Abstract The Internet of Things (IoT), an emerging paradigm, supports numerous devices to interconnect and exchange data via public networks. However, this paradigm remains weak in terms of security, as the exposed sensitive data has become a target for malicious attackers. Encrypting data before uploading it to remote devices is an effective measure. Attribute-Based Searchable Encryption (ABSE) ensures the confidentiality of data and possesses the capability to retrieve data. Unfortunately, previous ABSE schemes become impotent when attacked by quantum algorithms because they rely on classical assumptions. With this motivation, we present an ABSE scheme from Ideal Lattices (ABSEIL) for IoT, allowing authorized consumers to obtain desired data through the keyword search function. Furthermore, ABSEIL enables delegators to share data with delegates without disclosing any plaintext information by incorporating proxy re-encryption (PRE). We prove that ABSEIL is secure under the ring learning with errors (RLWE) and the ring small integer solution (RSIS) assumptions. Theoretical analyses state that our scheme involves appropriate computing and storage overhead, significantly reducing the transmission delay between IoT devices.

Keywords Attribute-based encryption, Keyword search, Proxy re-encryption, Ring learning with errors, Ring small integer solution

1 Introduction

The Internet of Things (IoT) has facilitated the development of numerous fields in our lives. It mainly focuses on framing a platform with extensive computing resources and storage services such that it is possible to exchange information among infrastructures [1, 2]. Fig. 1 illustrates the architecture of an IoT scenario. All members interact with the IoT platform through the Internet. Data owners outsource data to different IoT platforms. Data consumers located in the same or different locations can access the desired data in the IoT platforms through their applications. However, some devices in IoT environment lacks sufficient storage and data processing capabilities [1]. To this end, IoT platforms leverage the powerful computing capabilities of cloud servers to process the data within the IoT devices. As one of the momentous elements in the current IoT environment, cloud servers cannot provide a comprehensive method to guarantee data privacy as locally stored data.

Attribute-based encryption (ABE) provides flexible access control for data [3], such that data owners make unauthorized individuals lose access. Nevertheless, retrieving data becomes a thorny issue as the magnitude of data increases dramatically. In the standard ABE schemes, recipients are bound to download data from the cloud servers, decrypt it locally, and then retrieve the matching data [4]. This clearly imposes unnecessary consumption on resources. Attribute-Based Searchable Encryption (ABSE) offers flexible access control for data, enabling data owners to restrict unauthorized

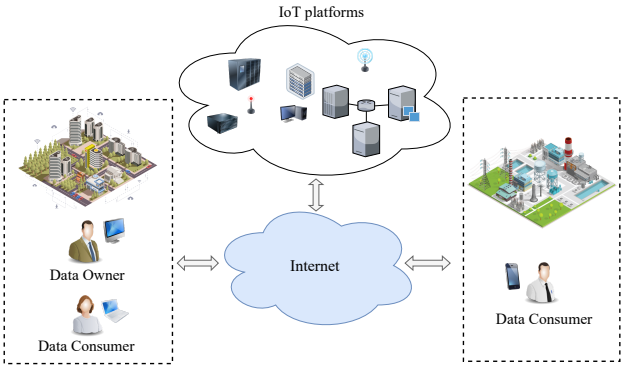

Fig. 1 Architecture of an IoT scenario

individuals from accessing data by imposing reasonable policies [5, 6]. Simultaneously, data consumers can directly retrieve desired data from the cloud server using search keys corresponding to keywords, instead of downloading the data locally for retrieval. In this context, both computation and storage costs for consumers are significantly improved.

However, when facing the threat of quantum algorithms, the underlying assumptions of the traditional ABSE appear powerless [7–9]. Although the platform operators are actively strengthening various measures for software and hardware of cloud servers to prevent malicious users from stealing data, it is challenging to apply complex protection methods to unsupervised edge devices in IoT scenarios due to their limited power capabilities. Furthermore, certain sensitive information is still monitored during the collection and transmission processes. Lattice-based ABSE is considered as a potential candidate in the post-quantum era [10–12]. From a real-world standpoint, efficiency is a critical metric, because IoT equipment generally has insufficient computing power. The aforesaid methods are inappropriate to practice because of involving quite heavy computational operations [13–15]. Therefore, efficient ABSE based on lattices play a critical role in ensuring data security within IoT scenarios.

Additionally, expecting IoT members to function continuously is impractical. For instance, in a healthcare system, each attending doctor needs to monitor the real-time health status of patients. The massive data complicates this motivation. Therefore, a practical solution is to integrate the proxy re-encryption technology [16], so that the doctor can designate an authorized assistant to assist with the task.

In this paper, we propose an offline/online attribute-based searchable encryption scheme from ideal lattices (ABSEIL). Benefiting from the keyword search function of ABSEIL, authorized consumers can efficiently retrieve the desired data with a lightweight search trapdoor. Through online/offline technology, complex arithmetic operations in encryption and key generation algorithms are pre-executed in the offline stage, and the online stage only involves a few arithmetic operations. Also, it incorporates the proxy re-encryption mechanism for completing end-to-end data sharing.

2 Preliminaries

We give the relevant notation required throughout the work.

2.1 Notation

Let $\mathcal{R} = \mathbb{Z}[x]/\langle x^n + 1\rangle$ be a cyclotomic polynomial ring, where each ring element is a polynomial of degree at most $n-1$, and its coefficients are integers. Throughout this paper, we take n as a power of 2. Also, $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$ denotes an irreducible integer polynomial, and q is the modulus. We state that for the security parameter λ, a function negl is negligible if $\text{negl}(\lambda) < 1/\lambda^c$ for a constant c. Moreover, we use several underlying notation as depicted in Table 1.

Table 1 Notation

Symbol	Definition
$\mathbf{a} \in \mathcal{R}_q^{1\times m}$	row vector whose elements are sampled from $\mathcal{R}_q$
$\mathbf{A} \in \mathcal{R}_q^{n\times m}$	matrix with n rows and m columns
$\mathbf{a}_i \in \mathcal{R}_q^n$	i-th column of matrix $\mathbf{A} \in \mathcal{R}_q^{n\times m}$
$[n]$	set $\{1, 2, \cdots, n\}$
$a \leftarrow \mathcal{S}$	a is sampled uniformly from a distribution (set) $\mathcal{S}$
$(\mathbf{a}\|\mathbf{b}) \in \mathcal{R}_q^{1\times(m_1+m_2)}$	the row concatenation of $\mathbf{a} \in \mathcal{R}_q^{1\times m_1}, \mathbf{b} \in \mathcal{R}_q^{1\times m_2}$
$(\mathbf{a};\mathbf{b}) \in \mathcal{R}_q^{(m_1+m_2)}$	the concatenation of $\mathbf{a} \in \mathcal{R}_q^{m_1}, \mathbf{b} \in \mathcal{R}_q^{m_2}$
$s_1(\mathbf{X})$	the spectral norm of the matrix $\mathbf{X}$
PPT	probabilistic polynomial time

2.2 Lattice

Given a basis $\mathbf{B} = (b_1, \cdots, b_n)$ in n-dimensional real space $\mathbb{R}^n$, a full rank lattice Λ is considered as an integer span of $\mathbf{B}$, defined as

$$\Lambda = \mathcal{L}(\mathbf{B}) = \left\{\mathbf{Bx} = \sum\nolimits_{i=1}^{n} x_i\mathbf{b}_i \mid \mathbf{x} \in \mathbb{Z}^n\right\}.$$

We also employ the two categories of q-ary lattices.

Definition 1. *For an uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and $\mathbf{u} \in \mathbb{Z}_q^n$, we define q-ary lattices as*

$$\Lambda_q^{\perp}(\mathbf{A}) = \{\mathbf{z} \in \mathbb{Z}^m \ s.t.\ \mathbf{Az} = 0 \ (\text{mod}q)\},$$
$$\Lambda_q^{\mathbf{u}}(\mathbf{A}) = \{\mathbf{z} \in \mathbb{Z}^m \ s.t.\ \mathbf{Az} = \mathbf{u} \ (\text{mod}q)\}.$$

2.3 Discrete gaussians

For a real $\sigma > 0$, the Gaussian function is defined as $\rho_{\mathbf{c},\sigma}(\mathbf{x}) = \exp\left(-\pi \|\mathbf{x}-\mathbf{c}\|/\sigma^2\right)$, where $\mathbf{c} \in \mathbb{R}^n$ is the center. Let $\rho_{\mathbf{c},\sigma}(\Lambda) = \sum_{\mathbf{x}\in\Lambda} \rho_{\mathbf{c},\sigma}(\mathbf{x})$ for lattice Λ. Define a discrete Gaussian distribution $\mathcal{D}_{\Lambda,\mathbf{c},\sigma}$ over Λ as:

$$D_{\Lambda,\mathbf{c},\sigma}(\mathbf{y}) = \frac{\rho_{\mathbf{c},\sigma}(\mathbf{y})}{\rho_{\mathbf{c},\sigma}(\Lambda)}, \ \forall \mathbf{y} \in \Lambda.$$

2.4 Hardness problems

Definition 2 (Decisional RLWE)**.** *Given a security parameter λ and an n-dimensional random cyclotomic polynomial $s \in \mathcal{R}_q$, the ring learning with errors assumption with the parameters n, q, σ requires that for any PPT adversary $\mathcal{A}$'s advantage*

$$Adv_{\mathcal{A}}(\lambda) = |\Pr[\mathcal{A}^{O_s} = 1] - \Pr[\mathcal{A}^{O_\$} = 1]|$$

is negligible for distinguishing the samples from oracles O_s and $O_\$$, where O_s samples $a \leftarrow \mathcal{R}_q$, $e \leftarrow \mathcal{D}_{\mathcal{R},\sigma}$ and returns $(a, as+e)$ for each query, while $O_\$$ samples $a \leftarrow \mathcal{R}_q$, $u \leftarrow \mathcal{R}_q$ and returns (a, u).

Definition 3 (RSIS)**.** *Given appropriate n, m, $\mathbf{a} \in \mathcal{R}_q^{1\times m}$ and $\tilde{B} < q$, the ring small integer solution (RSIS) problem requires to capture $\mathbf{r} \in \mathcal{R}_q^m$ conditioned on $\mathbf{ar} = 0$ and $0 < \|\mathbf{r}\| \leqslant \tilde{B}$. An algorithm $\mathcal{B}$'s advantage*

$$\Pr\left[0 = \mathbf{ar} \wedge 0 < \|\mathbf{r}\| \leqslant \tilde{B} \mid \mathbf{a} \in \mathcal{R}_q^{1\times m}, \mathbf{r} \leftarrow \mathcal{B}(\mathbf{a})\right]$$

in solving RSIS is $\text{negl}(\lambda)$.

2.5 Sampling algorithms

In this part, we represent the ring-based sampling algorithms of this work.

Theorem 1. *There is an algorithm RTrapGen that takes in a security parameter λ and a tag t, and outputs a ring vector $\mathbf{a}$ together with a trapdoor $\mathbf{T_a}$.*

The process of *RTrapGen* is stated in Algorithm 1. Step 1 relies on the parameter λ to determine the distribution parameter σ, the ring dimension n, the modulus q, and the dimension $\tilde{k}$ of the vector $\mathbf{g}^\top = (b^0, b^1, \cdots, b^{\tilde{k}-1})$. Steps 2 to 4 construct $\mathbf{a}$ and its trapdoor $\mathbf{T_a}$, where $\mathbf{T_a}$ contains two low-norm vectors $\boldsymbol{\varphi}$ and $\boldsymbol{\gamma}$.

Algorithm 1 RTrapGen

Input: The security parameter λ; The tag element $t \in \mathcal{R}_q$;
Output: The pseudo-random ring vector $\mathbf{a}$; The trapdoor $\mathbf{T_a}$ for $\mathbf{a}$;

1: Determine parameters σ, n, q, and calculate $\tilde{k} = \lfloor \log_b(q) + 1 \rfloor$ for $b \geqslant 2$.
2: Sample $a \leftarrow \mathcal{R}_q$ and construct $\tilde{\mathbf{a}} = [1, a]$.
3: Sample $\boldsymbol{\varphi} \leftarrow (\varphi_1, \cdots, \varphi_{\tilde{k}})$ and $\boldsymbol{\gamma} \leftarrow (\gamma_1, \cdots, \gamma_{\tilde{k}})$ for $\varphi_i, \gamma_i \in \mathcal{D}_{\mathcal{R},\sigma}$.
4: Compute $\mathbf{a} \leftarrow (\tilde{\mathbf{a}} \mid t\mathbf{g} - \tilde{\mathbf{a}}(\boldsymbol{\varphi};\boldsymbol{\gamma})^\top)$.
5: Return $(\mathbf{a}, \mathbf{T_a} = (\boldsymbol{\varphi};\boldsymbol{\gamma}))$.

Theorem 2. *There is an algorithm RSamplePre that takes in ring vector $\mathbf{a}$, a trapdoor $\mathbf{T_a}$, a syndrome u, distribution parameters σ, σ_s and a tag element t_p, and outputs a low-norm vector $\mathbf{y} \in \Lambda_q^u(\mathbf{a})$.*

The main objective of algorithm 2 is to produce a short ring vector $\mathbf{y}$, satisfying $\mathbf{ay} = u \in \mathcal{R}_q$, and the input $t_p = 1$ by default unless specified. The Perturb technology in step 1 is used to capture a perturbation vector $\mathbf{p}$. Then step 2 produces a vector $\mathbf{z}$ from G-lattice. By step 3, the vector $\mathbf{y}$ followed spherical distribution is generated.

Algorithm 2 RSamplePre

Input: The ring vector $\mathbf{a}$ and its trapdoor $\mathbf{T_a}$; The syndrome u; The distribution parameters σ and σ_s; The tag element t_p;

Output: The preimage $\mathbf{y}$;

1: Compute $\mathbf{p} \leftarrow \text{Perturb}(\mathbf{T_a}, (b+1)\sigma, \sigma_s, q, n)$.
2: Sample $\mathbf{z} \leftarrow \text{SampleG}\left(\sigma, q, t_p^{-1}(u - \mathbf{ap})\right)$.
3: Compute $\mathbf{y} \leftarrow [p_1 + \boldsymbol{\varphi}\mathbf{z}, p_2 + \boldsymbol{\gamma}\mathbf{z}, p_3 + z_1, \cdots, p_m + z_{\tilde{k}}]$.
4: Return $\mathbf{y}$.

Lemma 1. *Let $\mathbf{a} \in \mathcal{R}_q^{1\times m}$ and $\mathbf{b} \in \mathcal{R}_q^{1\times m_1}$ be two vectors. $\mathbf{T_a}$ and $\mathbf{T_b}$ are their trapdoors, $\mathbf{u} \in \mathcal{R}_q^{1\times \ell}$ is an uniform vector and $\mathbf{S} \in \{\pm 1\}^{m\times m_1}$. The low-norm vectors $\mathbf{e}$ and $\tilde{\mathbf{e}}$ sampled by RSampleLeft and RSampleRight respectively are indistinguishable.*

- *RSampleLeft$(\mathbf{a}, \mathbf{b}, \mathbf{T_a}, \mathbf{u}, \sigma, \sigma_s)$: On input ring vectors $\mathbf{a}$, $\mathbf{b}$, the trapdoor $\mathbf{T_a}$ of $\mathbf{a}$, a syndrome $\mathbf{u}$ and the parameters σ, σ_s, it produces a vector $\mathbf{e} \in \Lambda_q^{\mathbf{u}}(\mathbf{a} \mid \mathbf{b})$ such that $(\mathbf{a} \mid \mathbf{b}) \cdot \mathbf{e} = \mathbf{u} \pmod q$.*
- *RSampleRight$(\mathbf{a}, \mathbf{b}, \mathbf{T_b}, \mathbf{S}, \mathbf{u}, \sigma, \sigma_s)$: On input ring vectors $\mathbf{a}$, $\mathbf{b}$, the trapdoor $\mathbf{T_b}$ of $\mathbf{b}$, the matrix $\mathbf{S} \in \{\pm 1\}^{m\times m_1}$, a syndrome $\mathbf{u}$, and the parameters σ, σ_s, it produces a vector $\tilde{\mathbf{e}} \in \Lambda_q^{\mathbf{u}}(\mathbf{a} \mid \mathbf{aS} + \mathbf{b})$ such that $(\mathbf{a} \mid \mathbf{aS} + \mathbf{b}) \cdot \tilde{\mathbf{e}} = \mathbf{u} \pmod q$.*

Lemma 2. *Given any sequence of ring vectors $(\mathbf{b}_1, \cdots, \mathbf{b}_k) \in (\mathcal{R}_q^{1\times m})^k$, any family of functions $\mathcal{F} = \{f : \{0,1\}^k \to \{0,1\}\}$ with maximum depth d and $\mathbf{x} \in \{0,1\}^k$, there exists deterministic algorithms $(\text{Eval}_{pk}, \text{Eval}_{ct}, \text{Eval}_{sim})$ with the following properties:*

- $\mathbf{b}_f \leftarrow \text{Eval}_{pk}(f, \{\mathbf{b}_i\}_{i\in[k]})$*: It inputs a function $f \in \mathcal{F}$ and a sequence of ring vectors, outputs a ring vector $\mathbf{b}_f \in \mathcal{R}_q^{1\times m}$.*
- $\mathbf{c}_f \leftarrow \text{Eval}_{ct}(f, \{(\mathbf{b}_i, \mathbf{c}_i, x_i)\}_{i=1}^k)$*: It inputs a function f, $\mathbf{b}_i \in \mathcal{R}_q^{1\times m}$, $\mathbf{c}_i \in \mathcal{R}_q^m$ and $x_i \in \{0,1\}$ for $i \in [k]$, outputs a ring vector $\mathbf{c}_f \in \mathcal{R}_q^m$ satisfying $\mathbf{c}_f = (f(\mathbf{x})\mathbf{g} + \mathbf{b}_f)^\top s + \mathbf{e}_f$, if $\mathbf{c}_i = (\mathbf{b}_i + x_i\mathbf{g})^\top s + \mathbf{e}_i$ for $i \in [k]$, where $\mathbf{e}_f$ is a low-norm noise vector, and $\|\mathbf{e}_f\| < \Delta_f \cdot \max\{\|\mathbf{e}_i\|\}_{i\in[k]}$. The explanation of bound Δ_f refers to section 5.*
- $\mathbf{S}_f \leftarrow \text{Eval}_{sim}(f, \mathbf{a}, \{\mathbf{S}_i, x_i\}_{i\in[k]})$*: It inputs a function f, $\mathbf{a} \in \mathcal{R}_q^{1\times m}$, $\mathbf{S}_i \in \{\pm 1\}^{m\times m}$ and $x_i \in \{0,1\}$ for $i \in [k]$, outputs $\mathbf{S}_f \in \mathcal{R}_q^{m\times m}$ conditioned on $\mathbf{aS}_f + f(\mathbf{a})\mathbf{g} = \mathbf{b}_f$, where $\mathbf{b}_f = Eval_{pk}(f, \{\mathbf{aS}_i + x_i\mathbf{g}\}_{i\in[k]})$ and $\|\mathbf{S}_f\| < \Delta_f$.*

Lemma 3. *Given a matrix $\mathbf{X} \in \mathcal{R}^{m\times \ell}$ and a real $r > s_1(\mathbf{X})$, the $\mathbf{Xe} + NoiseReRand(\mathbf{X}, r)$ is distributed statistically close to $\mathcal{D}_{\mathcal{R}^m, 2\sigma r}$ for vector $\mathbf{e} \in \mathcal{D}_{\mathcal{R}^\ell, \sigma}$.*

We utilize Algorithm 3 to construct a challenge ciphertext with appropriate distribution. This enables the normal ciphertext to maintain the same distribution as the simulated one.

Algorithm 3 NoiseReRand [9]

Input: The matrix $\mathbf{X} \in \mathcal{R}^{m\times \ell}$; The distribution parameter $r \in \mathbb{R}^+$;

Output: The ring vector $\hat{\mathbf{e}} \in \mathcal{R}_q^m$;

1: Sample $\mathbf{e} \leftarrow \mathcal{D}_{\mathcal{R}^\ell, \sigma}$ and $\tilde{\mathbf{e}} \leftarrow \mathcal{D}_{\mathcal{R}^m, \sqrt{2}\sigma}$.
2: Compute $\mathbf{e}_1 = \mathbf{Xe} + \tilde{\mathbf{e}}\sqrt{(r^2\mathbf{1}_m - \mathbf{XX}^\top)}$.
3: Sample $\mathbf{e}_2 \leftarrow \mathcal{D}_{\mathcal{R}^m - \mathbf{e}_1, \sqrt{2}\sigma r}$.
4: Return $\hat{\mathbf{e}} = \mathbf{e}_1 + \mathbf{e}_2$.

3 The definition of ABSEIL

As described in Fig.2, ABSEIL involves five types of participants: central authority (*CA*), data owner (*DO*), data consumer (*DC*), proxy server (*PS*) and cloud server (*CS*). These entities are described as follows.

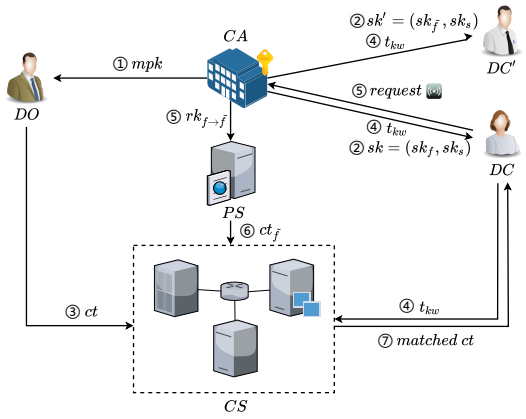

Fig. 2 The system model of ABSEIL

- *CA* plays the role of initializing the system. Then it generates the master public key (mpk) for all entities together with the master secret key (msk) for itself. Also, it grants secret key sk (sk') to all data consumers (See ① and ②).
- After capturing mpk, *DO* determines a keyword and encrypts data, then uploads the ciphertext ct to *CS* (See ③).
- Before decrypting ct, *DC* requests a searchable trapdoor t_{kw} from *CA* and forwards it to *CS* to search for matched ct. Moreover, an offline *DC* can initiate a request to *CA* to produce a re-encryption key $rk_{f\to\tilde{f}}$. Then *CA* transfers it to *PS*. In this setting, the *DC* delegates the decryption right to the *DC′* through the authorization of *CA* (See ④ and ⑤).
- *PS* utilizes $rk_{f\to\tilde{f}}$ to produce a fresh re-encrypted ciphertext $ct_{\tilde{f}}$, then transfers $ct_{\tilde{f}}$ to *CS* (See ⑥).
- *CS* supplies storage services. Furthermore, it utilizes t_{kw} to search for matched ct for *DC* (See ⑦).

3.1 Syntax of ABSEIL

For a function family $\mathcal{F}$, a predicate space $\mathcal{Y}$ and a message space $\mathcal{M}$, the syntax of an ABSEIL scheme is composed of the following algorithms:

$Setup(1^\lambda, 1^\ell) \rightarrow (mpk, msk)$: It is performed by CA. Given the security parameter λ, the length of message vectors ℓ, it generates a pair of master public key and master secret key as (mpk, msk).

$KeyGen_{off}(mpk, msk) \rightarrow sk_{off}$: It is performed by CA. Given mpk and msk, it outputs the offline secret key sk_{off}.

$KeyGen_{on}(mpk, msk, sk_{off}, f, \mathbf{y}) \rightarrow sk$: It is performed by CA. Given mpk, msk, sk_{off}, the policy function $f \in \mathcal{F}$, and a predicate vector $\mathbf{y}$, it produces secret key sk that contains decryption key sk_f and search key sk_s.

$Enc_{off}(mpk) \rightarrow ct_{off}$: It is performed by DO. Given mpk, it returns the offline ciphertext ct_{off}.

$Enc_{on}(mpk, ct_{off}, \mathbf{x}, \mathbf{m}, k_w) \rightarrow ct$: It is performed by DO. Given mpk, ct_{off}, the attribute vector $\mathbf{x} \in \{0,1\}^k$, the message $\mathbf{m} \in \mathcal{M}$ and the keyword $k_w \in \mathbb{Z}_q^n$, this algorithm generates the ciphertext ct.

$Trapdoor(mpk, msk, sk_s, k'_w) \rightarrow t_{kw}$: It is performed by CA. Given mpk, msk, search key sk_s and a keyword k'_w, it produces the search trapdoor t_{kw}.

$Search(mpk, ct, t_{kw}) \rightarrow (1/0)$: It is performed by CS. Given mpk, ct and t_{kw}, it outputs 1, if $k_w = k'_w$ holds; otherwise outputs 0.

$Decrypt(mpk, ct, sk_f) \rightarrow (\mu/\perp)$: It is performed by DC. Given mpk, ct and sk_f corresponding to the policy function f, DO can obtain the message μ, which represents the inner product of vectors $\mathbf{m}$ and $\mathbf{y}$ ($\mu = \langle \mathbf{m}, \mathbf{y} \rangle$), if $f(\mathbf{x}) = 0$. Otherwise, it returns a distinctive symbol $\perp$.

$ReKeyGen(mpk, msk, \tilde{f}) \rightarrow rk_{f \rightarrow \tilde{f}}$: It is performed by CA. Given mpk, msk and $\tilde{f}$, it produces the re-encryption key $rk_{f \rightarrow \tilde{f}}$.

$ReEnc(mpk, ct, rk_{f \rightarrow \tilde{f}}, \mathbf{x}) \rightarrow ct_{\tilde{f}}$: It is performed by PS. Given mpk, ct, $rk_{f \rightarrow \tilde{f}}$ and $\mathbf{x} \in \{0,1\}^k$, it generates a re-encrypted ciphertext $ct_{\tilde{f}}$ corresponding to the function $\tilde{f}$.

$ReDecrypt(ct_{\tilde{f}}, sk_{\tilde{f}}) \rightarrow (\mu/\perp)$: It is performed by DC. Given $ct_{\tilde{f}}$ and $sk_{\tilde{f}}$ of function $\tilde{f}$, it recovers μ, if $f(\mathbf{x}) = 0$, otherwise returns a distinctive symbol $\perp$.

Correctness. We state that ABSEIL is correct if the following constraints hold:

For $\forall (mpk, msk) \leftarrow Setup(1^\lambda, 1^\ell)$, $sk_{off} \leftarrow KeyGen_{off}(mpk, msk)$, $sk \leftarrow KeyGen_{on}(mpk, msk, sk_{off}, f, \mathbf{y})$, $ct_{off} \leftarrow Enc_{off}(mpk)$ and $ct \leftarrow Enc_{on}(mpk, ct_{off}, \mathbf{x}, \mathbf{m}, k_w)$, where $\mathbf{x} \in \{0,1\}^k$, $k_w \in \mathbb{Z}_q^n$, $\mathbf{y} \in \mathcal{Y}$ and $\mathbf{m} \in \mathcal{M}$. If $f(\mathbf{x}) = 0$ holds, we have that

$$\Pr\left[Decrypt(mpk, ct, sk_f) \rightarrow \mu\right] = 1 - \mathrm{negl}(\lambda).$$

For $\forall sk \leftarrow KeyGen_{on}(mpk, msk, sk_{off}, \tilde{f}, \mathbf{y})$, $rk_{f \rightarrow \tilde{f}} \leftarrow ReKeyGen(mpk, msk, \tilde{f})$, $ct_{\tilde{f}} \leftarrow ReEnc(mpk, ct, rk_{f \rightarrow \tilde{f}}, \mathbf{x})$, where $f, \tilde{f} \in \mathcal{F}$. If $f(\mathbf{x}) = 0$ holds, we have that

$$\Pr\left[ReDecrypt(ct_{\tilde{f}}, sk_{\tilde{f}}) \rightarrow \mu\right] = 1 - \mathrm{negl}(\lambda).$$

For $\forall t_{kw} \leftarrow Trapdoor(mpk, msk, sk_s, k'_w)$, where $k'_w \in \mathbb{Z}_q^n$. If $k'_w = k_w$ holds, we have that

$$\Pr\left[Search(mpk, ct, t_{kw}) \rightarrow 1\right] = 1 - \mathrm{negl}(\lambda).$$

3.2 Security definition

The security requirements of this work are to resist CPA and CKA. We now illustrate the security game of selective indistinguishability under CPA (Sel-IND-CPA), which contains a sequence of interacting operations between challenger C and adversary $\mathcal{A}$. In the reduction, let $KeyGen$ represent $KeyGen_{off}$ and $KeyGen_{on}$, and Enc represents Enc_{off} and Enc_{on}. The game $Game_{CPA}(\lambda, C, \mathcal{A})$ is depicted as follows.

Init. $\mathcal{A}$ declares a challenge attribute vector $\mathbf{x}^*$ and a challenge policy f^*.

Setup. Given the parameters λ and ℓ, C performs algorithm $(mpk, msk) \leftarrow Setup(1^\lambda, 1^\ell)$, then publishes the mpk to $\mathcal{A}$.

Query phase. $\mathcal{A}$ can adaptively perform the following queries:

- Key query. $\mathcal{A}$ submits $(f, \mathbf{y})$ to C. Then C returns $sk \leftarrow KeyGen(mpk, msk, f, \mathbf{y})$.
- Re-encryption key query. $\mathcal{A}$ sends $(f, \tilde{f}, \mathbf{y})$ to C. Then C returns $rk_{f \rightarrow \tilde{f}} \leftarrow ReKeyGen(mpk, msk, \tilde{f})$.
- Re-encryption query. $\mathcal{A}$ sends $(ct, \mathbf{x}, \mathbf{y}, f, \tilde{f})$ to C. Then C produces a re-encryption key $rk_{f \rightarrow \tilde{f}}$, and returns $ct_{\tilde{f}} \leftarrow ReEnc(mpk, ct, rk_{f \rightarrow \tilde{f}}, \mathbf{x})$.

Challenge query. $\mathcal{A}$ first sends two messages $(\mathbf{m}_0, \mathbf{m}_1)$ to C. Then C randomly picks $\beta \in \{0,1\}$, and returns $ct^* \leftarrow Enc(mpk, \mathbf{x}^*, \mathbf{m}_\beta, k_w)$.

Guess Phase. $\mathcal{A}$ outputs a guess bit β'.

The advantage of advantage $\mathcal{A}$ in the above game is represented as $Adv^{ABSEIL}_{CPA-\mathcal{A}}(\lambda) = |\Pr[\beta' = \beta] - 1/2|$.

Definition 4. *A scheme is Sel-IND-CPA secure if any adversary has at most a negligible advantage $Adv^{ABSEIL}_{CPA-\mathcal{A}}(\lambda)$ in winning $Game_{CPA}(\lambda, C, \mathcal{A})$.*

The definition of Sel-IND-CKA demonstrates that a malicious server cannot capture keyword information from the ciphertext, even if it has a search trapdoor. The game $Game_{CKA}(\lambda, C, \mathcal{A})$ is as follows.

Setup. On input the parameters λ and ℓ, C invokes $(mpk, msk) \leftarrow Setup(1^\lambda, 1^\ell)$ and published mpk to $\mathcal{A}$.

Query phase. $\mathcal{A}$ conducts the following queries:

- Key query. $\mathcal{A}$ queries to C on $(f, \mathbf{y})$, then obtains $sk \leftarrow KeyGen(mpk, msk, f, \mathbf{y})$ from C.
- Trapdoor query. $\mathcal{A}$ queries to C on search trapdoor t_{kw} of the keyword k_w, then obtains $t_{kw} \leftarrow Trapdoor(mpk, msk, sk_s, k_w)$ from C.

Ciphertext query. $\mathcal{A}$ first requests the ciphertext for $(k_w, \mathbf{x})$. Then C constructs the consequential result ct.

Forgery Phase. $\mathcal{A}$ forges a ciphertext ct^* relevant to keyword k^*_w and attribute vector $\mathbf{x}^*$. If $Search(mpk, ct^*, t_{kw}) \rightarrow 1$ holds, then $\mathcal{A}$ wins this game. The advantage of adversary $\mathcal{A}$ in the above game is represented as $Adv^{ABSEIL}_{CKA-\mathcal{A}}(\lambda)$.

Definition 5. *A scheme is Sel-IND-CKA secure if any adversary has at most a negligible advantage* $Adv^{ABSEIL}_{CKA-\mathcal{A}}(\lambda)$ *in winning* $Game_{CKA}(\lambda, C, \mathcal{A})$.

4 The concrete construction of ABSEIL

For a function family $\mathcal{F} = \{f : \{0,1\}^k \to \{0,1\}\}$, a predicate space $\mathcal{Y} = \{0, \cdots, B_y(\lambda)\}^\ell$ and a message space $\mathcal{M} = \{0, \cdots, B_m(\lambda)\}^\ell$, ABSEIL requires that the result $|\langle \mathbf{y}, \mathbf{m} \rangle| < K$, where the bound $K = \ell B_y B_m$.

$Setup(1^\lambda, 1^\ell) \to (mpk, msk)$: On input a security parameter λ, the length of attribute vectors ℓ, then perform:

1. Determine the parameters n, m, q, k, τ depending on λ, the hash function $H : \mathbb{Z}_q^n \to \mathcal{R}_q$, where $\forall \zeta, \nu \in \mathbb{Z}_q^n$, $H(\zeta) - H(\nu)$ is full rank.
2. Invoke $(\mathbf{a}, \mathbf{T_a}) \leftarrow RTrapGen(1^\lambda, t = 1)$, where $\mathbf{a} \in \mathcal{R}_q^{1\times m}$, $\mathbf{T_a} = (\boldsymbol{\varphi}, \boldsymbol{\gamma})$.
3. Perform $(\mathbf{a}_s, \mathbf{T}_{\mathbf{a}_s}) \leftarrow RTrapGen(1^\lambda, t_s = 0)$ and $(\mathbf{a}_u, \mathbf{T}_{\mathbf{a}_u}) \leftarrow RTrapGen(1^\lambda, t_u = 0)$, where $\mathbf{a}_s = (\hat{\mathbf{a}} \mid -\hat{\mathbf{a}}\mathbf{T}_{\mathbf{a}_s})$ and $\mathbf{a}_u = (\hat{\mathbf{a}} \mid -\hat{\mathbf{a}}\mathbf{T}_{\mathbf{a}_u})$. Then transfer $\mathbf{T}_{\mathbf{a}_u}$ to data owner DO.
4. Sample a vector $\mathbf{u} \leftarrow \mathcal{R}_q^{1\times\ell}$ and a sequence of ring vectors $(\mathbf{b}_1, \cdots, \mathbf{b}_k) \in (\mathcal{R}_q^{1\times m})^k$.
5. Construct $\mathbf{K}_s = [\hat{\mathbf{k}}_1; \hat{\mathbf{k}}_2] \in \mathcal{D}_{\mathcal{R}^{\ell\times 2}, \sigma_s}$ and $\mathbf{b}_s = [h_r; 1] \in \mathcal{R}_q^2$ for $h_r \in \mathcal{R}_q$, then compute $\mathbf{h}_s = \mathbf{u}\mathbf{K}_s$.
6. Output $mpk = (\mathbf{a}, \mathbf{a}_u, \mathbf{a}_s, \{\mathbf{b}_i\}_{i\in[k]}, \mathbf{h}_s, \mathbf{b}_s, \mathbf{u})$ and $msk = (\mathbf{T_a}, \mathbf{T}_{\mathbf{a}_s}, \mathbf{K}_s)$.

$KeyGen_{off}(mpk, msk) \to sk_{off}$: It receives mpk and msk, then conducts following processes:

1. Sample the matrix $\mathbf{E}_{off} \leftarrow \mathcal{D}_{\mathcal{R}^{m\times\ell}, \sigma_r}$, and the ring vectors $\mathbf{p}_i \in \mathcal{R}_q^m$ for $i \in [\ell]$ by invoking $\mathbf{p}_i \leftarrow \text{Perturb}(\mathbf{T_a}, (b+1)\sigma, \sigma_r, q)$ according step 1 of the $RSamplePre$.
2. Return $sk_{off} = (\mathbf{E}_{off}, \{\mathbf{p}_i\}_{i\in[\ell]})$.

$KeyGen_{on}(mpk, msk, sk_{off}, f, \mathbf{y}) \to sk$: It inputs mpk, msk, sk_{off}, a function $f \in \mathcal{F}$ and a predicate vector $\mathbf{y}$, then does the following processes:

1. Compute $\mathbf{b}_f \leftarrow \text{Eval}_{pk}(f, \{\mathbf{b}_i\}_{i\in[k]})$, where $\mathbf{b}_f \in \mathcal{R}_q^{1\times m}$.
2. Let $\boldsymbol{\rho} = \mathbf{u} - \mathbf{b}_f\mathbf{E}_{off} \in \mathcal{R}_q^{1\times\ell}$. For $i \in [\ell]$, generate $\mathbf{w}_i \in \mathcal{R}_q^{\tilde{k}}$ as $\mathbf{w}_i \leftarrow \text{SampG}(\rho_i - \mathbf{ap}_i, \sigma, q)$ by step 2 of the $RSamplePre$.
3. For $i \in [\ell]$, compute $\mathbf{r}_i \in \mathcal{D}_{\mathcal{R}^m, \sigma_s}$ as $\mathbf{r}_i = [p_1^{(i)} + \boldsymbol{\varphi}\mathbf{w}_i; p_2^{(i)} + \boldsymbol{\gamma}\mathbf{w}_i; p_3^{(i)} + w_1; \cdots; p_m^{(i)} + w_{\tilde{k}}]$.
4. Assemble $\mathbf{E}_{on} = [\mathbf{r}_1 \mid \cdots \mid \mathbf{r}_m] \in \mathcal{D}_{\mathcal{R}^{m\times\ell}, \sigma_s}$, then construct $\mathbf{E}_f = (\mathbf{E}_{on}; \mathbf{E}_{off})$ satisfying $[\mathbf{a} \mid \mathbf{b}_f] \cdot \mathbf{E}_f = \mathbf{u}$.
5. Produce decryption key $sk_f = \mathbf{E}_f\mathbf{y}$ and search key $sk_s = \mathbf{K}_s\mathbf{b}_s$.
6. Return $sk = (sk_f, sk_s)$.

$Enc_{off}(mpk) \to ct_{off}$: It takes in mpk, then performs:

1. Sample $s \leftarrow \mathcal{R}_q, \chi \leftarrow \mathcal{D}_{\mathcal{R},\sigma}, \mathbf{e}_1 \leftarrow \mathcal{D}_{\mathcal{R}^\ell,\sigma}, \mathbf{e}_2 \leftarrow \mathcal{D}_{\mathcal{R}^\ell,\tau}, \mathbf{e}_3 \leftarrow \mathcal{D}_{\mathcal{R}^m,\sigma}$ and $\mathbf{S}_i \leftarrow \{\pm 1\}^{m\times m}$ for $i \in [k]$.
2. Generate $\mathbf{e}_0 = (\mathbf{e}_3^\top \mid \mathbf{e}_3^\top\mathbf{S}_1 \mid \cdots \mid \mathbf{e}_3^\top\mathbf{S}_k)^\top$.
3. Return $ct_{off} = (s, \mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2, \{\mathbf{S}_i\}_{i\in[k]})$.

$Enc_{on}(mpk, ct_{off}, \mathbf{x}, \mathbf{m}, k_w) \to ct$: On input mpk, ct_{off}, an attribute vector $\mathbf{x} \in \{0,1\}^k$, a message $\mathbf{m} \in \mathcal{M}$ and a keyword $k_w \in \mathbb{Z}_q^n$, perform:

1. Construct $\mathbf{v} = (\mathbf{a} \mid \mathbf{b}_1 + x_1\mathbf{g} \mid \cdots \mid \mathbf{b}_k + x_k\mathbf{g})$.
2. Compute $\hat{\mathbf{c}}_0 = \mathbf{v}^\top s + \mathbf{e}_0 \in \mathcal{R}_q^{m(k+1)}$ and $\hat{\mathbf{c}}_1 = \mathbf{u}^\top s + \mathbf{e}_1 + \mathbf{e}_2 + \lfloor q/K \rfloor \mathbf{m} \in \mathcal{R}_q^\ell$.
3. Generate $t_w = H(k_w)$ and $\xi = \mathbf{h}_s\mathbf{b}_s$.
4. Compute $\mathbf{a}_w = \mathbf{a}_s + (\mathbf{0} \mid t_w\mathbf{g}) = (\hat{\mathbf{a}} \mid t_w\mathbf{g} - \hat{\mathbf{a}}\mathbf{T}_{\mathbf{a}_s})$ and $\mathbf{a}_\vartheta = \mathbf{a}_u + (\mathbf{0} \mid \xi\mathbf{g}) = (\hat{\mathbf{a}} \mid \xi\mathbf{g} - \hat{\mathbf{a}}\mathbf{T}_{\mathbf{a}_u})$.
5. Sample $\mathbf{c}_\vartheta \leftarrow RSamplePre(\mathbf{a}_\vartheta, \mathbf{T}_{\mathbf{a}_u}, 0, \sigma, \sigma_s, \xi)$ conditioned on $\mathbf{a}_\vartheta\mathbf{c}_\vartheta = 0 \pmod q$.
6. Set $\mathbf{c}_w = \mathbf{a}_w^\top s + \mathbf{e}_3$ and $\hat{c} = \xi s + \chi$.
7. Return $ct = (\hat{\mathbf{c}}_0, \hat{\mathbf{c}}_1, \mathbf{c}_w, \mathbf{c}_\vartheta, \hat{c})$.

$Trapdoor(mpk, msk, sk_s, k'_w) \to t_{kw}$: On input mpk, msk, sk_s and a keyword k'_w, it conducts:

1. Generate $t'_w = H(k'_w)$ and $\hat{\xi} = \mathbf{u} \cdot sk_s \in \mathcal{R}_q$.
2. Calculate $\hat{\mathbf{a}}_w = \mathbf{a}_s + (\mathbf{0} \mid t'_w\mathbf{g})$.
3. Sample $\mathbf{e}_w \leftarrow RSamplePre(\hat{\mathbf{a}}_w, \mathbf{T}_{\mathbf{a}_s}, \hat{\xi}, \sigma, \sigma_s, t'_w)$ conditioned on $\hat{\mathbf{a}}_w\mathbf{e}_w = \hat{\xi}$.
4. Return $t_{kw} = (\mathbf{e}_w, \hat{\xi})$.

$Search(mpk, ct, t_{kw}) \to (1/0)$: Takes as input mpk, ct and t_{kw}, then it performs:

1. Compute $\hat{\mathbf{a}}_\vartheta = \mathbf{a}_u + (\mathbf{0} \mid \hat{\xi}\mathbf{g}) = (\hat{\mathbf{a}} \mid \hat{\xi}\mathbf{g} - \hat{\mathbf{a}}\mathbf{T}_{\mathbf{a}_u})$.
2. Compute $\delta = \hat{c} - \mathbf{e}_w^\top\mathbf{c}_w$.
3. If $|\delta| < \lfloor q/4 \rfloor$ and $\hat{\mathbf{a}}_\vartheta\mathbf{c}_\vartheta = 0$ hold, it returns 1; otherwise returns 0.

$Decrypt(mpk, ct, sk_f) \to (\mu/\perp)$: It inputs mpk, ct and sk_f. Then execute the following procedures:

1. Parse $ct = (\hat{\mathbf{c}}_0, \hat{\mathbf{c}}_1, \mathbf{c}_w, \mathbf{c}_\vartheta, \hat{c})$, where the element $\hat{\mathbf{c}}_0 = \{\mathbf{c}_i\}_{i\in[0,k]} \in \mathcal{R}_q^{m(k+1)}$.
2. Perform $\mathbf{c}_f \leftarrow \text{Eval}_{ct}(f, \{(\mathbf{b}_i, \mathbf{c}_i, x_i)\}_{i\in[k]})$.
3. Output $\mu = Round_{\lfloor q/K \rfloor}\left(\mathbf{y}^\top\hat{\mathbf{c}}_1 - sk_f^\top(\mathbf{c}_0; \mathbf{c}_f)\right)$, where $Round_{\lfloor q/K \rfloor}$ denotes rounding to $\lfloor q/K \rfloor$.

$ReKeyGen(mpk, msk, \tilde{f}) \to rk_{f\to\tilde{f}}$: It takes in mpk, msk and $\tilde{f}$. Then proceeds as follows:

1. Perform $\mathbf{b}_f \leftarrow \text{Eval}_{pk}(f, \{\mathbf{b}_i\}_{i\in[k]})$ and $\mathbf{b}_{\tilde{f}} \leftarrow \text{Eval}_{pk}(\tilde{f}, \{\mathbf{b}_i\}_{i\in[k]})$.
2. Sample $\mathbf{E}_{f\to\tilde{f}} \leftarrow RSampleLeft(\mathbf{a}, \mathbf{b}_f, \mathbf{T_a}, (\mathbf{a}|\mathbf{b}_{\tilde{f}}), \sigma, \sigma_s)$, such that $(\mathbf{a} \mid \mathbf{b}_f)\mathbf{E}_{f\to\tilde{f}} = (\mathbf{a} \mid \mathbf{b}_{\tilde{f}})$ for $\mathbf{E}_{f\to\tilde{f}} \in \mathcal{D}_{\mathcal{R}_q^{2m\times 2m}, \sigma_s}$.
3. Generate $rk_{f\to\tilde{f}} = (\mathbf{E}_{f\to\tilde{f}}, f, \tilde{f})$.

$ReEnc(mpk, ct, rk_{f\to\tilde{f}}, \mathbf{x}) \to ct_{\tilde{f}}$: On input mpk, a ciphertext ct, $rk_{f\to\tilde{f}}$ and $\mathbf{x}$, perform:

1. Parse the ciphertext $ct = (\hat{\mathbf{c}}_0, \hat{\mathbf{c}}_1, \mathbf{c}_w, \mathbf{c}_\vartheta, \hat{c})$.
2. Execute $\mathbf{c}_f \leftarrow \text{Eval}_{ct}(f, \{(\mathbf{b}_i, \mathbf{c}_i, x_i)\}_{i\in[k]})$.
3. Compute $\tilde{\mathbf{c}} = (\mathbf{c}_0; \mathbf{c}_f) \in \mathcal{R}_q^{2m}$ and $\hat{\mathbf{c}}'_0 = \mathbf{E}_{f\to\tilde{f}}^\top\tilde{\mathbf{c}} \in \mathcal{R}_q^{2m}$.
4. Output $ct_{\tilde{f}} = (\hat{\mathbf{c}}'_0, \hat{\mathbf{c}}_1, \mathbf{c}_w, \mathbf{c}_\vartheta, \hat{c})$.

$ReDecrypt(ct_{\tilde{f}}, sk_{\tilde{f}}) \to (\mu/\perp)$: The algorithm inputs the re-encrypted ciphertext $ct_{\tilde{f}}$ and the secret key $sk_{\tilde{f}}$ corresponding to the function $\tilde{f}$. Perform as the following manners:

1. Parse the ciphertext $ct_{\tilde{f}} = (\hat{\mathbf{c}}'_0, \hat{\mathbf{c}}_1, \mathbf{c}_w, \mathbf{c}_\vartheta, \hat{c})$.
2. Output $\mu = Round_{\lfloor q/K \rfloor}\left(\mathbf{y}^\top\hat{\mathbf{c}}_1 - sk_{\tilde{f}}^\top\hat{\mathbf{c}}'_0\right)$.

5 Correctness and parameters of ABSEIL

We estimate $\sigma \approx \sqrt{\ln(2\tilde{n}_{\max}/\epsilon)}$, where $\tilde{n}_{\max}$ is the maximum ring dimension and $\epsilon \in (0,1)$ is the bound of error for rounding process. More precisely, we take $\sigma = 4.578$. The parameter σ_s satisfies $\sigma_s > C_0 \cdot \sigma^2 \cdot (\sqrt{n\tilde{k}} + \sqrt{2n} + 4.7)$, where $\tilde{k} = \lfloor \log_b(q) + 1 \rfloor$ and the constant $C_0 = 1.7$.

In the $KeyGen_{on}$ process, the matrices $\mathbf{E}_f$ and $\mathbf{K}_s$ follow the distributions $\mathcal{D}_{\Lambda_q^{\mathbf{u}}(\mathbf{a}|\mathbf{b}_f),\sigma_s}$ and $\mathcal{R}_q^{\ell\times 2}$, respectively. Therefore, we can estimate the bound of $\mathbf{E}_f$ as $\|\mathbf{E}_f\| < \sigma_s\sqrt{2nm\ell}$. Additionally, the bound of $\mathbf{K}_s$ is $\|\mathbf{K}_s\| < \sigma_s\sqrt{n\ell}$. In the $Trapdoor$ algorithm, $\mathbf{e}_w \in \mathcal{D}_{\mathcal{R}^m,\sigma_s}$, so the bound of $\mathbf{e}_w$ is $\sigma_s\sqrt{nm}$. Analogously, for matrix $\mathbf{E}_{f\to\tilde{f}} \in \mathcal{D}_{\mathcal{R}^{2m\times 2m},\sigma_s}$ in the re-encryption key $rk_{f\to\tilde{f}}$, we have $(\mathbf{a} \mid \mathbf{b}_f)\cdot\mathbf{E}_{f\to\tilde{f}} = (\mathbf{a} \mid \mathbf{b}_{\tilde{f}})$. Thus, the bound of $\mathbf{E}_{f\to\tilde{f}}$ satisfies $\|\mathbf{E}_{f\to\tilde{f}}\| < 2m\sigma_s$.

For the ciphertext ct, each element $\mathbf{c}_i = (\mathbf{b}_i + x_i\mathbf{g})^\top s + \mathbf{e}_i$, where $i \in [k]$ and $\mathbf{e}_i \in \mathcal{D}_{\mathcal{R}^m,\sigma}$. According to lemma 2, the bound for the error vector $\mathbf{e}_f$ is $\|\mathbf{e}_f\| < \Delta_f \cdot \max\{\|\mathbf{e}_i\|\}$, where $\Delta_f = C'\sigma(b\sqrt{nm})^d$ with a constant C', b is the base and $d = \lceil \log_2 k \rceil$. In the Enc_{off} algorithm, the error vectors $\mathbf{e}_1$, $\mathbf{e}_2$ and $\mathbf{e}_3$ are chosen from the distributions $\mathcal{D}_{\mathcal{R}^\ell,\sigma}$, $\mathcal{D}_{\mathcal{R}^\ell,\tau}$ and $\mathcal{D}_{\mathcal{R}^m,\sigma}$, respectively. Therefore, it follows that $\|\mathbf{e}_1\| < \sigma\sqrt{n\ell}$, $\|\mathbf{e}_2\| < \tau\sqrt{n\ell}$ and $\|\mathbf{e}_3\| < \sigma\sqrt{nm}$, where $\tau > C_0\sigma\sigma_s\Delta_f(2\sqrt{nm} + \sqrt{\ell})$. In the Enc_{on} phase, it requires that $\|\mathbf{c}_\vartheta\| < \sigma_s\sqrt{nm}$ if $\mathbf{a}_\vartheta\mathbf{c}_\vartheta = 0 \pmod q$ holds.

In the search process, if $k_w = k'_w$ holds, we have that

$$\begin{aligned}\delta &= \hat{c} - t_{kw}^\top \mathbf{c}_w \\ &= \xi s + \chi - \mathbf{e}_w^\top(\mathbf{a}_w^\top s + \mathbf{e}_3) \\ &= \chi - \mathbf{e}_w^\top \mathbf{e}_3,\end{aligned}$$

the δ is error. Hence, $|\delta| < \lfloor q/4 \rfloor$ holds, if the bound of $\mathbf{e}_w^\top\mathbf{e}_3$ satisfies $\|\mathbf{e}_w^\top\mathbf{e}_3\| < \sigma\sigma_s nm$.

During the decrypting process, if $f(\mathbf{x}) = 0$ holds, the result μ is computed as follows:

$$\begin{aligned}\mu &= \mathbf{y}^\top\hat{\mathbf{c}}_1 - sk_f^\top(\mathbf{c}_0;\mathbf{c}_f) \\ &= \mathbf{y}^\top(\mathbf{u}^\top s + \mathbf{e}_1 + \mathbf{e}_2 + \lfloor q/K \rfloor \mathbf{m}) \\ &- (\mathbf{E}_f\mathbf{y})^\top\Big((\mathbf{a} \mid (f(\mathbf{x})\mathbf{g} + \mathbf{b}_f))^\top s + (\mathbf{e}_3;\mathbf{e}_f)\Big) \\ &= \lfloor q/K \rfloor \langle \mathbf{y}, \mathbf{m}\rangle + \mathbf{e}_r\end{aligned}$$

where the error $\mathbf{e}_r = \mathbf{y}^\top(\mathbf{e}_1 + \mathbf{e}_2) + (\mathbf{E}_f\mathbf{y})^\top(\mathbf{e}_1;\mathbf{e}_f)$ and $(\mathbf{e}_1;\mathbf{e}_f) \in \mathcal{R}_q^{2m}$. Combined with the above explanation, we can determine the bound for $\mathbf{e}_r$ as $\|\mathbf{e}_r\| < n\ell V(\sigma+\tau) + 2C_1 nm\ell V\sigma\sigma_s(\sqrt{n} + (b\sqrt{nm})^d)$.

In the $ReDecrypt$ algorithm, if $f(\mathbf{x}) = 0$, we calculate μ as

$$\begin{aligned}\mu &= \mathbf{y}^\top\hat{\mathbf{c}}_1 - sk_{\tilde{f}}^\top\hat{\mathbf{c}}'_0 \\ &= \mathbf{y}^\top(\mathbf{u}^\top s + \mathbf{e}_1 + \mathbf{e}_2 + \lfloor q/K \rfloor \mathbf{m}) \\ &- (\mathbf{E}_f\mathbf{y})^\top\Big((\mathbf{a} \mid \mathbf{b}_{\tilde{f}})^\top s + \mathbf{E}_{f\to\tilde{f}}^\top(\mathbf{e}_3;\mathbf{e}_f)\Big) \\ &= \lfloor q/K \rfloor \langle \mathbf{y}, \mathbf{m}\rangle + \mathbf{e}'_r\end{aligned}$$

where $\mathbf{e}'_r = \mathbf{y}^\top(\mathbf{e}_1 + \mathbf{e}_2) + (\mathbf{E}_f\mathbf{y})^\top\mathbf{E}_{f\to\tilde{f}}^\top(\mathbf{e}_3;\mathbf{e}_f)$. Consequently, we determine the bound for $\mathbf{e}'_r$ as $\|\mathbf{e}'_r\| = n\ell V(\sigma+\tau) + 4C_1 nm^2\ell V\sigma\sigma_s^2(\sqrt{n} + (b\sqrt{nm})^d)$.

Therefore, for the correctness of decryption, it requires that $q > 4nK\ell V(\sigma+\tau) + 16C_1 nm^2\ell V\sigma\sigma_s^2(b\sqrt{nm})^{d+1}$, where the constant $C_1 = 128$.

Multi-hop. This property enables authorized DC to execute $ReKeyGen$ and $ReEnc$ algorithms multiple times by selecting appropriate parameters. This means that using a function p and a decryption key sk_f, one can perform $ReKeyGen$ to acquire the matrix $\mathbf{E}_{f\to p} \in \mathcal{D}_{\mathcal{R}_q^{2m\times 2m},\sigma_s}$ conditioned on $(\mathbf{a} \mid \mathbf{b}_f)\cdot\mathbf{E}_{f\to p} = (\mathbf{a} \mid \mathbf{b}_p)$. Then he/she constructs $rk_{f\to p} = (\mathbf{E}_{f\to p}, f, p)$, and acquires a re-encrypted ciphertext by invoking the $ReEnc$ operation.

Unidirectionality. This property suggests that the proxy can only convert the ciphertext for the function f into the ciphertext for function $\tilde{f}$, but not vice versa. Essentially, the $ReEnc$ algorithm does not involve the function $\tilde{f}$. Hence, the function f cannot be leaked by any party. In addition, it can be found that neither party can generate the component $\mathbf{E}_{\tilde{f}\to f}$ of the re-encryption key unless the function $\tilde{f}$ is compromised. In summary, we say that the proposed scheme satisfies unidirectionality.

5.1 Security

We demonstrate that our scheme is Sel-IND-CPA and Sel-IND-CKA secure under the RLWE assumption and the RSIS assumption.

Theorem 3. *Given appropriate parameters, if a PPT adversary breaks the ring variant of r-ALS with a negligible advantage, then our scheme is Sel-IND-CPA secure.*

Technically, we rely on the ring variant of work [8] (r-ALS) and the noise re-randomization technique of Lemma 3 to prove the indistinguishability of the challenge ciphertexts, where the operations of r-ALS are depicted in Fig.3. By a sequence of games, we demonstrate that if the r-ALS scheme is secure under the (Decisional) RLWE assumption, our scheme is secure for appropriate parameters.

Proof. As stated in Definition 4, the adversary $\mathcal{A}$ first declares the vector $\mathbf{x}^* = (x_1^*, \cdots, x_k^*)$ and challenge function f^*. We require the secret key query $(f, \mathbf{y})$ conditioned on $f(\mathbf{x}^*) = 1$ or $f = f^* \wedge \langle \mathbf{m}_0, \mathbf{y}\rangle = \langle \mathbf{m}_1, \mathbf{y}\rangle$, where $\mathbf{m}_0$ and $\mathbf{m}_1$ are two challenge messages. Note that, to simplify description, we denote $KeyGen_{off}$ and $KeyGen_{on}$ as $KeyGen$, the same for the Enc phase.

Game$_0$: This game is coincident with $Game_{CPA}(\lambda, C, \mathcal{A})$ of Section 2.

Game$_1$: In this game, we change the manner of generating the pseudo-random ring vector $\mathbf{a}$. For this, the vector $\mathbf{a}$ is sampled from the distribution $\mathcal{R}_q^{1\times m}$, instead of being generated by performing $(\mathbf{a}, \mathbf{T_a}) \leftarrow RTrapGen(1^\lambda)$. Then the challenger C enumerates a basis $\mathbf{T_a}$ to answer $\mathcal{A}$'s secret key query. The rest of operations remains the same. Therefore,

$\underline{Setup(1^\lambda, 1^\ell):}$	$\underline{Enc(mpk, \mathbf{m}):}$
$\mathbf{a} \leftarrow \mathcal{R}_q^m$	$s \leftarrow \mathcal{R}_q$
$\mathbf{Z} \leftarrow \mathcal{D}_{\mathcal{R}^{\ell\times m}, \sigma_s}$,	$\mathbf{e}_1 \leftarrow \mathcal{D}_{\mathcal{R}^m, \sigma}$
$\mathbf{u} \leftarrow \mathbf{Za} \in \mathcal{R}_q^\ell$	$\mathbf{e}_2 \leftarrow \mathcal{D}_{\mathcal{R}^\ell, \sigma}$
$mpk \leftarrow (\mathbf{a}, \mathbf{u})$	$ct_0 = \mathbf{a}s + \mathbf{e}_1$
$msk \leftarrow \mathbf{Z}$	$ct_1 = \mathbf{u}s + \mathbf{e}_2 + \lfloor q/K \rfloor \cdot \mathbf{m}$
Return (mpk, msk)	Return (ct_0, ct_1)
$\underline{KeyGen(msk, \mathbf{y}):}$	$\underline{Dec(ct_0, ct_1, sk_\mathbf{y}):}$
Return $(\mathbf{y}, sk_\mathbf{y} = \mathbf{y}^\top \mathbf{Z})$	$\mu = \mathbf{y}^\top ct_1 - sk_\mathbf{y} ct_0$
	$\bar{\mu} = \arg\min_{\bar{\mu} \in \{0,\cdots,K-1\}} \lvert \lfloor q/K \rfloor \mu - \bar{\mu} \rvert$
	Return $\bar{\mu}$

Fig. 3 The algorithms of the r-ALS scheme.

the $\mathbf{Game}_0$ and $\mathbf{Game}_1$ are statistically indistinguishable in $\mathcal{A}$'s view.

$\mathbf{Game}_2$: The main object of this game is to reprogram the parameter $\mathbf{u}$ of mpk and the component $\mathbf{E}_f$ in the *KeyGen* phase. First, C samples $\mathbf{D}_1, \mathbf{D}_2 \leftarrow \mathcal{D}_{\mathcal{R}^{m\times\ell}, \sigma_s}$, and calculates $\mathbf{u}_1 = \mathbf{aD}_1$. Then C performs the algorithm $\mathbf{b}_{f^*} \leftarrow \mathrm{Eval}_{pk}(f^*, \{\mathbf{b}_i\}_{i\in[k]})$, then computes $\mathbf{u} = \mathbf{u}_1 + \mathbf{b}_{f^*}\mathbf{D}_2$ and $\mathbf{D} = (\mathbf{D}_1; \mathbf{D}_2)$. In this context, the condition $\mathbf{u} = (\mathbf{a} \mid \mathbf{b}_{f^*}) \cdot \mathbf{D}$ holds. For a secret key query for $(f^*, \mathbf{y})$, C computes $sk_{f^*} = \mathbf{Dy}$ if $f^*(\mathbf{x}) = 0$; otherwise performs $\mathbf{E}_{f^*} \leftarrow RSampleLeft(\mathbf{a}, \mathbf{b}_{f^*}, \mathbf{T_a}, \mathbf{u}, \sigma, \sigma_s)$, then generates $sk_{f^*} = \mathbf{E}_{f^*}\mathbf{y}$ and sk_s^* by the way of *KeyGen*. At last, C returns sk^* to $\mathcal{A}$.

For this, we can observe that the element $\mathbf{u}_1$ is statistically close to $\mathcal{R}_q^{1\times\ell}$. Further, we have that $\mathbf{u} = \mathbf{u}_1 + \mathbf{b}_{f^*}\mathbf{D}_2$ follows the distribution $\mathcal{R}_q^{1\times\ell}$. Consequently, $\mathbf{Game}_1$ is statistically close to $\mathbf{Game}_2$.

$\mathbf{Game}_3$: In this game, we show how to reconstruct vectors $(\mathbf{b}_1, \cdots, \mathbf{b}_k)$, instead of sampling randomly from $\mathcal{R}_q^{1\times m}$. Specifically, C randomly samples a series of matrices $\mathbf{S}_i^* \in \{\pm 1\}^{m\times m}$ and sets $\mathbf{b}_i = \mathbf{aS}_i^* + x_i^*\mathbf{g}$ $\forall i \in [k]$. After completing the above operations, C produces the noise vector $\mathbf{e}_0 = (\mathbf{e}_3^\top \mid \mathbf{e}_3^\top\mathbf{S}_1^* \mid \cdots \mid \mathbf{e}_3^\top\mathbf{S}_k^*)^\top \in \mathcal{R}_q^{m(k+1)}$. Other operations remain the same as $\mathbf{Game}_2$.

By the leftover hash lemma [7], $\mathcal{A}$ can distinguish $(\mathbf{a}, \mathbf{aS}_i^*, \mathbf{e}_1^\top\mathbf{S}_i^*)$ from $(\mathbf{a}, \tilde{\mathbf{b}}, \mathbf{e}_1^\top\mathbf{S}_i^*)$ with the advantage $\mathrm{negl}(\lambda)$, where $\tilde{\mathbf{b}}$ is a ring vector over $\mathcal{R}_q^{1\times m}$. Moreover, the elements $(\mathbf{b}_1, \cdots, \mathbf{b}_k)$ of this game are statistically close to those in $\mathbf{Game}_2$. Thus, $\mathcal{A}$ has a negligible advantage to distinguish $\mathbf{Game}_2$ from $\mathbf{Game}_3$.

$\mathbf{Game}_4$: The operations of this game remains the same as $\mathbf{Game}_3$, except for changing sk_f and $rk_{f\to\tilde{f}}$ in *KeyGen* and *ReKeyGen*, respectively. For $f = f^* \wedge \langle \mathbf{m}_0, \mathbf{y} \rangle = \langle \mathbf{m}_1, \mathbf{y} \rangle$, C computes the element $\mathbf{D}$ relying on the method in the previous games, and sets $sk_{f^*} = \mathbf{Dy}$. To generate $rk_{f^*\to\tilde{f}}$ for $\mathcal{A}$, C performs $rk_{f^*\to\tilde{f}} \leftarrow ReKeyGen(mpk, \tilde{f}, sk_{f^*})$. For the constraint $f(\mathbf{x}^*) = 1$, C executes the following operations:

- Perform $\mathbf{S}_f \leftarrow \mathrm{Eval}_{sim}(f, \mathbf{a}, \{\mathbf{S}_i^*, x_i^*\}_{i\in[k]})$ conditioned on $\mathbf{aS}_f - f(\mathbf{x}^*)\mathbf{g} = \mathbf{b}_f$, where $\mathbf{b}_f = \mathrm{Eval}_{pk}(f, (\mathbf{aS}_1^* - x_1\mathbf{g}, \cdots, \mathbf{aS}_k^* - x_k\mathbf{g}))$.
- For key query, produce $\mathbf{E}_f \leftarrow RSampleRight(\mathbf{a}, \mathbf{g}, \mathbf{T_g}, \mathbf{S}_f, \mathbf{u}, \sigma, \sigma_s)$ such that $(\mathbf{a} \mid \mathbf{aS}_f + \mathbf{g})\mathbf{E}_f = \mathbf{u}$. Then construct $sk_f = \mathbf{E}_f\mathbf{y}$.
- For re-encryption key query, C receives $(f, \tilde{f})$ from $\mathcal{A}$. Then C executes *KeyGen* algorithm to obtain the secret key sk_f corresponding to the function f. Finally, C generates $rk_{f\to\tilde{f}} \leftarrow ReKeyGen(mpk, msk, \tilde{f})$.
- Re-encryption query, $\mathcal{A}$ submits the tuple $(f, \tilde{f}, ct, \mathbf{x})$ to C for $\mathbf{x} \neq \mathbf{x}^*$. Then, C produces $rk_{f\to\tilde{f}}$ and returns $ct_{\tilde{f}} \leftarrow ReEnc(mpk, ct, rk_{f\to\tilde{f}}, \mathbf{x})$.

From the above processes, the advantage of $\mathcal{A}$ in distinguishing between $\mathbf{Game}_3$ and $\mathbf{Game}_4$ is close to $\mathrm{negl}(\lambda)$. Therefore, $\mathbf{Game}_3$ and $\mathbf{Game}_4$ are considered statistically indistinguishable.

$\mathbf{Game}_5$: The objective of this game is to demonstrate the indistinguishability of the ciphertexts for message $\mathbf{m}_\beta (\beta \in \{0, 1\})$, relying on the security of the r-ALS. By interacting with the challenger in r-ALS (C_{r-ALS}), C obtains the master public key $mpk_\theta = (\mathbf{a}_\theta, \mathbf{u}_\theta)$ from C_{r-ALS}. Then C executes the following procedures.

Init. $\mathcal{A}$ declares a vector $\mathbf{x}^*$ and a function f^*.

Setup. Sample $\mathbf{D}_2 \leftarrow \mathcal{D}_{\mathcal{R}^{m\times\ell}, \sigma_s}$ and $\mathbf{S}_i^* \in \{\pm 1\}^{(m\times m)}$ $\forall i \in [k]$, then calculate $\mathbf{a} = \mathbf{a}_\theta^\top \in \mathcal{R}_q^{1\times m}$ and $\mathbf{u} = \mathbf{u}_\theta^\top + \mathbf{b}_{f^*}\mathbf{D}_2 \in \mathcal{R}_q^{1\times\ell}$. In addition, C generates $\mathbf{a}_s$, $\mathbf{h}_s$ and $\mathbf{b}_s$ relying on the original way of $\mathbf{Game}_4$, then produces $\{\mathbf{b}_i\}_{i\in[k]} \in \mathcal{R}_q^{(1\times m)k}$ as described in $\mathbf{Game}_3$ for producing mpk.

Query phase. For the tuple $(f, \mathbf{y})$ requested by $\mathcal{A}$, C requests C_{r-ALS} to obtain the secret key $sk_{\mathbf{y},\theta}$ of the r-ALS scheme.

If $f^*(\mathbf{x}) = 1$, C performs $\mathbf{S}_f \leftarrow \mathrm{Eval}_{sim}(f, \mathbf{a}, \{\mathbf{S}_i^*, x_i^*\}_{i\in[k]})$ and $\mathbf{E}_f \leftarrow RSampleRight(\mathbf{a}, \mathbf{g}, \mathbf{T_g}, \mathbf{S}_f, \mathbf{u}, \sigma, \sigma_s)$, and then constructs $\mathbf{E}_f\mathbf{y}$. Otherwise, compute the decryption key as $sk_{f^*} = (sk_{\mathbf{y},\theta}^\top; \mathbf{D}_2\mathbf{y}) \in \mathcal{R}_q^{2m}$. The rest operations remain the same as $\mathbf{Game}_4$.

Challenge query. $\mathcal{A}$ sends a pair of messages $(\mathbf{m}_0, \mathbf{m}_1)$ to C. Then C requests C_{r-ALS} to obtain a pair of ciphertexts $(ct_{0,\theta}, ct_{1,\theta})$, and computes $\mathbf{c}_{f^*} \leftarrow \mathrm{Eval}_{ct}(f^*, \{(\mathbf{b}_i, \mathbf{c}_i, x_i^*)\}_{i\in[k]})$. Next, C generates $(\hat{\mathbf{c}}_0^*, \hat{\mathbf{c}}_1^*)$ according to the following processes:

$$\hat{\mathbf{c}}_0^* = (ct_{0,\theta} \mid (\mathbf{S}_1^*)^\top ct_{0,\theta} \mid \cdots \mid (\mathbf{S}_k^*)^\top ct_{0,\theta}),$$
$$\hat{\mathbf{c}}_1^* = ct_{1,\theta}^* + \mathbf{D}_2^\top \cdot \mathbf{c}_{f^*} + \mathrm{NoiseReRand}(\mathbf{D}_2^\top, \sigma_s),$$

where $\tau > C_0\sigma\sigma_s\Delta_f(2\sqrt{nm} + \sqrt{\ell})$. At last, C sets $\mathbf{c}_w^* = \mathbf{S}_f \cdot ct_{0,\theta} \in \mathcal{R}_q^m$ and $\hat{c}^* \in \mathcal{R}_q$.

We show that $\mathbf{Game}_4$ and $\mathbf{Game}_5$ are considered indistinguishable. This means the advantage of $\mathcal{A}$ in distinguishing $\mathbf{Game}_4$ from $\mathbf{Game}_5$ is at most $\mathrm{negl}(\lambda)$. Since the element $\mathbf{Z}_\theta$ of r-ALS is sampled from the distribution $\mathcal{D}_{\mathcal{R}^{\ell\times m}, \sigma_s}$, for the parameter $\mathbf{u}$, we have

$$\begin{aligned}\mathbf{u} &= \mathbf{u}_\theta^\top + \mathbf{b}_{f^*}\mathbf{D}_2 \\ &= (\mathbf{Z}_\theta\mathbf{a}_\theta)^\top + \mathbf{b}_{f^*}\mathbf{D}_2 \\ &= (\mathbf{a} \mid \mathbf{b}_{f^*})\mathbf{D},\end{aligned}$$

where $\mathbf{D} = (\mathbf{Z}_\theta^\top; \mathbf{D}_2) \in \mathcal{D}_{\mathcal{R}_q^{2m\times\ell},\sigma_s}$. When $f = f^*$, $\mathcal{A}$ can obtain the decryption key $sk_{f^*} = (\mathbf{Z}_\theta^\top \mathbf{y}; \mathbf{D}_2\mathbf{y}) = \mathbf{D}\mathbf{y}$. Additionally, for $f^*(\mathbf{x}) = 1$, C performs the operations described in the *Query* phase. Therefore, the decryption key of this game remains indistinguishable from that of $\mathbf{Game}_4$. For the challenge ciphertext ct^*, we have that

$$\begin{aligned}\hat{\mathbf{c}}_0^* &= (ct_{0,\theta} \mid (\mathbf{S}_1^*)^\top ct_{0,\theta} \mid \cdots \mid (\mathbf{S}_k^*)^\top ct_{0,\theta}) \\ &= (\mathbf{a} \mid \mathbf{a}\tilde{\mathbf{S}}^*)^\top s + (\mathbf{e}_3 \mid (\tilde{\mathbf{S}}^*)^\top \mathbf{e}_1),\end{aligned}$$

where $\tilde{\mathbf{S}}^* = (\mathbf{S}_1^* \mid \cdots \mid \mathbf{S}_k^*)$. In this case, the elements $\hat{\mathbf{c}}_0$ and $\hat{\mathbf{c}}_0^*$ are statistically indistinguishable, since they follow the distribution $\mathcal{R}_q^{m(k+1)}$. For the element $\hat{\mathbf{c}}_1^*$, we have that

$$\begin{aligned}\hat{\mathbf{c}}_1^* &= ct_{1,\theta}^* + \mathbf{D}_2^\top \mathbf{c}_{f^*} + \mathrm{NoiseReRand}(\mathbf{D}_2^\top, \sigma_s) \\ &= (\mathbf{u} - \mathbf{b}_{f^*}\mathbf{D}_2)^\top s + \mathbf{e}_1 + \lfloor q/K \rfloor \cdot \mathbf{m}_b \\ &+ \mathbf{D}_1^\top (\mathbf{b}_{f^*}^\top s + \mathbf{e}_{f^*}) + \mathrm{NoiseReRand}(\mathbf{D}_2^\top, \sigma_s) \\ &= \mathbf{u}^\top s + \mathbf{e}_1 + \mathbf{D}_2^\top \mathbf{e}_{f^*} + \mathrm{NoiseReRand}(\mathbf{D}_2^\top, \sigma_s) + \lfloor q/K \rfloor \cdot \mathbf{m}_b.\end{aligned}$$

By Lemma 3, $\mathbf{D}_2^\top \mathbf{e}_{f^*} + \mathrm{NoiseReRand}(\mathbf{D}_2^\top, \sigma_s)$ follows the distribution $\mathcal{D}_{\mathcal{R}^\ell,\tau}$. Thus the noise is statistically close to $\mathbf{e}_2$.

Guess Phase. $\mathcal{A}$ outputs a guess bit β'.

From the above processes, we conclude that if the (Decisional) RLWE is hard, the advantage of $\mathcal{A}$ in breaking the proposed scheme is at most $\mathrm{negl}(\lambda)$.

Theorem 4. *Given appropriate parameters, our scheme achieves Sel-IND-CKA provided that the RSIS problem holds.*

Proof. Suppose the advantage $Adv_{CKA-\mathcal{A}}^{ABSEIL}(\lambda)$ of a PPT adversary is $1-\mathrm{negl}(\lambda)$, then a challenger C can solve the RSIS. The reduction is constructed relying on Definition 5.

Setup. C performs the equivalent operations as $Game_{CKA}(\lambda, C, \mathcal{A})$.

H-query. Let q_w be $\mathcal{A}$'s maximum number of trapdoor query, and L_w be the empty list about the query. When $\mathcal{A}$ initiates a query, C inspects whether the tuple $(\mathbf{a}_s^*, k_w^*, t_w^*)$ previously appeared in L_w. If not present, C produces $t_w^* = H(k_w^*)$, and programs $\mathbf{a}_s^*$ of mpk as $\mathbf{a}_s^* = (\hat{\mathbf{a}} \mid -t_w^*\mathbf{g} - \hat{\mathbf{a}}\mathbf{T}_{\mathbf{a}_s})$. Then inserts the tuple $(\mathbf{a}_s^*, k_w^*, t_w^*)$ into L_w. Otherwise, C sets $\mathbf{a}_s = \mathbf{a}_s^*$, and returns $\mathbf{a}_s$.

Query phase. For capturing sk and t_{kw}, $\mathcal{A}$ adaptively initiates the following queries:

- *Key query.* After catching the request of $\mathcal{A}$, C simulates sk_s by invoking $KeyGen_{off}$ and $KeyGen_{on}$. Then C programs sk_f depending on the operations of $\mathbf{Game}_3$ and $\mathbf{Game}_4$. In this context, $\mathcal{A}$ successfully forges sk_s with little advantage.
- *Trapdoor query.* To simulate t_{kw}, C sets $t_w = H(k_w)$ and $\mathbf{a}_w = \mathbf{a}_s + (\mathbf{0} \mid t_w\mathbf{g}) = (\hat{\mathbf{a}} \mid (t_w - t_w^*)\mathbf{g} - \hat{\mathbf{a}}\mathbf{T}_{\mathbf{a}_s})$. Then produces $\hat{\xi} = \mathbf{h}_s \cdot sk_s$ and $\mathbf{e}_w \leftarrow RSamplePre(\mathbf{a}_w, \mathbf{T}_{\mathbf{a}_s}, \hat{\xi}, \sigma, \sigma_s, (t_w - t_w^*))$. Since $(t_w - t_w^*) \in \mathcal{R}_q$ is an invertible tag, the condition $\mathbf{a}_w\mathbf{e}_w = \hat{\xi}$ holds.

Ciphertext query. $\mathcal{A}$ initiates a series of ciphertext requests for $(k_w, \mathbf{x})$. C produces $(\hat{\mathbf{c}}_0, \hat{\mathbf{c}}_1, \mathbf{c}_w, \hat{c})$ relying on the way in Enc_{on} process. Then sample $\mathbf{c}_\vartheta \leftarrow \mathcal{D}_{\mathcal{R}^m,\sigma_s}$ and generates $ct = (\hat{\mathbf{c}}_0, \hat{\mathbf{c}}_1, \mathbf{c}_w, \mathbf{c}_\vartheta, \hat{c})$.

Forgery phase. $\mathcal{A}$ forges a ciphertext $ct^* = (\hat{\mathbf{c}}_0^*, \hat{\mathbf{c}}_1^*, \mathbf{c}_w^*, \mathbf{c}_\vartheta^*, \hat{c}^*)$ about $(k_w^*, \mathbf{x}^*)$. It restricts that $(k_w^*, \mathbf{x}^*)$ was not queried in the ciphertext query phase.

We conclude that once $\mathcal{A}$ wins in $Game_{CKA}(\lambda, C, \mathcal{A})$, then the element $\mathbf{c}_\vartheta^*$ satisfying $0 < \|\mathbf{c}_\vartheta^*\| \leqslant \sigma_s\sqrt{nm}$ is a solution of RSIS, i.e. $\mathbf{a}_\vartheta\mathbf{c}_\vartheta^* = 0 \pmod q$ holds. Further, C captures a valid $\mathbf{c}_\vartheta^*$ with probability $1/q_w$. Consequently, $\mathcal{A}$ forges a ciphertext with advantage $\varepsilon = Adv_{CKA-\mathcal{A}}^{ABSEIL}(\lambda)$, then C can produce a solution of RSIS with advantage ε/q_w, contradicting to the fact. Theorem 4 holds.

6 Performance analysis

We present the efficiency analysis and evaluation results of the ABSEIL scheme. The symbols in the rest of this section are depicted in Table 2.

Table 2 List of symbols

Symbol	Definition
n_q	The value for $\log q$
$\tilde{n}_q$	The value for $\log_b(q)$
T_{pk}	The computational overhead of Eval_{pk}
T_{ct}	The computational overhead of Eval_{ct}
T_{sp}	The computational overhead of SamplePre
$\tilde{T}_{sp}$	The computational overhead of RSamplePre
$\tilde{T}_{sl}$	The computational overhead of RSampleLeft

6.1 Efficiency analysis

Table 3 explicates the feature comparison of the ABSEIL scheme with relevant works. We observe that the schemes [10] and [12] are inferior to other works in terms of policy flexibility. The schemes [11] and [12] involve searchable functions, but they exclude the PRE mechanism, which naturally incurs obstacles in the realistic application. The work [16] does not enjoy searchable capabilities. Retrieving the desired information from large volumes of data became their challenge. The Number Theoretic Transform (NTT) operations speed up the polynomial arithmetic. Thereby our scheme retains capable computing efficiency.

The performances on the storage overhead are described in Table 4. Note that the implicit condition is $\tilde{n}_q < n_q < n$. We remark that ABSEIL outperforms other related schemes in terms of storage cost of secret key sk and search trapdoor t_{kw}, which is quite acceptable to the recipients. Specifically, ABSEIL yields at least a 3-fold improvement in the storage size of sk compared with the work [10], and improves the t_{kw} size from $12nn_q$ in the reference [12] down to $n\tilde{n}_q$. For the storage space of master public key mpk, re-encryption key rk and re-encrypted ciphertext rct, our scheme also achieves prominent improvement compared with [10, 16]. Therefore, ABSEIL scheme reduces the transmission delay, benefiting from the low storage overhead.

We describe the approximate computational overhead of each algorithm about relevant schemes in Table 5, where

Table 3 List of futures

Scheme	Access Policy	Searchable	Unidirectional	Multi-hop	Assumption
Ref. [10]	And-Gates	✓	✓	×	LWE
Ref. [11]	Boolean expression	✓	×	×	LWE
Ref. [12]	Threshold	✓	×	×	LWE
Ref. [16]	Boolean expression	×	✓	✓	LWE
Ours	Boolean expression	✓	✓	✓	RLWE

Table 4 Comparison of storage requirement

Scheme	mpk	sk	ct	t_{kw}	rk	rct
Ref. [10]	$12kn^2n_q$	$6knn_q$	$12knn_q$	$6knn_q$	$72kn^2n_q^3$	$12knn_q$
Ref. [11]	$(k+3)n^2n_q$	$144n^2n_q^2$	$6(k+2)nn_q\ell$	$18nn_q$	–	–
Ref. [12]	$6(k+2)n^2n_q$	$36n^2n_q^2$	$6(k+1)nn_q$	$12nn_q$	–	–
Ref. [16]	$(k+2)n^2n_q$	$4n^2n_q^2$	$(k+2)nn_q$	–	$4n^2n_q^2$	$3nn_q$
Ours	$n\left((k+2)\tilde{n}_q+\ell\right)$	$n(\ell+2\tilde{n}_q)$	$n\left((k+3)\tilde{n}_q+\ell\right)$	$n\tilde{n}_q$	$4\tilde{n}_q^2$	$n(2\tilde{n}_q+\ell)$

$\tilde{n}=\log n$, T_b represents the overhead of basis delegation and $\tilde{T}_{sp} \ll T_{sp}$. The results indicate that the computational complexity of ABSEIL outperforms other correlative constructions. Because the complexity of the basis delegation enjoys high computational complexity, and the $RSamplePre$ of ABSEIL is more computationally efficient than $SamplePre$ in other schemes, such that ABSEIL completes notable improvements in $KeyGen$. Besides, we optimized the complexity of multiplication by the Number Theoretic Transform, resulting in a definite improvement in the execution efficiency of the Enc phase. The cost of ABSEIL for operations $Trapdoor$ and $Search$ are $n\tilde{n}\ell+\tilde{T}_{sp}$ and $2n\tilde{n}_q\tilde{n}$, respectively, which outperforms the methods [10–12]. In this setting, recipients can efficiently retrieve data from remote servers, even with lightweight equipment. Although our scheme involves the Eval_{ct} operation in the $Decrypt$ phase, it still maintains a similar time consumption to schemes [10] and [12]. Moreover, ABSEIL reaches simpler arithmetic operations in $ReKeyGen$ and $ReEnc$ phases than other constructions, relying on the optimized trapdoor form together with the sampling technique over ideal lattices. Taken together, the proposed scheme has theoretical merits in both storage and computing efficiency.

6.2 Evaluation results

The ABSEIL scheme is implemented using the Palisade Library [15] of version 1.11. We evaluated all these algorithms on a host with Intel(R) Core(TM) i5-8250U CPU@ 1.60 GHz, which runs Ubuntu 18.04 operating system. The results are taken from the average of 20 program executions.

Fig.4 summarizes the execution time of the $Setup$, $KeyGen$, Enc and $Decrypt$ phases when the parameter ℓ grows, where $KeyGen$ and Enc involve offline and online phases. In the experiments, we choose the ring dimension $n=1024$, base $b=512$, attribute number $k=8$, and the size of a ring vector $m=17$. The results demonstrate that ℓ is one of the factors affecting the runtime. For $KeyGen_{off}$, $KeyGen_{on}$ and Enc_{on}, execution time linearly grows as ℓ increases, and the algorithms $Setup$, Enc_{off} and $Decrypt$ maintain almost constant runtime. We remark that the offline phase of $KeyGen$ occupies dominant computing time with ℓ grows. In this setting, after pre-performing $KeyGen_{off}$ once, recipients merely need to invoke the lightweight $KeyGen_{on}$ algorithm for capturing secret key sk.

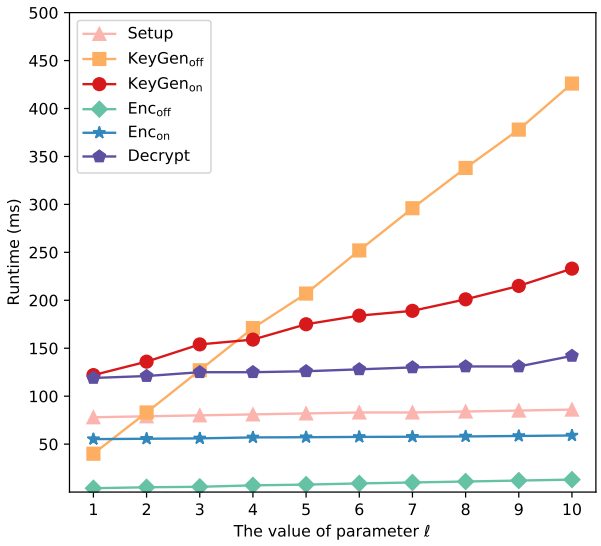

Fig. 4 The runtime for Setup, KeyGen, Enc and Decrypt when ℓ grows.

Further, the runtime of operations $Trapdoor$, $Search$, $ReKeyGen$ and $ReEnc$ with different ℓ is depicted in Table 6. For the search function, ABSEIL scheme enjoys gratifying runtime. From the view of a recipient, the desired data can be acquired from the remote servers in little time. Also, $ReKeyGen$ phase has a high runtime. The principal reason is that step 2 of this phase costs extensive time.

We investigate the impact of the number of attributes on the computational efficiency on the user side. From Fig.5, it can be observed that ABSEIL remains high-performance. The runtime of the $Decrypt$ phase is only 473 ms, even with the number of attributes $k=16$. In addition, operating offline reduces the computational load on both data holders and receivers. The efficiency of Enc and $KeyGen$ phases is im-

Table 5 Comparison of computational overhead

Scheme	KeyGen	Enc	Trapdoor	Search	Decrypt	ReKeyGen	ReEnc
Ref. [10]	T_{sp}	$12kn^2n_q + 2n$	T_{sp}	$6knn_q$	$6knn_q$	$2T_{sp}$	$72kn^2n_q^3$
Ref. [11]	$T_{pk} + T_b$	$6(k+2)n^2n_q\ell + T_{sp}$	$T_{pk} + T_b + T_{sp}$	$6nn_q(n+3\ell) + T_{ct}$	–	–	–
Ref. [12]	$T_{sp} + T_b$	$6(k+1)n^2n_q$	$2T_{sp}$	$24nn_q$	$12nn_q$	–	–
Ref. [16]	$T_{pk} + T_b$	$(k+2)n^2n_q$	–	–	$4n^2n_q^2 + T_{ct}$	$T_{pk} + T_{sp}$	$4n^2n_q^2 + T_{ct}$
Ours	$2n\tilde{n}_q\tilde{n}\ell + T_{pk} + \tilde{T}_{sl}$	$(k+2)n\tilde{n}n_q + n\tilde{n}\ell + \tilde{T}_{sp}$	$n\tilde{n}\ell + \tilde{T}_{sp}$	$2n\tilde{n}_q\tilde{n}$	$n(2\tilde{n}_q + \ell)\tilde{n} + T_{ct}$	$2T_{pk} + \tilde{T}_{sl}$	$4n\tilde{n}_q^2\tilde{n} + T_{ct}$

Table 6 The runtime (ms) for Trapdoor, Search, ReKeyGen and ReEnc

Parameter ℓ	Trapdoor	Search	ReKeyGen	ReEnc
3	40.2	1.4	1741.6	144.8
5	41.3	1.5	1742.8	148.6
7	42.0	1.6	1747.7	150.4
9	43.3	1.6	1774.5	152.3

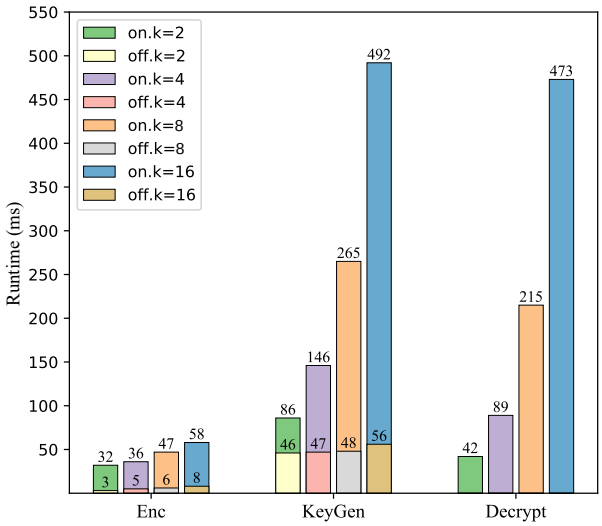

Fig. 5 The runtime for Enc, KeyGen and Decrypt when k grows.

proved by at least 9% and 11%, respectively, when the number of attributes $k < 16$.

As shown in Table 7, we evaluate the runtime of each algorithm in our scheme by choosing different bases. The results represent that the computational efficiency of all the algorithms increased by about 25% when base b is reduced to half of itself. For a rational security level, the scheme requires an appropriately large ring dimension as well as a small base [14]. We can employ a larger base to trade off security and efficiency in practical scenarios, such as base $b = 128, 256$ or 512. Furthermore, the efficiencies of *KeyGen*, *Decrypt* and *ReEnc* are mainly affected by operations Eval_{pk} and Eval_{ct}. Precisely, the time cost of Eval_{pk} occupies 55% of *KeyGen*. Likewise, the consumption of Eval_{ct} is approximately 98% and 92% of the runtime for *Decrypt* and *ReEnc*, respectively.

7 Conclusion

To achieve access control of post-quantum security, we proposed an offline/online attribute-based searchable encryption scheme from ideal lattices (ABSEIL) for IoT. It involves an efficient search function, adaptable access policy and multi-hop property. Theoretically, ABSEIL reaches appropriate performance for computing complexity and storage requirements. Notably, we deliver a specific implementation for ABSEIL. Through comprehensive evaluations, we elaborate that the proposed scheme is applicable to the application context.

As a future extension, we will investigate how to further strengthen the practicability of the proposed scheme, such as implementing the traceable mechanism. Moreover, we will be devoted to reducing the runtime of our scheme.

References

1. Li J, Zhang Y, Ning J, Huang X, Poh S P, Wang D. Attribute based encryption with privacy protection and accountability for cloudIoT. IEEE Transactions on Cloud Computing, 2020, 10(2): 762-773.
2. Lu Y, Li J, Wang F. Pairing-free certificate-based searchable encryption supporting privacy-preserving keyword search function for IIoTs. IEEE Transactions on Industrial Informatics, 2020, 17(4): 2696-2706.
3. Chen N, Li J, Zhang Y, Guo Y. Efficient CP-ABE scheme with shared decryption in cloud storage. IEEE Transactions on Computers, 2020, 71(1): 175-184.
4. Lu Y, Li J. Lightweight public key authenticated encryption with keyword search against adaptively-chosen-targets adversaries for mobile devices. IEEE Transactions on Mobile Computing, 2021, 21(12): 4397-4409.
5. Li J, Lin X, Zhang Y, Han J. KSF-OABE: Outsourced attribute-based encryption with keyword search function for cloud storage. IEEE Transactions on Services Computing, 2016, 10(5): 715-725.
6. Varri U S, Kasani S, Pasupuleti S K, Kadambari K V. FELT-ABKS: fog-enabled lightweight traceable attribute-based keyword search over encrypted data. IEEE Internet of Things Journal, 2022, 9(10): 7559-7571.
7. Agrawal S, Boneh D, Boyen X. Efficient lattice (H)IBE in the standard model. In: Proceedings of Annual International Conference on the Theory and Applications of Cryptographic Techniques, 2010, 553-572.
8. Agrawal S, Libert B, Stehlé D. Fully secure functional encryption for inner products, from standard assumptions. In: Proceedings of the 36th Annual International Cryptology Conference, 2016, 333-362.
9. Katsumata S, Yamada S. Partitioning via non-linear polynomial functions: More compact ibes from ideal lattices and bilinear maps. In:

Table 7 The runtime (ms) for algorithms of ABSEIL on different bases

Base	KeyGen	Enc	Trapdoor	Search	Decrypt	ReKeyGen	ReEnc
128	488.5	70.1	27.2	2.1	335.2	1915.9	335.6
256	393.8	51.4	25.7	1.8	266.0	1530.2	295.4
512	332.0	42.3	23.8	1.3	217.9	1190.1	231.8
1024	268.2	34.5	18.1	0.9	170.7	893.4	170.0

Proceedings of the 22nd International Conference on the Theory and Application of Cryptology and Information Security, 2016, 682-712.

10. Li J, Ma C, Zhang K. A novel lattice-based CP-ABPRE scheme for cloud sharing. Symmetry, 2019, 11(10): 1262-1281.
11. Li J, Ma M, Zhang J, Fan S, Li S. Attribute-based keyword search from lattices. In: Proceedings of the Information Security and Cryptology, 2020, 66-85.
12. Wang P, Chen B, Xiang T, Wang Z. Lattice-based public key searchable encryption with fine-grained access control for edge computing. Future Generation Computer Systems, 2022, 127: 373-383.
13. Micciancio D, Peikert C. Trapdoors for lattices: Simpler, tighter, faster, smaller. In: Proceedings of Annual International Conference on the Theory and Applications of Cryptographic Techniques, 2012, 700–718.
14. Genise N, Micciancio D. Faster gaussian sampling for trapdoor lattices with arbitrary modulus. In: Proceedings of Annual International Conference on the Theory and Applications of Cryptographic Techniques, 2018, 174–203.
15. Genise N, Micciancio D, Polyakov Y. Building an efficient lattice gadget toolkit: Subgaussian sampling and more. In: Proceedings of Annual International Conference on the Theory and Applications of Cryptographic Techniques, 2019, 655–684.
16. Luo F, Al-Kuwari S, Wang F, Chen K. Attribute-based proxy re-encryption from standard lattices. Theoretical Computer Science, 2021, 865: 52–62.