to L_3, and returns ϑ to $\mathcal{F}$. Otherwise, $\mathcal{C}$ generates $\xi_j \leftarrow \mathcal{D}_{\mathbb{Z}^m, \sigma_j}$ by running $\mathsf{SampleDom}$ in [24], and computes $h_j = U\xi_j \in \mathbb{Z}_q^n$, then $\mathcal{C}$ adds $(CT_{j2}, \gamma_j, h_j, \xi_j)$ to L_3.

Trapdoor oracle: Upon receiving the query on a keyword w in time period j from $\mathcal{F}$. $\mathcal{C}$ first looks into L_2 list, if $(w\|j, A_{r\|j}(R_{w\|j})^{-1}, R_{w\|j}, T_{w\|j})$ is in L_2 list, $\mathcal{C}$ gets $T_{w\|j}$, and runs $\mathsf{SamplePre}(A_{r\|j}(R_{w\|j})^{-1}, T_{w\|j}, \mu, \sigma_j)$ to generate the trapdoor $t_{w\|j}$, and returns to $\mathcal{F}$.

Searchable ciphertext oracle: $\mathcal{F}$ submits a keyword w and a binary string $\gamma \in \{0,1\}^{\ell}$ in time period j for searchable ciphertext query. $\mathcal{C}$ randomly chooses a uniform matrix $B_j \leftarrow \mathbb{Z}_q^{n\times m}$, a noise matrix $V_j \in \mathbb{Z}_q^{m\times \ell}$, and computes CT_{j1}, CT_{j2} in a normal way. Then $\mathcal{C}$ chooses $\xi_j \leftarrow \mathbb{D}_{\mathbb{Z}^m,\sigma_j}$ by running the algorithm $\mathsf{SampleDom}$. Finally, $\mathcal{C}$ returns $CT_j = (CT_{j1}, CT_{j2}, \xi_j)$ to the adversary $\mathcal{F}$.

Break-in phase: This phase models the possibility of key exposure. $\mathcal{F}$ queries on the private key for the data sender in time period j, with the restriction that the time period $j > j^*$, where $j^* = \jmath$ is the break-in time period. As we assume $\mathcal{F}$ may have made relative queries to H_1 on $A_s\|j$, $\mathcal{C}$ could provide $\mathcal{F}$ with corresponding private key $T_{s\|j}$ as follows.

For the previous time period i query, we note that $i = \jmath + 1$ is the smallest time period when $H_1(A_s\|i) \neq S_i^*$. Since we have assumed that $\mathcal{F}$ would have made H_1 query on $A_s\|i$ as before, we could retrieve the saved tuple $(A_s\|i, A_{s\|i}, S_i, T_{s\|i})$ from L_1 list. We could construct $A_{s\|i} = A_{s\|\jmath+1} = A_s(S_1^*)^{-1}\cdots(S_{\jmath}^*)^{-1}(H_1(A_s\|\jmath{+}1))^{-1}$ and $T_{s\|i}$ is a short random basis for $\Lambda_q^{\perp}(A_{s\|i})$. $\mathcal{C}$ could compute the low norm matrix $S_{s\|i\to j} = H_1(A_s\|j)\cdots H_1(A_s\|i+1)$ as before. Then $\mathcal{C}$ runs $\mathsf{NewBasisDel}(A_{s\|i}, S_{s\|i\to j}, T_{s\|i}, \delta_j)$ to generate the public key $A_{s\|j} = A_{s\|i}(S_{s\|i\to j})^{-1}$ and a short random lattice basis $T_{s\|j}$ as the private key of the data sender in time period j, and returns $T_{s\|j}$ to $\mathcal{F}$.

Forgery phase: In this phase, the adversary $\mathcal{F}$, in the role of the misbehaved cloud server, may try to forge the searchable ciphertext in time period $\jmath = j^*$. More specifically, $\mathcal{F}$ returns $\mathcal{C}$ a forged searchable ciphertext $CT_{\jmath}^* = (CT_{\jmath1}^*, CT_{\jmath2}^*, \xi_{\jmath}^*)$ of $\gamma_{\jmath}^*$ associated with w^*. With the restriction that $\gamma_{\jmath}^*$ associated with $w_{\jmath}^*$ cannot be submitted to the searchable ciphertext oracle.

Note that the adversary $\mathcal{F}$ could query $\mathcal{C}$ to get $(w^*\|\jmath, A_{r\|\jmath}(R_{w^*\|\jmath})^{-1}, R_{w^*\|\jmath}, T_{w^*\|\jmath})$ in L_2 list, $\mathcal{F}$ could further generate $t_{w^*\|\jmath} \leftarrow \mathsf{SamplePre}(A_{r\|\jmath}\beta_{\jmath}^{-1}, T_{w^*\|\jmath}, \mu, \sigma_{\jmath})$.

Now $\mathcal{C}$ recovers $\gamma_{\jmath}^*$ by computing $\gamma_{\jmath}^* \leftarrow CT_{\jmath1}^* - t_{w^*\|\jmath}^{\top} CT_{\jmath2}^*$ under the searchable trapdoor $t_{w^*\|\jmath}$, and outputs $\xi_{\jmath}^*$ as a forged signature of $\gamma_{\jmath}^*$ associated with the keyword w^*. Thus, $\mathcal{C}$ outputs $\xi_{\jmath}^*$ as its answer to the ISIS problem $(U, \vartheta) \in \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^n$. We know that $\mathcal{F}$ could win the game only if $\xi_{\jmath}^*$ is a valid signature of $\gamma_{\jmath}^*$ associated with the keyword w^*, thus we have $A_{s\|\jmath}\xi_{\jmath}^* = H_3(CT_{\jmath2}^*\|\gamma_{\jmath}^*)$, where $0 < \|\xi_{\jmath}^*\| \leq \sigma_{\jmath}\sqrt{m}$. Recall that $A_s = US_{\jmath}^*\cdots S_1^*$, thus we have $A_{s\|\jmath}\xi_{\jmath}^* = A_s(S_1^*)^{-1}\cdots(S_{\jmath}^*)^{-1}\xi_{\jmath}^* = U\xi_{\jmath}^*$. Moreover, note that $\mathcal{C}$ could successfully guess that $H_3(CT_{\jmath2}^*\|\gamma_{\jmath}^*) = \vartheta$ with probability $1/M$. The challenger $\mathcal{C}$ could successfully guess the break-in time $j^* = \jmath$ with probability $1/\eta$. Thus, if $\mathcal{F}$ succeeds in forging a valid searchable ciphertext with a non-negligible probability ϵ, then $\mathcal{F}$ could find a solution $\xi_{\jmath}^*$, such that $U\xi_{\jmath}^* = \vartheta$ and $0 < \|\xi_{\jmath}^*\| \leq \sigma_{\jmath}\sqrt{m}$, with a non-negligible probability $\mathsf{Adv}_{\mathcal{F}}^{\mathcal{C}}(\kappa) = \epsilon/\eta M$, which contradicts to the hardness assumption of ISIS problem. Consequently, the extension of FS-PEKS achieves IKGA resistance from the misbehaved cloud server. □

6 Performance Evaluation

In this section, we firstly provide performance comparison among FS-PEKS and existing PEKS schemes [9], [11], [12], [33], including the communication overhead and computational costs. Then, we further compare the performance of the extension of FS-PEKS achieving IKGA with the other PEKS schemes [10], [13], [14], which also have such security property. All the experiments are run on a laptop with Window 10 system with an Intel Core 2 i5 CPU and 8GB DDR 3 of RAM. We utilize C language and MIRACL Library version 5.6.1. We employ an MNT curve with base field size 159 bits and embedding degree 6. As the lattice-based algorithms are based on parameters n, m, q, to achieve the security of the q-module integer lattices, the parameters need to satisfy $m \geq 2n\lceil \log q \rceil$. To compare with existing schemes, here we give an instance for FS-PEKS and its extended scheme, and set the security level $\ell = 10$. All the results of experiments are represented 30 trials on average. Now we firstly evaluate the performance comparison among FS-PEKS and existing PEKS schemes [9], [11], [12], [33]. Let $|\mathbb{G}_1|$, $|\mathbb{G}_T|$ denote the bit size of an element in group $\mathbb{G}_1$, $\mathbb{G}_T$, respectively. Let $|p|$, $|q|$ denote the bit size of an element in $\mathbb{Z}_p$, $\mathbb{Z}_q$, respectively. The notations of operations are given in Table 1.

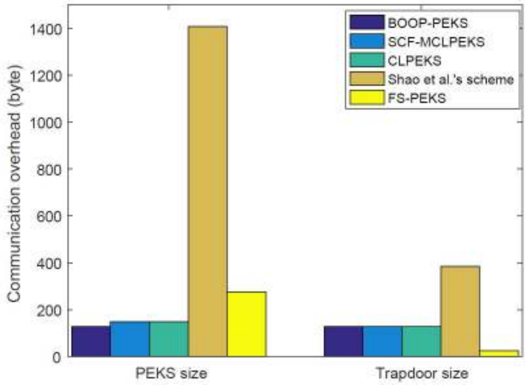

Fig. 2. Communication overhead comparison

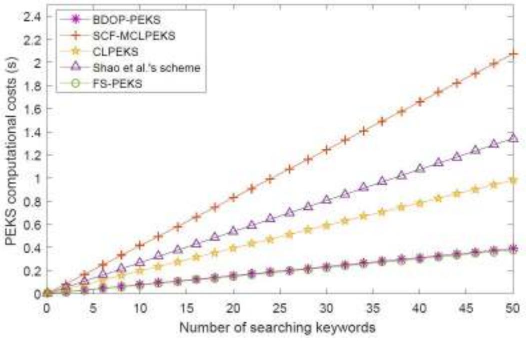

Fig. 3. PEKS computational costs comparison

Table 2 lists the communication overhead in detail. As shown in Fig. 2, the trapdoor size of FS-PEKS is much less than that of other schemes. Although the PEKS size of FS-PEKS is slightly bigger than that of [9], [11], [12], only

TABLE 1
Notations of Operations

Notations	Operations
T_{Pa}	The running time of a bilinear pairing
T_{Ex}	The running time of an modular exponentiation
T_{Ad}	The running time of a point addition
T_{Mu}	The running time of a scalar multiplication
T_{mu}	The running time of a general multiplication
T_{Ha}	The running time of a hash-to-point
T_{ha}	The running time of a general hash function

TABLE 2
Communication Overhead

Schemes	PEKS size	Trapdoor size
BDOP-PEKS [9]	$\vert\mathbb{G}_1\vert + \ell$	$\vert\mathbb{G}_1\vert$
SCF-MCLPEKS [11]	$\vert\mathbb{G}_1\vert + \vert p\vert$	$\vert\mathbb{G}_1\vert$
CLPEKS [12]	$\vert\mathbb{G}_1\vert + \vert p\vert$	$\vert\mathbb{G}_1\vert$
Shao et al.'s scheme [33]	$5\vert\mathbb{G}_1\vert + 3\vert\mathbb{G}_T\vert$	$3\vert\mathbb{G}_1\vert$
FS-PEKS	$(\ell + m\ell)\vert q\vert$	$m\vert q\vert$

TABLE 3
Computational Costs

Schemes	PEKS computational costs	Testing time
BDOP-PEKS [9]	$T_{Pa} + 2T_{Ex} + 2T_{ha}$	$T_{Pa} + T_{ha}$
SCF-MCLPEKS [11]	$3T_{Pa} + T_{Ad} + 4T_{Mu} + 3T_{Ha} + T_{ha} + 2T_{mu}$	$T_{Pa} + 2T_{Ad} + T_{Mu} + 2T_{Ha} + T_{ha}$
CLPEKS [12]	$T_{Pa} + 2T_{Ad} + 4T_{Mu} + T_{Ha} + 3T_{ha}$	$T_{Pa} + T_{ha}$
Shao et al.'s scheme [33]	$3T_{Pa} + 9T_{Ex} + 3T_{ha}$	$4T_{Pa} + 5T_{Ex} + T_{ha}$
FS-PEKS	$T_{ha} + (n\ell + nm^2 + nm\ell)T_{mu}$	$m\ell T_{mu}$

TABLE 4
Testing Time Comparison

Schemes	Testing time
PAEKS [10]	$2T_{Pa} + T_{mu}$
CLPAEKS [13]	$2T_{Pa} + 2T_{Ad} + 2T_{Mu} + 2T_{ha} + T_{mu}$
DS-PAEKS [14]	$7T_{Ex} + 3T_{mu}$
Extended scheme	$T_{ha} + (m\ell + nm)T_{mu}$

FS-PEKS can achieve forward security. Thus, FS-PEKS just adds reasonable PEKS size overhead. Moreover, we give an analysis of computational costs among FS-PEKS and other existing schemes [9], [11], [12], [33] in Table 3.

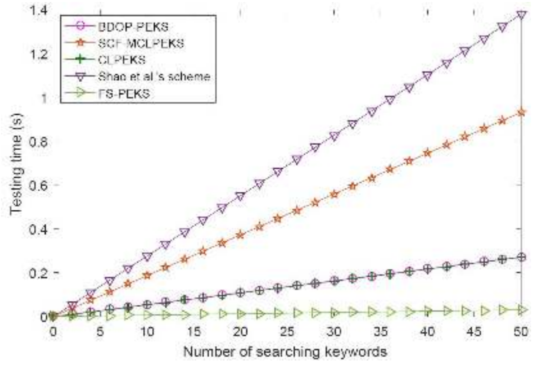

Fig. 4. Testing time comparison

The results of computational costs, including detailed PEKS computational costs and testing time, are shown in Fig. 3 and Fig. 4, respectively. According to the computational costs comparison, with the increasing number of searching keywords, FS-PEKS achieves almost the same as [9] in PEKS computational costs, and is much more efficient than that of [11], [12], [33]. FS-PEKS is much more efficient

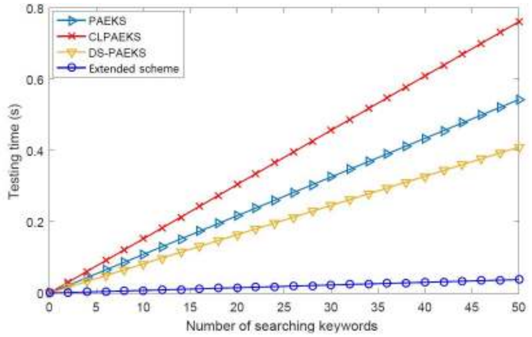

Fig. 5. The comparison of testing time

than all the existing schemes in testing time, and only FS-PEKS can achieve quantum resistance. Now we further compare the extension of FS-PEKS with existing PEKS schemes [10], [13], [14], all of which can resist IKGA. In particular, here we evaluate the performance of the end-to-end delay from the cloud server to the data receiver, and focus on the comparison of testing time in Table 4. The experiment results of testing time with the increasing number of searching keywords are illustrated in Fig. 5. Similar to the computational costs in Fig. 4, the extended scheme is also more lightweight than other existing schemes. This is mainly because FS-PEKS and its extended scheme are built on lattice, which only needs simple addition and multiplication operations over a moderate module, without time-consuming bilinear pairing and modular exponentiation operations. In addition, only our extended scheme could achieve forward security, quantum attacks and IKGA resistance simultaneously.

7 Related Work

With the rapid development of cloud computing, cloud storage technologies have become increasingly prevalent. Cloud storage data auditing services [34], [35] are employed to ensure the integrity of data. While data encryption services are used to ensure the data confidentiality [36]. To achieve the goal of searching over encrypted data outsourced to the cloud server without leaking any information about the messages shared by the data sender, searchable encryption is one of the essential approaches. Searchable encryption is divided into symmetry searchable encryption and asymmetric searchable encryption. The symmetry searchable encryption (SSE) scheme was firstly introduced in [37]. Following that, many SSE schemes with distinct features have been proposed in the literature [38], [39], [40], [41]. Although these SSE schemes are with relatively high computational efficiency, they are only suitable for a single user model, but could not be well applied in multi-user model.

On the contrary, public-key encryption with keyword search (PEKS) scheme contributes to key distribution and data sharing with multiple users. The first PEKS scheme was proposed by Boneh et al. [9], which enables a cloud server to search on a collection of encrypted data with a trapdoor from a data receiver. Following Boneh et al.'s work [9], many PEKS variants [42], [43], [44], [45], [46] with different features were proposed. However, Baek et al. [27] pointed out that the scheme in [9] needs to establish a secure channel for transmitting the trapdoor. To tackle with this issue, Baek et al. [27] constructed a PEKS scheme without secure channel, and the trapdoor could be transmitted via a public channel. Byun et al. [47] have shown that these PEKS schemes are vulnerable to off-line keyword guessing attack due to the low-entropy keyword space. After that, Rhee et al. [28] proposed a designated tester PEKS scheme removing the secure channel, where only the designated server could test whether the ciphertext of the keyword matches the trapdoor generated by the data receiver. Rhee et al. [48] gave a generic construction of designated tester PEKS scheme, it not only achieves ciphertext indistinguishability but also trapdoor security, such that any outside adversary cannot distinguish whether two trapdoors are generated by the same keyword. Moreover, Fang et al. [49] constructed a PEKS scheme against keyword guessing attacks under the standard model. Ma et al. [50] proposed a public-key encryption with equality test scheme supporting flexible authorization. In addition, to avoid key management and the key escrow problem in cloud-assisted IIoT, some certificateless PEKS schemes for cloud-assisted IIoT have also been proposed [12], [13].

Although some feasible techniques [28], [48], [49] have been proposed to resist KGA, we also observe that a misbehaved cloud server could perform insider keyword guessing attacks (IKGA). In order to resist such attacks, a PEKS scheme [29] with fuzzy keyword search has been constructed. However, heavy communication overhead and computational costs are also introduced to the data receiver. Chen et al. [14] proposed a dual-server PEKS scheme, which utilizes two servers and requires the two servers do not collude. Sun et al. [51] proposed a PEKS scheme with resistance against IKGA by using indistinguishability obfuscation [52], which is inefficient to execute, making it impractical. More recently, Huang et al. [10] proposed an authenticated PEKS scheme succeeding in resisting IKGA. The certificateless authenticated PEKS scheme for cloud-assisted IIoT in [13] could also achieve KGA resistance against the misbehaved cloud server.

With the explosive use of mobile intelligent terminal devices with limited key protection, the key exposure of the individual for the search on encrypted data stored in the cloud server might occur, and the adversary could capture the private key and break the forward security. Recently, to address this security issue, some SSE schemes supporting forward security in the literature [26], [53], [54], [55], [56] have emerged. However, PEKS schemes achieving forward security have remained much less in previous research. To the best of our knowledge, there is only one public-key broadcast searchable encryption [57] with this security property.

Nevertheless, due to the security proof in [15], the conventional public key cryptographic algorithms will be insecure once quantum computers become realistic, thus the security of those PEKS schemes mentioned above will also be threatened. Fortunately, lattice-based cryptography [22] has been a promising technique for resisting quantum attacks, since it holds very strong security guarantee based on worst-case hardness. In this paper, we construct lattice-based PEKS scheme by leveraging a hierarchy identity-based encryption based on the hardness assumption of deciding LWE problem [23] and the idea of the forward secure lattice-based signature [58]. The LWE problem was introduced in Regev's seminal work [25]. As a novel hardness assumption, it lead to a series of different LWE-based cryptosystems and applications [59], [60], which by extension are also post-quantum secure.

8 Conclusions and Future Work

In this paper, we have proposed a forward secure public-key encryption with keyword search (FS-PEKS) scheme from lattice, which is secure against the quantum attacks. The proposed FS-PEKS is provable secure and achieves forward security, making it suitable for secure searching over encrypted industrial data in cloud-assisted IIoT. Furthermore, we have extended FS-PEKS to resist IKGA from the misbehaved cloud server. Compared with existing PEKS schemes, our proposed solution has higher computational efficiency at the cloud server side and lower communication overhead in trapdoor size.

For the future work, we plan to investigate how to construct an efficient lattice-based key-exposure resilient PEKS that not only protects against past keys in the event of key exposure, but also future keys. Moreover, we will investigate how to utilize novel lattice-based cryptographic technologies to enhance cloud-assisted IIoT in terms of security, performance, and functionality.

References

[1] K. Ashton, "That internet of things thing," *RFID Journal*, vol. 22, no. 7, pp. 97-114, 2009.

[2] J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645-1660, 2013.

[3] A. R. Sadeghi, C. Wachsmann, and M. Waidner, "Security and privacy challenges in industrial internet of things," in *Proceedings of DAC*. IEEE, 2015, pp. 1-6.

[4] K. K. R. Choo, "Cloud computing: Challenges and future directions," *Trends and Issues in Crime and Criminal Justice*, vol. 400, pp. 1-6, 2010.

[5] M. S. Hossain and G. Muhammad, "Cloud-assisted industrial internet of things (iiot)-enabled framework for health monitoring," *Computer Networks*, vol. 101, no. 4, pp. 192?02, 2016.

[6] C. Esposito, A. Castiglione, B. Martini, and K. K. R. Choo, "Cloud manufacturing: Security, privacy, and forensic concerns," *IEEE Cloud Computing*, vol. 3, pp. 16-22, 2016.

[7] J. Li, Z. Liu, X. Chen, X. Tan, D. S. Wong, "L-EncDB: A lightweight framework for privacy-preserving data queries in cloud computing," *Knowledge-based Systems*, vol. 79, pp. 18-26, 2015.

[8] Y. Zhang, C. Xu, H. Li, K. Yang, J. Zhou, and X. Lin, "HealthDep: An efficient and secure deduplication scheme for cloud-assisted eHealth systems," *IEEE Transactions on Industrial Informatics*, to appear, doi: 10.1109/TII.2018.2832251.

[9] D. Boneh, G. D. Crescenzo, R. Ostrovsky, G. Persiano, "Public key with keyword search," in *Proceedings of EUROCRYPT*. Springer, 2004, pp.506-522.

[10] Q. Huang, H. Li, "An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks," *Information Sciences*, 403-404, pp. 1-14, 2017.

[11] M. Ma, D. He, N. Kumar, K. K. R. Choo, J. Chen, "Certificateless searchable public Key encryption scheme for Industrial Internet of Things," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 2, pp. 759-767, 2017.

[12] M. Ma, D. He, M. K. Khan, and J. Chen, "Certificateless searchable public key encryption scheme for mobile healthcare system," *Computers and Electrical Engineering*, vol. 65, pp. 413-424, 2017.

[13] D. He, M. Ma, S. Zeadall, N. Kumar, K. Liang, "Certificateless public key authenticated encryption with keyword search for Industrial Internet of Things," *IEEE Transactions on Industrial Informatics*, PP(99): 1-1, 2017.

[14] R. Chen, Y. Mu, G. Yang, F. Guo, X. Wang, "Dual-server public-key encryption with keyword search for secure cloud storage," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 4, pp. 789-798, 2016.

[15] P. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1484-1509, 1997.

[16] T. D. Ladd, F. Jelezko, R. Laflamme, Y. Nakamura, C. Monroe, J. L. O. Brien, "Quantum computers," *Nature*, vol. 464, no. 7285, pp. 45-53, 2012.

[17] Bloomberg, "IBM makes breakthrough in race to commercialize quantum computers," https://m.cacm.acm.org/, 2017.

[18] Microsoft, "Key management," Available: http://technet.microsoft.com/en-us/library/cc961626.aspx, 2014.

[19] U.S. Computer Emergency Readiness Team, "OpenSSL 'heartbleed' vulnerability (CVE-2014-0160)," https://www.uscert.gov/ncas/alerts/TA14-098A, April 2014.

[20] J. Li, C. Qin, P. P. C. Lee, J. Li, "Rekeying for encrypted deduplication storage," in *Proceedings of DSN*. IEEE, 2016, pp. 618-629.

[21] W. C. Yau, S. H. Heng, B. M. Goi, "Off-line keyword guessing attacks on recent public key encryption with keyword search schemes," in *Proceedings of ATC*. Springer, 2008, pp. 100-105.

[22] D. Micciancio, O. Regev, "Lattice-based cryptography," in *Proceedings of CRYPTO*. Springer, 2006, pp. 131-141.

[23] S. Agrawal, D. Boneh, X. Boyen, "Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE," in *Proceedings of CRYPTO*. Springer, 2010, pp. 98-115.

[24] C. Gentry, C. Peikert, V. Vaikuntanathan, "Trapdoors for hard lattices and new cryptographic constructions," in *Proceedings of TOC*. ACM, 2008, pp. 197-206.

[25] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *Journal of the ACM*, vol. 56, no. 6, pp. 1-40, 2009.

[26] Y. Zhang, J. Katz, C. Papamanthou, "All your queries are belong to us: The power of file-injection attacks on searchable encryption," in *Proceedings of USENIX Security Symposium*, 2006, pp. 707-720.

[27] J. Baek, R. Safavi-Naini, W. Susilo, "Public key encryption with keyword search revisited," in *Proceedings of ACISP*. Springer, 2006, pp. 1249-1259.

[28] H. S. Rhee, J. H. Park, W. Susilo, D. H. Lee, "Trapdoor security in a searchable public-key encryption scheme with a designated tester," *Journal of Systems and Software*, vol. 83, no. 5, pp. 763-771, 2010.

[29] P. Xu, H. Jin, Q. Wu, and W. Wang, "Public-key encryption with fuzzy keyword search: A provably secure scheme under keyword guessing attack," *IEEE Transactions on Computers*, vol. 62, no. 11, pp. 2266-2277, 2013.

[30] M. Ajtai, "Generating hard instances of the short basis problem," in *Proceedings of ALP*. Springer, 1999, pp. 1-9.

[31] C. Peikert, "Public-key cryptosystems from the worst-case shortest vector problem," in *Proceedings of TOC*. ACM, 2009, pp. 333-342.

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

[33] Z. Y. Shao, B. Yang, "On security against the server in designated tester public key encryption with keyword search," *Information Processing Letters*, vol. 115, no. 12, pp. 957-961, 2015.

[34] Y. Zhang, C. Xu, X. Liang, H. Li, Y. Mu, X. Zhang, "Efficient public verification of data integrity for cloud storage systems from indistinguishability obfuscation," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 3, pp. 676-688, 2017.

[35] X. Zhang, H. Wang, C. Xu, "Identity-based key-exposure resilient cloud storage public auditing scheme from lattices," *Information Sciences*, vol. 472, pp. 223-234, 2019.

[36] X. Liu, B. Qin, R. H. Deng, R. Lu, J. Ma, "A privacy-preserving outsourced functional computation framework across large-scale multiple encrypted domains," *IEEE Transactions on Computers*, vol. 65, no. 12, pp. 3567-3579, 2016.

[37] D. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proceedings of IEEE Security and Privacy*. IEEE, 2000, pp. 44-55.

[38] F. Hahn and F. Kerschbaum, "Searchable encryption with secure and efficient updates," in *Proceedings of ACM CCS*. ACM, 2014, pp. 310-320.

[39] 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.

[40] H. Li, Y. Yang, T. H. Luan, X. Liang, L. Zhou, and X. Shen, "Enabling fine-grained multi-keyword search supporting classified sub-dictionaries over encrypted cloud data," *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 3, pp. 312-325, 2016.

[41] Z. Xia, X. Wang, X. Sun, and Q. Wang, "A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 2, pp. 340-352, 2016.

[42] K. Liang, W. Susilo, "Searchable attribute-based mechanism with efficient data sharing for secure cloud storage," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 9, pp. 1981-1992, 2015.

[43] R. Chen, Y. Mu, G. Yang, F. Guo, X. Huang, "Server-aided public key encryption with keyword search," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 12, pp. 2833-2842, 2016.

[44] K. Yang, K. Zhang, X. Jia, M. A. Hasan, and X. Shen, "Privacy preserving attribute-keyword based data publish-subscribe service on cloud platforms," *Information Sciences*, vol. 387, pp. 116-131, 2017.

[45] Y. Miao, J. Ma, X. Liu, X. Li, Z. Liu, H. Li, "Practical attribute-based multi-keyword search scheme in mobile crowdsourcing," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 3008-3018, 2018.

[46] Y. Miao, J. Ma, X. Liu, J. Weng, H. W. Li, H. Li, "Lightweight fine-grained search over encrypted data in fog computing," *IEEE Transactions on Services Computing*, http://doi.org/10.1109/TSC.2018.2823309, 2018.

[47] J. Byun, H. Rhee, H.-A. Park, and D. Lee, "Off-line keyword guessing attacks on recent keyword search schemes over encrypted data," *Secure Data Management*, pp. 75-83, 2006.

[48] H. S. Rhee, J. H. Park, and D. H. Lee, "Generic construction of designed tester public-key encryption with keyword search," *Information Sciences*, vol. 205, pp. 93-109, 2012.

[49] L. Fang, W. Susilo, C. Ge, and J. Wang, "Public key encryption with keyword search secure against keyword guessing attacks without random oracle," *Information Sciences*, vol. 238, pp. 221-241, 2013.

[50] S. Ma, Q. Huang, M. Zhang, and B. Yang, "Efficient public key encryption with equality test supporting flexible authorization," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 3, pp. 458-470, 2015.

[51] L. Sun, C. Xu, M. Zhang, K. Chen, and H. Li, "Secure searchable public key encryption against insider keyword guessing attacks from indistinguishability obfuscation," *Science China Information Sciences*, vol. 61, no. 3, p. 038106, 2018.

[52] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters, "Candidate indistinguishability obfuscation and functional encryption for all circuits," *SIAM Journal on Computing*, vol. 45, no. 3, pp. 882-929, 2016.

[53] R. Bost, "$\Sigma o\phi o\varsigma$-Forward secure searchable encryption," in *Proceedings of CCS*. ACM, 2016, pp. 1143-1154.

[54] R. W. Lai, S. S. Chow, "Forward-secure searchable encryption on labeled bipartite graphs," in *Proceedings of ACNS*. Springer, 2017, pp. 478-497.

[55] K. S. Kim, M. Kim, D. Lee, J. H. Park, W. H. Kim, "Forward secure dynamic searchable symmetric encryption with efficient updates," in *Proceedings of SIGSAC*. ACM, 2017, pp. 1449-1463.

[56] X. Song, C. Dong, D. Yuan D, Q. Xu, M. Zhao, "Forward private searchable symmetric encryption with optimized I/O efficiency," *IEEE Transactions on Dependable and Secure Computing*, PP(99): 1-1, 2018.

[57] N. Attrapadung, J. Furukawa, H. Imai, "Forward-secure and searchable broadcast encryption with short ciphertexts and private keys," in *Proceedings of CRYPTO*. Springer, 2006, pp. 161-177.

[58] X. Zhang, C. Xu, C. Jin, R. Xie, "Efficient forward secure identity-based shorter signature from lattice," *Computers and Electrical Engineering*, vol. 40, no. 6, pp. 1963-1971, 2014.

[59] X. Zhang, C. Xu, C. Jin, R. Xie, J. Zhao, "Efficient fully homomorphic encrypti on with an extension to a threshold encryption scheme," *Future Generation Computer Systems*, vol. 36, no. 7, 180-186, 2014.

[60] Z. Brakerski1, R. Perlman, "Lattice-based fully dynamic multi-key FHE with short ciphertexts," in *Proceedings of CRYPTO*. Springer, 2016, pp. 190-213, 2016.

Xiaojun Zhang received the B.Sc. degree in mathematics and applied mathematics from Hebei Normal University, Shijiazhuang, China, in 2009, the M.Sc. degree in pure mathematics from Guangxi University, Nanning, China, in 2012, and the Ph.D. degree in information security from the University of Electronic Science Technology of China (UESTC), Chengdu, China, in 2015. He is a Lecturer in the School of Computer Science, Southwest Petroleum University, Chengdu, China. He also works as a Postdoctoral Fellow in University of Electronic Science Technology of China from 2016. He is a research scholar in the School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore. His research interests include cryptography, network security, and cloud computing security.

Chunxiang Xu received her B.Sc., M.Sc. and Ph.D. degrees at Xidian University, in 1985, 1988 and 2004 respectively, 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). She is a member of the IEEE.

Huaxiong Wang received the PhD degree in mathematics from the University of Haifa, Israel, in 1996 and the PhD degree in computer science from the University of Wollongong, Australia, in 2001. He joined Nanyang Technological University in 2006 and is currently an associate professor in the Division of Mathematical Sciences. He is also an honorary fellow at Macquarie University, Australia.His research interests include cryptography, information security, coding theory, combinatorics, and theoretical computer science. He has been on the editorial board of three international journals: Designs, Codes and Cryptography(2006-2011), the Journal of Communications (JCM), and Journal of Communications and Networks. He was the program cochair of Ninth Australasian Conference on Information Security and Privacy (ACISP 04) in 2004 and Fourth International Conference on Cryptology and Network Security (CANS 05) in 2005, and has served in the program committee for more than 70 international conferences. He received the inaugural Award of Best Research Contribution from the Computer Science Association of Australasia in 2004.

Yuan Zhang received his B.Sc. degree in University of Electronic Science Technology of China (UESTC) in 2013, P.R.China. He is currently a Ph.D candidate in School of Computer Science and Engineering at University of Electronic Science Technology of China. His research interests are cryptography, network security, and cloud computing security. He is a student member of the IEEE.

Shixiong Wang received the B.Sc. degree in mathematics in 2013 from Tsinghua University, Beijing, China, and the M.Sc. degree in mathematics in 2015 from National University of Defense Technology, Changsha, China. He is now a Ph.D. student with the College of Computer, National University of Defense Technology. His research fields include cryptography and coding theory.