Lattice-based proxy-oriented identity-based encryption with keyword search for cloud storage

Xiaojun Zhang[a,b,c,*], Yao Tang[a], Huaxiong Wang[b], Chunxiang Xu[c], Yinbin Miao[b,d], Hang Cheng[b,e]

[a]*School of Computer Science, Southwest Petroleum University, Chengdu 610500, China*
[b]*School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore*
[c]*Center for Cyber Security, School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China*
[d]*School of Cyber Engineering, Xidian University, Xi'an 710071, China*
[e]*College of Mathematics and Computer Science, Fuzhou University, Fu'zhou 350108, China*

Abstract

Public-key encryption with keyword search (PEKS) enables users to search over encrypted data and retrieve target data efficiently. However, most of existing PEKS schemes are vulnerable to adversaries equipped with quantum computers in the near future, and even incur complex certificate management procedures due to the public key infrastructure (PKI). To this end, we propose a proxy-oriented identity-based encryption with keyword search scheme (PO-IBEKS) from lattices for cloud storage, which is post-quantum secure. In PO-IBEKS, an original data owner authorizes a proxy to encrypt sensitive data as well as corresponding keywords and upload them to clouds, which alleviates the data processing burden on the original data owner. Besides, PO-IBEKS can resist inside keyword guessing attacks (IKGA) from misbehaved cloud servers by integrating the learning with errors (LWE) encryption and preimage sampleable function. Each entity in PO-IBEKS is identified with her/his recognizable information, thereby eliminating managing certificates. Formal security analysis proves that PO-IBEKS can achieve ciphertext indistinguishability, existential unforgeability, and delegation security. Experimental results demonstrate PO-IBEKS is much more practical

*Corresponding author: School of Computer Science, Southwest Petroleum University, Chengdu 610500, China; E-mail: zhangxjdzkd2012@163.com

compared with existing schemes.

Keywords: proxy-oriented, identity-based encryption, keyword search, lattices, post-quantum secure, inside keyword guessing attacks

1. Introduction

With the sharp increase in massive data, cloud computing has been integrated into various types of network technologies to improve massive data processing and management capabilities [1]. In cloud computing, one of the most significant services is cloud storage [2], which possesses the advantages of nearly unlimited storage space, and enables users to remotely upload their data to the cloud storage server. Besides, cloud storage services contribute to kernel data sharing among users with no geographic constraints, which can achieve flexible access anywhere.

Although cloud storage services have brought about a great many benefits to users, some serious security concerns in data outsourcing have emerged [3]. Without the physical control of the data stored in the remote cloud server, users would not fully trust the cloud server. As the cloud server may be malicious or corrupted, the outsourced data may be eavesdropped, tampered, or deleted, thereby undermining the data confidentiality, integrity, and reliability. From the perspective of users, the most important security issue is data confidentiality, especially for those sensitive data which need to be encrypted before being uploaded to the cloud server [4, 5]. Simultaneously, original data owners also want to share encrypted data with other data receivers. Hence, the issue how to search over the ciphertexts and retrieve target data in the cloud server efficiently has been raised.

Public-key encryption with keyword search (PEKS) [6] is a cryptographic primitive that integrates keyword search functionality into public key encryption, which can achieve the goals of searching and retrieving target data efficiently. However, most of existing PEKS schemes are designed on conventional cryptographic hardness assumptions, which may be broken in the near future according to the research on quantum computers [7]. The fact about some breakthrough research results [8, 9] in recent years indicates that quantum computers would be probably constructed, which makes post-quantum secure PEKS schemes more critical than ever before.

We also observe that existing PEKS schemes lack a controlled way of delegatable outsourcing. In some scenarios, original data owners may only

have limited computation resources, or have no ability to access the Internet flexibly. For instance, as a chronic disease patient, she/he needs long-term treatments. During this period of treatments, continuous recording health status of the patient with medical sensor devices will generate massive medical data, among which, some medical data are critical and sensitive. As the patient cannot flexibly process these massive medical data in real time, on behalf of the patient, a trusted proxy (e.g., medical worker) needs to be authorized to encrypt these sensitive medical data as well as corresponding keywords, and upload them to the cloud server associated with medical information systems. Thus, it enables the patient to share these sensitive medical data with data receivers (e.g., chief doctors) for further precision analysis and diagnosis. Consequently, constructing an efficient proxy-oriented PEKS scheme in cloud storage systems is indispensable [10, 11], especially for those original data owners with constrained data processing capability.

As far as we are concerned, a majority of existing PEKS schemes are designed on certificate-based systems [12, 13], which incur complex certificate management procedures in public key infrastructure (PKI), including time-consuming certificate generation, certificate storage, certificate update, and certificate revocation. Additionally, some drawbacks in the security procedures of various certificate authorities have also jeopardized trust in the entire PKI, which might hinder the secure deployment of PEKS in cloud storage in practice. While an identity-based cryptographic system, first introduced by Shamir [14], can eliminate the establishment of the PKI. In such a system, a trusted key generator center (KGC) can generate the private key according to any known information of an individual, e.g., telephone number, email or IP address. Hence, it can remove the need for explicit certification and all associated costs, making an identity-based cryptographic system particularly appealing.

Along with the aforementioned hindrances in the deployment of PEKS in cloud storage, as pointed out in Huang et al.'s work [12], due to the low entropy of keywords, most of existing PEKS schemes are vulnerable to inside keyword guessing attacks (IKGA) from misbehaved cloud servers. In particular, to perform such IKGA, a misbehaved cloud server encrypts all keywords under a data receiver's public key by exhausting a small keyword space off-line. With a searchable trapdoor from the data receiver, the misbehaved cloud server can identify the ciphertext which matches the targeted trapdoor exhaustively, and recover the keyword hidden in the trapdoor to violate data privacy eventually.

To address the above issues, in this paper we propose a proxy-oriented identity-based encryption with keyword search scheme from lattice assumptions for cloud storage, providing a post-quantum secure promise. Lattice-based cryptography [16] inherently owns distinct advantages, it enjoys very strong security proofs based on worst-case hardness, efficient implementations, and has been considered as the best promising post-quantum cryptography. Specifically, the contributions of this work are elaborated as follows.

- We propose an efficient proxy-oriented identity-based encryption with keyword search scheme, called PO-IBEKS. PO-IBEKS is constructed by leveraging an identity-based encryption based on the hardness assumption of deciding learning with errors (LWE) problem [17], and is secure against quantum computers attacks. Moreover, we integrate the LWE encryption and preimage sampleable function [17] into PO-IBEKS to resist inside keyword guessing attacks (IKGA) from misbehaved cloud servers.

- PO-IBEKS enables an original data owner to authorize a trusted proxy to process the kernel data by generating the signed warrant. Once validating the signed warrant, the proxy can further encrypt the kernel data as well as corresponding keywords, and upload the ciphertext to the cloud server. Since the warrant includes the relative rights and information of an original data owner and a proxy, any unauthorized one cannot encrypt the data and outsource them to the cloud server on behalf of the original data owner.

- We provide provable security of PO-IBEKS, including ciphertext indistinguishability, extensional unforgeability, and delegation security. We conduct a comprehensive performance evaluation, compared with existing schemes, PO-IBEKS is much more efficient in terms of communication overhead and computational costs. Specifically, in the test process, without needing much more time-consuming cryptographic operations, such as bilinear pairing and modular exponentiation operations, the cloud server only needs to execute simple addition and multiplication operations over a moderate module. Hence, PO-IBEKS dramatically reduces the end-to-end computation delay from the cloud server to the data receiver, making it quite practical for post-quantum secure cloud storage systems.

The remainder of this paper is organized as follows. In Section 2, we review the related work. In Section 3, we introduce preliminaries, including lattice-based cryptography, system model of PO-IBEKS, threat model, and design goals. In Section 4, we propose PO-IBEKS. In Section 5, we provide the security proof of PO-IBEKS. In Section 6, we conduct the performance evaluation. In Section 7, we draw the conclusions and future work.

2. Related work

Nowadays, cloud storage service plays an important role in data outsourcing. Simultaneously, many security concerns for users have emerged [18, 19, 20, 21, 22], such as data integrity, data reliability, data confidentiality. So far, many public verification techniques have been proposed [23, 24] to ensure the integrity of outsourced data. Besides, some other cryptographic techniques [25, 26, 27, 28] have been exploited to achieve the reliability of the data stored in the cloud server. From the perspective of users, one of the most important security issues is data confidentiality. In fact, kernel data (e.g., emails with sensitive keywords) are considered as being strictly confidential by many individuals and organizations. Thus, these kernel data need to be encrypted before being outsourced to the cloud server. Hence, how to search over the encrypted data and retrieve the target outsourced data efficiently becomes a challenging issue.

Searchable encryption (SE) [29] was first proposed to allow users to search over encrypted data. In general, SE can be divided into symmetric searchable encryption (SSE) and asymmetric searchable encryption (ASE). More recently, many symmetric searchable encryption schemes with novel features have been proposed [30, 31, 32, 33]. Although SSE has a higher execution efficiency, it is only suitable for a single-user model, in which a user outsources her/his encrypted data to the storage server, and later she/he searches the encrypted data by keywords. Thus, SSE has low scalability and cannot been widely deployed in multi-user model.

Boneh et al. [6] designed the first public-key encryption with keyword search (PEKS) scheme, which can break the limitation of SSE. However, Baek et al. [34] pointed out that Boneh et al.'s model needs to establish a secure channel, which incurs high communication overhead. To tackle this problem, they proposed a channel-free PEKS scheme. Following up, Rhee et al. [35] proposed a generic construction of designated tester PEKS scheme, it not only achieves ciphertext indistinguishability but also trapdoor security.

Fang et al. [36] constructed a channel-free PEKS, which achieves secure against keyword guessing attack under the standard model. In recent, many PEKS variants with novel properties were proposed [10, 11, 37, 38, 39, 40]. Specifically, the schemes in [10, 11] support proxy re-encryption with keyword search under the assistance of the proxy.

Actually, the aforementioned PEKS scheme are vulnerable to keyword guessing attacks launched by a malicious system insider, e.g., a misbehaved cloud server, called IKGA. To resist against such IKGA, Xu et al. [41] constructed a public-key encryption with fuzzy keyword search scheme, in which each trapdoor corresponds to an extract keyword searchable trapdoor and a fuzzy keyword searchable trapdoor. However, it is impractical since the heavy communication overhead and computational costs are also introduced to the data receiver. After that, Chen et al. [42] proposed a dual-server PEKS scheme, but it relies on two servers and requires them do not collude with each other. Huang et al. [12] introduced the concept of public-key authenticated encryption with keyword search, where a data sender not only encrypts each keyword, but also authenticates it. We also observe that, most of these aforementioned schemes are designed on certificate systems, introducing complex key management procedures. Fortunately, the identity-based cryptographic systems [14] can simplify key management process in PKI. More recently, Li et al. [15] proposed a designated-server identity-based authenticated encryption with keyword search, which can also resist against IKGA. In addition, to eliminate complex key management and key escrow problem, certificateless PEKS schemes have also been proposed [43, 44, 45].

Nevertheless, according to the security analysis in [7], the conventional public key cryptographic algorithms will be threatened. Thus, those schemes mentioned above will be broken by quantum computers. As a promising technique for resisting quantum computers attacks, lattice-based cryptography holds very strong security proofs based on worst-case hardness [16, 46]. Up to date, few lattice-based searchable encryption schemes in public-key cryptographic systems have been proposed [47, 48], but these schemes still cannot resist against IKGA from misbehaved cloud servers.

3. Preliminaries

3.1. Notations

Now we provide some notations that will be used throughout the paper. We denote by $\mathbb{R}$, $\mathbb{Z}$, $\mathbb{Z}_q$ the set of real number, integer, and the cyclic group

$\{0, 1, \cdots, q\}$ with addition modulo q, respectively. For a positive integer m, $[m]$ is denoted by $\{1, 2, \cdots, m\}$. By convention, vectors are assumed to be in column form and are written using lower-case letters, e.g., b, the i-th component of b is denoted by b_i. Matrices are written as capital letters, e.g., B, and the i-th column vector of a matrix B is denoted by b_i. We denote by $\|B\|$ or $\|b\|$ the norm of a matrix B or vector b, we denote by $\widetilde{B}$ by the Gram-Schmidt ordered orthogonalization of B, and its norm by $\|\widetilde{B}\|$. Let $B\|B'\|$ denote the concatenation of the sets B, B'.

We use standard big-O notation to classify the growth of functions, and say that $f(n) = \tilde{O}(g(n))$ if $f(n) = O(g(n) \cdot \log^c n)$ for some fixed constant c. Denote by $poly(n)$ an unspecified function $f(n) = O(n^c)$ for some constant c. An expression is exponentially small in n, it means that it is at most $2^{\Omega(n)}$. A negligible function denoted by $negl(n)$ is that it is smaller than all polynomial fractions for sufficiently large n.

3.2. Lattices

Now, we first provide the definitions of integer lattices as follows.

Let $B = \{b_1, \cdots, b_m\} \in \mathbb{R}^{m \times m}$ be an $(m \times m)$-dimension matrix whose columns are linearly independent vectors $b_1, \cdots, b_m \in \mathbb{R}^m$. B is actually a basis of the m-dimensional full-rank lattice Λ, here $\Lambda = \{\nu \in \mathbb{R}^m : \exists \lambda = (\lambda_1, \cdots, \lambda_m)^\top \in \mathbb{Z}^m, \nu = B\lambda = \sum_{i \in [m]} \lambda_i b_i\}$.

Definition 1. *For a prime q, a matrix $A \in \mathbb{Z}_q^{n \times m}$, and a vector $\nu \in \mathbb{Z}_q^n$, the q-modular integer lattices in [49] are defined as follows:*

1. $\Lambda_q(A) = \{z \in \mathbb{Z}_q^m : \exists \lambda \in \mathbb{Z}_q^n, z = A^\top \lambda \ mod \ q\}$.
2. $\Lambda_q^\perp(A) = \{z \in \mathbb{Z}_q^m : Az = 0 \ mod \ q\}$.
3. $\Lambda_q^\nu(A) = \{z \in \mathbb{Z}_q^m : Az = \nu \ mod \ q\}$.

Now, we introduce discrete Gaussians as follows. $\rho_{\delta,\upsilon}(y) = \exp(-\pi\|y - \upsilon\|^2/\delta^2)$ denotes a Gaussian function on $\mathbb{R}^m$ with center υ and parameter $\delta > 0$. Let L be a subset of $\mathbb{Z}^m$, $\forall y \in L$, $\rho_{\delta,\upsilon}(L) = \sum_{y \in L} \rho_{\delta,\upsilon}(y)$ denotes the sum of $\rho_{\delta,\upsilon}$ over L. $\forall y \in L$, $\mathcal{D}_{L,\delta,\upsilon}(y) = \rho_{\delta,\upsilon}(y)/\rho_{\delta,\upsilon}(L)$ denotes the discrete Gaussian distribution over L with center υ and parameter δ.

The ciphertext indistinguishability of PO-IBEKS reduces to the following learning with errors (LWE) problem.

Definition 2. *For a prime q, a positive integer n, and a Gaussian noise distribution χ over $\mathbb{Z}_q$. An LWE instance consists of access to an unspecified*

challenge oracle $\mathcal{O}_{lwe}$, being either a noisy pseudo-random sampler $\mathcal{O}_1$, or a truly random sampler $\mathcal{O}_2$. Their behaviors are described respectively as follows:

$\mathcal{O}_1$: It outputs samples of the form $(v_k, u_{k1}, \cdots, u_{k\ell}) = (v_k, u_k^\top x_1 + t_1, \cdots, u_k^\top x_\ell + t_\ell) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^\ell$, where each $x_l \in \mathbb{Z}_q^n$ $(l = 1, \cdots, \ell)$ is a uniformly distributed persistent value is invariant across invocations, each $t_l \in \mathbb{Z}_q$ is a fresh sample from χ, and v_k is uniform vector in $\mathbb{Z}_q^n$.

$\mathcal{O}_2$: It outputs truly uniform samples from $\mathbb{Z}_q^n \times \mathbb{Z}_q^\ell$.

The LWE problem allows to repeatedly query to the oracle $\mathcal{O}_{lwe}$. An adversary $\mathcal{A}$ can succeed in deciding the hardness assumption of LWE problem if $\mathsf{LWE}_{\mathsf{adv}}[\mathcal{A}] := |\mathsf{Pr}[\mathcal{A}^{\mathcal{O}_1} = 1] - \mathsf{Pr}[\mathcal{A}^{\mathcal{O}_2} = 1]|$ is non-negligible. According to the security proof in [50, 51], the LWE problem is as hard as the worst-case SIVP and GapSVP under a quantum reduction.

Definition 3. *The inhomogeneous small integer solution (ISIS) problem is defined as follows: Given a positive real number $\hbar$, a matrix $Q \in \mathbb{Z}_q^{n \times m}$, and a uniform vector $\vartheta \leftarrow \mathbb{Z}_q^n$, the goal of ISIS problem is to find the solution $\theta \in \mathbb{Z}_q^m$, such that $Q\theta = \vartheta \bmod q$ and $0 < \|\theta\| \leq \hbar$.*

According to the security proof in [17], for any prime $q > \hbar \cdot \omega(\sqrt{n \log n})$, and any poly-bounded $\hbar = poly(n)$, the average-case hardness assumption of ISIS problem is as hard as approximating the problem SIVP in the worst case to within certain factor $\hbar \cdot \tilde{O}(\sqrt{n})$.

We introduce the preimage sampleable functions [17], which are given by a tuple of probabilistic polynomial-time (PPT) algorithms (TrapGen, SampleDom, SampleDom).

- TrapGen: The PPT algorithm $\mathsf{TrapGen}(q, n)$ in [52] generates $A \in \mathbb{Z}_q^{m \times n}, T_A \in \mathbb{Z}_q^{m \times m}$, such that A is statistically close to a uniform matrix in $\mathbb{Z}_q^{n \times m}$, and T_A is a basis of $\Lambda_q^\perp(A)$, satisfying $\|\widetilde{T_A}\| \leq O(\sqrt{n \log q})$, and $\|T_A\| \leq O(\sqrt{n \log q})$ with all but negligible probability in n.

- SampleDom: Sample an x from distribution $\mathcal{D}_{\mathbb{Z}^m, \delta}$, the distribution of $f_A(x)$ is uniform over $\mathbb{R}^n$.

- SamplePre: For $q \geq 2$, $m \geq 2n\lceil \log q \rceil$, taking as inputs a matrix $A \in \mathbb{Z}_q^{n \times m}$, a basis $T_A \in \mathbb{Z}_q^{m \times m}$ of $\Lambda_q^\perp(A)$, a vector $\nu \in \mathbb{Z}_q^n$, and a parameter $\delta \geq \|\widetilde{T_A}\| \cdot \omega(\sqrt{\log m})$, the PPT algorithm SamplePre outputs

a sample $z \in \Lambda_q^{\nu}(A)$ sampled from a distribution statistically close to $\mathcal{D}_{\Lambda_q^{\nu}(A),\delta}$, where $\mathcal{D}_{\Lambda_q^{\nu}(A),\delta}$ is a discrete Gaussian distribution over $\Lambda_q^{\nu}(A)$ with parameter δ, such that $Az = \nu \ mod \ q$.

Then we introduce the lattice basis delegation NewBasisDel [53], which is a key tool to generate the proxy-oriented private key in PO-IBEKS. 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 1. *Taking as inputs a matrix $A \in \mathbb{Z}_q^{n\times m}$, a $\mathbb{Z}_q$-invertible matrix R sampled from $\mathcal{D}_{m\times m}$, a short lattice basis T_A, and parameter $\sigma \geq \|\widetilde{T_A}\| \cdot \delta_R\sqrt{m}\omega(\log^{3/2} m)$, the* PPT *algorithm* NewBasisDel *outputs a random short lattice basis T_B of $\Lambda_q^{\perp}(B)$, where $B = AR^{-1}$.*

To prove that our scheme achieves ciphertext indistinguishability and existential unforgeability, we need a PPT algorithm SampleR, referring to [53], to sample matrices from a distribution, which are statistically close to $\mathcal{D}_{m\times m}$ over $\mathbb{Z}_q^{m\times m}$.

Additionally, the provable security of the proposed scheme employs a PPT algorithm SampleRwithBasis [53] to simulate a random short lattice basis in the following lemma.

Lemma 2. *Given a prime $q > 2$, $m \geq 2n\lceil\log q\rceil$, for all but at most a q^{-n} fraction of rank n matrix A in $\mathbb{Z}_q^{n\times m}$, the* PPT *algorithm* SampleRwithBasis(A) *outputs a low-norm matrix $R \in \mathbb{Z}_q^{m\times m}$ which is sampled from a distribution statistically close to $\mathcal{D}_{m\times m}$, and a random short lattice basis T_B of $\Lambda_q^{\perp}(B)$ with $B = AR^{-1}$, such that $\|\widetilde{T_B}\| \leq \delta_R/\omega(\sqrt{\log m})$.*

3.3. Basic system model and security definition

Now we introduce the system model of PO-IBEKS for cloud storage in Figure 1, which has five different entities: original data owner, proxy, data receiver, cloud server, and key generation center (KGC).

1. Original data owner: Her/his sensitive data as well as corresponding keywords need to be encrypted and uploaded to the cloud server by the proxy.

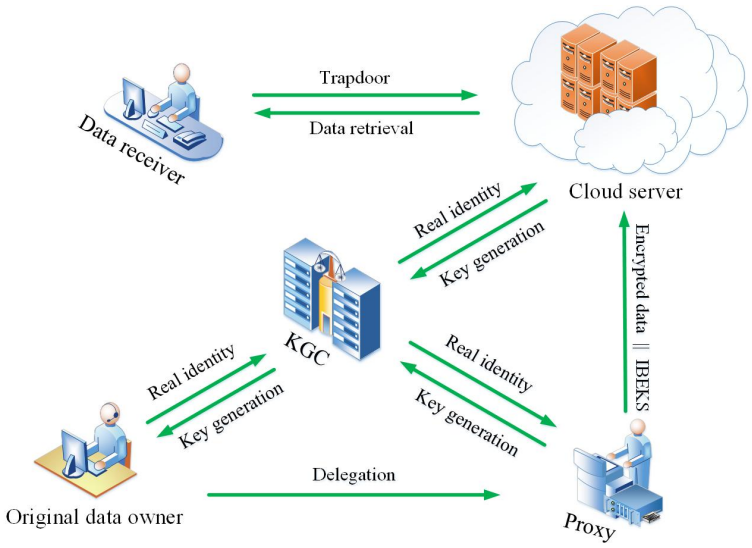

Figure 1: The system model of PO-IBEKS

2. Proxy: It is authorized by an original data owner. Once validating the signature of the warrant, on behalf of the original data owner, a proxy encrypts the sensitive data as well as keywords contained in the data under the public key of the intended data receiver, and further uploads them to the remote cloud server.
3. Data receiver: As a user, the data receiver employs the private key to generate the trapdoor associated with the specific keyword, and sends it to the cloud server to retrieve the intended encrypted data.
4. Cloud server: It provides computing power and cloud storage services. Once receiving a trapdoor associated with a specific keyword from a data receiver, the cloud server provides the data receiver with an efficient and secure way to search the ciphertexts with the trapdoor, and forwards corresponding encrypted data to the data receiver.
5. KGC: It is controlled by an authority. Using the master public-secret key, the KGC can generate the public/private key pair of any identity, such as an original data owner, the proxy, or a data receiver.

A formal definition of PO-IBEKS is given as follows.

Definition 4. *PO-IBEKS consists of six polynomial-time algorithms,* ***Setup***, ***KeyExtract***, ***Proxy-oriented key generation***, ***IBEKS***, ***Trapdoor***,

Test.

Setup: The PPT algorithm takes as inputs a secure parameters κ, outputs the system public parameters, and the master public/secret key pair $(\mathsf{Mpk}, \mathsf{Msk})$ of KGC.

KeyExtract: This PPT algorithm is performed by the KGC. Taking as inputs system public parameters, the master public/secret key pair $(\mathsf{Mpk}, \mathsf{Msk})$, and an identity id, the KGC generates corresponding public/private key SK_{id} of id.

Proxy-oriented key generation: This phase is to generate the proxy-oriented public/private key pair of the proxy. For the delegation of the proxy-oriented encryption rights, we denote a warrant W which records the delegation policy, valid period of delegation, and the identities of the original data owner id_o and the proxy id_p.

id_o firstly generates the signature of the warrant W by using her/his private key SK_{id_o}, and sends it to id_p, then id_p validates the signature of W. Based on the signature of W, id_p further generates the proxy-oriented public/private key pair $(\mathsf{PK}_{pro}, \mathsf{SK}_{pro})$ using her/his private key SK_{id_p}.

IBEKS: This PPT algorithm is performed by the proxy id_p. Taking as inputs the system public parameters, a keyword w, a data receiver's identity id_r, and the proxy-oriented private key SK_{pro}, id_p outputs a searchable ciphertext C associated with the keyword w.

Trapdoor: This PPT algorithm is performed by a data receiver id_r. Taking as inputs the system public parameters, the private key SK_{id_r} of the data receiver id_r, and a keyword w, id_r outputs a trapdoor d_w associated with the keyword w.

Test: This deterministic polynomial-time algorithm is performed by the cloud server, it takes as inputs a trapdoor d_w, a searchable ciphertext C, outputs 1 if C and d_w contain the same keyword w, and 0 otherwise.

Correctness consistency: PO-IBEKS requires that for any honestly generated proxy-oriented public/private key pair $(\mathsf{PK}_{pro}, \mathsf{SK}_{pro})$ of the proxy id_p, the private key SK_{id_r} of id_r, and for any keyword w, $\mathsf{Test}(d_w, C, \mathsf{PK}_{pro}, id_r) = 1$ holds, where $C \leftarrow \mathsf{IBEKS}(w, \mathsf{SK}_{pro}, id_r)$ and $d_w \leftarrow \mathsf{Trapdoor}(w, \mathsf{SK}_{id_r}, \mathsf{PK}_{pro})$.

3.4. Threat model

In the threat model of PO-IBEKS, here we consider security threats from three different aspects: malicious data receiver, adversarial cloud storage server, and outside adversary.

1. Malicious data receiver: Any malicious data adversary can break the ciphertext indistinguishability, such that it can guess the specific keyword and target corresponding intended encrypted data.
2. Adversarial cloud storage server. The adversarial cloud storage server would perform IKGA to violate the confidentiality of the outsourced keywords by forging an authenticated searchable ciphertext.
3. Outside adversary. Any outside adversary may impersonate an original data owner to authorize the proxy to encrypt the data as well as corresponding keywords, and uploads them to the cloud server, by forging the signature of warrant.

To prove the security of PO-IBEKS under the above threat model, we follow the security definitions that contain ciphertext indistinguishability, existential unforgeability, and delegation security. Details will be provided in Section 5.

3.5. Design goals

To construct a practical PO-IBEKS under the aforementioned model, the following objects should be achieved.

1. Security. PO-IBEKS should achieve ciphertext indistinguishability, existential unforgeability, and delegation security. Moreover, enabling PO-IBEKS to be secure against quantum computer attacks will be significant in the near future.
2. Efficiency. The communication overhead and computational costs should be as little as possible, especially at the side of the cloud server, decreasing the end-to-end computation delay would be an economic and favorable in practice. Moreover, eliminating the substantial certificate management of the PKI will be great benefit to the deployment of PO-IBEKS in mobile cloud storage systems.

4. Proxy-oriented identity-based encryption with keyword search scheme from lattices

4.1. The construction of PO-IBEKS

Now we first provide mathematical symbols of the proxy-oriented identity-based encryption with keyword search scheme (PO-IBEKS) from lattices, in Table 1. Specifically, the search and test process of PO-IBEKS is shown

Table 1: Mathematical symbol

Symbol	Description
χ	The discrete Gaussian distribution
σ, δ	Gaussian parameters
Mpk	The master public key
Msk	The master secret key
$\nu \leftarrow \mathbb{Z}_q^n$.	Sample a uniform vector from $\mathbb{Z}_q^n$
$\lfloor q/2 \rfloor$	The maximum integer less than or equal to $q/2$
H	Cryptographic hash function
$id \in \{0,1\}^{\ell_1}$	The size of identity is ℓ_1 bits
A_{pro}, T_{pro}	Proxy-oriented public/private key
KGC	The generation center

in Figure 2. PO-IBEKS consists of the following six polynomial-time algorithms, **Setup**, **KeyExtract**, **Proxy-oriented key generation**, **IBEKS**, **Trapdoor**, **Test**.

Setup: Taking as input a security parameter κ, the KGC determines the system parameters in the following steps:

- Determine the discrete Gaussian distribution χ and security Gaussian parameters σ, δ.
- Run $\mathsf{TrapGen}(q, n)$ to generate the master public key $\mathsf{Mpk} = A \in \mathbb{Z}_q^{n \times m}$ together with the master secret key $\mathsf{Msk} = T_A \in \mathbb{Z}_q^{m \times m}$.
- Select a uniform random vector $\nu \leftarrow \mathbb{Z}_q^n$.
- Set five secure cryptographic hash functions: $H_1 : \{0,1\}^{\ell_1} \rightarrow \mathbb{Z}_q^{m \times m}$, $H_2 : \{0,1\}^{\ell_1} \times \{0,1\}^{\ell_1} \times \{0,1\}^{\ell_2} \times \mathbb{Z}_q^n \rightarrow \mathbb{Z}_q^n$, $H_3 : \{0,1\}^{\ell_1} \times \{0,1\}^{\ell_1} \times \{0,1\}^{\ell_2} \times \mathbb{Z}_q^m \rightarrow \mathbb{Z}_q^{m \times m}$, $H_4 : \{0,1\}^{\ell_1} \times \{0,1\}^{\ell_1} \times \{0,1\}^{\ell_3} \rightarrow \mathbb{Z}_q^{m \times m}$, and $H_5 : \{0,1\}^{\ell} \times \mathbb{Z}_q^{m \times \ell} \rightarrow \mathbb{Z}_q^n$, where the outputs of H_1, H_3 and H_4 are distributed in $\mathcal{D}_{m \times m}$.

The system public parameters are $\Gamma = (A, \nu, H_1, H_2, H_3, H_4, H_5, \sigma, \delta)$, the KGC secretly keeps the master secret key $\mathsf{Msk} = T_A$.

KeyExtract: Taking as inputs system public parameters Γ, the master public/secret key pair (A, T_A), and an identity $id \in \{0,1\}^{\ell_1}$, here the general identity id can be termed an original data owner, a proxy, or a data receiver, the KGC performs as follows:

- Compute $R_{id} = H_1(id)$ and $A_{id} = A(R_{id})^{-1} \in \mathbb{Z}_q^{n \times m}$ as the public key of id.

- Run $\mathsf{NewBasisDel}(A, R_{id}, T_A, \sigma)$ to generate a random short lattice basis $T_{id} \in \mathbb{Z}_q^{m \times m}$ of $\Lambda_q^{\perp}(A_{id})$ as the private key of id.

The KGC sends the private key T_{id} to id via a secure channel.

Proxy-oriented key generation: To generate the proxy-oriented public/private key pair, an original data owner id_o will interact with a proxy id_p as follows:

- id_o creates the warrant $W \in \{0,1\}^{\ell_2}$ according to its requirements. There is an explicit description of the relative rights and information of an original data owner and a proxy in the warrant W. Specifically, the warrant W also includes the information of the intended data receiver, so that id_p can know about whom the original data owner wants to share the encrypted data with.

- id_o selects a uniform random vector $r \leftarrow \mathbb{Z}_q^n$, computes $\mu = H_2(id_o \| id_p \| W \| r)$, and runs $\mathsf{SamplePre}(A_{id_o}, T_{id_o}, \mu, \delta)$ to generate $\beta_W \in \mathbb{Z}_q^m$. Then id_o sends (W, r, β_W) directly to id_p. Here, everybody can validate the signature of the warrant W.

- Upon receiving (W, r, β_W) from id_o, id_p checks the validity of the signed warrant W by computing $A_{id_o}\beta_W = H_2(id_o \| id_p \| W \| r)$, where β_W is distributed in $\mathcal{D}_{\Lambda_q^{\mu}(A_{id_o}),\delta}$. If the verification equation does not hold, id_p rejects it and informs id_o; otherwise, id_p proceeds to compute $R_W = H_3(id_o \| id_p \| W \| \beta_W)$, and runs $\mathsf{NewBasisDel}(A_{id_p}, R_W, T_{id_p}, \sigma)$ to generate (A_{pro}, T_{pro}) as the proxy-oriented public/private key pair of id_p, where $A_{pro} = A_{id_p}(R_W)^{-1} \in \mathbb{Z}_q^{n \times m}$.

IBEKS: Taking as inputs Γ, (A_{pro}, T_{pro}) of id_p, a data receiver's identity id_r, and a keyword $w \in \{0,1\}^{\ell_3}$, id_p performs in the following steps:

- Randomly choose a uniform matrix $F \in \mathbb{Z}_q^{n \times \ell}$, and a random binary string $\tau = (\tau_1, \tau_2, \cdots, \tau_\ell) \in \{0,1\}^{\ell}$.

- Sample a random noise vector $\eta = (\eta_1, \eta_2, \cdots, \eta_\ell) \leftarrow \chi^{\ell}$. Sample ℓ random noise vectors $s_1, s_2, \cdots, s_\ell \leftarrow \chi^m$, and set the noise matrix $S = (s_1, s_2, \cdots, s_\ell) \in \mathbb{Z}_q^{m \times \ell}$.

Data receiver (A_{id_r}, T_{id_r}, w)		**Cloud server** (Γ, A_{pro}, C)
1) Set $\gamma = H_4(id_p \Vert id_r \Vert w)$, and compute $D_w = \mathsf{NewBasisDel}(A_{id_r}, \gamma, T_{id_r}, \sigma)$; 2) Compute trapdoor $d_w = \mathsf{SamplePre}(A_{id_r}\gamma^{-1}, D_w, \nu, \delta)$;	$\xrightarrow{Trapdoor = d_w}$	
		3) Compute $\tau \leftarrow \zeta - d_w^\top \xi$, if τ_j is close to $\lfloor q/2 \rfloor$, set $\tau_j \leftarrow 1$, else $\tau_j \leftarrow 0$, $j = 1, \cdots, \ell$; 4) Set $\tau = (\tau_1, \tau_2, \cdots, \tau_\ell)$, and compute $h = H_5(\tau \Vert \xi)$; 5) Check $A_{pro}\theta = h$ whether or it holds. If it holds, the cloud server returns 1, and returns corresponding ciphertext CT; otherwise, the cloud server returns 0;
	$\xleftarrow{CT \Vert \mathsf{PO-IBEKS}}$	
6) Use SK_{id_r} to decrypt CT.		

Figure 2: Search and test process of PO-IBEKS

- Compute $\xi = (A_{id_r}\gamma^{-1})^\top F + S$, $\zeta = \nu^\top F + \eta + (\tau_1, \tau_2, \cdots, \tau_\ell)\lfloor q/2 \rfloor$, where $\gamma = H_4(id_p \Vert id_r \Vert w) \in \mathbb{Z}_q^{m \times m}$.
- Compute $h = H_5(\tau \Vert \xi)$, and run $\mathsf{SamplePre}(A_{pro}, T_{pro}, h, \delta)$ to generate $\theta \in \mathbb{Z}_q^m$.

Finally, on behalf of id_o, id_p uploads the proxy-oriented searchable ciphertext $C = (\xi, \zeta, \theta)$ to the cloud server.

Trapdoor: Taking as inputs Γ, (A_{id_r}, T_{id_r}) of id_r, and a keyword $w \in \{0,1\}^{\ell_3}$, id_r performs as follows:

- Compute $\gamma = H_4(id_p \Vert id_r \Vert w) \in \mathbb{Z}_q^{m \times m}$, and run $\mathsf{NewBasisDel}(A_{id_r}, \gamma, T_{id_r}, \sigma)$ to generate a random short lattice basis $D_w \in \mathbb{Z}_q^{m \times m}$ of $\Lambda_q^\perp(A_{id_r}\gamma^{-1})$.
- Run $\mathsf{SamplePre}(A_{id_r}\gamma^{-1}, D_w, \nu, \delta)$ to generate the trapdoor $d_w \in \mathbb{Z}_q^m$.

Note that $A_{id_r}\gamma^{-1}d_w = \nu \in \mathbb{Z}_q^n$, and the trapdoor d_w is distributed in $\mathcal{D}_{\Lambda_q^\nu(A_{id_r}\gamma^{-1}),\delta}$. Finally, id_r sends the trapdoor d_w to the cloud server to retrieve the target encrypted data.

Test: Taking as inputs Γ, A_{pro} of id_p, the searchable ciphertext C, and a trapdoor d_w from id_r, the cloud server performs as follows:

- Compute $\tau \leftarrow \zeta - d_w^\top \xi \in \mathbb{Z}_q^\ell$. For $j = 1, 2, \cdots, \ell$, compare each τ_j and $\lfloor q/2 \rfloor$, treating them as integers in $\{1, 2, \cdots, q\} \subset \mathbb{Z}$. If they are close, i.e., if $|\tau_j - \lfloor q/2 \rfloor| < \lfloor q/4 \rfloor$ in $\mathbb{Z}$, set $\tau_j \leftarrow 1$; otherwise, output $\tau_j \leftarrow 0$, then output $\tau = (\tau_1, \tau_2, \cdots, \tau_\ell) \in \{0,1\}^\ell$.

- Compute $h = H_5(\tau \| \xi)$, and check whether the equation $A_{pro}\theta = h$ holds, where θ is distributed in $\mathcal{D}_{\Lambda_q^h(A_{pro}),\delta}$. If the equation holds, the cloud server returns 1; otherwise, it returns 0.

4.2. Correctness.

Let w be the keyword contained in the searchable ciphertext C and w' be that in $d_{w'}$. The correctness of PO-IBEKS is elaborated as follows.

In **Test**, with the trapdoor $d_{w'}$, the cloud server can easily compute $\tau' = (\tau_1', \tau_2', \cdots, \tau_\ell') \leftarrow \zeta - d_{w'}^\top \xi \in \mathbb{Z}_q^\ell$. We take into account the following two cases:

1. If the keywords w and w' are the same, i.e., $w = w'$, we have that $\tau' \leftarrow \zeta - d_{w'}^\top \xi = (\tau_1, \tau_2, \cdots, \tau_\ell)\lfloor q/2 \rfloor + \eta - d_{w'}^\top S$. Actually, $\eta - d_{w'}^\top S$ is an error vector, to decrypt correctly, we need to guarantee each component of error vector is less than $q/5$ discussed in [53]. Thus, $\tau' = \tau$, we have $h' = H_5(\tau' \| \xi) = H_5(\tau \| \xi)$. Therefore, $A_{pro}\theta = h'$ holds, where θ is distributed in $\mathcal{D}_{\Lambda_q^h(A_{pro}),\delta}$. The cloud server returns 1.
2. If the keywords w and w' are distinct, i.e., $w \neq w'$, as $\tau' \leftarrow \zeta - d_{w'}^\top \xi \in \mathbb{Z}_q^\ell \neq (\tau_1, \tau_2, \cdots, \tau_\ell)\lfloor q/2 \rfloor + \eta - d_{w'}^\top S$, the searchable ciphertext cannot be decrypted as $\tau = (\tau_1, \tau_2, \cdots, \tau_\ell) \in \{0,1\}^\ell$. Thus, we have $h' = H_5(\tau' \| \xi) \neq H_5(\tau \| \xi)$ and the equation $A_{pro}\theta \neq h'$. The cloud server returns 0.

Therefore, PO-IBEKS satisfies correctness consistency and the cloud server can believe that the searchable ciphertext $C = (\xi, \zeta, \theta)$ together with the trapdoor $d_{w'}$ contain the same keyword w. The cloud server responds with corresponding encrypted data associated with the keyword w to the data receiver id_r. Thus, id_r can further decrypt it with the private key T_{id_r}, and get the primitive data shared by the proxy id_p.

5. Security proof

Now we demonstrate that PO-IBEKS achieves semantically security, provided that PO-IBEKS achieves ciphertext indistinguishability, existential unforgeability, and delegation security simultaneously. Thus, PO-IBEKS resists against IKGA from misbehaved cloud servers. The details of the provable security processes could be provided as follows.

5.1. Ciphertext indistinguishability

Now, we prove that PO-IBEKS achieves ciphertext indistinguishability under the selective-ID security in the random oracle model. It is defined by the following interactive game between an adversary $\mathcal{A}_1$ (a malicious data receiver) and a challenger $\mathcal{C}$.

Game 1: The game is interactive between $\mathcal{A}_1$ and $\mathcal{C}$.

Setup: Input a security parameter κ, $\mathcal{C}$ outputs the system public parameters, the master public key of the KGC, and returns them to $\mathcal{A}_1$. $\mathcal{A}_1$ submits to $\mathcal{C}$ a target identity id_r^* which will be challenged ahead of time. Then $\mathcal{A}_1$ is allowed to adaptively issue queries to the following oracles.

KeyExtract oracle: Once receiving a query on an identity id (an original data owner id_o, a proxy id_p, or a data receiver id_r), $\mathcal{C}$ generates corresponding private key SK_{id} and returns it to $\mathcal{A}_1$. With the restriction that $\mathcal{A}_1$ cannot query the private key of id_r^*.

Trapdoor oracle: Once receiving a query on a keyword w, $\mathcal{C}$ generates corresponding trapdoor d_w and returns it to $\mathcal{A}_1$.

Challenge phase: At some point, $\mathcal{A}_1$ adaptively chooses two challenge keywords (w_0^*, w_1^*) as well as target identity id_r^*, which have not been queried for **Trapdoor oracle** or **KeyExtract oracle**, and submits them to $\mathcal{C}$. $\mathcal{C}$ randomly chooses a bit $\jmath \in \{0, 1\}$, generates corresponding searchable ciphertext C^* in a normal way, and returns it to $\mathcal{A}_1$.

$\mathcal{A}_1$ 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, $\mathcal{A}_1$ outputs a bit $\jmath' \in \{0, 1\}$. It wins the game if and only if $\jmath' = \jmath$.

The probability of $\mathcal{A}_1$ in winning the above game is defined as $\mathsf{Adv}_{\mathcal{A}_1}(\kappa) = |\mathsf{Pr}[\jmath' = \jmath] - 1/2|$.

Now we provide the detailed security proof of ciphertext indistinguishability as follows.

Theorem 1. *PO-IBEKS achieves ciphertext indistinguishability under the selective-ID security in the random oracle model, provided that the hardness assumption of deciding LWE problem holds.*

Proof. We suppose $\mathcal{A}_1$ (a malicious data receiver) can break ciphertext indistinguishability with non-negligible probability ϵ_1, we will demonstrate that $\mathcal{C}$ can successfully decide the hardness assumption of LWE problem with non-negligible probability ϵ_1'.

Setup: $\mathcal{C}$ requests from LWE oracle $\mathcal{O}_{lwe}$ for $m+1$ times, and obtains each instance $(v_k, u_{k1}, \cdots, u_{k\ell}) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^\ell$, where $k = 0, 1, \cdots, m$. Meanwhile, to maintain consistency, $\mathcal{C}$ sets five lists L_1 ,L_2, L_3, L_4, L_5, respectively. Set q_{H_i} to be the maximum number of $\mathcal{A}_1's$ queries to H_i for $i = 1, 2, 3, 4, 5$. $\mathcal{C}$ randomly chooses $I_1^* \in [q_{H_1}]$, and $I_4^* \in [q_{H_4}]$. Finally, $\mathcal{C}$ prepares system public parameters for $\mathcal{A}_1$ as follows:

- $\mathcal{C}$ samples two random matrices $P_1^*, P_2^* \leftarrow \mathcal{D}_{m\times m}$ by running $\mathsf{SampleR}$.
- $\mathcal{C}$ assembles a random matrix $P^* \in Z_q^{n\times m}$ from m of the LWE instances, by letting the k-th column of P^* be the vector $v_k \in \mathbb{Z}_q^n$ for all $k = 1, \cdots, m$.
- $\mathcal{C}$ sets $A = P^* P_2^* P_1^*$, where A is uniform in $\mathbb{Z}_q^{n\times m}$, since $P_1^*, P_2^* \in \mathbb{Z}_q^{m\times m}$ are $\mathbb{Z}_q$-invertible and P^* is uniform in $\mathbb{Z}_q^{n\times m}$. $\mathcal{C}$ sets $\nu = v_0 \in \mathbb{Z}_q^n$, and sets $\Gamma = (A = P^* P_2^* P_1^*, \nu, H_1, H_2, H_3, H_4, H_5)$. Finally, $\mathcal{C}$ returns Γ to $\mathcal{A}_1$. Ahead of time, $\mathcal{A}_1$ submits to $\mathcal{C}$ a target identity id_r^* which will be challenged.

We first assume that no matter when $\mathcal{A}_1$ performs the **KeyExtract oracle** query, it has made all relevant H_1 queries before. Now, $\mathcal{A}_1$ performs the following queries.

H_1 query: For the l-th query, here $l = 1, 2, \cdots, q_{H_1}$, $\mathcal{A}_1$ queries to H_1 on any identity id adaptively. If $l = I_1^*$, such that $id = id_r^*$, $\mathcal{B}$ sets $H_1(id) = P_1^*$ and returns it to $\mathcal{A}_1$. Otherwise, $\mathcal{C}$ runs $\mathsf{SampleRwithBasis}(A)$ to generate a low norm invertible matrix $R_{id} \in \mathbb{Z}_q^{m\times m}$, and a short lattice basis T_{id} of $\Lambda_q^\perp(A(R_{id})^{-1})$. Then $\mathcal{C}$ adds $(id, A(R_{id})^{-1}, R_{id}, T_{id})$ to list L_1, and returns R_{id} to $\mathcal{A}_1$.

H_2 query: For the l-th query, here $l = 1, 2, \cdots, q_{H_2}$, $\mathcal{A}_1$ queries on a distinct $(id_o \| id_p \| W \| r)$ to $\mathcal{C}$. Then $\mathcal{C}$ first checks if the value of H_2 was previously defined. If it was, the previously defined value is returned. Otherwise,

$\mathcal{C}$ randomly chooses a vector $\mu \leftarrow \mathbb{Z}_q^n$, and adds it to list L_2, then returns it to $\mathcal{A}_1$.

H_3 query: For the l-th query, here $l = 1, 2, \cdots, q_{H_3}$, $\mathcal{A}$ queries on a distinct (id_o, id_p, W, β_W) to $\mathcal{C}$. $\mathcal{C}$ returns R_W to $\mathcal{A}_1$, if there exists $(id_o, id_p, W, \beta_W, R_W, A_{pro}, T_{pro})$ in list L_3. Otherwise, $\mathcal{C}$ searches $(id_p, A(R_{id_p})^{-1}, R_{id_p}, T_{id_p})$ in list L_1, randomly chooses a matrix $R_W \leftarrow \mathcal{D}_{m\times m}$, and runs $\mathsf{NewBasisDel}(A_{id_p}, R_W, T_{id_p}, \sigma)$ to generate a short lattice basis T_{pro} for $A_{pro} = A_{id_p}(R_W)^{-1} \in \mathbb{Z}_q^{n\times m}$. Finally, $\mathcal{C}$ adds $(id_o, id_p, W, \beta_W, R_W, A_{pro}, T_{pro})$ to L_3, and returns R_W to $\mathcal{A}_1$.

H_4 query: For the l-th query, here $l = 1, 2, \cdots, q_{H_4}$, $\mathcal{A}_1$ queries on a distinct (id_p, id_r, w), here we consider the following two cases.

If $id_r = id_r^*$, $\mathcal{C}$ performs as follows:

If $l = I_4^*$, such that $w = w^*$, the challenger $\mathcal{C}$ sets $H_4(id_p \| id_r \| w) = P_2^*$, adds $(id_p, id_r^*, w^*, A(P_1^*)^{-1}(P_2^*)^{-1}, \perp, P_2^*)$ to list L_4, and returns P_2^* to $\mathcal{A}_1$. Otherwise, the challenger $\mathcal{C}$ runs $\mathsf{SampleRwithBasis}(A(P_1^*)^{-1})$ to generate a random $m \times m$-dimension matrix $R_w \leftarrow \mathcal{D}_{m\times m}$ and a short lattice basis D_w for $A(P_1^*)^{-1}(R_w)^{-1}$. Finally, $\mathcal{C}$ adds $(id_p, id_r, w, A(P_1^*)^{-1}(R_w)^{-1}, D_w, R_w)$ to L_4, and returns R_w to $\mathcal{A}_1$.

If $id_r \neq id_r^*$, $\mathcal{C}$ performs as follows:

If $l = I_4^*$, such that $w = w^*$, $\mathcal{C}$ directly returns P_2^* to $\mathcal{A}_1$; otherwise, $\mathcal{C}$ looks into list L_1 to find $(id_r, A(R_{id_r})^{-1}, R_{id_r}, T_{id_r})$, randomly chooses a matrix $R_w \leftarrow \mathcal{D}_{m\times m}$, and runs $\mathsf{NewbasisDel}(A(R_{id_r})^{-1}, R_w, T_{id_r}, \sigma)$ to generate a short basis D_w for $A(R_{id_r})^{-1}(R_w)^{-1}$. Finally, $\mathcal{C}$ adds $(id_p, id_r, w, A(H_1(id_r))^{-1} \cdot (R_w)^{-1}, D_w, R_w)$ to L_4, and returns R_w to $\mathcal{A}_1$.

H_5 query: Upon $\mathcal{A}_1$ queries on a distinct (τ, ξ), $\mathcal{C}$ first checks if the value of H_5 was previously defined. If it was, the previously defined value is returned; otherwise, $\mathcal{C}$ random chooses $h_5 \leftarrow Z_q^n$, and returns it to $\mathcal{A}_1$.

KeyExtract oracle: Upon receiving the KeyExtract query on id, with the restriction that $id \neq id_r^*$, $\mathcal{C}$ checks list L_1 to find $(id, A(R_{id})^{-1}, R_{id}, T_{id})$, if it is found, $\mathcal{C}$ returns T_{id} to $\mathcal{A}_1$. If not, $\mathcal{C}$ performs as the same to the H_1 query on the id, outputs $(id, A(R_{id})^{-1}, R_{id}, T_{id})$, and returns T_{id} to $\mathcal{A}_1$.

Trapdoor oracle: Upon receiving the query on a keyword w from $\mathcal{A}_1$, with the restriction that $id \neq id_r^*$. $\mathcal{C}$ first checks list L_4 to find $(id_p, id_r, w, A(H_1(id_r))^{-1}(R_w)^{-1}, D_w, R_w)$, and runs $\mathsf{SamplePre}(A(H_1(id_r))^{-1}(R_w)^{-1}, D_w, \nu, \delta)$ to generate d_w, and returns to $\mathcal{A}_1$.

Challenge phase: The adversary $\mathcal{A}_1$ sends $(id_p, id_r^*, w_0^*, w_1^*)$ to $\mathcal{C}$, where w_0^* and w_1^* are two challenge keywords, id_r^* is a challenge identity. $\mathcal{A}_1$ randomly selects $\jmath \leftarrow \{0, 1\}$. If $\jmath = 0$, $\mathcal{C}$ returns a random searchable ciphertext $C^* = (\xi^*, \zeta^*, \theta^*)$ associated with the keyword w_1^* to $\mathcal{A}_1$. Otherwise, $\mathcal{C}$ per-

forms as follows:

- For each $k = 0, 1 \cdots, m$, retrieve $u_{k0}, u_{k1}, \cdots, u_{k\ell} \in \mathbb{Z}_q$ from LWE instances, set $u_k = (u_{k1}, \cdots, u_{k\ell})^\top \in \mathbb{Z}_q^\ell$, $u^* = (u_1, \cdots, u_m)^\top \in \mathbb{Z}_q^{m\times\ell}$, and randomly choose a binary string $\tau^* = (\tau_1^*, \tau_2^*, \cdots, \tau_\ell^*) \in \{0,1\}^\ell$.
- Compute $\xi^* = u_0 + \tau^* \lfloor q/2 \rfloor \in \mathbb{Z}_q^\ell$ and set $\zeta^* = u^* \in \mathbb{Z}_q^{m\times\ell}$.

Then $\mathcal{C}$ computes $h^* = H_5(\tau^* \| \xi^*) \in \mathbb{Z}_q^n$ and gets the proxy-oriented private key T_{pro} of the data sender id_p in a similar manner of answering the H_3 query on (id_o, id_p, W, β_W), and then runs $\mathsf{SamplePre}(A_{pro}, T_{pro}, h^*, \delta)$ to generate $\theta^* \in \mathbb{Z}_q^m$. Finally, $\mathcal{C}$ returns $C^* = (\xi^*, \zeta^*, \theta^*)$ associated with the keyword w_0^* to $\mathcal{A}_1$.

Guess: At last, $\mathcal{A}_1$ outputs $\jmath' \leftarrow \{0,1\}$.

Now, we evaluate the probability of $\mathcal{A}_1$ in breaking ciphertext indistinguishability under the selective-ID security in the random model. Essentially, the goal of $\mathcal{A}_1$ is to decide which keyword is involved in the challenge searchable ciphertext C^*. Based on the setting of the public parameters Γ, since $A = P^* P_2^* P_1^*$, we have the equation $A(H_1(id_r^*))^{-1} H_4(id_p \| id_r^* \| w_0^*)^{-1} = A(P_1^*)^{-1}(P_2^*)^{-1} = P^*$ and $\xi^* = u^* = (P^*)^\top F^* + S^*$ for some random matrices $F^* \in \mathbb{Z}_q^{n\times\ell}$ and $S^* \in \mathbb{Z}_q^{m\times\ell}$ with Gaussian distribution. Therefore, ξ^*, ζ^* have the correct forms. We consider the case that $\mathcal{A}_1$ can successfully guess that the keyword w_0^* is involved in C^* with non-negligible probability ϵ_1, where the keyword w_0^* is just the I_4^*-th H_4 query, it means that $w_0^* = w^*$, this case occurs with the probability $1/q_{H_4}$. While id_r^* is just the I_1^*-th H_1 query, this case occurs with the the probability $1/q_{H_1}$. Moreover, $\mathcal{C}$ can return the searchable ciphertext which is really associated with the keyword w_0^* with the probability 1/2. Thus, if $\mathcal{A}_1$ with non-negligible probability ϵ_1 can break ciphertext indistinguishability of the proposed scheme under the selective-ID security in the random oracle model, $\mathcal{C}$ has the non-negligible probability $\epsilon_1' = \epsilon_1/(2q_{H_1}q_{H_4})$ to solve the hardness assumption of deciding LWE problem by running the adversary $\mathcal{A}_1$ as a subroutine, it leads to a contradiction. Consequently, $\mathsf{Adv}_{\mathcal{A}_1}(\kappa)$ is negligible, PO-IBEKS achieves ciphertext indistinguishability under the selective-ID security in the random oracle model.

5.2. Existential unforgeability

As we described in the threat model, a misbehaved cloud server may select a proxy's identity ahead of any query, guess the target keyword, and try

to generate a forged searchable ciphertext, so that the searchable ciphertext can pass the test process. Now we prove that PO-IBEKS achieves existential unforgeability, so that a misbehaved cloud server cannot forge a valid searchable ciphertext to perform the IKGA.

Existential unforgeability is defined by the following interactive game between an adversary $\mathcal{A}_2$ (a misbehaved cloud server) and a challenger $\mathcal{C}$.

Game 2: The game is interactive between $\mathcal{A}_2$ and $\mathcal{C}$.

Setup: Input a security parameter κ, $\mathcal{C}$ outputs the system public parameters, the master public key of the KGC, and returns them to $\mathcal{A}_2$. $\mathcal{A}_2$ submits to $\mathcal{C}$ a target identity id_p^* which will be challenged ahead of time. Then $\mathcal{A}_2$ is allowed to adaptively issue queries to the following oracles.

KeyExtract oracle: Once receiving a query on an identity id (an original data owner id_o, a proxy id_p, or a data receiver id_r), $\mathcal{C}$ generates corresponding private key SK_{id} and returns it to $\mathcal{A}_2$. With the restriction that $\mathcal{A}_2$ cannot query the private key of id_p^*.

Trapdoor oracle: Once receiving a query on a keyword w, $\mathcal{C}$ generates corresponding trapdoor d_w and returns it to $\mathcal{A}_2$.

Authenticated searchable ciphertext oracle: $\mathcal{A}_2$ queries to $\mathcal{C}$ on any keyword w under the identities id_p, id_r. $\mathcal{C}$ generates corresponding authenticated searchable ciphertext associated with the keyword w, with the restriction that $id_p \neq id_p^*$.

Forgery phase: Finally, $\mathcal{A}_2$ outputs a forged authenticated searchable ciphertext of τ^* associated with (w^*, id_p^*, id_r), which can pass the test process.

The probability of $\mathcal{A}_2$ in winning the above game is defined as $\mathsf{Adv}_{\mathcal{A}_2}(\kappa)$.

Now we provide the detailed security proof of existential unforgeability as follows.

Theorem 2. *PO-IBEKS achieves existential unforgeability under the selective-ID security in the random oracle model, provided that the hardness problem of ISIS assumption holds.*

Proof. Suppose that an adversary $\mathcal{A}_2$ (a misbehave cloud server) can break existential unforgeability in the random model with non-negligible probability ϵ_2, we will prove that $\mathcal{C}$ can solve the hardness assumption of ISIS problem also with non-negligible probability ϵ_2', by running the adversary $\mathcal{A}_2$ as a subroutine.

Setup: $\mathcal{C}$ receives an instance of ISIS problem $(Q, \vartheta) \in \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^n$. $\mathcal{C}$ simulates the game of existential unforgeability with $\mathcal{A}_2$, tries to find a

solution $\theta^* \in \mathbb{Z}_q^m$, such that $0 < \|\theta^*\| \leq \delta\sqrt{m}$ and $Q\theta^* = \vartheta$. Meanwhile, to maintain consistency, $\mathcal{C}$ sets five lists L_1 ,L_2, L_3, L_4, L_5 respectively. Set q_{H_i} to be the maximum number of $\mathcal{A}_1's$ queries to H_i for $i = 1, 2, 3, 4, 5$. $\mathcal{C}$ randomly chooses $J_1^* \in [q_{H_1}]$, $J_3^* \in [q_{H_3}]$, and $J_5^* \in [q_{H_5}]$. Finally, $\mathcal{C}$ prepares system public parameters for $\mathcal{A}_2$ as follows:

- $\mathcal{C}$ samples two random matrices $Q_1^*, Q_2^* \leftarrow \mathcal{D}_{m\times m}$ by running the algorithm SampleR.
- $\mathcal{C}$ sets the matrix $A = QQ_2^*Q_1^*$, randomly chooses a vector $\nu \leftarrow \mathbb{Z}_q^n$, and returns $\Gamma = (A, \nu, H_1, H_2, H_3, H_4, H_5)$ to $\mathcal{A}_2$.

Initially, $\mathcal{A}_2$ submits to $\mathcal{C}$ a target identity id_p^* which will be challenged. We assume that no matter when $\mathcal{A}_2$ performs the KeyExtract query, it has made all relevant H_1 queries before. $\mathcal{A}_2$ performs the following queries.

H_1 query: For the l-th query, here $l = 1, 2, \cdots, q_{H_1}$, $\mathcal{A}_2$ queries to H_1 on any identity id adaptively. If $l = J_1^*$, such that $id = id_p^*$, $\mathcal{C}$ sets $H_1(id) = Q_1^*$ and returns it to $\mathcal{A}_2$. Otherwise, $\mathcal{C}$ runs SampleRwithBasis(A) to generate a low norm invertible matrix $R_{id} \in \mathbb{Z}_q^{m\times m}$, and a short lattice basis T_{id} of $\Lambda_q^{\perp}(A(R_{id})^{-1})$. Then $\mathcal{C}$ adds $(id, A(R_{id})^{-1}, R_{id}, T_{id})$ to list L_1, and returns R_{id} to $\mathcal{A}_2$.

H_2 query: $\mathcal{A}_2$ queries on a distinct $(id_o\|id_p\|W\|r)$ to $\mathcal{C}$. Then $\mathcal{C}$ first checks if the value of H_2 was previously defined. If it was, the previously defined value is returned. Otherwise, $\mathcal{C}$ randomly chooses a vector $\mu \leftarrow \mathbb{Z}_q^n$, and adds it to list L_2, then returns it to $\mathcal{A}_2$.

H_3 query: For the l-th query, here $l = 1, 2, \cdots, q_{H_3}$, $\mathcal{A}_2$ queries on a distinct (id_o, id_p, W, β_W) to $\mathcal{C}$. $\mathcal{C}$ returns R_W, if there exists $(id_o, id_p, W, \beta_W, R_W, A_{pro}, T_{pro})$ in list L_3. If $l = J_3^*$, such that $id_p = id_p^*$, $\mathcal{C}$ adds $(id_o, id_p, W, \beta_W, Q_2^*, A_{pro}, \perp)$ to list L_3 and returns Q_2^* to $\mathcal{A}_2$. Otherwise, $\mathcal{C}$ searches $(id_p, A(R_{id_p})^{-1}, R_{id_p}, T_{id_p})$ in list L_1, then $\mathcal{C}$ randomly chooses a matrix $R_W \leftarrow \mathcal{D}_{m\times m}$, and runs NewBasisDel$(A_{id_p}, R_W, T_{id_p}, \sigma)$ to generate a short lattice basis T_{pro} for $A_{pro} = A_{id_p}(R_W)^{-1} \in \mathbb{Z}_q^{n\times m}$. Finally, $\mathcal{C}$ adds $(id_o, id_p, W, \beta_W, R_W, A_{pro}, T_{pro})$ to L_3, and returns R_W to $\mathcal{A}_2$.

H_4 query: $\mathcal{A}_2$ queries on a distinct (id_p, id_r, w), $\mathcal{C}$ looks into list L_1 and finds $(id_r, A(R_{id_r})^{-1}, R_{id_r}, T_{id_r})$, randomly chooses a matrix $R_w \leftarrow \mathcal{D}_{m\times m}$, then runs NewbasisDel$(A(R_{id_r})^{-1}, R_w, T_{id_r}, \sigma)$ to generate a short basis D_w for $A(R_{id_r})^{-1}(R_w)^{-1}$. Finally, $\mathcal{C}$ adds $(id_p, id_r, w, A(H_1(id_r))^{-1}(R_w)^{-1}, D_w, R_w)$ to list L_4, and returns R_w to $\mathcal{A}_2$.

H_5 query: For the l-th query, here $l = 1, 2, \cdots, q_{H_5}$, $\mathcal{A}_2$ queries on a distinct (τ, ξ), $\mathcal{C}$ returns h_5 to $\mathcal{A}_2$, if (τ, ξ) exists in list L_5. If $l = J_5^*$, such that $\tau = \tau^*$, ξ^* is just the first component of ciphertext C associated with the keyword w^*, under some uniform matrix $F^* \leftarrow Z_q^{n \times m}$, some noise matrix $S^* \in Z_q^{m \times \ell}$, $\mathcal{C}$ adds $(\tau^*, \xi^*, \perp, \vartheta)$ to L_5, and returns ϑ to $\mathcal{A}_2$. Otherwise, $\mathcal{C}$ generates $\theta \leftarrow \mathcal{D}_{\mathbb{Z}^m, \delta}$ by runing $\mathsf{SampleDom}$, and computes $h_5 = Q\theta \in \mathbb{Z}_q^n$, then $\mathcal{C}$ adds (τ, ξ, h, θ) to L_5, and returns h_5 to $\mathcal{A}_2$.

KeyExtract oralce: Upon receiving the KeyExtract query on id, with the restriction that $id \neq id_p^*$. $\mathcal{C}$ will check list L_1 to find its hash value, if it is found, the previous value is returned. If it is not found, $\mathcal{C}$ performs as the same to the H_1 query on id, outputs $(id, A(R_{id})^{-1}, R_{id}, T_{id})$, returns T_{id} to $\mathcal{A}_2$.

Trapdoor oracle: Once receiving the query on a keyword w from $\mathcal{A}_2$. $\mathcal{C}$ first looks into list L_4, if $(id_p, id_r, w, A(H_1(id_r))^{-1}(R_w)^{-1}, D_w, R_w)$ is in L_4, $\mathcal{C}$ can get D_w, and runs the algorithm $\mathsf{SamplePre}(A(H_1(id_r))^{-1}(R_w)^{-1}, D_w, \nu, \delta)$ to generate d_w, and returns it to $\mathcal{A}_2$.

Authenticated searchable ciphertext oracle: $\mathcal{A}_2$ submits a keyword w and a binary string $\tau \in \{0,1\}^\ell$ for the authenticated searchable ciphertext query. $\mathcal{C}$ randomly chooses a uniform matrix $F \leftarrow Z_q^{n \times m}$, chooses a noise matrix $S \in Z_q^{m \times \ell}$, and computes ξ, ζ in a normal way. Then, $\mathcal{C}$ chooses $\theta \leftarrow \mathcal{D}_{Z^m, \delta}$ by running the algorithm $\mathsf{SampleDom}$. Finally, $\mathcal{C}$ returns $C = (\xi, \zeta, \theta)$ to $\mathcal{A}_2$.

Forgery phase: $\mathcal{A}_2$ eventually returns $\mathcal{C}$ a forged searchable ciphertext $C^* = (\xi^*, \zeta^*, \theta^*)$ of τ^* associated with (w^*, id_p^*, id_r). With the restriction that τ^* associated with (w^*, id_p^*, id_r) cannot be submitted to authenticated searchable ciphertext oracle.

Note that $\mathcal{A}_2$ can query to $\mathcal{C}$ to get $(id_p^*, id_r, w^*, A(H_1(id_r))^{-1}(R_{w^*})^{-1}, D_{w^*}, R_{w^*})$ in list L_4, $\mathcal{A}_2$ further generates $d_{w^*} \leftarrow \mathsf{SamplePre}(A_{id_r}\gamma^{-1}, D_{w^*}, \nu, \delta)$. Now $\mathcal{C}$ performs as follows:

- Recover τ^* by computing $\tau^* \leftarrow \zeta^* - d_{w^*}\xi^*$ under the searchable trapdoor d_{w^*}.
- Output θ^* as a forged signature of τ^* associated with the keyword w^*.

Thus, $\mathcal{C}$ outputs θ^* as its answer to the ISIS instance (Q, ϑ). We know that the adversary $\mathcal{A}_2$ can win the game only if θ^* is a valid proxy-oriented signature of τ^* associated with the keyword w^* for the proxy id_p^*, thus we

have $A_{id_p^*}(R_W)^{-1}\theta^* = H_5(\tau^*\|\xi^*)$, where $0 < \|\theta^*\| \leq \delta\sqrt{m}$. Note that if $\mathcal{C}$ successfully guesses, such that $H_5(\tau^*\|\xi^*) = \vartheta$ and $H_3(id_o\|id_p^*\|W\|\beta_W) = Q_2^*$. As $A_{id_p^*}(R_W)^{-1} = A(R_{id_p^*})^{-1}(R_W)^{-1} = A(Q_1^*)^{-1}(Q_2^*)^{-1} = Q$, $\mathcal{A}_2$ succeeds in forging a valid authenticated searchable ciphertext with non-negligible probability ϵ_2, then $\mathcal{C}$ can find a solution θ^*, such that $Q\theta^* = \vartheta$ and $0 < \|\theta^*\| \leq \delta\sqrt{m}$, with non-negligible probability $\epsilon_2' = \epsilon_2/(q_{[H_1]}q_{[H_3]}q_{[H_5]})$, which contradicts to the hardness assumption of ISIS problem. Consequently, $\mathsf{Adv}_{\mathcal{A}_2}(\kappa)$ is negligible, PO-IBEKS achieves existential unforgeability under the selective-ID security in the random oracle model.

5.3. Delegation security

For the delegation of the proxy-oriented encryption rights, as the warrant W records the delegation policy, valid period of delegation, and the identities of an original data owner and the proxy, only the rights of id_p satisfy the contents of the warrant W, can id_p encrypt the data as well as the keywords, and upload them to the remote cloud server on behalf of the original data owner. Now we further prove the delegation security of PO-IBEKS as follows.

Theorem 3. *PO-IBEKS achieves delegation security, provided that the hardness problem of ISIS assumption holds.*

Proof. Since the provable security process is similar to Theorem 2, for simplicity, here we omit the description of Game 3, we directly provide the security analysis. Once the adversary $\mathcal{A}_3$ captures the valid authorized information (W, r, β_W) from an original data owner, we assume that $\mathcal{A}_3$ with an advantage $\mathsf{Adv}_{\mathcal{A}_3}(\kappa)$ can forge a valid authorized information (W', r', β_W') without the private key T_{id_o} of the original data owner, such that $A_{id_o}\beta_W' = H_2(id_o\|id_p\|W'\|r') = h_2'$, where β_W' is distributed in $\mathcal{D}_{\Lambda_q^{h_2'}(A_{id_o}),\delta}$. As a matter of fact, the authorized information (W, r, β_W) satisfies the equation $A_{id_o}\beta_W = H_2(id_o\|id_p\|W\|r) = h_2$, provided that β_W is generated under the private key T_{id_o} of the original data owner. Thus, according to the security proof in [17], $\mathcal{A}_3$ can find the solution $\beta_W^* = \beta_W - \beta_W'$ of the ISIS instance, such that $A_{id_o}\beta_W^* = h_2 - h_2'$, which contradicts to the hardness problem of ISIS assumption, thus $\mathsf{Adv}_{\mathcal{A}_3}(\kappa)$ is negligible. Therefore, PO-IBEKS achieves delegation security, any outside adversary cannot generate the forged signature of the warrant, or cannot personate the original data owner to authorize the proxy to encrypt the sensitive data as well as corresponding keywords, nor upload the ciphertext to the cloud server.

Table 2: Communication overhead comparison

Schemes	Trapdoor	Ciphertext
PAEKS [12]	$L_{\mathbb{G}_2}$	$2L_{\mathbb{G}_1}$
dIBAEKS [15]	$2L_{\mathbb{G}_1}$	$2L_{\mathbb{G}_1} + L_{\mathbb{G}_2}$
CLPAEKS [45]	$L_{\mathbb{G}_2}$	$2L_{\mathbb{G}_1}$
PO-IBEKS	mL_q	$(m\ell + \ell + m)L_q$

Table 3: Test costs comparison

Schemes	Test costs
PAEKS [12]	$2Pair + mul$
dIBAEKS [15]	$2Pair + 2Exp + mul$
CLPAEKS [45]	$2Pair + 2Add + 2Mul + mul + 2hash$
PO-IBEKS	$(nm + m\ell)mul + hash$

5.4. Performance analysis

To deploy a practical PO-IBEKS in mobile cloud storage systems, a computationally efficient test process is a critical requirement to minimize the end-to-end computation delay. Now we conduct an elaborate performance evaluation of PO-IBEKS, compared with existing schemes [12, 45, 15]. The key factors that affect the performance of a practical PO-IBEKS in mobile cloud storage systems, are actually in terms of the communication overhead and the testing costs on the end-to-end delay from a cloud server to a data receiver.

All the implementations are conducted on a laptop with Window 10 systems with an Intel Core 2 i5 CPU and 8GB DDR 3 of RAM. We exploit C language and MIRACL Library version 5.6.1. The elliptic curve is an MNT curve, its base field size is 159 bits and the embedding degree is 6. As the lattice-based cryptographic algorithms are based on these parameters n, m, q, to achieve the provable security, we give an instance of a concrete PO-IBEKS with appropriate parameters which satisfy $m \geq 2n\lceil \log q \rceil$. We compare a concrete PO-IBEKS to the existing schemes, we also set the security level $\ell = 10$. All the results of implementations are represented 30 trials on average.

Now we specify some notations to represent corresponding cryptographic operations. In particular, $L_{\mathbb{G}_1}$, $L_{\mathbb{G}_2}$ denote the bit length of an element in cycle group $\mathbb{G}_1, \mathbb{G}_2$ respectively, L_q denotes the bit length of an element in $\mathbb{Z}_q$. Moreover, $Pair$ denotes a bilinear pairing executing time, Exp denotes a modular exponentiation executing time, Add denotes a point addition exe-

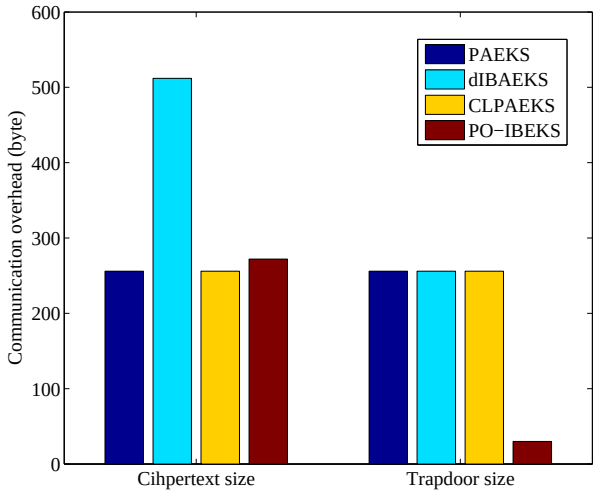

Figure 3: The comparison of communication overhead

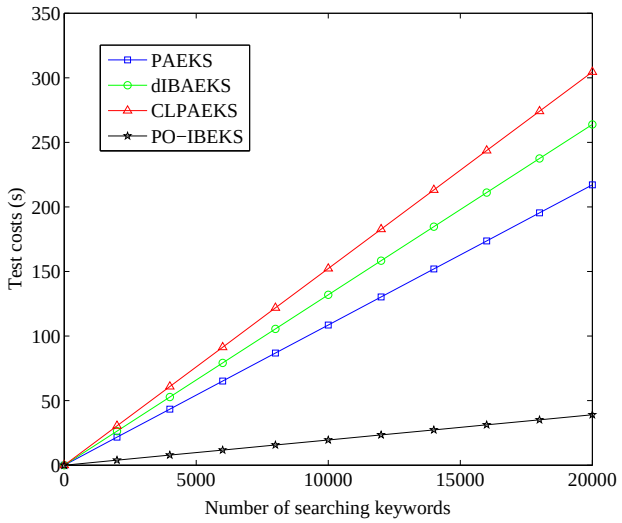

Figure 4: The comparison of test costs

cuting time in $\mathbb{G}_1$, Mul denotes a scalar multiplication executing time, mul denotes a general multiplication executing time, $hash$ denotes a general hash function executing time. The communication overhead comparison is listed in Table 2, including trapdoor and ciphertext size. The test costs comparison is also listed in Table 3. The implementation results in Figure 3 show that PO-IBEKS owns the ciphertext size almost as the same as existing schemes, while PO-IBEKS achieves much more efficient in terms of trapdoor size. Moreover, the implementation results in Figure 4 show that PO-IBEKS is much more light-weight than existing schemes in the comparison of test costs with the increasing number of searching keywords. In addition, PO-IBEKS resists against quantum computers and IKGA simultaneously, which can be much more practical in post-quantum secure cloud storage systems.

6. Conclusions and future work

In this paper, we have proposed an efficient proxy-oriented identity-based encryption with keyword search scheme (PO-IBEKS) for cloud storage. PO-IBEKS is constructed on lattice-based cryptography, achieving quantum computers resistance. PO-IBEKS enables an original data owner to authorize a trusted proxy to encrypt the kernel data as well as corresponding keywords, and upload them to the remote cloud server. We have proved the semantically security of PO-IBEKS, including ciphertext indistinguishability, existential unforgeability, and delegation security. Therefore, PO-IBEKS can resist against IKGA from misbehaved cloud servers. The security proof and performance evaluation demonstrate that PO-IBEKS is much more practical for post-quantum secure cloud storage systems. To further preserve data privacy against proxies, we will design an efficient proxy re-encryption with keyword search from lattices in our future work.

7. Acknowledgements

This work is supported by National Key R&D Program of China (No.2017 YFB0802000), National Natural Science Foundation of China (No.61872060), China Postdoctoral Science Foundation Funded Project (No.2017M623008), Sichuan Science and Technology Program (No. 2018GZ0102), Scientific Research Starting Project of SWPU (No.2017QHZ023), Natural Science Foundation of Fujian Province (2017J01502), and the State Scholarship Fund of China Scholarship Council (CSC).

8. References

[1] D. Song, E. Shi, I. Fischer, and U. Shankar, Cloud data protection for the masses, Computer. 45(1)(2012)39-45.

[2] Y. Cui, Z. Lai, X. Wang, N. Dai, and C. Miao, Quicksync: improving synchronization efficiency for mobile cloud storage services, in: Proceedings of International Conference on Mobile Computing and Networks, ACM, 2015, pp. 592-603.

[3] D. Zissis, D. Lekkas, Addressing cloud computing security issues, Future Generation Computer Systems. 28(3)(2012)583-592.

[4] 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. 36(7)(2014)180-186.

[5] C. Esposito, A. Castiglione, B. Martini, and K. K. R. Choo, Cloud manufacturing: Security, privacy, and forensic concerns, IEEE Cloud Computing. 3(2016)16-22.

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

[7] P. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, SIAM Journal on Computing. 26(5)(1997)1484-1509.

[8] T. D. Ladd, F. Jelezko, R. Laflamme, Y. Nakamura, C. Monroe, J. L. O. Brien, Quantum computers, Nature. 464(7285)(2012)45-53.

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

[10] J. Shao, Z. Cao, X. Liang, H. Lin, Proxy re-encryption with keyword search, Information Sciences. 180(2010)2576-2587.

[11] Li. Fang, W. Susilo, C. Ge, J. Wang, Chosen-ciphertext secure anonymous conditional proxy re-encryption with keyword search, Theoretical Computer Science. 462(2012)39-58.

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

[13] 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. 11(4)(2016)789-798.

[14] A. Shamir, Identity-based cryptosystems and signature schemes, in Proceedings of Advances in cryptology-CRYPTO 1984, Springer Berlin Heidelberg, 1984, pp. 47-53.

[15] H. Li, Q. Huang, J. Shen, G. Yang, W. Susilo. Designated-server identity-based authenticated encryption with keyword search for encrypted emails. Information Sciences. 481(2019)330-343.

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

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

[18] J. Li, X. Huang, J. Li, X. Chen, and Y. Xiang, Securely outsourcing attribute-based encryption with checkability, IEEE Transactions on Parallel and Distributed Systems. 25(8)(2014)2201-2210.

[19] J. Li, J. Li, X. Chen, C. Jia, and W. Lou, Identity-based encryption with outsourced revocation in cloud computing, IEEE Transactions on Computers. 64(2)(2015)425-437.

[20] Y. Zhang, D. Zheng, R. H. Deng, Security and privacy in smart health: efficient policy-hiding attribute-based access control, IEEE Internet of Things Journal. 5(3)(2018)2130-2145.

[21] X. Fu, X. Nie, T. Wu, F. Li, Large universe attribute based access control with efficient decryption in cloud storage system, Journal of Systems and Software. 135(2018)157-164.

[22] Y. Zhang, C. Xu, H. Li, K. Yang, J. Zhou, X. Lin, HealthDep: An efficient and secure deduplication scheme for cloud-assisted eHealth systems, IEEE Transactions on Industrial Informatical. 2018, PP(99): 1-1.

[23] 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. 12(3)(2017)676-688.

[24] X. Zhang, H. Wang, C. Xu. Identity-based key-exposure resilient cloud storage public auditing scheme from lattices, Information Sciences, 2018, doi: 10.1016/j.ins.2018.09.013.

[25] X. Chen, J. Li, J. Ma, J. Weng, and W. Lou, Verifiable computation over large database with incremental updates, IEEE Transactions on Computers. 65(10)(2016)3184-3195.

[26] Y. Zhang, X. Chen, J. Li, D. S. Wong, H. Li, and I. You, Ensuring attribute privacy protection and fast decryption for outsourced data security in mobile cloud computing, Information Sciences. 379(2017)42-61.

[27] X. Liu, B. Qin, R. H. Deng, and Y. Li, An efficient privacy-preserving outsourced computation over public data, IEEE Transactions on Services Computing. 10(5)(2017)756-770.

[28] X. Liu, K. K. R. Choo, R. H. Deng, R. Lu, and J. Weng, Efficient and privacy-preserving outsourced calculation of rational numbers, IEEE Transactions on Dependable and Secure Computing. 2018, PP(99), 1-1.

[29] 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.

[30] 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. 3(3)(2016)312-325.

[31] J. Wang, X. Chen, J. Li, J. Zhao, and J. Shen, Towards achieving flexible and verifiable search for outsourced database in cloud computing, Future Generation Computer Systems. 67(2017)266-275.

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

[33] 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. 2018, PP(99): 1-1.

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

[35] H. S. Rhee, J. H. Park, and D. H. Lee, Generic construction of designed tester public-key encryption with keyword search, Information Sciences. 205(2012)93-109.

[36] 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. 238(2013)221-241.

[37] K. Liang, W. Susilo, Searchable attribute-based mechanism with efficient data sharing for secure cloud storage, IEEE Transactions on Information Forensics and Security. 10(9)(2015)1981-1992.

[38] 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. 387(2017)116-131.

[39] Y. Miao, J. Ma, X. Liu, X. Li, Q. Jiang and J. Zhang, Attribute-based keyword search over hierarchical data in cloud computing, IEEE Transactions on Services Computing, 2017, doi: 10.1109/TSC.2017.2757467.

[40] Y. Miao, J. Weng, X. Liu, K. K. R. Choo, Z. Liu and H. Li, Enabling verifiable multiple keywords search over encrypted cloud data,Information Sciences, 2018, doi: 10.1016/j.ins.2018.06.066.

[41] 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. 62(11)(2013)2266-2277.

[42] 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. 11(12)(2016)2833-2842.

[43] 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. 14(2)(2017)759-767.

[44] M. Ma, D. He, M. K. Khan, and J. Chen, Certificateless searchable public key encryption scheme for mobile healthcare system, Computers and Electrical Engineering. 65(2017)413-424.

[45] 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. 2017, PP(99): 1-1.

[46] Q. Lai, B. Yang, Y. Yu, Y. Chen, and J. Bai, Novel smooth hash proof systems based on lattices, Computer Journal. 61(4)(2018)561-574.

[47] X. Zhang, C. Xu, L. M, J. Zhao. Identity-based encryption with keyword search from lattice assumption, China Communications. 15(4)(2018)164-178.

[48] X. Zhang, C. Xu, Trapdoor security lattice-based public-key searchable encryption with a designated cloud server, Wireless Personal Communications. 100(3)(2018)907-921.

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

[50] O. Regev, On lattices, learning with errors, random linear codes, and cryptography, Journal of the ACM. 56(6)(2009)1-40.

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

[52] J. Alwen, C. Peikert, Generating shorter bases for hard random lattices, Theory of Computing Systems. 48(3)(2011)535-553.

[53] 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.