Multi-Authority Attribute-Based Multi-Keyword Searchable Encryption with Dynamic Membership from Lattices*

Er-Shuo Zhuang, Wei-Lin Lai, Xu-Zhi Li, and Chun-I Fan†

National Sun Yat-sen University, Kaohsiung, Taiwan
{zhuanges, williamlai22062203, az645252}@gmail.com, cifan@mail.cse.nsysu.edu.tw

Abstract

As quantum computing advances, many traditional encryption mechanisms will be vulnerable, making cryptosystems resistant to quantum attacks increasingly important. Among these, lattice-based cryptography is recognized as an efficient and practical alternative. In communication systems, Attribute-Based Encryption (ABE) schemes are more suitable for environments with a large number of users. ciphertext-policy ABE (CP-ABE) allows senders to specify which users with particular attributes can decrypt the ciphertext, making it ideal for real-world applications. Additionally, searchable encryption enables users to search encrypted data without revealing keywords to the cloud server. To enhance user management, dynamic membership management is incorporated, making the system more flexible and adaptable. Moreover, Multi-Authority (MA) approaches reduce the risk of a single authority being compromised by an attacker. Building on these properties, this paper proposes a lattice-based MA-CP-ABE scheme that supports multi-keyword searches and dynamic membership. The scheme is proven to be secure under the Decision Learning-With-Errors (D-LWE) assumption, providing both strong security and flexibility. The proposed scheme outperforms many existing schemes in computation costs and provides robust functionality, ensuring secure and efficient data sharing and searching in post-quantum cryptography environments.

1 Introduction

With the growing demand for data storage, a secure, reliable, and flexible system is essential. Cloud storage can support a large number of users and is increasingly adopted by enterprises. However, some sensitive data should remain hidden from cloud servers. Functional encryption, such as attribute-based encryption (ABE) and searchable encryption (SE), addresses this concern. ABE enables fine-grained access control for multi-user environments, while SE allows searching over encrypted data.

ABE is classified into key-policy ABE (KP-ABE) [2] and ciphertext-policy ABE (CP-ABE) [18]. KP-ABE embeds the attribute structure into the private key, with the sender defining the ciphertext attributes. In contrast, CP-ABE allows the sender to set the access structure during encryption, ensuring only eligible users can decrypt the ciphertext. CP-ABE is generally more flexible and suited for dynamic access control scenarios.

SE enables users to search encrypted data. The data owner encrypts messages and keywords before uploading them to the cloud service provider (CSP). While CSP cannot access the

*Proceedings of the 9th International Conference on Mobile Internet Security (MobiSec'25), Article No. 23, December 16-18, 2025, Sapporo, Japan.

†Corresponding Author

message content, data users can generate search tokens to retrieve matching ciphertexts, which they decrypt using private keys.

Dynamic membership enhances system practicality by enabling efficient user management, such as registration, revocation, and attribute updates. To formalize this concept, Fan *et al.* [6] defined four key properties:

1. **Expandability**: New users can be added.
2. **Revocability**: An Attribute Authority (AA) can revoke a user's key to prevent decryption after revocation.
3. **Renewability**: AA can update users' attributes and keys; outdated keys cannot decrypt new ciphertexts.
4. **Independence**: Updating or revoking one user's attributes does not impact others.

To avoid a single point of failure, the multi-authority (MA) model was introduced by Chase [3], where multiple attribute authorities (AAs) jointly manage attributes. Although a central authority aggregates key components, this model mitigates risks associated with a single compromised AA and distributes the computational load.

One of the significant advantages of lattice-based cryptography is its support for functional encryption, and its quantum resistance has led to wide adoption in post-quantum standards. In 2011, Agrawal *et al.* [1] introduced the first fuzzy identity-based encryption scheme based on lattices, wherein each user's identity functions as an independent attribute, possessing corresponding public and private keys. In the same year, Zhang *et al.* [18] proposed the first lattice-based CP-ABE scheme, incorporating a threshold access structure. Subsequently, in 2012, Boyen *et al.* [2] introduced the first lattice-based KP-ABE scheme, employing a Linear Secret Sharing Scheme (LSSS) for its access structure.

In 2013, Hou extitet al. [9] developed the first lattice-based searchable encryption (SE) scheme supporting single-keyword searches. Later, in 2015, Zhang extitet al. [17] proposed a lattice-based MA-ABE scheme, which, despite incorporating a central authority, utilizes it solely for generating public parameters that attribute authorities (AAs) leverage to generate user private keys. More recently, in 2023, Lin [10] proposed a CP-ABE scheme with multi-keyword search functionality, employing a Bloom filter as the search structure.

1.1 Contributions

As summarized in Table 1, the proposed scheme simultaneously incorporates the following properties:

- It utilizes second-type (type-2) trapdoor functions [7, 11]. Compared to first-type (type-1) trapdoor functions [8], which offer greater functionality and facilitate scheme design, type-2 trapdoor functions significantly reduce computation and transmission costs [11].
- As a CP-ABE scheme, the proposed approach provides flexible access control.
- A tree-based multi-keyword search mechanism is employed to support both OR and AND gates, enhancing the flexibility and accuracy of keyword searches.
- The multi-authority (MA) technique mitigates the key escrow problem, which arises in cryptosystems where a trusted third party holds all users' private keys. If compromised, such a third party poses significant security and privacy risks.

- Attribute Authorities (AAs) are only involved in key generation, allowing them to remain offline once keys are issued.

- The proposed scheme supports multi-bit encryption, reducing both computational and transmission costs in the encryption phase.

- The scheme facilitates dynamic membership management, offering expandability, revocability, renewability, and independence, enabling flexible modifications to system memberships.

- The proposed scheme's security relies on the decisional learning-with-errors (D-LWE) assumption. It achieves ciphertext indistinguishability under chosen-plaintext attacks (IND-CPA) and keyword privacy under chosen-keyword attacks (IND-CKA). Due to the strict 20-page limit of the MobiSec 2025 submission format (including appendices, EasyChair style), the complete security proofs are omitted in this version but will be made available upon request.

Table 1: Properties of Various LWE-based ABE Schemes

Reference	Trapdoor Functions	ABE Type	Structure		Authentication†		MA	CR	Offline *AA*	MBE	DM
			Access	Search	Access	Search					
Wang *et al.* [16]	Type-1	KP♠	Subset	Single Keyword	YES	YES	NO	YES	YES	NO	NO
Chen *et al.* [4]	Type-1	CP	Tree♡	-	YES	-	NO	NO	YES	NO	NO
Zhuang *et al.* [19]	Type-2	CP	Tree	-	YES	-	YES	YES	YES	NO	YES
Varri *et al.* [14]	Type-1	CP	LSSS♢	Single Keyword	NO	YES	NO	NO	NO	NO	NO
Chen [5]	Type-2♣	CP	LSSS♢	Single Keyword	YES	YES	NO	YES	YES	NO	YES
Wang [15]	Type-2♣	CP	LSSS♢	LSSS♢	YES	YES	NO	YES	YES	NO	YES
Shen *et al.* [13]	Type-1	CP	LSSS♢	Single Keyword	YES	NO	YES	NO	NO	NO	YES
Lin [10]	Type-1	CP	Tree	Bloom Filter	YES	YES	YES	YES	YES	NO	YES
Ours	Type-2	CP	Tree	Tree	YES	YES	YES	YES	YES	YES	YES

*KP stands for key-policy, CP for ciphertext-policy, MA for multi-authority, CR for collusion resistance, MBE for multi-bit encryption, and DM for dynamic membership.
"-" indicates that this option is not considered because the search functionality is not provided by this scheme.
† Authentication refers to whether a user's attributes need to be verified. 'No Authentication' means that all users in the system can access this functionality without requiring attribute verification.
♠ Although Wang *et al.* claim that the data owner can specify an access policy, the access policy is placed in the user's secret keys. Therefore, their scheme is KP-ABE.
♡ Although Chen *et al.* claim that their scheme uses an LSSS-based access structure, it is constructed in tree-based form. So we mark it as tree-based for fairness.
♢ Schemes based on the LSSS structure require Gaussian elimination during decryption. However, ensuring that the coefficients computed through Gaussian elimination remain sufficiently small is challenging. This may lead to increased noise, potentially causing non-negligible search or decryption failures.
♣ Chen's and Wang's schemes use type-1 trapdoor functions, but those functions can be changed to type-2 trapdoor functions. So we mark them as type-2 for fairness.

2 Related Works

In this section, we review several lattice-based CP-ABE and MA-ABE schemes. Table 1 summarizes and compares the properties of existing LWE-based ABE constructions that support keyword search, dynamic membership, or multi-authority settings.

Due to the limited number of lattice-based schemes combining ABE and searchable encryption, some referenced works are unpublished manuscripts or master's theses. Although not peer-reviewed in standard cryptography venues, we include them for functional comparison, as they represent rare efforts integrating similar features within a lattice-based framework. Their status and limitations are clearly indicated for reference.

Wang *et al.* [16] proposed an ABE scheme with a subset access structure, where a data user's attributes must be a subset of those chosen by the sender. The scheme supports single-keyword search via search tokens. Although claimed to allow data owner-specified policies, the access structure is embedded in user secret keys, thus falling under the KP-ABE category.

Chen *et al.* [4] tackled noise accumulation in LSSS-based decryption by using a structured matrix limited to 0, 1, -1, multiplied by an all-one vector during decryption to suppress noise. Although claimed as LSSS-based, the structure is implemented in a tree-like form.

Zhuang *et al.* [19] presented a collusion-resistant CP-ABE scheme with a tree-based access structure. AAs generate per-user keys to prevent secret key aggregation among users.

Varri *et al.* [14] introduced a CP-ABE scheme with an LSSS structure and single-keyword search, but only verify user permissions during search. Thus, any user can access ciphertexts regardless of attributes.

Chen [5] proposed an LSSS-based CP-ABE scheme where AAs issue independent public and secret keys to resist collusion.

Wang [15] extended Chen's work to support multi-keyword search while preserving the LSSS search structure.

Shen *et al.* [13] designed an MA-CP-ABE scheme with an LSSS-based structure and single-keyword search. While user permission is verified, CSPs do not validate token permissions, allowing unrestricted ciphertext search.

Lin [10] proposed an MA-CP-ABE scheme based on lattices, supporting dynamic membership updates and per-user key modifications.

3 Preliminaries

In this section, we first introduce some background knowledge about the proposed scheme such as the definition of the lattice.

3.1 Lattice

The definition of a q-ary lattice $\Lambda_q^{\mathbf{v}}(\mathbf{A})$ is shown as follows:

Definition 3.1. n, m are positive integers, Given a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, a prime number q, and a vector $\mathbf{v}$,

$$\Lambda_q^{\mathbf{v}}(\mathbf{A}) := \{\mathbf{x} \in \mathbb{Z}^m \mid \mathbf{A}\mathbf{x} = \mathbf{v} \mod q\}$$

3.2 Learning With Errors (LWE)

Given a distribution χ, a prime number q, and a positive integer n, let $\mathcal{O}$ be an unspecified oracle, where $\mathcal{O}_\Psi$ is a truly random sampler and $\mathcal{O}_\chi$ is a pseudo-random noise sampler with

random secret vector $\mathbf{s} \in \mathbb{Z}_q^n$. The definition of $(\mathbb{Z}_q, n, \chi)$-LWE problem [12] is as follow:

- $\mathcal{O}_\Psi$: Outputs truly uniform-random samples $(\mathbf{y}_i, v_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$.
- $\mathcal{O}_\chi$: Outputs pseudo-random samples $(\mathbf{y}_i, v_i = \mathbf{y}_i^{\mathrm{T}}\mathbf{s} + e_i)$, with a noise value $e_i \in \mathbb{Z}_q$ sampled from χ, a uniformly random vector $\mathbf{y}_i \in \mathbb{Z}_q^n$, and a fixed secret vector $\mathbf{s} \in \mathbb{Z}_q^n$.

Definition 3.2. The Decisional Learning-With-Errors (D-LWE) problem:
"From $\mathcal{O}$, given a polynomial number of samples, determine whether $\mathcal{O}$ is $\mathcal{O}\chi$ or $\mathcal{O}\psi$."

Definition 3.3. The advantage of an adversary $\mathcal{A}$ breaking the D-LWE problem is :

$$\mathbf{Adv}_{D-LWE}(\mathcal{A}) := |Pr[\mathcal{A}^{\mathcal{O}_\chi} = 1] - Pr[\mathcal{A}^{\mathcal{O}_\Psi} = 1]|$$

Definition 3.4. D-LWE Assumption: There does not exist a polynomial time algorithm with a non-negligible $\mathbf{Adv}_{D-LWE}$.

3.3 Lattice Trapdoor Functions

The proposed scheme employs the type-2 trapdoor functions [11] [7]. Therefore, we introduce them here.

Definition 3.5. $(\mathbf{A}, \mathbf{R}) \leftarrow \mathbf{GenTrap}(\mathbf{A}_0, \mathbf{H})$
Given $\mathbf{A}_0 \in \mathbb{Z}_q^{n \times m'}$ and any invertible matrix $\mathbf{H} \in \mathbb{Z}_q^{n \times n}$, this function will output a matrix $\mathbf{A} = [\mathbf{A}_0 | -\mathbf{A}_0\mathbf{R} + \mathbf{HG}] \in \mathbb{Z}_q^{n \times m}$ and its trapdoor $\mathbf{R} \in \mathbb{Z}_q^{m' \times z}$, where $\mathbf{G}$ is a gadget matrix, $z = n\lceil \log_2 q \rceil$, and $m = z + m'$. Furthermore, the trapdoor quality is assured, guaranteeing that $s(\mathbf{R}) \leq \omega(\sqrt{m \log q})$. Here $\omega()$ denotes asymptotic notation and $s()$ represents the extraction of the Euclidean length of an input.

Definition 3.6. $\mathbf{x} \leftarrow \mathbf{SampleD}(\mathbf{R}, \mathbf{A} = [\mathbf{A_0} | -\mathbf{A_0}\mathbf{R} + \mathbf{HG}], \mathbf{H}, \mathbf{t}, \sigma)$ Given a trapdoor $\mathbf{R} \in \mathbb{Z}^{m' \times w}$ of $\Lambda_q^\perp(\mathbf{A})$, a matrix $\mathbf{A} = [\mathbf{A}_0 | -\mathbf{A}_0\mathbf{R} + \mathbf{HG}] \in \mathbb{Z}_q^{n \times m}$ an invertible matrix $\mathbf{H} \in \mathbb{Z}_q^{n \times n}$, a target vector $\mathbf{t} \in \mathbb{Z}_q^n$, and a Gaussian parameter σ, this function will output a vector $\mathbf{x}$ that satisfies $\mathbf{Ax} = \mathbf{t} \in \mathbb{Z}_q^n$.

3.4 Tree-Based Access and Search Structures

The proposed scheme adopts a tree-based structure for both access control in encryption/decryption and keyword matching in searchable encryption. In CP-ABE, the data owner defines an access structure that determines authorized users, while in SE, the data user generates a search token based on a structure that identifies matching index entries.

Figure 1 illustrates an example of a tree-based access structure. The structure is defined as $\tau = (attr1 \cap attr2 \cap attr3) \cap (attr4 \cup attr5)$, where the i-th attribute is denoted as $attr_i$. The generation of r_i follows several steps. Initially, the root node value is set as $r = 5$. This root node represents an **AND** gate and has two child nodes, $r_1^{'}$ and $r_2^{'}$. The values of these nodes (e.g., $r_1^{'} = 7$ and $r_2^{'} = -2$) can be chosen by the user as long as they are logically consistent. Next, node $r_1^{'}$ (also an **AND** gate) has three leaf nodes (r_1, r_2, r_3), and their values (e.g., $r_1 = 9$, $r_2 = -4$, and $r_3 = 2$) can also be decided by the user. Finally, node $r_2^{'}$, representing an **OR** gate, has two leaf nodes r_4 and r_5, which the user can assign values to (e.g., $r_2^{'}, r_4, r_5 = -2$).

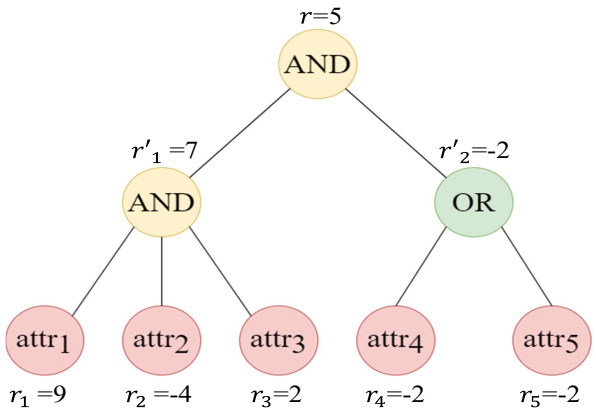


Figure 1: Access Structure

3.5 System Model

Figure 2 shows the roles and the algorithms in the system. There are five roles as follows:

- **Initializer:** A specific attribute authority that initializes the system by generating global public parameters (such as the attribute universe and keyword domain). It does not access any master secret keys of other authorities.
- **Attribute Authority (AA):** Each AA manages a subset of attributes and issues corresponding private keys to users. It also supports updates such as attribute addition and revocation.
- **Data Owner (DO):** Encrypts a message under an access policy and a set of keywords. The resulting ciphertext and keyword index are uploaded to the CSP.
- **Cloud Service Provider (CSP):** Stores encrypted data and performs keyword-based searches using search tokens submitted by users. It is assumed to be honest-but-curious.
- **Data User (DU):** Receives attribute-based secret keys from AAs, generates search tokens for chosen keywords, and decrypts retrieved ciphertexts if authorized.

The proposed scheme comprises nine algorithms shown as follows:

- $PP \leftarrow \textbf{\textit{Setup}}(\lambda, U, D)$: Input a security parameter λ, the set of all attributes U, and a set of all AA D. This algorithm will output the public parameter PP.
- $(PK_\theta, MK_\theta) \leftarrow \textbf{\textit{AASetup}}(PP, U_\theta)$: Input PP and attribute set monitored by AA_θ. This algorithm will output the public keys PK_θ and the master private keys MK_θ for each attribute managed by $AA_\theta \in D$.

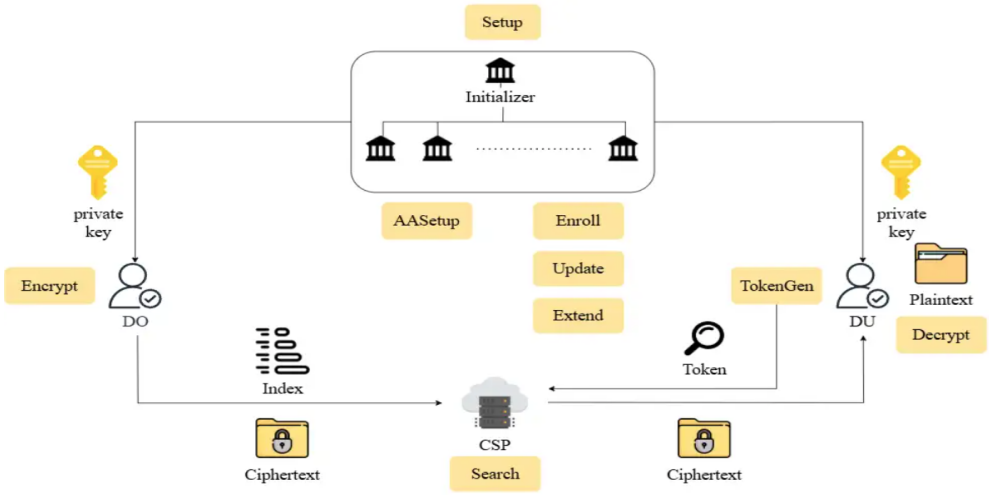


Figure 2: The System Model

- $(SK^{ID}) \leftarrow \textbf{\textit{Enroll}}(PP, PK, MK, U^{ID}, ID)$: Input PP, PK, MK, attribute set of ID U^{ID}, and the user's identity ID. This algorithm will output the user's private keys SK^{ID} and updates PP.
- $(SK'^{ID}) \leftarrow \textbf{\textit{Update}}(PP, PK, MK, U'^{ID}, ID)$:Input PP, PK, MK, new user attribute set U'^{ID} and ID. This algorithm will output SK'^{ID} and update PP.
- $(SK_t^{ID}) \leftarrow \textbf{\textit{Extend}}(PP, PK_t, MK_t, t, ID)$: Input PP, PK_t, MK_t, target attribute t, and ID. This algorithm will output SK_t^{ID}.
- $(CT, I) \leftarrow \textbf{\textit{Encrypt}}(PP, \mathbf{b}, \tau, \{PK_d\}_{d \in D_C}, U_W)$: Input PP, a message vector $\mathbf{b}$, an access structure τ, and $\{PK_d\}_{d \in D_C}$. D_C denotes the attribute authority set that monitor the attributes in τ, and a keyword set associated with the index U_W. This algorithm will output the ciphertext $CT = (C_{ID,j}, C_i, \tau)$, index $I = (I_{ID,w}, I_i)$ and sends them to CSP.
- $(TK) \leftarrow \textbf{\textit{TokenGen}}(PP, PK_d, SK^{ID}, \mathbf{W}', \tau')$: This algorithm takes PP, PK_d, SK^{ID}, a keyword set $\mathbf{W}'$, and a search structure τ' as input. This algorithm will output the search token $TK = \{\{\mathbf{TK}_{1,w}\}_{w \in \mathbf{W}'}, \{\mathbf{TK}_{2,i}\}_{a_i \in U^{ID}}, \tau'\}$, then sends them to CSP.
- $(CT \text{ or } \perp) \leftarrow \textbf{\textit{Search}}(CT, TK, I, U^{ID})$:
  Input CT, TK, I, and U^{ID}. If the search is successful, this algorithm will output the corresponding ciphertext CT. Otherwise, the algorithm aborts.
- $(\mathbf{b} \text{ or } \perp) \leftarrow \textbf{\textit{Decrypt}}(CT, U^{ID}, SK^{ID})$: Input CT, SK^{ID}, and U^{ID}. If the decrypt is successful, this algorithm will output $\mathbf{b}$. Otherwise, the algorithm aborts.

3.6 The Security Model

The scheme achieves ciphertext indistinguishability under chosen-plaintext attacks (IND-CPA) and keyword search security under chosen-keyword attacks (IND-CKA). Owing to the submission's space limitations, detailed proofs are not included here but can be made available upon request.

4 Construction

This proposed scheme contains nine algorithms defined in section 3.5(Setup, AASetup, Enroll+Update+Extend, Encrypt, TokenGen, Search, Decrypt). This section presents the details of these algorithms. Table 2 shows the notations used in the proposed scheme.

Table 2: The Notations

Notation	Meaning
AA_d	The d-th attribute authority
ID	The identity of some user
U	The set of all attributes
D	The set of all AA
m	The count of vectors comprising a basis
z	$z = n \lceil \log_2 q \rceil$
n	The number of components in a vector
q	A prime modulus
L	The set of all ID
l	The total number of keywords in the system
σ	A Gaussian parameter
H_0, H_1	Hash functions
χ	A Gaussian distribution
η	The length of a message vector
a_i	The i-th attribute
k_w	The w-th keyword
U_d	The attribute set monitored by AA_d
U_d^{ID}	The attribute set of ID monitored by AA_d
$U_i'^{ID}$	The updated attribute set of ID monitored by AA_i
U^{ID}	The attribute set of ID
U'^{ID}	The updated attribute set of ID
U_C	The attribute set associated with the CT
τ	An access structure
W	A keyword set
U_W	The keyword set associated with the index
τ'	A keyword structure
D_C	The attribute authority set monitoring attributes in U_C
PK_d	The public key of AA_d
SK^{ID}	The private key of ID
$\lvert \cdot \rvert$	The number of the elements in some set
$\|$	The symbol $\|$ denotes column-wise concatenation.

4.1 Setup

$PP \leftarrow \textbf{\textit{Setup}}(\lambda, U, D)$
The Initializer inputs the security parameter λ, the set of all attributes U, and the set of all attribute authorities D. Then, the Initializer generates the public parameter PP by performing the steps below:

1. With λ, select a prime number q, two positive integers n, m, the count of keyword in the system l, a Gaussian parameter σ, a Gaussian distribution χ, and length of message vector η.
2. Choose two hash functions $H_0 : \{0,1\}^* \to \mathbb{Z}_q^{n\times n} \setminus GL(n,\mathbb{Z})$, $H_1 : \{0,1\}^* \to \mathbb{Z}_q$.
3. Set $\mathbf{U}_{ID} = \emptyset$.
4. Set $\mathbf{u}_{ID} = \emptyset$.
5. For $i = 1$ to η:
   - set $\mathbf{u}_{ID,i} = \emptyset$.
6. Set $\mathbf{U}_1 = \{\mathbf{u}_{ID,1}, \mathbf{u}_{ID,2}, ..., \mathbf{u}_{ID,\eta}\}, \mathbf{U}_2 = \{\mathbf{U}_{ID}, \mathbf{u}_{ID}\}$.
7. Set the user set $L = \emptyset$.
8. Output $PP = \{q, n, m, l, \sigma, H_0, H_1, D, U, L, \chi, \eta, \mathbf{U}_1, \mathbf{U}_2\}$.

4.2 AASetup

$(PK_d, MK_d) \leftarrow \boldsymbol{AASetup}(PP, U_d)$
After the Initializer publishes PP, each AA_d generates PK_d and MK_d by performing the steps below:

1. For each attribute $a_i \in U_d$:
   - Choose an invertible matrix $\mathbf{H}_i \in \mathbb{Z}_q^{n\times n}$.
   - Randomly choose two matrices $\mathbf{A}_{i,0}, \mathbf{B}_{i,0} \in \mathbb{Z}_q^{n\times m'}$.
   - $(\mathbf{A}_i, \mathbf{R}_{\mathbf{A}_i}) \leftarrow GenTrap(\mathbf{A}_{i,0}, \mathbf{H}_i)$. ($GenTrap$ is defined in Section 3.3.)
   - $(\mathbf{B}_i, \mathbf{R}_{\mathbf{B}_i}) \leftarrow GenTrap(\mathbf{B}_{i,0}, \mathbf{H}_i)$.
2. Publish $PK_d = \{\mathbf{A}_i, \mathbf{B}_i\}_{a_i \in U_d}$.
3. Set $\{\mathbf{R}_{\mathbf{A}_i}, \mathbf{R}_{\mathbf{B}_i}\}_{a_i \in U_d}$ as MK_d.

4.3 Enroll

$(SK^{ID}) \leftarrow \boldsymbol{Enroll}(PP, PK, MK, U^{ID}, ID)$
After receiving an enrollment request from a user ID, the Initializer executes the user enrollment using the following steps.

1. If the user set L contains ID, return $\perp$.
2. Update $L = L \cup ID$.
3. Randomly choose vectors $\mathbf{u}_{ID}, \mathbf{u}_{ID,1}, \mathbf{u}_{ID,2}, ..., \mathbf{u}_{ID,\eta} \in \mathbb{Z}_q^n$.
4. Randomly choose a random matrix $\mathbf{U}_{ID,0} \in \mathbb{Z}_q^{n\times m'}$.
5. Choose an invertible matrix $\mathbf{H}_{ID} \in \mathbb{Z}_q^{n\times n}$.

6. $(\mathbf{U}_{ID}, \mathbf{R}_{\mathbf{U}_{ID}}) \leftarrow GenTrap(\mathbf{U}_{ID,0}, \mathbf{H}_{ID})$.
7. Update $\mathbf{U}_1 = \mathbf{U}_1 \cup \{\mathbf{u}_{ID,1}, ..., \mathbf{u}_{ID,\eta}\}, \mathbf{U}_2 = \mathbf{U}_2 \cup \{\mathbf{U}_{ID}, \mathbf{u}_{ID}\}$.
8. Set $SK_1^{ID} = \mathbf{R}_{\mathbf{U}_{ID}} \in \mathbb{Z}_q^{m' \times z}$.
9. Send SK_1^{ID} to ID.

Then, each AA_d receives from the Initializer a set of attributes U_d^{ID} and ID that are monitored by AA_d. The private keys for U_d^{ID} is generated by each AA_d by performing the steps below:

1. $\forall a_i \in U_d^{ID}$:
   - Choose an invertible matrix $\mathbf{H}_i \in \mathbb{Z}_q^{n \times n}$.
   - For $j = 1$ to η:
     - $\mathbf{x}_{i,j} \leftarrow SampleD(\mathbf{R}_{\mathbf{A}_i}, \mathbf{A}_i, \mathbf{H}_i, \mathbf{u}_{ID,j}, \sigma)$, such that $\mathbf{A}_i \cdot \mathbf{x}_{i,j} = \mathbf{u}_{ID,j}$.
   - $\mathbf{TK}_{2,i} \leftarrow SampleD(\mathbf{R}_{\mathbf{B}_i}, \mathbf{B}_i, \mathbf{H}_i, H_0(ID) \cdot \mathbf{u}_{ID}, \sigma)$, such that $\mathbf{B}_i \cdot \mathbf{TK}_{2,i} = H_0(ID) \cdot \mathbf{u}_{ID}$.
2. Set $SK_{2,d}^{ID} = \{\mathbf{x}_{i,j}^{\mathrm{T}}\}_{a_i \in U_d^{ID}, 1 \le j \le \eta} \in \mathbb{Z}_q^m$.
3. Set $SK_{3,d}^{ID} = \{\mathbf{TK}_{2,i}\}_{a_i \in U_i^{ID}} \in \mathbb{Z}_q^m$.
4. Set $SK_d^{ID} = \{SK_{2,d}^{ID}, SK_{3,d}^{ID}\}$.
5. Send SK_d^{ID} to ID.

Finally, ID sets $SK^{ID} = SK_1^{ID} \cup \{SK_d^{ID}\}_{a_d \in U_d^{ID}}$.

Note: With the ***Enroll*** algorithm, the Initializer can add new users to the system without affecting other users, thereby achieving the expandability and independence properties.

4.4 Update

$(SK'^{ID}) \leftarrow \textbf{\textit{Update}}(PP, PK, MK, U'^{ID}, ID)$
After receiving the update request from a user ID. The initializer executes the user update using the following steps.

1. Randomly choose vectors $\mathbf{u'}_{ID}, \mathbf{u'}_{ID,1}, ..., \mathbf{u'}_{ID,\eta} \in \mathbb{Z}_q^n$.
2. Randomly choose a matrix $\mathbf{U'}_{ID,0} \in \mathbb{Z}_q^{n \times m'}$.
3. Choose an invertible matrix $\mathbf{H}_{ID} \in \mathbb{Z}_q^{n \times n}$.
4. $(\mathbf{U'}_{ID}, \mathbf{R'}_{\mathbf{U}_{ID}}) \leftarrow GenTrap(\mathbf{U'}_{ID,0}, \mathbf{H}_{ID})$.
5. Update $\mathbf{U}_1 = (\mathbf{U}_1 - \{\mathbf{u}_{ID,1}, ..., \mathbf{u}_{ID,\eta}\}) \cup \{\mathbf{u'}_{ID,1}, ..., \mathbf{u'}_{ID,\eta}\}, \mathbf{U}_2 = (\mathbf{U}_2 - \{\mathbf{U}_{ID}, \mathbf{u}_{ID}\}) \cup \{\mathbf{U'}_{ID}, \mathbf{u'}_{ID}\}$.
6. Set $SK'^{ID}_1 = \mathbf{R'}_{\mathbf{U'}_{ID}} \in \mathbb{Z}_q^{m' \times z}$.
7. Send SK'^{ID}_1 to ID.

AA_d updates ${SK'}_d^{ID}$ by performing the steps below:

1. $\forall a_i \in {U'}_d^{ID}$:
   - Choose an invertible matrix $\mathbf{H}_i \in \mathbb{Z}_q^{n \times n}$.
   - For $j = 1$ to η:
     - $\mathbf{x'}_{i,j} \leftarrow SampleD(\mathbf{R}_{\mathbf{A}_i}, \mathbf{A}_i, \mathbf{H}_i, \mathbf{u'}_{ID,j}, \sigma)$, such that $\mathbf{A}_i \cdot \mathbf{x'}_{i,j} = \mathbf{u'}_{ID,j}$.
   - $\mathbf{TK'}_{2,i} \leftarrow SampleD(\mathbf{R}_{\mathbf{B}_i}, \mathbf{B}_i, \mathbf{H}_i, H_0(ID) \cdot \mathbf{u'}_{ID}, \sigma)$, such that $\mathbf{B}_i \cdot \mathbf{TK'}_{2,i} = H_0(ID) \cdot \mathbf{u'}_{ID}$.
2. Set ${SK'}_{2,d}^{ID} = \{\mathbf{x'}_{i,j}^{\mathrm{T}}\}_{a_i \in {U'}_i^{ID}, 1 \le j \le \eta} \in \mathbb{Z}_q^m$.
3. Set ${SK'}_{3,d}^{ID} = \{\mathbf{TK'}_{2,i}\}_{a_i \in {U'}_d^{ID}} \in \mathbb{Z}_q^m$.
4. Set ${SK'}_d^{ID} = \{{SK'}_{2,d}^{ID}, {SK'}_{3,d}^{ID}\}$.
5. Send ${SK'}_d^{ID}$ to ID.

Finally, ID sets ${SK'}^{ID} = {SK'}_1^{ID} \cup \{{SK'}_d^{ID}\}_{a_d \in U_d^{ID}}$ and updates SK^{ID} by ${SK'}^{ID}$.

Note:

1. With the ***Update*** algorithm, AAs can independently generate new public and private keys for a user without affecting other users, thereby achieving the independence, revocability, and renewability properties.
2. Once the public key is updated, search tokens generated under the old public key remain valid for ciphertexts encrypted before the update but cannot be used to search ciphertexts created with the new public key. Only tokens generated under the updated key can access the corresponding new ciphertexts. This design maintains backward compatibility and forward security without requiring any re-encryption or re-indexing operations.

4.5 Extend

$(SK_t^{ID}) \leftarrow \textbf{\textit{Extend}}(PP, PK_t, MK_t, t, ID)$
After receiving the extend request from a user ID. The extension of the attribute is executed by AA_d involving an attribute a_t. AA_d generates a new private key of a_t for ID by performing the steps below:

1. Choose an invertible matrix $\mathbf{H}_t \in \mathbb{Z}_q^{n \times n}$.
2. For $j = 1$ to η :
   - $\mathbf{x}_{t,j} \leftarrow SampleD(\mathbf{R}_{\mathbf{A}_t}, \mathbf{A}_t, \mathbf{H}_t, \mathbf{u}_{ID,j}, \sigma)$, such that $\mathbf{A}_t \cdot \mathbf{x}_{t,j} = \mathbf{u}_{ID,j}$.
3. $\mathbf{TK}_{2,t} \leftarrow SampleD(\mathbf{R}_{\mathbf{B}_t}, \mathbf{B}_t, \mathbf{H}_t, H_0(ID) \cdot \mathbf{u}_{ID}, \sigma)$, such that $\mathbf{B}_t \cdot \mathbf{v}_t = H_0(ID) \cdot \mathbf{u}_{ID}$.
4. Set $SK_{2,t}^{ID} = \{\mathbf{x}_{t,j}^{\mathrm{T}}\}_{1 \le j \le \eta}$
5. Set $SK_{3,t}^{ID} = \mathbf{TK}_{2,t}^{\mathrm{T}}$
6. Set $SK_t^{ID} = \{{SK'}_{2,t}^{ID}, {SK'}_{3,t}^{ID}\}$.

7. Send SK_t^{ID} to ID.

Finally, ID updates $SK^{ID} = SK^{ID} \cup SK_t^{ID}$.

Note:

1. The ***Extend*** algorithm adds a single attribute for a user at a lower cost than the ***Update*** algorithm, as it does not require updating public keys or SK_1^{ID}.
2. The ***Enroll*** and ***Update*** algorithms enable dynamic membership support (expandability, revocability, renewability, and independence) as defined in [6].

4.6 Encrypt

$(CT, I, \tau) \leftarrow$ ***Encrypt***$(PP, \mathbf{b}, \tau, \{PK_d\}_{d \in U_C}, U_W)$
DO encrypts a message vector $\mathbf{b} = (b_1, b_2, \ldots, b_\eta) \in \{0,1\}^\eta$ with an access structure τ by performing the steps below:

1. Randomly choose a value $r \in \mathbb{Z}_q$.
2. Generate $\{r_i\}_{a_i \in \tau} \in \mathbb{Z}_q$ according to τ (See Section 3.4 for more details).
3. Randomly choose a secret vectors $\mathbf{s}_1 \in \mathbb{Z}_q^n$.
4. Randomly choose noises $\{e_{1,1}, e_{1,2}, \ldots, e_{1,j}\} \in \mathbb{Z}_q, \{\mathbf{e}_{2,1}, \mathbf{e}_{2,2}, \ldots, \mathbf{e}_{2,i}\} \in \mathbb{Z}_q^m$.
5. For each $ID \in L$:
   - for $(j = 1; j <= \eta; j++)$
     - Compute $C_{ID,j} = r \cdot \mathbf{u}_{ID,j}^{\mathrm{T}} \cdot \mathbf{s}_1 + e_{1,j} + b_j \cdot [q/2] \in \mathbb{Z}_q$.
6. For each $a_i \in \tau$:
   - Compute $C_i = r_i \cdot \mathbf{A}_i^{\mathrm{T}} \cdot \mathbf{s}_1 + \mathbf{e}_{2,i} \in \mathbb{Z}_q^m$.
7. Set $CT = (\{C_{ID,j}\}_{ID \in L, 1 \le j \le \eta}, \{C_i\}_{a_i \in \tau}, \tau)$.
8. Randomly choose a secret vector $\mathbf{s}_2 \in \mathbb{Z}_q^n$.
9. Randomly choose noises $\{\mathbf{e}_{3,1}, \mathbf{e}_{3,2}, \ldots, \mathbf{e}_{3,w}\} \in \mathbb{Z}_q^m, \{\mathbf{e}_{4,1}, \mathbf{e}_{4,2}, \ldots, \mathbf{e}_{4,i}\} \in \mathbb{Z}_q^m$.
10. For each $ID \in L$:
    - For $w = 1$ to l:
      - If $k_w \in U_W$:
        (a) Compute $I_{ID,w} = r \cdot H_1(k_w) \cdot \mathbf{U}_{ID}^{\mathrm{T}} \cdot H_0(ID)^{\mathrm{T}} \cdot \mathbf{s}_2 + \mathbf{e}_{3,w} \in \mathbb{Z}_q^m$.
      - Else:
        (a) Randomly choose $I_{ID,w} \in \mathbb{Z}_q^m$.
11. For each $a_i \in \tau$:
    - Compute $I_i = r_i \cdot \mathbf{B}_i^{\mathrm{T}} \cdot \mathbf{s}_2 + \mathbf{e}_{4,i} \in \mathbb{Z}_q^m$.
12. Set $I = (\{I_{ID,w}\}_{ID \in L, 1 \le w \le l}, \{I_i\}_{a_i \in \tau})$.

13. Upload (CT, I) to CSP.

Note: By using a tree-based structure defined in Section 3.4, when a set of attributes $\{a'_1, a'_2, \ldots, a'_j\}$ exactly satisfy τ, then $\sum_{i=1}^{j} r_i = r$. Consequently, the private keys $\{\mathbf{x}_{i,1}\}_{a_i \in U_C^{ID}}$ can exactly satisfy the equation: $\sum_{i=1}^{j} \mathbf{x}_{i,1}^{\mathrm{T}} \cdot r_i \cdot \mathbf{A}_i^{\mathrm{T}} = \sum_{i=1}^{j} r_i \cdot (\mathbf{A}_i \cdot \mathbf{x}_{i,1})^{\mathrm{T}} = \sum_{i=1}^{j} r_i \cdot \mathbf{u}_1^{\mathrm{T}} = r \cdot \mathbf{u}_1^{\mathrm{T}}$.

4.7 TokenGen

$(TK) \leftarrow \boldsymbol{TokenGen}(PP, SK^{ID}, \mathbf{W}', \tau')$
DU generates a search token with a search structure τ' by performing the steps:

1. Generate $\{r'_w\} \in \mathbb{Z}_q$ according to τ' (See Section 3.4 for more details).
2. For each keyword $k'_w \in \mathbf{W}'$:
   - $\mathbf{TK}_{1,w} \leftarrow SampleD(\mathbf{R}_{\mathbf{U}_{ID}}, \mathbf{U}_{ID}, \mathbf{H}_{ID}, H_1(k'_w)^{-1} \cdot r'_w \cdot \mathbf{u}_{ID}, \sigma)$, such that $\mathbf{U}_{ID} \cdot \mathbf{TK}_{1,w} = H_1(k'_w)^{-1} \cdot r'_w \cdot \mathbf{u}_{ID}$.
3. Set $TK = \{\{\mathbf{TK}_{1,w}\}_{k'_w \in \mathbf{W}'}, \{\mathbf{TK}_{2,i}\}_{a_i \in U^{ID}}, \tau'\}$
4. Send TK to CSP.

Note:

1. DU can use its private keys to generate search tokens. Thus, AA does not need to assist users in token generation.
2. By using a tree-based structure defined in Section 3.4, when a set of keywords $\{k_1^*, k_2^*, \ldots, k_j^*\}$ exactly satisfy τ', then $\sum_{w=1}^{j} r'_w = 1$. Consequently, the search tokens $\{\mathbf{TK}_{1,w}\}_{k_w \in U_W}$ can exactly satisfy the equation: $\sum_{w=1}^{j} \mathbf{TK}_{1,w}^{\mathrm{T}} \cdot H_1(k'_w) \cdot \mathbf{U}_{ID}^{\mathrm{T}} = \sum_{w=1}^{j} H_1(k'_w) \cdot (\mathbf{U}_{ID} \cdot \mathbf{TK}_{1,w})^{\mathrm{T}} = \sum_{w=1}^{j} H_1(k'_w) \cdot H_1(k'_w)^{-1} \cdot r'_w \cdot \mathbf{u}^{\mathrm{T}} = \sum_{w=1}^{j} r'_w \cdot \mathbf{u}^{\mathrm{T}} = \mathbf{u}^{\mathrm{T}}$.

4.8 Search

$(CT \text{ or } \perp) \leftarrow \boldsymbol{Search}(CT, TK, I, U^{ID})$
CSP receives a search token TK from DU. If U^{ID} does not exactly satisfy τ, CSP returns $\perp$. Otherwise, CSP can choose a set of attribute U_C^{ID} that can exactly satisfy τ, and performs the steps below:

1. For any set of keyword set $\{k'_1, k'_2, ..., k'_w\}$ satisfy τ':
   - Compute $z = \Sigma_{k_w \in U_W} \mathbf{TK}_{1,w}^{\mathrm{T}} \cdot I_{ID,w} - \Sigma_{a_i \in U_C^{ID}} \mathbf{TK}_{2,i}^{\mathrm{T}} \cdot I_{2,i}$.
   - If $|z| < \frac{q}{4}$, send the corresponding ciphertext CT to DU.
2. Otherwise, return $\perp$ to DU, indicating that TK did not match I.

Note:

1. The CSP, having U^{ID}, can select an attribute set U_C^{ID} that exactly satisfies τ. In the current scheme, CSP thus knows user attributes during searches; protecting these attributes is an important future research direction.

2. The CSP cannot access the keywords encrypted in I and must check all keyword combinations that satisfy τ'. The search cost depends on the structure of τ', with each OR gate increasing the number of satisfying keyword combinations the CSP must consider. Importantly, the CSP does not need to know the value of any r_i; it only verifies which keywords satisfy the search policy.

4.9 Decrypt

($\mathbf{b}$ or $\perp$) $\leftarrow$ ***Decrypt***(CT, U^{ID}, SK^{ID})
DU receives the CT from CSP. If U^{ID} can not exactly satisfy τ, DU returns $\perp$. Otherwise, DU can choose a set of attribute U_C^{ID} that can exactly satisfy τ, and decrypt the ciphertext by performing the steps below:

1. $\forall a_i \in U_C^{ID}$:

   - For every bit b_l in the message vector:
     - Compute $C_l^* = \Sigma_{a_i \in U_C^{ID}} \mathbf{x}_{i,l}^{\mathrm{T}} \cdot C_i$.
     - Compute $b_l' = C_{ID,l} - C_l^*$.
     - If $|b_l'| < \frac{q}{4}$, set $b_l = 0$; otherwise, set $b_l = 1$.
   - Return $\mathbf{b} = \{b_1, b_2, \ldots, b_l\}$.

4.10 Correctness

The proposed scheme satisfies correctness if the data user possesses attribute-based private keys that satisfy the ciphertext's access structure and submits a search token matching the keyword index. In this case, the CSP returns the corresponding ciphertexts, and the decryption algorithm will recover the original plaintext. Otherwise, the decryption fails or the result is empty.

Assume that an attribute set U_C^{ID} can exactly satisfy τ, indicating that $\Sigma_{i \in U_C^{ID}} r_i = r$. On the other hand, assume that a keyword set U_W can exactly satisfy τ', indicating that $\Sigma_{w \in U_W} w_w = 1$.

- Search Correctness

$$\begin{aligned}
z &= \Sigma_{k_w \in U_W} \mathbf{TK}_{1,w}^{\mathrm{T}} \cdot I_{ID,w} - \Sigma_{a_i \in U_C^{ID}} \mathbf{TK}_{2,i}^{\mathrm{T}} \cdot I_i \\
&= \Sigma_w \mathbf{TK}_{1,w}^{\mathrm{T}} \cdot r \cdot H_1(k'_w) \cdot \mathbf{U}_{ID}^{\mathrm{T}} \cdot H_0(ID)^{\mathrm{T}} \cdot \mathbf{s}_2 \\
&\quad + \Sigma_w \mathbf{TK}_{1,w}^{\mathrm{T}} \cdot \mathbf{e}_{3,w} - \Sigma_i \mathbf{TK}_{2,i}^{\mathrm{T}} \cdot r_i \cdot \mathbf{B}_i^{\mathrm{T}} \cdot \mathbf{s}_2 \\
&\quad - \Sigma_i \mathbf{TK}_{2,i}^{\mathrm{T}} \cdot \mathbf{e}_{4,i} \\
&= \Sigma_w r \cdot H_1(k'_w) \cdot H_1(k'_w)^{-1} \cdot r'_w \cdot \mathbf{u}_{ID}^{\mathrm{T}} \cdot H_0(ID)^{\mathrm{T}} \cdot \mathbf{s}_2 \\
&\quad + \Sigma_w \mathbf{TK}_{1,w}^{\mathrm{T}} \cdot \mathbf{e}_{3,w} - \Sigma_i \mathbf{TK}_{2,i}^{\mathrm{T}} \cdot r_i \cdot \mathbf{B}_i^{\mathrm{T}} \cdot \mathbf{s}_2 \\
&\quad - \Sigma_i \mathbf{TK}_{2,i}^{\mathrm{T}} \cdot \mathbf{e}_{4,i} \\
&= r \cdot \mathbf{u}_{ID}^{\mathrm{T}} \cdot H_0(ID)^{\mathrm{T}} \cdot \mathbf{s}_2 + \Sigma_w \mathbf{TK}_{1,w}^{\mathrm{T}} \cdot \mathbf{e}_{3,w} \\
&\quad - \Sigma_i \mathbf{TK}_{2,i}^{\mathrm{T}} \cdot r_i \cdot \mathbf{B}_i^{\mathrm{T}} \cdot \mathbf{s}_2 - \Sigma_i \mathbf{TK}_{2,i}^{\mathrm{T}} \cdot \mathbf{e}_{4,i} \\
&= r \cdot \mathbf{u}_{ID}^{\mathrm{T}} \cdot H_0(ID)^{\mathrm{T}} \cdot \mathbf{s}_2 + \Sigma_w \mathbf{TK}_{1,w}^{\mathrm{T}} \cdot \mathbf{e}_{3,w} \\
&\quad - \Sigma_i r_i \cdot \mathbf{u}_{ID}^{\mathrm{T}} \cdot H_0(ID)^{\mathrm{T}} \cdot \mathbf{s}_2 - \Sigma_i \mathbf{TK}_{2,i}^{\mathrm{T}} \cdot \mathbf{e}_{4,i} \\
&= r \cdot \mathbf{u}_{ID}^{\mathrm{T}} \cdot H_0(ID)^{\mathrm{T}} \cdot \mathbf{s}_2 + \Sigma_w \mathbf{TK}_{1,w}^{\mathrm{T}} \cdot \mathbf{e}_{3,w} \\
&\quad - r \cdot \mathbf{u}_{ID}^{\mathrm{T}} \cdot H_0(ID)^{\mathrm{T}} \cdot \mathbf{s}_2 - \Sigma_i \mathbf{TK}_{2,i}^{\mathrm{T}} \cdot \mathbf{e}_{4,i} \\
&= \underbrace{\Sigma_w \mathbf{TK}_{1,w}^{\mathrm{T}} \cdot \mathbf{e}_{3,w} - \Sigma_i \mathbf{TK}_{2,i}^{\mathrm{T}} \cdot \mathbf{e}_{4,i}}_{\text{error term}}
\end{aligned}$$

To ensure the correctness of search results, error terms must hold the following formula:

$$\left| \Sigma_{k_w \in U_W} \mathbf{TK}_{1,w}^{\mathrm{T}} \cdot \mathbf{e}_{3,w} - \Sigma_{a_i \in U_C^{ID}} \mathbf{TK}_{2,i}^{\mathrm{T}} \cdot \mathbf{e}_{4,i} \right| < \frac{q}{4}.$$

- Decryption Correctness

$$\begin{aligned}
b_j &= C_{ID,j} - \Sigma_{a_i \in U_C^{ID}} \mathbf{x}_{i,j}^{\mathrm{T}} \cdot C_i \\
&= r \cdot \mathbf{u}_{ID,j}^{\mathrm{T}} \cdot \mathbf{s}_1 + b_j \cdot [q/2] + e_{1,j} \\
&\quad - \Sigma_{a_i \in U_C^{ID}} (\mathbf{x}_{i,j}^{\mathrm{T}} \cdot r_i \cdot \mathbf{A}_i^{\mathrm{T}} \cdot \mathbf{s}_1 + \mathbf{x}_{i,j}^{\mathrm{T}} \cdot \mathbf{e}_{2,i}) \\
&= r \cdot \mathbf{u}_{ID,j}^{\mathrm{T}} \cdot \mathbf{s}_1 + b_j \cdot [q/2] + e_{1,j} \\
&\quad - \Sigma_{a_i \in U_C^{ID}} (r_i \cdot \mathbf{u}_{ID,j}^{\mathrm{T}} \cdot \mathbf{s}_1 + \mathbf{x}_{i,j}^{\mathrm{T}} \cdot \mathbf{e}_{2,i}) \\
&= r \cdot \mathbf{u}_{ID,j}^{\mathrm{T}} \cdot \mathbf{s}_1 + b_j \cdot [q/2] + e_{1,j} - (r \cdot \mathbf{u}_{ID,j}^{\mathrm{T}} \cdot \mathbf{s}_1 \\
&\quad \Sigma_{a_i \in U_C^{ID}} \mathbf{x}_{i,j}^{\mathrm{T}} \cdot \mathbf{e}_{2,i}) \\
&= b_j \cdot [q/2] + \underbrace{e_{1,j} - \Sigma_{a_i \in U_C^{ID}} \mathbf{x}_{i,j}^{\mathrm{T}} \cdot \mathbf{e}_{2,i}}_{\text{error term}}
\end{aligned}$$

To ensure the correctness of decrypt results, error terms must hold the following formula:

$$\left| e_{1,j} - \Sigma_{a_i \in U_C^{ID}} \mathbf{x}_{i,j}^{\mathrm{T}} \cdot \mathbf{e}_{2,i} \right| < \frac{q}{4}.$$

5 Security Proof

We formally demonstrate that the proposed scheme achieves ciphertext indistinguishability under chosen-plaintext attacks (IND-CPA) and keyword privacy against chosen-keyword attacks

(IND-CKA), both relying on the computational hardness of the decisional learning with errors (D-LWE) problem. For clarity, Fig. 3 illustrates the IND-CPA security game that underpins our proof framework. Due to space constraints, the complete formal proofs are deferred to a full version, which can be made available upon request.

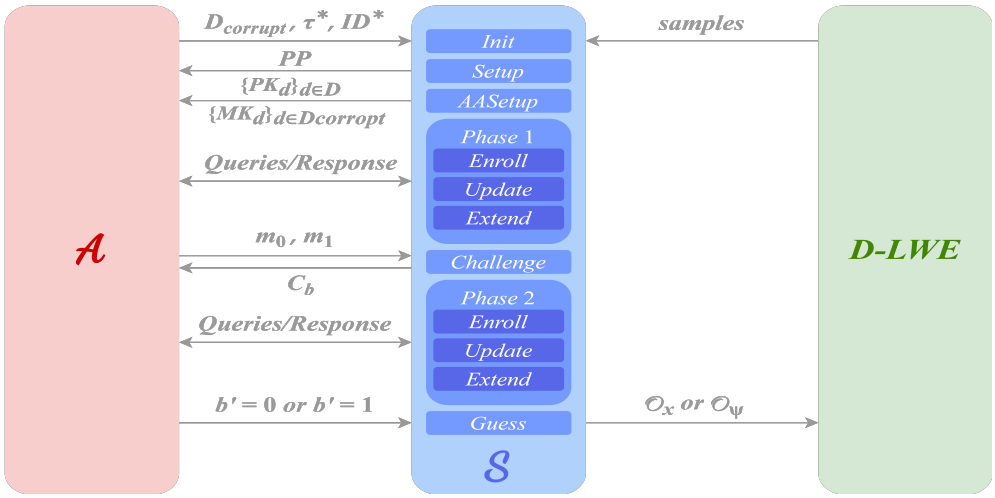


Figure 3: The IND-CPA Game

6 Comparisons

This section analyzes the communication and computation costs of the proposed scheme and compares them with related works. Schemes that do not support searchable encryption are excluded from the comparisons involving index length, token length, index encryption, and search operations. The properties of various LWE-based ABE schemes are summarized in Section 2.

As shown in Table 3, under the same security level, the parameter sizes required by the Type-2 trapdoor are significantly smaller than those of the Type-1 trapdoor [11]. Thus, adopting the Type-2 trapdoor can notably reduce computational and transmission overhead. However, it supports fewer functions than the Type-1 variant, increasing the challenge in scheme construction. Table 4 presents parameter selections aligned with real-world scenarios to ensure the practicality and representativeness of the performance evaluation.

Table 3: Parameters Setting for Trapdoor Types

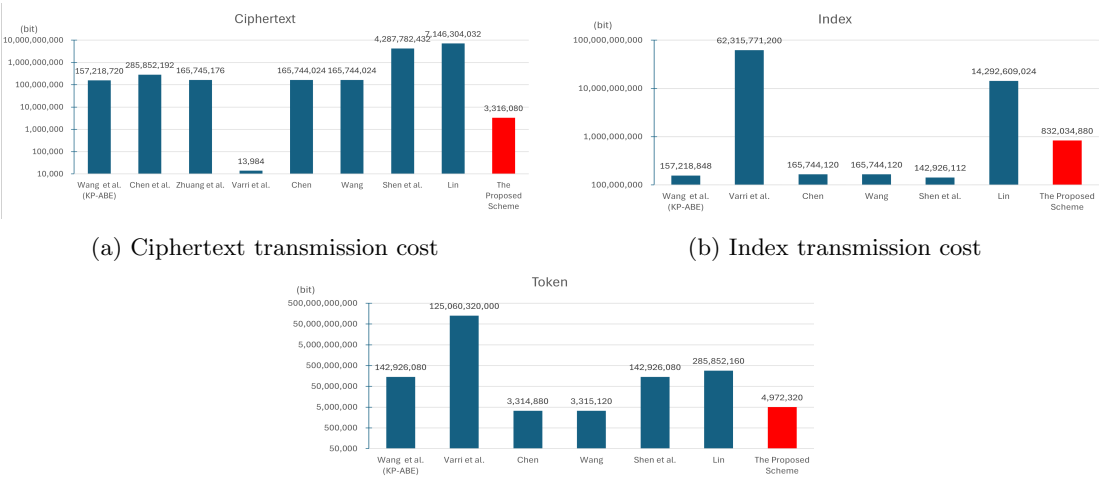
	Type-1		Type-2	
Description	Notation	Value	Notation	Value
Modulo	q_1	2^{32}	q_2	2^{24}
The number of components in a vector	n_1	436	n_2	284
The count of vectors comprising a basis	m_1	446644	m_2	13812

Reference: [11].

Table 4: Parameters Setting

Description	Notation	Value
The count of a user's attributes	l	10
The count of users in the system	u	50
The count of keywords in the ciphertext	w	5
The count of keywords in the system	w'	50
The size of the array in a bloom filter	k	32

6.1 Transmission Cost



(a) Ciphertext transmission cost

(b) Index transmission cost

(c) Token transmission cost

Figure 4: Transmission cost comparison for ciphertext, index, and token (lower values indicate better performance).

In Figure 4a, Varri *et al.*'s scheme [14] achieves the lowest ciphertext transmission cost. However, since their ciphertexts do not embed an access structure and decryption lacks access control, all users can decrypt them. Excluding this scheme, the proposed scheme achieves the best ciphertext transmission efficiency among the remaining works.

In Figure 4b, although the proposed scheme incurs higher index transmission costs compared to LSSS-based schemes [5, 13, 15], the use of a tree-based structure mitigates noise, reducing search errors. Moreover, the proposed scheme reduces index size by a factor of 17 compared to Lin's scheme [10]. Shen *et al.*'s scheme achieves the smallest index cost due to skipping attribute verification during search.

In Figure 4c, the proposed scheme maintains a comparable search token transmission cost while providing a flexible search structure. Although Chen's [5] and Wang's [15] schemes have smaller token sizes, they rely on LSSS-based structures, which may suffer from noise accumulation and search failures.

6.2 Computation Cost

We compare the computation costs of the ***Encrypt***, ***Search***, and ***Decrypt*** algorithms. The evaluation was conducted on a VMware virtual machine running Kali Linux 2024.1, equipped with a 4-core AMD Ryzen 5 5600 CPU @ 3.5GHz, 10GB RAM, and Python 3.11.9 with NumPy 1.24.2.

To ensure fairness, we simulated the time required for various operations under the parameters defined in Table 3 and Table 4. As shown in Table 5, matrix-matrix operations involving m introduce significant computation overhead due to the large size of m. In contrast, vector operations are comparatively lightweight. Benefiting from smaller parameter sizes, computations under the Type-2 trapdoor configuration require substantially less time than those under Type-1.

Table 5: The Operations Cost

Under Type-1 Trapdoor		Under Type-2 Trapdoor	
Multiplication Operation	Cost (Milliseconds)	Multiplication Operation	Cost (Milliseconds)
$\mathbb{Z}_{q_1} \times \mathbb{Z}_{q_1}$	0.002	$\mathbb{Z}_{q_2} \times \mathbb{Z}_{q_2}$	0.002
$\mathbb{Z}_{q_1}^{1\times n_1} \times \mathbb{Z}_{q_1}^{n_1\times 1}$	0.237	$\mathbb{Z}_{q_2}^{1\times n_2} \times \mathbb{Z}_{q_2}^{n_2\times 1}$	0.214
$\mathbb{Z}_{q_1}^{m_1\times n_1} \times \mathbb{Z}_{q_1}^{n_1\times 1}$	90,488.355	$\mathbb{Z}_{q_2}^{m_2\times n_2} \times \mathbb{Z}_{q_2}^{n_2\times 1}$	2,351.266
$\mathbb{Z}_{q_1}^{m_1\times n_1} \times \mathbb{Z}_{q_1}$	81,138.582	$\mathbb{Z}_{q_2}^{m_2\times n_2} \times \mathbb{Z}_{q_2}$	1,616.059
$\mathbb{Z}_{q_1}^{1\times m_1} \times \mathbb{Z}_{q_1}^{m_1\times 1}$	207.145	$\mathbb{Z}_{q_2}^{1\times m_2} \times \mathbb{Z}_{q_2}^{m_2\times 1}$	5.095
$\mathbb{Z}_{q_1} \times \mathbb{Z}_{q_1}^{m_1}$	186.404	$\mathbb{Z}_{q_2} \times \mathbb{Z}_{q_2}^{m_2}$	3.949
$\mathbb{Z}_{q_1}^{2m_1\times n_1} \times \mathbb{Z}_{q_1}^{n_1\times 1}$	178,536.294	$\mathbb{Z}_{q_2}^{2m_2\times n_2} \times \mathbb{Z}_{q_2}^{n_2\times 1}$	4,798.257
$\mathbb{Z}_{q_1}^{2m_1\times n_1} \times \mathbb{Z}_{q_1}$	167,260.665	$\mathbb{Z}_{q_2}^{2m_2\times n_2} \times \mathbb{Z}_{q_2}$	3,205.449
$\mathbb{Z}_{q_1}^{1\times 2m_1} \times \mathbb{Z}_{q_1}^{2m_1\times 1}$	401.815	$\mathbb{Z}_{q_2}^{1\times 2m_2} \times \mathbb{Z}_{q_2}^{2m_2\times 1}$	10.257
$\mathbb{Z}_{q_1} \times \mathbb{Z}_{q_1}^{2m_1}$	360.881	$\mathbb{Z}_{q_2} \times \mathbb{Z}_{q_2}^{2m_2}$	7.782
$\mathbb{Z}_{q_1}^{l} \times \mathbb{Z}_{q_1}^{l}$	0.013	$\mathbb{Z}_{q_2}^{l} \times \mathbb{Z}_{q_2}^{l}$	0.013
$\mathbb{Z}_{q_1}^{n_1\times 2m_1} \times \mathbb{Z}_{q_1}^{2m_1\times 1}$	207,941.511		

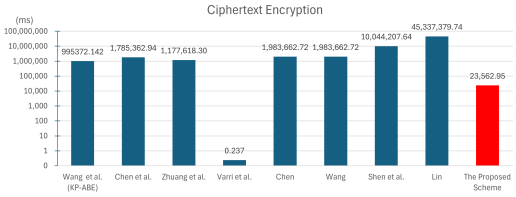
Figure 5a presents the computation cost of ciphertext encryption. Schemes such as Zhuang *et al.* [19], Chen [5], Wang [15], and Lin [10] generate independent ciphertexts per user to prevent collusion, which significantly increases computation. In contrast, the proposed scheme maintains collusion resistance with much lower encryption overhead.

Figure 5b shows that the proposed scheme outperforms others in index encryption efficiency while supporting both OR and AND gate queries. In contrast, schemes such as Chen *et al.* [4] and Zhuang *et al.* [19] do not provide searchable encryption features.

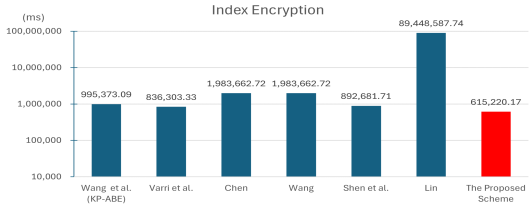
Figure 5c illustrates that the proposed scheme supports both AND and OR logic for multi-keyword search, unlike Lin [10] which supports only OR gates, and also achieves lower computation costs, highlighting its enhanced expressiveness and efficiency.

Figure 5d compares decryption costs. The proposed scheme performs on par with the most efficient schemes. In contrast, schemes using Type-1 trapdoor functions [4, 10, 13, 14, 16] suffer higher decryption overhead due to larger parameter sizes required for equivalent security levels.

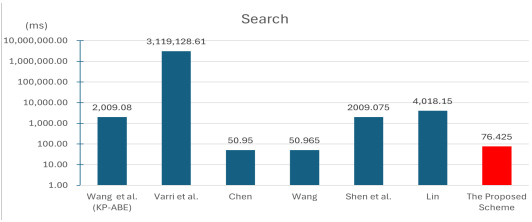
In summary, the proposed scheme achieves multi-authority and multi-keyword search with dynamic membership. It maintains low computation cost, supports flexible access and keyword policies, and prevents collusion attacks. Moreover, it demonstrates how multi-bit message encryption can be realized without significantly increasing the computation cost.



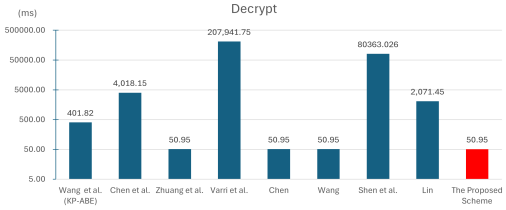
(a) Computation cost of Ciphertext encryption



(b) Computation cost of Index encryption



(c) Computation cost of Search



(d) Computation cost of Decryption

Figure 5: Computation cost comparison for ciphertext encryption, index encryption, search, and decryption (lower values indicate better performance).

7 Conclusion

We propose a novel lattice-based Attribute-Based Encryption (ABE) scheme, summarized in Table 1, which is the first tree-based CP-ABE from lattices supporting multi-keyword searches with both AND and OR gates. To mitigate key escrow, the scheme decentralizes authority and ensures each user has a unique public key component, preventing user collusion. Access control is enforced during the ***Search*** and ***Decrypt*** phases, enhancing both security and functionality.

The scheme supports offline attribute authorities, dynamic membership, and multi-bit message encryption, offering flexibility, scalability, and practical applicability. In terms of performance, it improves computational efficiency while maintaining competitive transmission costs and introduces functionalities beyond existing approaches. Security is guaranteed under IND-CPA and IND-CKA, relying on the hardness of the D-LWE problem. Due to space constraints, complete proofs are deferred to the full version and are available upon request.

In conclusion, our lattice-based ABE scheme provides an efficient, secure, and scalable framework for attribute-based encryption with multi-keyword search, suitable for a wide range of secure data-sharing applications.

8 Acknowledgments

This work was partially supported by the National Science and Technology Council (NSTC) of Taiwan under grants 114-2634-F-110-001-MBK and 114-2221-E-110-040. It also was supported by the MediaTek Advanced Research Center under the research contract MTKC-2024-1133. Additional financial support was provided by the Information Security Research Center at National Sun Yat-sen University in Taiwan and the Intelligent Electronic Commerce Research Center from the Featured Areas Research Center Program through the Framework of the Higher Education Sprout Project by the Ministry of Education in Taiwan.

References

[1] Shweta Agrawal, Xavier Boyen, Vinod Vaikuntanathan, Panagiotis Voulgaris, and Hoeteck Wee. Fuzzy identity based encryption from lattices. Cryptology ePrint Archive, Paper 2011/414, 2011.

[2] Xavier Boyen. Attribute-based functional encryption on lattices. In *Theory of Cryptography*, pages 122–142. Springer Berlin Heidelberg, 2013.

[3] Melissa Chase. Multi-authority attribute based encryption. In *Theory of Cryptography*, pages 515–534. Springer Berlin Heidelberg, 2007.

[4] E Chen, Yan Zhu, Kaitai Liang, and Hongjian Yin. Secure remote cloud file sharing with attribute-based access control and performance optimization. *IEEE Transactions on Cloud Computing*, 11(1):579–594, 2023.

[5] Xin-Jie Chen. Lattice-based searchable attribute-based encryption supporting dynamic membership management. Master thesis, National Sun Yat-sen University, 2021.

[6] Chun-I Fan, Vincent Shi-Ming Huang, and He-Ming Ruan. Arbitrary-state attribute-based encryption with dynamic membership. *IEEE Transactions on Computers*, 63(8):1951–1961, 2014.

[7] Nicholas Genise and Daniele Micciancio. Faster gaussian sampling for trapdoor lattices with arbitrary modulus. In *Advances in Cryptology – EUROCRYPT 2018*, pages 174–203. Springer International Publishing, 2018.

[8] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, page 197–206. Association for Computing Machinery, 2008.

[9] Changjiang Hou, Fei Liu, Hongtao Bai, and Lanfang Ren. Public-key encryption with keyword search from lattice. In *2013 Eighth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, pages 336–339, 2013.

[10] Lin. Multi-authority attribute-based multi-keyword searchable encryption with dynamic membership from lattices. Master thesis, National Sun Yat-sen University, 2023.

[11] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *Advances in Cryptology – EUROCRYPT 2012*, pages 700–718. Springer Berlin Heidelberg, 2012.

[12] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *STOC '05: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, 56(6), September 2009.

[13] Xiajiong Shen, Xiaoran Li, Hongjian Yin, Chaoyang Cao, and Lei Zhang. Lattice-based multi-authority ciphertext-policy attribute-based searchable encryption with attribute revocation for cloud storage. *Computer Networks*, 250:110559, 2024.

[14] Uma Sankararao Varri, Syam Kumar Pasupuleti, and KV Kadambari. CP-ABSEL: Ciphertext-policy attribute-based searchable encryption from lattice in cloud storage. *Peer-to-Peer Networking and Applications*, 14:1290–1302, 2021.

[15] De-Rong Wang. Multi-keyword searchable attribute-based encryption supporting dynamic membership management from lattices. Master thesis, National Sun Yat-sen University, 2022.

[16] Peng Wang, Biwen Chen, Tao Xiang, and Zhongming Wang. Lattice-based public key searchable encryption with fine-grained access control for edge computing. *Future Generation Computer Systems*, 127(C):373–383, February 2022.

[17] Guoyan Zhang, Jing Qin, and Shams Qazi. Multi-authority attribute-based encryption scheme from lattices. *JUCS - Journal of Universal Computer Science*, 21(3):483–501, 2015.

[18] Jiang Zhang, Zhenfeng Zhang, and Aijun Ge. Ciphertext policy attribute-based encryption from lattices. In *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '12, page 16–17. Association for Computing Machinery, 2012.

[19] Er-Shuo Zhuang, Chun-I Fan, and I-Hua Kuo. Multiauthority attribute-based encryption with dynamic membership from lattices. *IEEE Access*, 10:58254–58267, 2022.