Secure Remote Cloud File Sharing with Attribute-based Access Control and Performance Optimization

E Chen, Yan Zhu, *Member, IEEE,* Kaitai Liang, *Member, IEEE,* and Hongjian Yin

Abstract—The increasing popularity of remote Cloud File Sharing (CFS) has become a major concern for privacy breach of sensitive data. Aiming at this concern, we present a new resource sharing framework by integrating enterprise-side Attribute-Based Access Control/eXtensible Access Control Markup Language (ABAC/XACML) model, client-side Ciphertext-Policy Attribute-Based Encryption (CP-ABE) scheme, and cloud-side CFS service. Moreover, the framework workflow is provided to support the encrypted-file writing and reading algorithms in accordance with ABAC/XACML-based access policy and attribute credentials. However, an actual problem of realizing this framework is that policy matrix, derived from access policy, seriously affects the performance of existing CP-ABE from Lattice (CP-ABE-L) schemes. To end it, we present an optimal generation algorithm of Small Policy Matrix (SPM), which only consists of small elements, and generates an all-one reconstruction vector. Based on such a matrix, the improved CP-ABE-L scheme is proposed to reduce the cumulative errors to the minimum and prevent the enlargement of error bounds. Furthermore, we give the optimal estimation of system parameters to implement a valid Error Proportion Allocation (EPA). Our experimental results indicate that our scheme has short size of parameters and enjoys efficient computation and storage overloads. Thus, our new framework with optimization methods is conducive to enhancing the security and efficiency of remote work on CFS.

Index Terms—Security, Cloud File Sharing, ABAC/XACML, Attribute-Based Encryption, Small Policy Matrix.

1 Introduction

Cloud File Sharing (CFS), also called online file sharing, is a popular service in which a user is allocated storage space on a cloud server, and reading and writing files are carried out over the Internet.By keeping users own documents and media in cloud, the CFS is rapidly increasing because it provides seamless access to these data via any Internet-capable device, e.g., laptop, smartphone, tablet, from any location. This high demand for storage has nurtured the growth of a thriving cloud service industry that offers affordable, easy-to-use and remotely-accessible cloud services. Therefore, the CFS services, e.g., Amazon WorkDocs, iCloud Drive, Box, Dropbox, Egnyte, Google Drive, OneDrive, have been the primary choice of individuals and enterprises with increasing competitiveness.

The other advantage of CFS is to meet the requirement of remote working. The COVID-19 pandemic has created an abrupt need for employees to be moved out of corporate facilities and into virtual environments. According to Gallup [1] report, the percentage of full-time employees working from home due to COVID-19 closures jumped from 33% to 61% throughout the second half of March 2020. Moreover, 88% of organizations have encouraged or required their employees to work from home, according to a March 17 Gartner, Inc. survey of 800 global human resources (HR) executives. As a powerful tool, the CFS not only provides flexibility for employees remote work needs, but also helps saves business thousands of dollars in IT investments.

But as with every new technology, there are several security risks on the use of third-party remote cloud services, including:

- **Loss of control over sensitive data**, is that when using third-party file sharing services, the data privacy settings are beyond the control of the enterprise.
- **Concern of having the data leaked**, stems from the fact that the cloud is a multi-user environment, the data is potentially at risk of being viewed or mishandled by the provider, as well as a number of external threats.
- **Threats from hacker's snooping,** originate from the risk that files in the cloud are among the most susceptible to being hacked without security measures in place, and even if the CFS provides storage-oriented encryption for files, data can still be intercepted on route to its destination.

To avoid the above risks, some CFS services, such as Google Drive, can keep the cloud-bound documents protected with a password, but the password is still easy to crack. There are third-party client-oriented encryption tools, e.g., Boxcryptor, GarbleCloud, Veracypt, that promise to encrypt documents before uploading them, but the other users or collaborators having access to the file will need to provide the password, derived from the owner, to open the file. Considering this requires additional channels for key distribution, it is a fairly complex process that is not guaranteed. Moreover, these existing encryption tools do not support flexible data access and policy-based sharing on the enterprise-side. Thus, the best way is to ensure that the enterprises, rather than the CFS service, can encrypt the employee's files during storage and transmission under the control of their security experts. Therefore, in this paper we concentrate on an effective enterprise-side encryption framework with flexible access

- *E Chen, Y. Zhu and H. Yin were with the Department of School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing, 10083 China. E-mail: chene5546@163.com, zhuyan@ustb.edu.cn, honjanyin@163.com*
- *K. Liang joined the Cybersecurity group at Delft University of Technology, Van Mourik Broekmanweg 6, 2628 XE Delft, Netherlands. E-mail: Kaitai.Liang@tudelft.nl*

1. https://www.gallup.com/workplace/307622/leaders-responding-covid-workplace-disruption.aspx

control mechanism built on the CFS.

There exist a large number of classical public-key encryption schemes used as our candidates, but the schemes against quantum computing may be more worthy of consideration. The reason is that the traditional cryptosystems do not have the ability to cope with the potential risks incurred by the "quantum attacks", and post-quantum cryptography can guarantee *long-term* data security because the data may be stored on the network for a long time without being deleted. Furthermore, National Security Agency (NSA) pointed out the necessity for transition to quantum-resistant schemes is increasing, and then issued a policy statement on the development standards for post-quantum cryptography in 2015 [1], [2]. In 2016, National Institute of Standards and Technology (NIST) also announced a call for post-quantum standard submissions, including encryption schemes, digital signatures, and key-encapsulation mechanisms, for replacing the currently cryptosystems such as RSA, ECC [3]. Thus, we now must begin to prepare information security systems so as to be able to resist quantum computing.

Challenges. In order to provide access control mechanism for the CFS, Attribute-based Access Control (ABAC) may be the best choice to manage sensitive data under the control of enterprise. Strictly speaking, ABAC is a policy-based approach of controlling the authentication process based on attributes. Attributes are considered as descriptions of subjects, objects, actions, and environmental factors that are combined to create access policies and access requests. An example standard to realize ABAC is the eXtensible Access Control Markup Language (XACML), as shown in Fig. 1. The ABAC/XACML model supports and encourages the separation of enforcement (PEP), decision making (PDP), and management (PAP) of the authorization, meanwhile attribute credentials for authorization are specified from PIP in terms of the XACML requests.

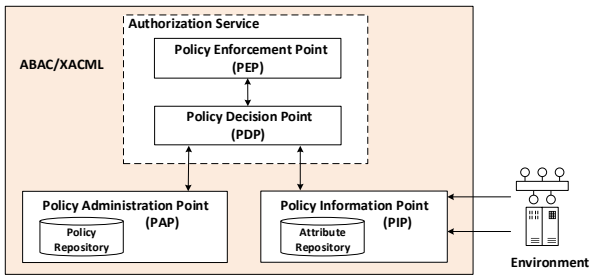


Fig. 1. The ABAC/XACML model.

One of the most suitable cryptosystems for ABAC/XACML is Attribute-based Encryption (ABE), that is a group-oriented cryptographic technique by integrating access policies with data encryption. In a Ciphertext-Policy ABE (CP-ABE) system, the user's private keys described by an attribute set are produced by a trust authority, and a ciphertext generated by a sender is associated with an access policy over attributes. Then, a user can decrypt this ciphertext only if there is a match between the attributes of the user's key and the policy of ciphertext. Therefore, the combination of ABE and ABAC/XACML will be adopted to guarantee security of migrating sensitive data to the CFS.

Despite good foundations on ABE and ABAC/XACML, there are still the following challenges for us:

- How to establish more effective solution to integrate enterprise-side ABAC/XACML management, client-side ABE encryption, and cloud-side CFS service into a complete data migration system; and
- How to improve the performance of ABE cryptosystem to encrypt sensitive data against the future security risks, including quantum attacks.

Thus, we will focus our attention to the solution of these two challenges in this paper.

Related works. Bethencourt et al. [4] proposed the first CP-ABE scheme using a threshold secret sharing, however, this scheme was only proved secure under the generic group heuristic. A general approach to convert KP-ABE systems into a CP-ABE system was presented by Goyal et al. [5] with a security proof based on the standard decisional bilinear Diffie-Hellman assumption. Herranz et al. [6] proposed the first dynamic (t, N)-threshold CP-ABE scheme with constant-size ciphertext, which was more expressive than merely AND gates. However, the above schemes were constructed on bilinear mapping techniques [7], [8]. In addition, Liu et al. [9] presented an algorithm that converts an arbitrary (t, N)-threshold access tree to the corresponding matrix of Linear Secret Sharing Scheme (LSSS), which provides a meaningful reference to design highly expressive CP-ABE schemes.

To integrate ABE and ABAC together, Zhu et al. [10] presented an efficient temporal access control encryption scheme for cloud services with the help of cryptographic integer comparisons and a proxy-based re-encryption mechanism on the current time. Soon afterwards, a practical Cryptographic ABAC (CABAC) framework was introduced by them [11] to support provable policy decision-making and verifiable attribute Tokens among multiple distributed authorities. In summary, those above works provides us good tools to establish a cryptographic solution to integrate ABAC and ABE.

In the post-quantum cryptography, seeing that the hardness of lattice problem is considered to resist quantum attacks, researchers are inspired to construct ABE schemes with lattices cryptographic primitive. Zhang et al. [12] first utilized the lattice theory to develop an ABE scheme, which was secure under the LWE assumption. However, the scheme had an increase in the size of system parameters due to the introduction of default attributes and two "trapdoor" techniques. Boyen [13] introduced the linear secret sharing scheme into a monotone access structure to construct a KP-ABE scheme, but the size of private key and ciphertext were too long. Wang [14] presented a lattice CP-ABE scheme under the LWE assumption, but the scheme only supported AND-gates on attributes and the size of public key was relatively long that would lead to high storage and communication costs. Dai et al. [15] proposed an Ring LWE-based construction of KP-ABE, which supported both ciphertext and public key homomorphism. Chen et al. [16] proposed two CP-ABE schemes under the Ring LWE assumption, which only supported a single threshold structure. Also, in our improvement, we refer to the lattice trapdoor sampling optimization proposed recently in [17], [18], [19], [20].

Approach. The major problem of previous design on lattice-based ABE is considerable size of parameters caused by access policies. We attempt to develop a practical ap-

proach to design a more expressive CP-ABE scheme but with small size of parameters. This requires us to design an algorithm of generating a policy matrix $\mathbf{W}$ whose inverse has small coefficients. We call it a Small Policy Matrix (SPM). For a given policy, the policy matrix is public but not unique, so we need to optimize it to meet our requirement. Considering that the error item e_i is introduced into the aforementioned equation by using the LWE construction, for a small ϵ, the decryption process will rely on the equation $\| \sum_i (\mathbf{W}^{-1})_{1i} e_i \| \leq \epsilon$, so each component of $(\mathbf{W}^{-1})_1$ must be as small as possible [2]. On the contrary, for lattices in $\mathbb{Z}_q$, the coefficients of $(\mathbf{W}^{-1})_1$ is uniformly distributed on $\mathbb{Z}_q$ since $\mathbf{W}^{-1}$ is non-optimal in the existing ABE schemes, so it will result in the failure of the decryption process with non-negligible probability.

Our approach constructs a small policy matrix $\mathbf{W}$ supporting both logic AND and OR gates, which is an approximate "sparse matrix" containing only $-1, 0$ and 1. This is obviously different from the existing schemes [9] on policy matrix construction, e.g., Vandermonde matrix. Moreover, this ensures that the inverse of candidate submatrix $(\mathbf{W}^*)^{-1}$ only contains small integer coefficients. Thus, the absolute value of the determinant of $(\mathbf{W}^*)^{-1}$ can only be 1, which is commonly far less than that of other schemes, e.g., the determinant of inverse of Vandermonde matrix [9]. Furthermore, we prove that the coefficients of $(\mathbf{W}^{-1})_1$ are all ONEs in this paper, thus the policy matrix generated by our scheme is optimal and meet the requirement of minimizing the error amplification range in the lattice-based cryptosystem. At this time, we obtain the minimum solution

$$\min(\| \sum\nolimits_i (\mathbf{W}^{-1})_{1i} e_i \|) = \| \sum\nolimits_i e_i \|$$

for a distribution $e_i \leftarrow \chi$ and all validated $\mathbf{W}$ in any dimensional space. Compared with the aforementioned schemes [12], [21], our scheme achieves shorter parameters, such as $q \geq 2^{46}$ in our scheme vs. $q \geq 2^{124}$ in [12], [21] for $m = 2^{15}$, from the results in Section 5.4.

Contribution. We focus on a new solution to protect sensitive data in remote CFS against privacy breach and quantum attacks. To further improve the performance of solution, we optimize the existing CP-ABE from Lattice (CP-ABE-L) by using optimal policy matrix and minimized cumulative errors. Our work is summarized as follows.

- We present a new resource sharing framework by integrating enterprise-side ABAC/XACML model, client-side CP-ABE scheme, and cloud-side CFS service. This framework workflow is provided to support encrypted-file writing and reading operations in accordance with ABAC/XACML based access policy and attribute credentials.
- Aiming at the problem that policy matrix seriously affects the performance of existing CP-ABE-L, we present an optimal generation algorithm of small policy matrix, which only consists of elements in $\{-1, 0, 1\}$, and generates an ALL-ONE reconstruction vector. Based on such a matrix, the improved CP-ABE-L scheme is proposed to reduce the cumulative errors to the minimum.
- To further optimize the performance of CP-ABE-L, we analyze the bound of error term and give the optimal estimation of system parameters (e.g., modulus' size is less than 64 bits) to implement a valid Error Proportion Allocation (EPA). Our experimental results indicates that our scheme has short size of parameters and enjoys efficient computation and storage overloads.

Organization. Some related preliminaries and backgrounds are reviewed in Section 2. Our system model is described in Section 3. In Section 4, we propose a construction of SPM and a improved CP-ABE-L scheme. In Section 5, we provide the parameters optimization and performance analysis. The paper concludes in Section 6.

2 Preliminaries and Backgrounds

Vectors will be written in column form and by bold lowercase letters (e.g., $\mathbf{x}$). Denote x_i be the i-th component of $\mathbf{x}$. Roman T represents the transposition of vector. Set matrix denoted by bold capital letters (e.g., $\mathbf{A}$) as the set of its column vectors. And, define $|\mathbf{A}|$ be the determinant of $\mathbf{A}$. ℓ_2 and ℓ_∞ norm will be denoted by $\|\cdot\|$ and $\|\cdot\|_\infty$, respectively. We view $\|\mathbf{A}\|$ as the Euclidean norm of the longest column in matrix $\mathbf{A}$, i.e. $\|\mathbf{A}\| = \max_i \|\mathbf{a}_i\|$. Let $\tilde{\mathbf{A}}$ be the Gram-Schmidt orthogonalization of $\mathbf{A}$. For $\mathbf{A} \in \mathbb{R}^{n \times m}$ and $\mathbf{B} \in \mathbb{R}^{n \times l}$, define $(\mathbf{A}\|\mathbf{B}) \in \mathbb{R}^{n \times (m+l)}$ be the concatenation of $\mathbf{A}$'s columns followed by $\mathbf{B}$'s columns.

2.1 Lattices

A m-dimensional lattice Λ is defined as $\Lambda = \{\sum_{i=1}^m x_i \mathbf{a}_i | x_i \in \mathbb{Z}\}$, where the sequence of m linearly independent vectors $\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_m \in \mathbb{R}^n$ is a basis of Λ. For a prime q and any fixed $\mathbf{u} \in \mathbb{Z}_q^n$, we describe three full-rank m-dimensional integer lattices defined by $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_m] \in \mathbb{Z}_q^{n \times m}$ as follows,

$$\begin{aligned} \Lambda_q(\mathbf{A}) &= \{\mathbf{e} \in \mathbb{Z}^m s.t. \exists \mathbf{s} \in \mathbb{Z}_q^n, \mathbf{A}^{\mathrm{T}}\mathbf{s} = \mathbf{e} \mod q\}, \\ \Lambda_q^{\perp}(\mathbf{A}) &= \{\mathbf{e} \in \mathbb{Z}^m s.t. \mathbf{A}\mathbf{e} = 0 \mod q\}, \\ \Lambda_q^{\mathbf{u}}(\mathbf{A}) &= \{\mathbf{e} \in \mathbb{Z}^m s.t. \mathbf{A}\mathbf{e} = \mathbf{u} \mod q\}. \end{aligned}$$

Discrete Gaussian. For any vector $\mathbf{c} \in \mathbb{R}^m$ and any positive parameter $\sigma > 0$, define $\rho_{\sigma,\mathbf{c}}(\mathbf{x}) = \exp(-\pi \frac{\|\mathbf{x}-\mathbf{c}\|^2}{\sigma^2})$ be the Gaussian function on $\Lambda \subset \mathbb{Z}^n$ with center $\mathbf{c}$ and parameter σ. Let the sum of $\rho_{\sigma,\mathbf{c}}$ over Λ as $\rho_{\sigma,\mathbf{c}}(\Lambda) = \sum_{\mathbf{x} \in \Lambda} \rho_{\sigma,\mathbf{c}}(\mathbf{x})$. And, for $\forall \mathbf{y} \in \Lambda$, define $\mathcal{D}_{\Lambda,\sigma,\mathbf{c}}(\mathbf{y}) = \frac{\rho_{\sigma,\mathbf{c}}(\mathbf{y})}{\rho_{\sigma,\mathbf{c}}(\Lambda)}$ be the discrete Gaussian distribution over Λ with center $\mathbf{c}$ and parameter σ. Specifically, $\rho_{\sigma,0}$ and $\mathcal{D}_{\Lambda,\sigma,0}$ are abbreviated as ρ_σ and $\mathcal{D}_{\Lambda,\sigma}$ when σ and $\mathbf{c}$ are 1 and 0, respectively. Moreover, $\mathcal{D}_{\Lambda,\sigma,\mathbf{c}}$ is always defined over the lattice $\Lambda_q^{\perp}$ for $\mathbf{A}$ or $\Lambda_q^{\mathbf{u}}(\mathbf{A})$.

2.2 Sampling Algorithms

The following algorithms are used to sample short vectors from specific lattices. Looking ahead, we will use **SampleLeft** and **SampleRight** algorithms to generate the private keys in real system and respond the private key queries made by the adversary in simulation, respectively.

Definition 1 (Sampling Algorithms [22]) *For $q > 2$ and $m > n$, the algorithms are defined as follows.*

2. $(\mathbf{W}^{-1})_1$ denotes the 1-st row of inverse of $\mathbf{W}$ and $(\mathbf{W}^{-1})_{1i}$ is the i-th element of the above row.

- *Algorithm* ***SampleLeft***$(\mathbf{A}, \mathbf{B}, \mathbf{T_A}, \sigma, \mathbf{u})$*: takes a full rank matrix* $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$*, a matrix* $\mathbf{B} \in \mathbb{Z}_q^{n\times m_1}$*, a short basis* $\mathbf{T_A}$ *of* $\Lambda_q^{\mathbf{u}}(\mathbf{A})$*, a Gaussian parameter* $\sigma > \|\tilde{\mathbf{T}}_\mathbf{A}\|\omega(\sqrt{\log(m+m_1)})$*, and a vector* $\mathbf{u}$ *as inputs, outputs a vector* $\mathbf{e} \in \mathbb{Z}_q^{m+m_1}$ *distributed statistically close to* $\mathcal{D}_{\Lambda_q^{\mathbf{u}}(\mathbf{F}),\sigma}$*, i.e.,* $\mathbf{Fe} = \mathbf{u}$*, where* $\mathbf{F} = (\mathbf{A}\|\mathbf{B})$.
- *Algorithm* ***SampleRight***$(\mathbf{A}, \mathbf{B}, \mathbf{R}, \mathbf{T_B}, \sigma, \mathbf{u})$*: takes matrices* $\mathbf{A}, \mathbf{B} \in \mathbb{Z}_q^{n\times m}$*, and a uniform random matrix* $\mathbf{R} \in \{-1, 1\}^{m\times m}$*, a short basis* $\mathbf{T_B}$ *of* $\Lambda_q^{\mathbf{u}}(\mathbf{B})$*, a Gaussian parameter* $\sigma > \|\tilde{\mathbf{T}}_\mathbf{B}\|\sqrt{m}\omega(\sqrt{\log(m)})$*, and a vector* $\mathbf{u}$ *as inputs, outputs a vector* $\mathbf{e} \in \mathbb{Z}_q^{2m}$ *distributed statistically close to* $\mathcal{D}_{\Lambda_q^{\mathbf{u}}(\mathbf{F}),\sigma}$*, i.e.,* $\mathbf{Fe} = \mathbf{u}$*, where* $\mathbf{F} = (\mathbf{A}\|\mathbf{AR} + \mathbf{B})$.

The following lemma is also needed for our security proof.

Lemma 1 ([22]) *For a prime* q*,* $m > (n+1)\log_2 q + \omega(\log n)$*, randomly choose matrices* $\mathbf{A}, \mathbf{B} \in \mathbb{Z}_q^{n\times m}$*. Let* $\mathbf{R}$ *be an* $m \times m$ *matrix chosen uniformly in* $\{-1, 1\}^{m\times m}$ *under* $\mathbb{Z}_q$*. Then for all vectors* $\mathbf{w} \in \mathbb{Z}_q^m$*, the distribution* $(\mathbf{A}, \mathbf{AR}, \mathbf{R}^{\mathrm{T}}\mathbf{w})$ *is statistically close to the distribution* $(\mathbf{A}, \mathbf{B}, \mathbf{R}^{\mathrm{T}}\mathbf{w})$.

2.3 Existing Researches

We now review the LWE problem defined by Regev [23].

Let $\mathbb{T} = \mathbb{R}/\mathbb{Z}$ be the group of reals $[0, 1)$ with modulo 1 addition. For a real $\alpha \in \mathbb{R}^+$, Ψ_α is the distribution on $\mathbb{T}$ of a normal variable with mean 0 and standard deviation $\alpha/\sqrt{2\pi}$, and reduced modulo 1. Let $\lfloor x \rceil = \lfloor x + \frac{1}{2} \rfloor$ be a nearest integer to $x \in \mathbb{R}$. For an integer q, define $\bar{\Psi}_\alpha$ as the discrete distribution over $\mathbb{Z}_q$ of the random variable $\lfloor qX \rceil \mod q$, where $X \in \mathbb{T}$ has distribution Ψ_α.

For a modulus $q \geq 2$ and a distribution χ on $\mathbb{Z}_q$, the LWE problem asks to recover a secret $s \in \mathbb{Z}_q^n$ given any samples $(\mathbf{a}, \mathbf{a}^{\mathrm{T}}\mathbf{s}+e)$ from distribution $A_{\mathbf{s},\chi}$ on $\mathbb{Z}_q^n \times \mathbb{Z}_q$, where $\mathbf{a} \in \mathbb{Z}_q^n$ and $e \leftarrow \chi$. Regev [23] shows for certain noise distributions χ, denoted by $\bar{\Psi}_\alpha$, the LWE problem is as hard as the worst-case SIVP and GapSVP under a quantum reduction.

Theorem 1 ([23]) *Let* $\alpha = \alpha(n) \in (0, 1)$ *and* $q = q(n)$ *be a prime such that* $\alpha q > 2\sqrt{n}$*. If there exists an efficient, possibly quantum algorithm that solves* $LWE_{q,\bar{\Psi}_\alpha}$ *problem, then there exists an efficient quantum algorithm for approximating the SIVP and GapSVP problems, to within* $\tilde{O}(n/\alpha)$ *factors in the* ℓ_2 *norm, in the worst case.*

The following theorem, derived from Theorem 3.2 of [17] taking $\delta = 1/3$, is an improved sampling algorithm that samples an essentially uniform matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ with an associated basis $\mathbf{S}$ of $\Lambda_q^{\perp}(\mathbf{A})$ with low Gram-Schmidt norm.

Theorem 2 ([17]) *Let* $q \geq 3$ *be odd and* $m = \lceil 6n \log q \rceil$*. There is a PPT algorithm* ***TrapGen***(q, n) *that outputs a pair* $(\mathbf{A} \in \mathbb{Z}_q^{n\times m}, \mathbf{S} \in \mathbb{Z}^{m\times m})$ *such that* $\mathbf{A}$ *is statistically close to a uniform matrix in* $\mathbb{Z}_q^{n\times m}$ *and* $\mathbf{S}$ *is a basis for* $\Lambda_q^{\perp}(\mathbf{A})$ *satisfying*

$$\|\tilde{\mathbf{S}}\| \leqslant O(\sqrt{n \log q}) \ \text{ and } \ \|\mathbf{S}\| \leqslant O(n \log q) \tag{1}$$

with all but negligible probability in n.

Micciancio and Regev [24] state an additional property of discrete Gaussian distribution as follows.

Lemma 2 ([24]) *For any* n*-dimensional lattice* Λ*, vector* $\mathbf{c} \in \mathbb{R}^n$*, and reals* $\epsilon \in (0, 1), \sigma \geq \eta_\epsilon(\Lambda)$*, we have*

$$\Pr_{\mathbf{x}\sim\mathcal{D}_{\Lambda,\sigma,\mathbf{c}}}[\|\mathbf{x} - \mathbf{c}\| > \sigma\sqrt{n}] \leq \frac{1+\epsilon}{1-\epsilon} \cdot 2^{-n}. \tag{2}$$

The following lemma about the distribution $\bar{\Psi}_\alpha$ is used to ensure that decryption works correctly.

Lemma 3 ([19], Lemma 3) *Let* $\mathbf{y}$ *be some vector in* $\mathbb{Z}^m$ *and let* $\mathbf{e} \leftarrow \bar{\Psi}_\alpha^m$*. Then the quantity* $|\mathbf{y}^{\mathrm{T}}\mathbf{e}|$ *treated as an integer in* $[0, q-1]$ *satisfies*

$$|\mathbf{y}^{\mathrm{T}}\mathbf{e}| \leq \|\mathbf{y}\|q\alpha\omega(\sqrt{\log m}) + \|\mathbf{y}\|\sqrt{m}/2, \tag{3}$$

with all but negligible probability in m*. As a special case, if* $e \leftarrow \bar{\Psi}_\alpha$ *is treated as an integer in* $[0, q-1]$*, then* $|e| \leq q\alpha\omega(\sqrt{\log m}) + 1/2$ *with all but negligible probability in* m.

2.4 CP-ABE from Lattices

A CP-ABE from Lattice (CP-ABE-L) is presented in accordance with the constructions in [12], [21]. The CP-ABE-L scheme consists of four algorithms, including **Setup, Extract, ABE-Enc** and **ABE-Dec**, as follows.

—-Initialization. The system manager chooses a security parameter n and an attribute set $\mathcal{U} = \{att_1, \cdots, att_N\}$, where the number of attributes N is unlimited. As described in Algorithm 1, the **Setup** algorithm is executed to generate a pair of the public key pk and the master secret key msk.

Algorithm 1 Setup$(n, \mathcal{U})$

Input: a positive integer n, and an attribute set $\mathcal{U}$.
Output: the public key $pk = (\mathbf{A}, \mathbf{B}, \mathbf{E}, \mathbf{u})$, and the master secret key $msk = \mathbf{T_A}$.

1: Generate the parameters m, q and σ;
2: Compute $(\mathbf{A}, \mathbf{T_A}) \leftarrow$ **TrapGen**(q, n);
3: (*$\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and $\mathbf{T_A} \in \mathbb{Z}_q^{m\times m}$ is a short basis for $\Lambda_q^{\perp}(\mathbf{A})$*)
4: Choose two random matrices $\mathbf{B}, \mathbf{E} \leftarrow \mathbb{Z}_q^{n\times m}$;
5: Choose a random vector $\mathbf{u} = (u_1, \cdots, u_n)^{\mathrm{T}} \leftarrow \mathbb{Z}_q^n$.;
6: **return** $(pk = (\mathbf{A}, \mathbf{B}, \mathbf{E}, \mathbf{u}), msk = \mathbf{T_A})$;

—-Extract Key. As shown in Algorithm 2, the **Extract** algorithm is used to generate the user's private key for a given attribute set $S \subseteq \mathcal{U}$ by using pk, msk and a cryptographic hash function $H : \{0, 1\}^* \rightarrow \mathbb{Z}_q^{n\times n}$.

Algorithm 2 Extract(pk, msk, S)

Input: the public key pk, the master secret key msk and a user's attribute set $S \subseteq \mathcal{U}$.
Output: the private key $sk = (\{\mathbf{d}_i\}_{\forall att_i \in S}, S)$.

1: **for** each att_i in S **do**
2: Set $\mathbf{A}_i = (\mathbf{A}\|\mathbf{E} + H(att_i)\mathbf{B}) \in \mathbb{Z}_q^{n\times 2m}$;
3: $\mathbf{d}_i \leftarrow$ **SampleLeft**$(\mathbf{A}, \mathbf{E}+H(att_i)\mathbf{B}, \mathbf{T_A}, \sigma, \mathbf{u})$;
4: (*$\mathbf{A}_i\mathbf{d}_i = \mathbf{u}$ and $\mathbf{d}_i \in \mathbb{Z}_q^{2m}$ is a short vector*)
5: **end for**
6: **return** $sk = (\{\mathbf{d}_i\}_{\forall att_i \in S}, S)$;

—-Encryption. In the CP-ABE-L scheme, the ciphertext is associated with an access policy Π expressed by a policy matrix $\mathbf{W} \in \mathbb{Z}_q^{k\times l}$ over $\mathcal{U}$, and the i-th row of $\mathbf{W}$ corresponds to $att_i \in \mathcal{U}$. Let $\mathbf{W}^*$ be a candidate invertible submatrix in $\mathbf{W}$. The **ABE-Enc** algorithm only needs to encrypt a message M by using pk and an integer $D = \mathrm{LCM}_{\forall \mathbf{W}^* \subseteq \mathbf{W}}(|(\mathbf{W}^*)^{-1}|)$,

Algorithm 3 ABE-Enc$(pk, \mathbf{W}, M)$

Input: the public key pk, a policy matrix $\mathbf{W} \in \mathbb{Z}_q^{k\times l}$ of the access policy Π, and a message bit $M \in \{0, 1\}$.
Output: the ciphertext $C = (c, \mathbf{Z}, \mathbf{W})$.

for each att_i belongs to Π **do**
  Calculate $D = \mathrm{LCM}_{\forall \mathbf{W}^* \subseteq \mathbf{W}}(|(\mathbf{W}^*)^{-1}|)$ in terms of $\mathbf{W}$;
  Set $\mathbf{A}_i = (\mathbf{A} \| \mathbf{E} + H(att_i)\mathbf{B})$;
  Choose l random vectors $\mathbf{s}, \mathbf{r}_2, \cdots, \mathbf{r}_l \leftarrow \mathbb{Z}_q^n$;
  Set $\mathbf{v} = (\mathbf{s}, \mathbf{r}_2, \cdots, \mathbf{r}_l)^{\mathrm{T}}$;
  Pick $e \leftarrow \chi$ and compute $c = \mathbf{u}^{\mathrm{T}}\mathbf{s} + De + M\lfloor \frac{q}{2} \rfloor \in \mathbb{Z}_q$;
  for $i = 1$ to k **do**
    Set $\mathbf{W}_i$ as the i-th row of $\mathbf{W}$;
    Compute $\lambda_i = \mathbf{W}_i \times \mathbf{v} \in \mathbb{Z}_q^n$;
    Pick a random noise vector $\mathbf{e}_i \leftarrow \chi^{2m}$;
    Compute $\mathbf{z}_i = \mathbf{A}_i^{\mathrm{T}}\lambda_i + D\mathbf{e}_i \in \mathbb{Z}_q^{2m}$;
  end for
  Generate $\mathbf{Z} = (\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_k) \in \mathbb{Z}_q^{2m\times k}$;
end for
return $C = (c, \mathbf{Z}, \mathbf{W})$;

where LCM denotes the lowest common multiple function. The above process is described in Algorithm 3.

—-Decryption. The **ABE-Dec** algorithm (described in Algorithm 4) is composed of policy decision-making and message retrieval. At last, the algorithm use the private key sk_S to retrieve the the decrypted message M. The correctness of this algorithm is presented in the supplementary file.

Algorithm 4 ABE-Dec(C, sk)

Input: the ciphertext C and the private key sk.
Output: the Message M.

if $S \vdash \mathbf{W}$ **then**
  Exist a candidate $\mathbf{W}^* \in \mathbb{Z}_q^{t\times t}$ to meet $S \vdash \mathbf{W}^*$;
  for each i in $[1, t]$ **do**
    Compute $w_i = \mathbf{z}_i^{\mathrm{T}}\mathbf{d}_i \in \mathbb{Z}_q$;
  end for
  $\mathbf{w} = (w_1, \cdots, w_t)^{\mathrm{T}} \in \mathbb{Z}_q^t$;
  Set $(\mathbf{W}^*)_1^{-1}$ as the 1-st row of the inverse of $\mathbf{W}^*$;
  Compute $c' = (\mathbf{W}^*)_1^{-1} \cdot \mathbf{w} \in \mathbb{Z}_q$;
  Compute $r = c - c' \in \mathbb{Z}$;
  Set $M = 1$;
  if r is closer to 0 than to $\lfloor \frac{q}{2} \rfloor$ **then**
    Set $M = 0$;
  end if
else
  return $\perp$;
end if
return M;

The core approach of the CP-ABE-L scheme is to multiply a sufficiently large constant $D = (T!)^2$ associated with policy matrix $\mathbf{W}$ into the noise vector, where T is the size of identity set of attributes $\{1, 2, \cdots, T\}$. Thus, the magnitude of noise vector is amplified D times. To ensure the correctness of decryption, the modulus q is conservatively estimated at $q \geq m^3 \log m \cdot 2^{5T}$ in Agrawal's scheme [21]. Let $T = O(\log m)$. It is easy to see $q \geq 1.88 \times 2^{123}$ for $m = 2^{15}$ at 112-bit security ($n = 112$). The parameters of the above CP-ABE-L scheme, constructed on the Shamir's secret sharing scheme, are much larger than those of general lattice cryptosystems. This inevitably brings the increase of computation and storage overheads, which negatively affects the practical use of ABE.

3 System Model

3.1 Potential Applications

In many application scenarios, CP-ABE-L provides a secure approach for sharing resources, especially for an open untrusted environment where the data is no longer under the resource owner's control. Today, this kind of open environment is becoming more and more popular, such as cloud computing, blockchain, and IoT, so CP-ABE-L plays an increasingly important role in protecting data privacy. For clarity, we will take secure cloud file sharing (CFS) services as an example to introduce how the CP-ABE-L scheme enforces an access control mechanism in cloud.

Although most remote CFS services, such as *Box*, *Dropbox*, *Microsoft*, and *Google*, claim to adopt encryption to protect data, there are always leaks or bugs in their software, which make user's private data vulnerable to hackers, let alone CFS providers themselves. To solve this problem, client-side encryption has been proposed because it is infeasible for CFS service provider to search or modify any document encrypted by client.

However, this brings a new problem: the traditional encryption systems are difficult to realize the sharing among generous users with different responsibilities. To solve it, we consider introducing Attribute-based Access Control (ABAC) and ABAC-friendly cryptosystem, e.g., CP-ABE, into CFS to express and implement vastly diverse access control policies for various types of shared data. For example, Al-Dahhan et al. [25] analyzed the major issues of CP-ABE in the IoT paradigm and discussed benefits, requirements, challenges, and weaknesses of outsourcing and sharing data in cloud. More importantly, ABAC-friendly cryptosystem must be able to resist quantum computing with lower storage overheads and computational complexity. Thus, we introduce CP-ABE-L into our system framework to ensure the security of CFS in the post-quantum era.

3.2 System Framework

A cryptography approach that integrates ABAC/XACML-L system into existing cloud storage service is proposed to achieve secure cloud file sharing, where XACML is a standard for the implementation of ABAC. The system framework contains the following three parts:

- **Enterprise-side ABAC/XACML model**, which performs a lightweight enterprise-side authorization management independent of the cloud according to the rules defined in policies;
- **Client-side CP-ABE scheme**, which enforces client-side encryption/decryption under ABAC/XACML policy;
- **Cloud-side CFS service**, which provides a black-box cloud-based file storage and sharing for tenants.

In this framework, as shown in Fig. 2, the CP-ABE scheme is placed on the client side to protect the tenant's privacy against potential untrusted servers, where the data is no longer under the resource owner's control. Moreover, authorization center based on ABAC/XACML can derive the CP-ABE scheme to automatically encrypt or decrypt the resources, so that it does not need too much manual intervention for these client-side operations. More specifically, the authorization center can provide access policies

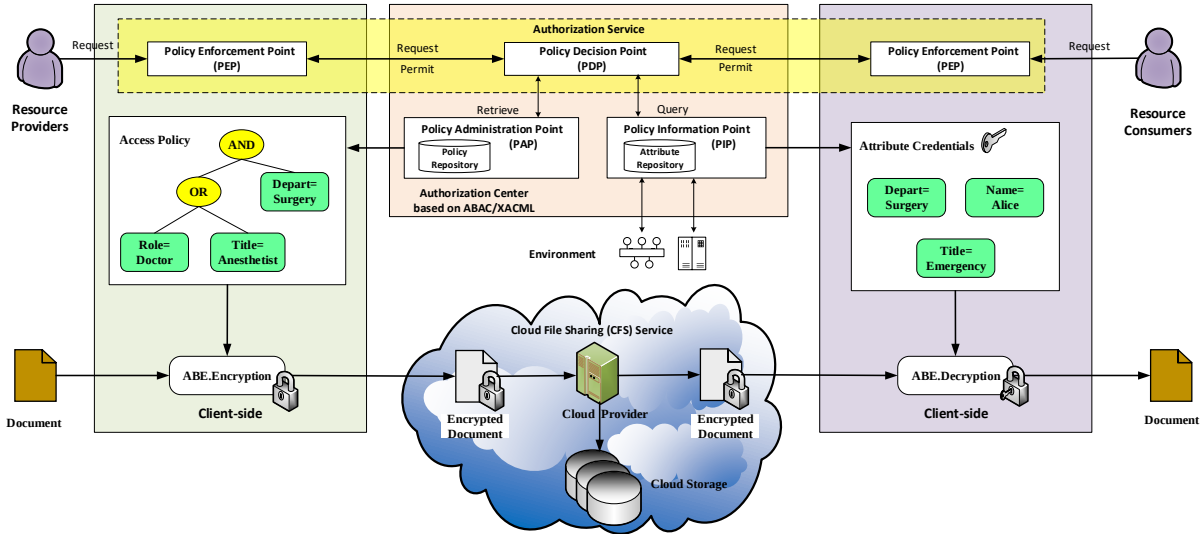


Fig. 2. The system framework of secure cloud file sharing based on the CP-ABE scheme and the ABAC/XACML model.

and attribute credentials for encryption and decryption in CP-ABE, respectively. The integration of CP-ABE, ABAC/X-ACML, and CFS, is conducive to policy-based data sharing within enterprise and organization.

3.3 ABAC/XACML Model

The ABAC/XACML's reference architecture (applied to *SIC-SACML, IBM TSPM, Sun XACML, XEngine, WSO2*, etc.) is shown in wheat and yellow box of the Fig.2. In this architecture, the functionality of *Policy Enforcement Point (PEP)* and *Policy Decision Point (PDP)* can be distributed or centralized, and they constitute so-called Authorization Service (AS). In our framework, the PEPs are deployed in a distributed way into client side in order to interpret the cloud tenants' access request. The PDP is more often used as an access point of authorization service for receiving requests from the PEPs. The other points, *Policy Information Point (PIP)* and *Policy Administration Point (PAP)*, are placed inside the enterprise to take responsibility for policy and attribute repository, respectively. For an access request, the workflow of this architecture is described as follows:

- For an access request (S, A, O, E) from an authenticated subject, the PEP interprets this request, expressed by the function $Q \leftarrow \mathbf{PEP}(S, A, O, E)$, and sends the request Q to the PDP. Here, (S, A, O, E) denotes that the subject S intents to enforce the operation A on the object O under the environment E.
- In terms of the require Q, the PDP requires the PAP to retrieve the corresponding access policy Π, expressed by the process $\Pi \leftarrow \mathbf{PDP}^{\mathrm{PAP}}(Q)$. Furthermore, the PDP queries the PIP to obtain the attribute set A for subject, object, and environment according to the policy Π, expressed by $\mathcal{A} \leftarrow \mathbf{PDP}^{\mathrm{PIP}}(Q, \Pi)$. At last, the PDP makes access decision, expressed by $\mathcal{A} \vdash \Pi$, according to Π and $\mathcal{A}$. Note that, $\mathbf{PDP}^{\mathrm{X}}(\cdot)$ denotes the PDP perform functions by querying the entity X.
- The final decision result (either permit or deny) given by the PDP is sent to the client-side PEP, and then the PEP fulfills the operation A of the access request according to the decision of PDP.

For a basic CFS system, assume that the user's access requests for the PEP just consist of two basic operations, Read and Write, on the sensitive files in cloud. As far as encrypted file is concerned, our solution must provide a transparent way to implement *encryption (when writing) and decryption (when reading)* corresponding to the "write" and "read" requests, which is the focus of this paper.

3.4 System Workflow

This paper briefly introduces how to integrate CP-ABE scheme and ABAC/XACML model to complete encrypted file sharing. Assume that the ABAC/XACML administrator establishes CP-ABE by invoking $\mathbf{Setup}()$ in the initial stage, so that any user in the system can share the cryptosystem and unified public key. Moreover, any user in the system can own his privacy key with attribute credentials generated by the function $\mathbf{Extract}()$ from the authorization center.

Without loss of generality, assume that the CFS involves only two basic file operations, Read and Write, as mentioned before. The specific processes of algorithms 5 and 6 are described as follows.

3.4.1 *Encrypted-file writing process*

When PEP interprets an encrypted-file write request from a resource provider, it completes the conversion of the request in an ABAC/XACML format, i.e., $Q \leftarrow \mathbf{PEP}(S, \text{“}W\text{”}, O, E)$, and then sends it to the PDP in the authorization center. Next, the PDP follows normal ABAC/XACML routine to retrieve access policy from the PAP, i.e., $(\Pi_S, \Pi_O) \leftarrow \mathbf{PDP}^{\mathrm{PAP}}(Q)$, where the policy consists of the subject's Π_S and object's Π_O. The former is used to define the permission of resource provider's writing action, and the latter is to define the rules on the written file, which will be embedded into the encrypted file. For Π_S, the PDP continues to obtain the set of attributes of subject, object, and environment from the PIP, i.e., $\mathcal{A} \leftarrow \mathbf{PDP}^{\mathrm{PIP}}(Q, \Pi_S)$. And then, the PDP makes access decision according to Π_S and $\mathcal{A}$. If the result is true, the PDP will permit the PEP to perform the write operation; otherwise, the operation is denied.

Algorithm 5 WriteEncryptedFile(S, O, E)

Input: a subject S, an object O, and the environment E.
Output: True or False.

Generate a request $Q \leftarrow$ **PEP**$(S, \text{“}W\text{”}, O, E)$;
Retrieve access policy $(\Pi_S, \Pi_O) \leftarrow$ **PDP**$^{\text{PAP}}(Q)$;
Query the current attribute set $\mathcal{A} \leftarrow$ **PDP**$^{\text{PIP}}(Q, \Pi_S)$;
if $\mathcal{A} \vdash \Pi_S$ **then**
    Compute $\mathbf{W} \leftarrow$ **SPMGen**$(\Pi_O, \cdot)$ invoked by PEP;
    Generate a random session key ek;
    Compute $C_{ek} \leftarrow$ **ABE-Enc**$(pk, \mathbf{W}, ek)$ invoked by PEP;
    Obtain the object's file $O.file$;
    Compute $C_f \leftarrow$ **SymEnc**$(ek, O.file)$ invoked by PEP;
    Store (C_{ek}, C_f) into the cloud;
    return True;
else
    return False;
end if

After receiving the permission, the client-side PEP converts the object's policy Π_O into a policy matrix $\mathbf{W}$, i.e., $\mathbf{W} \leftarrow$ **SPMGen**$(\Pi_O, \cdot)$, see Section 4 for details. After obtaining the public key pk reserved in the authorization center, the PEP invokes the algorithm **ABE-Enc** to encrypt a random session key ek according to the policy matrix $\mathbf{W}$, i.e., $C_{ek} =$ **ABE-Enc**$(pk, \mathbf{W}, ek)$. After that, for a given object's file $O.file$, the PEP continues to make use of ek to encrypt the file by using a symmetric encryption algorithm, i.e., $C_f =$ **SymEnc**$(ek, O.file)$. For example of Fig. 2, the access policy for specifying a medical document is **“**$((Role =$ **Doctor**$)$ **OR** $(Title =$ **Anesthetist**$)$**AND** $(Depart. =$ **Surgery**$))$**”** built on Boolean logic. Next, the PEP can require the cloud provider to store the encrypted file (C_{ek}, C_f) because the request is granted by PDP as mentioned above.

3.4.2 Encrypted-file reading process

As shown in Algorithm 6, the encrypted-file reading process is essentially the same as that of writing process. At first, resource consumer submits an access request on “read file” to the PEP, then the PEP interprets the request (i.e., $Q \leftarrow$ **PEP**$(S, \text{“}R\text{”}, O, E)$) and sends Q to the PDP. Next, the PDP makes access decision according to the policy derived from PAP (i.e., $\Pi_O \leftarrow$ **PDP**$^{\text{PAP}}(Q)$) and the attribute credentials associated with the policy Π_O, obtained by querying the PIP, i.e., $\mathcal{A} \leftarrow$ **PDP**$^{\text{PIP}}(Q, \Pi_O)$.

Algorithm 6 ReadEncryptedFile(S, O, E)

Input: a subject S, an object O, and the environment E.
Output: True or False.

Generate a request $Q \leftarrow$ **PEP**$(S, \text{“}R\text{”}, O, E)$;
Retrieve access policy $\Pi_O \leftarrow$ **PDP**$^{\text{PAP}}(Q)$;
Query the current attribute set $\mathcal{A} \leftarrow$ **PDP**$^{\text{PIP}}(Q, \Pi_O)$;
if $\mathcal{A} \vdash \Pi_O$ **then**
    Obtain the user's private key sk_S to PEP;
    Compute $ek \leftarrow$ **ABE-Dec**(sk_S, C_{ek}) invoked by PEP;
    Obtain the encrypted file C_f from the cloud;
    Compute $file \leftarrow$ **SymDec**(ek, C_f) invoked by PEP;
    Send $file$ to the user;
    return True;
else
    return False;
end if

If the decision is permitted (i.e., $\mathcal{A} \vdash \Pi_O$), the PDP send the permission to the PEP. The client-side PEP invokes the algorithm **ABE-Dec** to decrypt the ciphertext C_{ek} with the help of consumer's private key sk_S, i.e., $ek \leftarrow$ **ABE-Dec**(sk_S, C_{ek}). In the above example, an user assigns the private key with the following attributes: **“**$(Depart. =$ **Surgery**$), (Name =$ **Alice**$), (Title =$ **Emergency**$)$**”**. Clearly, in this case, the CP-ABE decryption will fail due to the failure of matching the above policy. This does not usually happen if the PEP's decision result on $\mathcal{A} \vdash \Pi_O$ is true, but may happen when ABAC/XACML is abnormally executed or the attacker forges the PDP's permission to the PEP. Thus, it is an important security mechanism for cryptograhphic policy matching on ABE to prevent unauthorized access besides ABAC/XACML.

Finally, after downloading encrypted file from cloud platform, the PEP takes the session key ek as a certificate to decrypt the file by invoking the symmetric decryption algorithm, i.e., $file \leftarrow$ **SymDec**(ek, C_f).

Through the explanation of system workflow, it is not difficult to see that the CP-ABE scheme is able to provide more complete control over managing and protecting high-value or sensitive data among the resource providers, the consumers, and the cloud service provider. However, as mentioned in the related work of Section 1, there are still two practical problems to be improved in the candidate CP-ABE-L schemes [14], [16] against quantum attacks. One problem is the lack of an optimization approach to convert access policy into policy matrix, which leads to excessive error terms in CP-ABE-L. Another problem is the lack of a parameter optimization approach to reduce the storage and computation overloads of CP-ABE-L. These two problems will be analyzed and improved in the subsequent sections.

4 Construction of Small Policy Matrix

The most innovative aspect of our work is to design an optimization approach of access policies for lattice-based cryptosystem. For a given policy Π, our approach is to construct a corresponding policy matrix $\mathbf{W}$ that is an approximate “sparse matrix” containing only $-1, 0$ and 1. The benefit is that its determinant could be either 1 or -1. This further ensures that the inverse of the policy matrix, $\mathbf{W}^{-1}$, contains only small integer coefficients. Such a policy matrix $\mathbf{W}$ is called Small Policy Matrix (SPM), which is defined formally as follows.

Definition 2 (Small Policy Matrix, SPM) *For an integer q, a matrix $\mathbf{W}$ is called Small Policy Matrix (SPM) in $\mathbb{Z}_q$ if*

1) *an access policy Π can be converted into a policy matrix $\mathbf{W} \in \mathbb{Z}_q^{k \times l}$ by using Algorithm **SPMGen**$(\cdot)$;*
2) *the matrix $\mathbf{W}$ consists of elements in $\{-1, 0, 1\}$;*
3) *there exists a candidate submatrix $\mathbf{W}^* \subseteq \mathbf{W}$, where the determinant of its inverse satisfies $|(\mathbf{W}^*)^{-1}| = \pm 1$, and the 1-st row elements of its inverse $(\mathbf{W}^*)^{-1}$ consist of all ones, i.e., $(\mathbf{W}^*)_1^{-1} = (1, 1, \cdots, 1)$.*

Based on this idea, we present a practical optimization algorithm **SPMGen**$(\cdot)$ to convert access policy Π into small policy matrix in this section. More importantly, the generated SPM is optimal because the coefficients of $(\mathbf{W}^{-1})_1$ are all ONEs. Such a property of SPM can satisfy the requirement of minimizing the range of error amplification in the

lattice-based cryptosystem. In the last part, we provide the improved CP-ABE-L scheme by adopting the SPM to one given in Section 2.4.

4.1 Small Policy Matrix Generation

We now provide an effective method to convert attribute-oriented access policy into a cryptographic form that is used in the CP-ABE-L scheme. Based on the previous research results [13], such an access policy can be converted into a share-generating matrix. For clarity, this matrix is called *policy matrix*, which plays a key role in our improved scheme since it is used to encrypt a message in a direct way.

For a policy expressed by Boolean logic, a set of attribute predicates in it can be set as $\{P_1, P_2, \cdots, P_n\}$. Each predicate contains an attribute name and a constant, e.g., "$Depart. =$ Surgery", which means access is permitted if a user's department is surgery and then outputs True. The different attribute predicates may correspond to the same attribute, e.g., two predicates, "$Depart. =$ Surgery" and "$Depart. =$ Pediatrics", are used to restrict the scope of the authorized departments.

Adapted from [13], a policy matrix can be realized by a Linear Secret Sharing Scheme (LSSS). In this case, the secret sharing refers to methods for distributing a secret amongst a group of predicates in policy, each of which is allocated one share of the secret. A secret sharing scheme derived from policy Π over $\{P_1, \cdots, P_l\}$ consists of two aspects, share generation (Definition 3) and secret reconstruction (Definition 4). The process of share generation is described as follows.

Definition 3 (Share Generation in LSSS, [13]) *For the secret s that will be shared over $\mathbb{Z}_q$, the access policy Π can be converted to a $k \times n$ share-generating matrix $\mathbf{W}$, the i-th row of which is labeled by a predicate P_i. And the share λ_i which belongs to predicate P_i will form a vector λ over $\mathbb{Z}_q$. Randomly choose $r_1, \cdots, r_{n-1} \in \mathbb{Z}_q$, the vector λ is the k shares of s by using $\lambda = \mathbf{W}\mathbf{v}$, where $\mathbf{v} = (s, r_1, \cdots, r_{n-1})^{\mathrm{T}}$.*

In this definition, the core process of share generation is to convert the access policy Π to a share-generating matrix $\mathbf{W}$. This conversion are divided into two phases: the conversion from access policy into policy tree, and then into policy matrix. We will describe these two phases as follows.

1) Convert access policy Π into policy tree.

At first, we assume that access policy Π is expressed by a monotone Boolean formula with "AND ($\wedge$)" and "OR ($\vee$)" gates over some attribute predicates, which are named by P_i from left to right. For example, define a policy Π as "the department may be Surgery or Dental", i.e., $\Pi = P_1 \vee P_2$, where P_1 means "$Depart. =$ Surgery", and P_2 means "$Depart. =$ Dental".

It is easy to convert the policy into a policy binary tree in which each internal node corresponds to a logic operator and each leaf node corresponds to a predicate. As an example, let us give an illustration in the left of Fig. 3. Here, the policy is defined as $\Pi = [P_1 \vee (P_2 \wedge P_3)] \wedge (P_4 \vee P_5 \wedge P_6)$, and the leaf nodes are simple predicates, which are recorded as $P_1, P_2, \cdots, P_6$. Note that, considering that the binary tree produced from the policy is not unique, the policy matrix generated from it is also not unique. This step will be used in the initial stage (see step 2) of Algorithm 7.

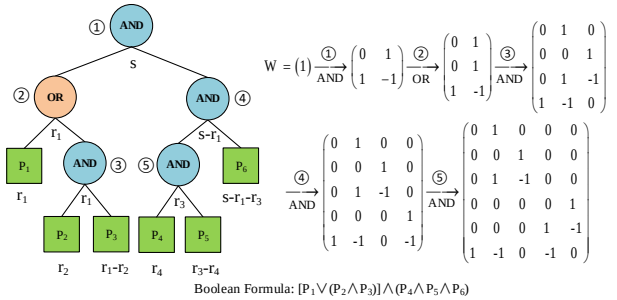


Fig. 3. The policy tree and evolution process of its policy matrix.

2) Convert policy tree into $\mathbf{W}$ by using Definition 3.

According to Definition 3, we design the **SPMGen**$(\cdot)$, described in Algorithm 7, to realize the convert from policy tree into policy matrix. The algorithm **SPMGen**$(\cdot)$ is a recursive algorithm with an access policy Π, a current node pointer p in the policy binary tree, the addresses of three variables, $\mathbf{W}$, k and l, where k denotes the row corresponding to the current node, and l denotes the number of columns in the matrix.

Given a target policy Π, the initial invocation is set to **SPMGen**$(\Pi, \mathrm{NULL}, \mathbf{W}, k, l)$, where the initial node pointer p is NULL. According to step 1 in this algorithm, the condition $p ==$ NULL ensures that three input variables, $\mathbf{W}$, k and l, are initialized as $\mathbf{W} = (1)$, $k = 1$, and $l = 1$, in terms of step 2-4. In addition, let the secret s be the value of the root node, as shown in Fig. 3, so that $\mathbf{v} = (s)$.

Next, we turn our attention to the process of internal nodes in policy tree. The procedure of this process is executed by step 7-20 in this algorithm. Starting with the above initial state, we take an simple example to illustrate this procedure, which consists of two cases:

- For an AND gate with s, a random value r_1 and $s - r_1$ are assigned to its left and right child node, respectively. Then, according to step 8-10, we convert the initial matrix $\mathbf{W} = (1)$ into $\begin{pmatrix} 0 & 1 \\ 1 & -1 \end{pmatrix}$ and $\mathbf{v}' = (s, r_1)^{\mathrm{T}}$, such that $\lambda' = (r_1, s - r_1)^{\mathrm{T}}$. Next, we make recursive calls to the left child of the current node via **SPMGen**$(\Pi, p \to left, \mathbf{W}, 1, 2)$, where $k = 1$ denotes the left child corresponding to the 1-th row. Similarly, the right child is processed via **SPMGen**$(\Pi, p \to right, \mathbf{W}, 2, 2)$, where $k = 2$ denotes the right child corresponding to the 2-th row. Meanwhile, the number of rows and columns in $\mathbf{W}$ is increased by 1, respectively.
- For an OR gate with s, the same value s is assigned to all of its child nodes. Then, according to step 15-16, we convert the initial matrix $\mathbf{W} = (1)$ into $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$ and $\mathbf{v}' = (s)$, such that $\lambda' = (s, s)^{\mathrm{T}}$. Next, we make recursive calls to the left child of the current node via **SPMGen**$(\Pi, p \to left, \mathbf{W}, 1, 1)$, where $k = 1$ denotes the left child corresponding to the 1-th row. Similarly, the right child is processed via **SPMGen**$(\Pi, p \to right, \mathbf{W}, 2, 1)$, where $k = 2$ denotes the right child corresponding to the 2-th row. Here, the number of rows in $\mathbf{W}$ is increased by 1, but the columns remain unchanged.

Algorithm 7 SPMGen$(\Pi, p, \&\mathbf{W}, \&k, \&l)$

Input: an access policy Π, the current node pointer p with the initial value NULL, the address of policy matrix $\&\mathbf{W}$, the address of the current row $\&k$, and the address of the current column $\&l$.
Output: the policy matrix $\mathbf{W}$.

if $p ==$ NULL **then**
  Convert Π to a binary tree, where the root pointer is p;
  Generate initial policy matrix $\mathbf{W} = (1)$;
  Set $k \leftarrow 1$ and $l \leftarrow 1$;
end if
if the node pointed by p is not a leaf node **then**
  if $p.op ==$ Operator.AND **then**
    Append an all zero vector $\mathbf{W}_k$ before the k-row;
    Append an all zero vector $\mathbf{W}_{l+1}^{\mathrm{T}}$ after the l-column;
    Set $l = l + 1$, $\mathbf{W}_{kl} = 1$, and $\mathbf{W}_{(k+1)l} = -1$;
    Invoke $\mathbf{W} \leftarrow$ **SPMGen**$(\Pi, p \rightarrow left, \mathbf{W}, k, l)$;
    Set $k = k + 1$;
    Invoke $\mathbf{W} \leftarrow$ **SPMGen**$(\Pi, p \rightarrow right, \mathbf{W}, k, l)$;
  else
    Append an all zero vector $\mathbf{W}_{k+1}$ after the k-row;
    Copy k-th vector $\mathbf{W}_k$ into $(k+1)$-th vector $\mathbf{W}_{k+1}$;
    Invoke $\mathbf{W} \leftarrow$ **SPMGen**$(\Pi, p \rightarrow left, \mathbf{W}, k, l)$;
    Set $k = k + 1$;
    Invoke $\mathbf{W} \leftarrow$ **SPMGen**$(\Pi, p \rightarrow right, \mathbf{W}, k, l)$;
  end if
end if
return $\mathbf{W}$;

The **SPMGen**$(\cdot)$ has a recursive procedure to traverse a tree using pre-order traversal, and repeat the above process until all nodes in the tree are filled with shares. For example, the evolution process of $\mathbf{W}$ is shown in the right of Fig. 3.

At last, the **SPMGen**$(\cdot)$ outputs a 6×5 final policy matrix

$$\mathbf{W} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & -1 \\ 1 & -1 & 0 & -1 & 0 \end{pmatrix},$$

where the number of rows and columns is equal to the number of leaf nodes and the dimension of $\mathbf{v}$, respectively. More importantly, the final matrix has the following features: 1) it is an approximate "sparse matrix" containing only -1, 0, and 1; 2) its determinant is either 1 or -1 (Theorem 3).

According to $\lambda_i = \mathbf{W}_i\mathbf{v}$ in Definition 3, the share of leaf node can be computed as follows. Assume the final secret vector $\mathbf{v}$ be $(s, r_1, r_2, r_3, r_4)^{\mathrm{T}}$, where r_1, r_2, r_3, r_4 are randomly chosen. For $i = 1, \cdots, 6$, the value placed on the leaf node P_i represents its share λ_i, i.e., $\lambda_1 = r_1, \lambda_2 = r_2, \lambda_3 = r_1 - r_2, \lambda_4 = r_4, \lambda_5 = r_3 - r_4$, and $\lambda_6 = s - r_1 - r_3$.

Our **SPMGen**$(\cdot)$ algorithm can express the general (t, N)-threshold policy by means of the conversion between policy matrices. There exist some researches on the conversion methods, e.g., the paper [9] illustrates the different forms for describing access policies in detail, and points out that any monotone access structure on Boolean formula can be converted to AND-OR-gate access tree. Exactly, the given (t, N)-threshold policy is first converted to a minimal form access structure, then to a monotone Boolean formula by using equivalent Disjunctive Normal Form (DNF), and finally to an AND-OR-gate access tree. In this way, the **SPMGen**$(\cdot)$ algorithm can deal with the AND-OR-gate access tree derived from the (t, N)-threshold policy. Note that an equivalent AND-OR-gate access tree will have more leaf nodes than the original threshold-gate access tree. This will lead to larger size of policy matrix, and then affects performance negatively.

4.2 Small Policy Matrix Reconstruction

For a policy matrix $\mathbf{W}$ generated by **SPMGen**$(\cdot)$, we can employ it to design an encryption algorithm associated with the share generation in LSSS. Furthermore, the decryption algorithm can be designed on the secret reconstruction in LSSS, which is described as follows.

Definition 4 (Secret Reconstruction in LSSS [13]) *For any authorized set U, define $I \subset \{1, 2, \cdots, k\}$ be $I = \{i : P_i \in U\}$. There exist constants $\{\omega_i \in \mathbb{Z}_q\}_{i\in I}$ such that, if $\{\lambda_i\}$ are any valid shares of a secret s according to share generation in LSSS, then s is retrieved by $\sum_{i\in I}\omega_i\lambda_i = s$. Also, $\{\omega_i\}$ can be found in polynomial time in the size of the share-generating matrix $\mathbf{W}$. Furthermore, for any unauthorized set, no such constants $\{\omega_i\}$ exists, i.e., s should be information theoretically hidden.*

According to Definition 4, we prove that our proposed Algorithm **SPMGen**$(\cdot)$ has a remarkable feature: **the reconstruction vector is the vector of all ones** from Theorem 3.

In the following description, we will still consider the policy tree mentioned in Fig. 3. We first define $U = \{P_2, P_3, P_4, P_5, P_6\}$ as an authorized set of this example. Therefore, we have $I = \{i : P_i \in U\} = \{2, 3, 4, 5, 6\}$, and select these rows $\mathbf{W}_2, \mathbf{W}_3, \mathbf{W}_4, \mathbf{W}_5$, and $\mathbf{W}_6$ in $\mathbf{W}$ corresponding to P_2, P_3, P_4, P_5, and P_6 as the reconstruction matrix. And then, we observe whether there is a reconstruction vector $\omega = (\omega_2, \omega_3, \omega_4, \omega_5, \omega_6)$ to satisfy

$$\begin{aligned} & \omega_2\mathbf{W}_2 + \omega_3\mathbf{W}_3 + \omega_4\mathbf{W}_4 + \omega_5\mathbf{W}_5 + \omega_6\mathbf{W}_6 \\ = \ & (\omega_2, \omega_3, \omega_4, \omega_5, \omega_6)\begin{pmatrix} \mathbf{W}_2 \\ \mathbf{W}_3 \\ \mathbf{W}_4 \\ \mathbf{W}_5 \\ \mathbf{W}_6 \end{pmatrix} \\ = \ & (1, 0, 0, 0, 0). \end{aligned} \tag{4}$$

According to the above equation, the values, $\omega_2, \omega_3, \omega_4, \omega_5$ and ω_6, can be calculated in polynomial time, i.e.,

$$\begin{aligned} (\omega_2, \omega_3, \omega_4, \omega_5, \omega_6) &= (1, 0, 0, 0, 0)\begin{pmatrix} \mathbf{W}_2 \\ \mathbf{W}_3 \\ \mathbf{W}_4 \\ \mathbf{W}_5 \\ \mathbf{W}_6 \end{pmatrix}^{-1} \\ &= (1, 0, 0, 0, 0)\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} \\ &= (1, 1, 1, 1, 1). \end{aligned} \tag{5}$$

Thus, we deduce $\omega_2 = \omega_3 = \omega_4 = \omega_5 = \omega_6 = 1$, i.e., ω is the vector of all ones. Therefore, for the valid shares $\lambda_2, \lambda_3, \lambda_4, \lambda_5$, and λ_6 derived from P_2, P_3, P_4, P_5, and P_6

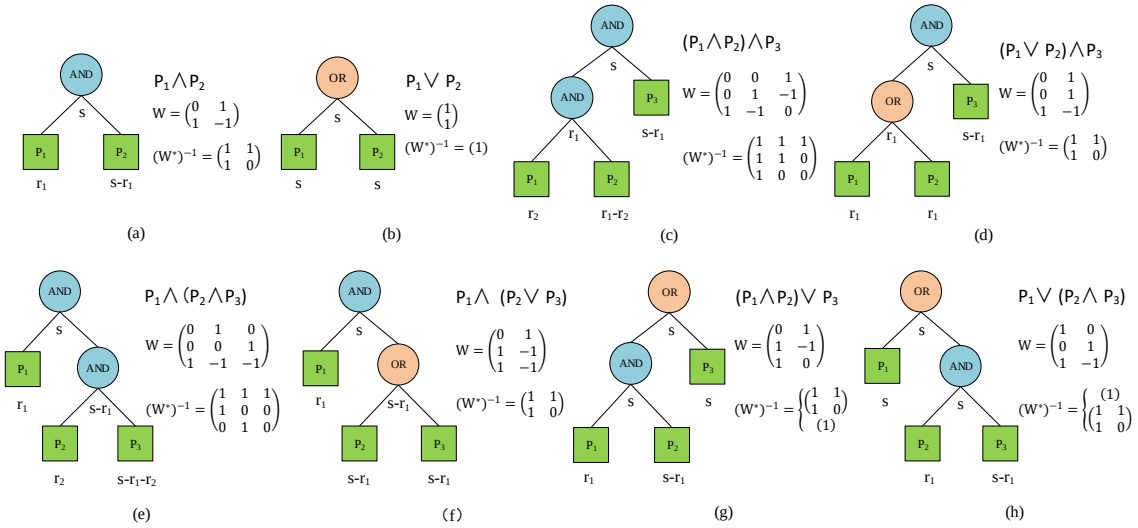


Fig. 4. Examples of access policy trees, policy matrices, and their inverses with two or three leaf nodes.

respectively, the secret s can be reconstructed according to $\lambda_i = \mathbf{W}_i\mathbf{v}$ and Equation (4) as

$$\begin{aligned}\sum_{i\in I}\omega_i\lambda_i &= (\omega_2,\omega_3,\omega_4,\omega_5,\omega_6)(\lambda_2,\lambda_3,\lambda_4,\lambda_5,\lambda_6)^{\mathrm{T}}\\ &= (\omega_2,\omega_3,\omega_4,\omega_5,\omega_6)\begin{pmatrix}\mathbf{W}_2\\ \mathbf{W}_3\\ \mathbf{W}_4\\ \mathbf{W}_5\\ \mathbf{W}_6\end{pmatrix}\begin{pmatrix}s\\ r_1\\ r_2\\ r_3\\ r_4\end{pmatrix} \quad (6)\\ &= (1,0,0,0,0)(s,r_1,r_2,r_3)^{\mathrm{T}} = s.\end{aligned}$$

Thus, we can use the reconstructed secret s to design the decryption algorithm. The key to achieving effective reconstruction is to calculate the inverse of the reconstruction matrix. For instance, in the above process, we gain the reconstruction matrix

$$\mathbf{W}^* = \begin{pmatrix}0 & 0 & 1 & 0 & 0\\ 0 & 1 & -1 & 0 & 0\\ 0 & 0 & 0 & 0 & 1\\ 0 & 0 & 0 & 1 & -1\\ 1 & -1 & 0 & -1 & 0\end{pmatrix}, \quad (7)$$

which is a candidate submatrix of $\mathbf{W}$, and its inverse is

$$(\mathbf{W}^*)^{-1} = \begin{pmatrix}1 & 1 & 1 & 1 & 1\\ 1 & 1 & 0 & 0 & 0\\ 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 1 & 0\\ 0 & 0 & 1 & 0 & 0\end{pmatrix}. \quad (8)$$

We find that it is a special matrix consisting of only 0 and 1.

In addition, in order to further study the above conclusions, we have listed the access policy trees with almost two or three leaf nodes in Fig. 4. In the figure, subfigures (a) and (b) describe the case of two leaf nodes, and subfigures (c)-(h) represent the case of three leaf nodes. Note that there is also a case of three leaf nodes, in which the root and internal nodes are OR gates, but this is omitted in Fig. 4 since only the same row vectors are added to the policy matrix.

For all cases, we use the **SPMGen**$(\cdot)$ to convert the policy tree into $\mathbf{W}$. Then, the inverse $(\mathbf{W}^*)^{-1}$ of their candidate submatrices $\mathbf{W}^* \subseteq \mathbf{W}$ are listed in Fig. 4, respectively. We observe that *all candidate submatrices have such a feature that the 1-st row elements of their inverses are all ones*.

Theorem 3 *The access policies, expressed as a monotone Boolean formula over logic AND or OR, can be converted into a policy tree. Using the* ***SPMGen****$(\cdot)$ algorithm described above, this tree can be further converted into a small policy matrix $\mathbf{W}$, where the determinant of the inverse of a candidate submatrix $\mathbf{W}^* \subseteq \mathbf{W}$ satisfies $|(\mathbf{W}^*)^{-1}| = \pm 1$, and the 1-st row elements of $(\mathbf{W}^*)^{-1}$ consist of all ones, i.e., $(\mathbf{W}^*)_1^{-1} = (1,1,\cdots,1)$.*

The proof of the theorem is presented in the supplement file.

4.3 Improved CP-ABE-L Scheme with SPM

We now turn our attention to the CP-ABE-L scheme in Section 2.4. This scheme always introduces Shamir's k-out-of-l secret sharing scheme to establish the access policy matrix by employing *Vandermonde matrix* on the identity set of attributes $\{I_1,\cdots,I_T\}$ in $\mathbb{Z}$, where T indicates the maximum number of attributes. In decryption, Shamir's scheme uses a linear combination of shares to reconstruct the secret. By using the fractional Lagrangian coefficients $L_i = \prod_{j=1,i\neq j}^{T}\frac{-I_j}{I_j-I_i} \mod q$, the cumulative error term is bounded to $x' - \sum_{i\in S} L_i e_i^{\mathrm{T}} x_i < q/k$ in the process of secret reconstruction, where k is a constant integer and S indicates the subset of shares.

However, the value of L_i may be arbitrarily large because it is an element in $\mathbb{Z}_q$, even if both the numerator and denominator in L_i are restricted to integer fractions. To eliminate the denominator, Agrawal et. al. ([21], Lemma 3) proposed a common idea, which is to multiply the noise vector by a sufficiently large constant $D = (T!)^2$ to hold

$$Dx' - \sum\nolimits_{i\in S} DL_i e_i^{\mathrm{T}} x_i < q/k$$

with overwhelming probability, where $DL_i \in \mathbb{Z}$ for $\forall i \in S$ and $|DL_i| \leq D^2 \leq (T!)^4$ for $\forall I_i \in [1,T]$. But the magnitude of the error term $e_i^{\mathrm{T}} x_i$ is amplified by $(T!)^4 \leq 2^{4T\log T}$ times. In Agrawal's Fuzzy IBE/ABE scheme [21], the modulus q is conservatively estimated at $q \geq m^3 \log m \cdot 2^{5T}$. Let $T = O(\log m)$, it is easy to have $q \geq 1.88 \times 2^{123}$ for $m = 2^{15}$ at 112-bit security ($n = 112$).

In light of complex access policies, the parameters of ABE scheme become much larger than those of general lattice cryptosystems. This brings about an increase

in computation and storage overheads. In fact, the constant D can be computed in terms of $\mathbf{W}$, i.e., $D = \text{LCM}_{\forall \mathbf{W}^* \subseteq \mathbf{W}}(|(\mathbf{W}^*)^{-1}|) \in \mathbb{Z}$. Fortunately, we have already got the conclusion $|(\mathbf{W}^*)^{-1}| = \pm 1$ for $\forall \mathbf{W}^* \subseteq \mathbf{W}$ according to Theorem 3. Thus, the SPM has the ability to reduce the constant D to ± 1, i.e., $D = \pm 1$. This means that the SPM is the optimal expression of policies in ABE.

According to the above analysis, we introduce the SPM into the CP-ABE-L scheme, which can be optimized to a great extent. To avoid repetition, we only describe the revised parts about encryption and decryption algorithms in the CP-ABE-L scheme of Section 2.4, as follows.

Encryption. The encryption process remains the same as the given CP-ABE-L scheme except for the step 3). Exactly, the step 3) is revised as follows: pick $e \leftarrow \chi$ and k noise vectors $\mathbf{e}_i \leftarrow \chi^{2m}$ for $i \in [1, k]$, and compute

$$\begin{cases} c &= \mathbf{u}^{\mathrm{T}}\mathbf{s} + e + M\lfloor \frac{q}{2} \rfloor \in \mathbb{Z}_q, \\ \mathbf{z}_i &= \mathbf{A}_i^{\mathrm{T}}\lambda_i + \mathbf{e}_i \in \mathbb{Z}_q^{2m},\ i = 1, \ldots, k, \end{cases} \tag{9}$$

where $\mathbf{Z} = (\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_k) \in \mathbb{Z}_q^{2m \times k}$. Finally, it returns the ciphertext $C = (c, \mathbf{Z}, \mathbf{W})$.

Decryption. The decryption process also remains the same as the given CP-ABE-L scheme except for the step 2). The revised part in the step 2) is as follows: compute

$$c' = (\mathbf{W}^*)_1^{-1} \cdot \mathbf{w} = \sum\nolimits_{i=1}^{t} w_i \in \mathbb{Z}_q \tag{10}$$

for $(\mathbf{W}^*)_1^{-1} = (1, 1, \cdots, 1)$.

Considering that different leaf nodes may correspond to the same attributes, the size of policy matrix is nonlinear with the number of attributes in the policy. In addition, the size of ciphertext in CP-ABE-L is directly linear to the number of rows in policy matrix, and the latter is equal to the number of leaf nodes in policy tree. In the view of the improved scheme, the ciphertext $\mathbf{Z}$ is a k-dimensional vector, $\mathbf{W}$ is a $k \times l$ matrix and the corresponding policy tree has k leaf nodes. This means that the larger policy matrix, the larger the ciphertext of CP-ABE-L. However, the policy matrix does not affect the number of private keys.

Then, we show our improved CP-ABE-L scheme meets the correctness. We do the following steps if $S \vdash \mathbf{W}^*$.

- Invokes the **SampleLeft** algorithm, we obtain $\mathbf{A}_i\mathbf{d}_i = \mathbf{u}$. Then, we compute $w_i = \mathbf{z}_i^{\mathrm{T}}\mathbf{d}_i = (\lambda_i^{\mathrm{T}}\mathbf{A}_i + \mathbf{e}_i^{\mathrm{T}})\mathbf{d}_i = \lambda_i^{\mathrm{T}}\mathbf{u} + \mathbf{e}_i^{\mathrm{T}}\mathbf{d}_i$ for $i = 1, \cdots, t$. Therefore, the vector $\mathbf{w}$ can be denoted as

$$\begin{aligned} \mathbf{w} &= (\mathbf{z}_1^{\mathrm{T}}\mathbf{d}_1, \cdots, \mathbf{z}_t^{\mathrm{T}}\mathbf{d}_t)^{\mathrm{T}} = \begin{pmatrix} \lambda_1^{\mathrm{T}}\mathbf{u} + \mathbf{e}_1^{\mathrm{T}}\mathbf{d}_1 \\ \lambda_2^{\mathrm{T}}\mathbf{u} + \mathbf{e}_2^{\mathrm{T}}\mathbf{d}_2 \\ \cdots \\ \lambda_t^{\mathrm{T}}\mathbf{u} + \mathbf{e}_t^{\mathrm{T}}\mathbf{d}_t \end{pmatrix} \\ &= \begin{pmatrix} \lambda_1^{\mathrm{T}} \\ \lambda_2^{\mathrm{T}} \\ \cdots \\ \lambda_t^{\mathrm{T}} \end{pmatrix}\mathbf{u} + \begin{pmatrix} \mathbf{e}_1^{\mathrm{T}}\mathbf{d}_1 \\ \mathbf{e}_2^{\mathrm{T}}\mathbf{d}_2 \\ \cdots \\ \mathbf{e}_t^{\mathrm{T}}\mathbf{d}_t \end{pmatrix} = \lambda\mathbf{u} + \begin{pmatrix} \mathbf{e}_1^{\mathrm{T}}\mathbf{d}_1 \\ \mathbf{e}_2^{\mathrm{T}}\mathbf{d}_2 \\ \cdots \\ \mathbf{e}_t^{\mathrm{T}}\mathbf{d}_t \end{pmatrix}. \end{aligned} \tag{11}$$

- By using the secret reconstruction in LSSS, we have $\mathbf{v} = (\mathbf{W}^*)^{-1}\lambda = (\mathbf{s}, \mathbf{r}_2, \cdots, \mathbf{r}_t)^{\mathrm{T}}$, so we can retrieve the first element $\mathbf{s}^{\mathrm{T}} = (\mathbf{W}^*)_1^{-1}\lambda$. Then, we compute

$$\begin{aligned} c' &= (\mathbf{W}^*)_1^{-1}\mathbf{w} = (\mathbf{W}^*)_1^{-1}\lambda\mathbf{u} + (\mathbf{W}^*)_1^{-1}\begin{pmatrix} \mathbf{e}_1^{\mathrm{T}}\mathbf{d}_1 \\ \mathbf{e}_2^{\mathrm{T}}\mathbf{d}_2 \\ \cdots \\ \mathbf{e}_t^{\mathrm{T}}\mathbf{d}_t \end{pmatrix} \\ &= (\mathbf{W}^*)_1^{-1}\mathbf{W}\begin{pmatrix} \mathbf{s}^{\mathrm{T}} \\ \mathbf{r}_2^{\mathrm{T}} \\ \cdots \\ \mathbf{r}_n^{\mathrm{T}} \end{pmatrix}\mathbf{u} + (\mathbf{W}^*)_1^{-1}\begin{pmatrix} \mathbf{d}_1^{\mathrm{T}}\mathbf{e}_1 \\ \mathbf{d}_2^{\mathrm{T}}\mathbf{e}_2 \\ \cdots \\ \mathbf{d}_t^{\mathrm{T}}\mathbf{e}_t \end{pmatrix} \\ &= (1, 0, \cdots, 0)\begin{pmatrix} \mathbf{s}^{\mathrm{T}} \\ \mathbf{r}_2^{\mathrm{T}} \\ \cdots \\ \mathbf{r}_n^{\mathrm{T}} \end{pmatrix}\mathbf{u} + \sum_{i=1}^{t}(\mathbf{W}^*)_{1i}^{-1}(\mathbf{d}_i^{\mathrm{T}}\mathbf{e}_i) \\ &= \mathbf{s}^{\mathrm{T}}\mathbf{u} + \sum_{i=1}^{t}(\mathbf{W}^*)_{1i}^{-1}(\mathbf{d}_i^{\mathrm{T}}\mathbf{e}_i) = \mathbf{s}^{\mathrm{T}}\mathbf{u} + e', \end{aligned} \tag{12}$$

where we define $e' = \sum_{i=1}^{t}(\mathbf{W}^*)_{1i}^{-1}(\mathbf{d}_i^{\mathrm{T}}\mathbf{e}_i)$.

- Thus, we compute $r = c - c' = M\lfloor \frac{q}{2} \rfloor + e - e'$. If $|e - e'| < \lfloor \frac{q}{4} \rfloor$, then the algorithm can decrypt the ciphertext correctly and output the right message M.

To ensure the correctness of our scheme, the overall noise terms, including e and e', must be small enough to meet the condition $|e - e'| < \lfloor \frac{q}{4} \rfloor$ with overwhelming probability.

Furthermore, Learning with Errors (LWE) will be used to reduce the security of our improved CP-ABE-L scheme, the proof of which is presented in the supplement file.

We observe that the CP-ABE-L scheme will be more effective if multi-bit messages can be encrypted. Chen et al. [26] proposed a multi-key fully homomorphic encryption scheme over ring, which can encrypt a ring element rather than a single bit. With a similar algebraic structure of [26], our construction can be extended to support multi-bit messages over ring, such that the performance will be further improved. Therefore, it is a more practical solution for using ring-LWE to build our CP-ABE-L scheme in order to achieve secure CFS.

5 Parameter Optimization

In this section, we will analyze how to make cumulative errors minimum and improve the performance of our scheme, including storage and computation complexity. We also provide a performance comparison between the previous researches and our scheme.

5.1 Minimized Cumulative Errors

This paper analyzes how to choose appropriate parameters to ensure the correctness and security of the scheme. Before proceeding, we define some symbols as shown in TABLE 1.

TABLE 1
Definition of symbols in our improved CP-ABE-L scheme.

Symbol	Description
n	the security parameter
m	the dimension of Lattice
s	the total number of attributes
t	the number of attributes in decryption ($t \leq k \leq s$)
k	the number of attributes in policy
l	the number of logic in policy

We define t as the number of attributes in access policy, and T is the maximum of t, i.e., $t \leq T$. Without loss of

generality, we use $k \log m$ as the upper-bound on the growth of T for some $k > 0$, i.e., $T = O(\log m)$. Note that, the total number of attributes is not limited in our CP-ABE-L.

To ensure the correctness of decryption, we need to satisfy the following equation according to Section 4.3,

$$r = c - c' = \underbrace{e - e'}_{\text{error term}} + M\lfloor \frac{q}{2} \rfloor, \quad (13)$$

where $e' = \sum_{i=1}^{t} (\mathbf{W}^*)_{1i}^{-1} (\mathbf{d}_i^{\mathrm{T}} \mathbf{e}_i) = \sum_{i=1}^{t} \mathbf{d}_i^{\mathrm{T}} \mathbf{e}_i$ in terms of the constant vector $(\mathbf{W}^*)_1^{-1} = (1, 1, \cdots, 1)$ in Theorem 3. Then, we estimate the magnitude of error term $|e - e'| = |e - \sum_{i=1}^{t} \mathbf{d}_i^{\mathrm{T}} \mathbf{e}_i|$ as follows.

To analyze the bound of error term, we use an accumulate vector approach to combine multiple dot products of two vectors into a dot product of two long vectors. Let $\mathbf{d}_i = (d_{i,1}, \cdots, d_{i,2m}) \in \mathbb{Z}_q^{2m}$ and $\mathbf{e}_i = (e_{i,1}, \cdots, e_{i,2m}) \leftarrow \chi^{2m}$. We have $\sum_{i=1}^{t} \mathbf{d}_i^{\mathrm{T}} \mathbf{e}_i = \sum_{i=1}^{t} \sum_{j=1}^{2m} d_{i,j} \cdot e_{i,j}$. Considering that e has the same distribution as e_{ij}, we further define

$$e - e' = e - \sum_{i=1}^{t} \sum_{j=1}^{2m} d_{i,j} \cdot e_{i,j} = \sum_{k=0}^{2mt} \bar{d}_k \cdot \bar{e}_k = \bar{\mathbf{d}}^{\mathrm{T}} \bar{\mathbf{e}}, \quad (14)$$

where all elements are expressed into two vectors, $\bar{\mathbf{d}} = (\bar{d}_0, \cdots, \bar{d}_{2mt})^{\mathrm{T}} = (1, d_{1,1}, \cdots, d_{t,2m})^{\mathrm{T}} \in \mathbb{Z}_q^{2mt+1}$ and $\bar{\mathbf{e}} = (\bar{e}_0, \cdots, \bar{e}_{2mt})^{\mathrm{T}} = (e, e_{1,1}, \cdots, e_{t,2m})^{\mathrm{T}} \leftarrow \chi^{2mt+1}$.

By Lemma 2, we obtain $\|\bar{\mathbf{d}}\| \leq \sigma\sqrt{2mt+1}$ with overwhelmingly probability. Then, by Lemma 3, we have $|\bar{\mathbf{d}}^{\mathrm{T}} \bar{\mathbf{e}}| \leq \|\bar{\mathbf{d}}\|(\alpha q \omega(\sqrt{\log(2mt+1)}) + \sqrt{2mt+1}/2)$. Therefore, we have the inequation

$$\begin{aligned} |e - e'| &\leq (\sigma\sqrt{2mt+1}) \cdot (\alpha q \cdot \omega(\sqrt{\log(2mt+1)}) \\ &\quad + \sqrt{2mt+1}/2) \\ &= \alpha\sigma q \cdot \sqrt{2mt+1} \cdot \omega(\sqrt{\log(2mt+1)}) \\ &\quad + \sigma(2mt+1)/2 \\ &\leq \alpha\sigma q \cdot \omega(\sqrt{\Gamma \log \Gamma}) + \sigma \cdot \Gamma/2, \end{aligned} \quad (15)$$

where $\Gamma = 2mT + 1$. According to the requirement of successful decryption, the error term in Equation (15) is less than $\lfloor q/4 \rfloor$ with high probability. Since the error consists of two parts, the problem of allocating the proportion of them is called *error proportion allocation* (EPA). The solution to this problem must ensure that *three following lattice-generation conditions* hold with overwhelming probability.

1) For the **TrapGen** algorithm that can operate correctly, we need $m \geq \lceil 6n \log q \rceil$, $\|\tilde{\mathbf{T}}_{\mathbf{A}}\| \leq O(\sqrt{m})$ and $\|\tilde{\mathbf{T}}_{\mathbf{B}}\| \leq O(\sqrt{m})$ in terms of Theorem 2.
2) A sufficiently large σ is needed for **SampleLeft** and **SampleRight** (Definiton 1), such that we have $\sigma > \|\tilde{\mathbf{T}}_{\mathbf{A}}\| \omega(\sqrt{\log(2m)})$ and $\sigma > \|\tilde{\mathbf{T}}_{\mathbf{B}}\| \sqrt{m} \cdot \omega(\sqrt{\log m})$ used in the security proof of CP-ABE-L.
3) To ensure the security of our scheme, we need the relation $m > (n+1)\log_2 q + \omega(\log n)$ (Lemma 1). And we apply Regev's reduction (Theorem 1) to the hardness of $\mathrm{LWE}_{q,2m,\bar{\Psi}_\alpha}$, i.e., $q > 2\sqrt{2m}/\alpha$ for $\chi = \bar{\Psi}_\alpha^{2m}$.

We take n as the security parameter, round up m to the nearest larger integer and q to the nearest larger prime. To meet these conditions, we firstly divide the probability $\lfloor \frac{q}{4} \rfloor > \frac{\sqrt{2}q}{6} + \frac{q}{2^9}$ into two following items:

$$\sigma \cdot \Gamma/2 < q/2^9, \quad (16)$$

$$\alpha\sigma q \cdot \omega(\sqrt{\Gamma \log \Gamma}) < \sqrt{2}q/6. \quad (17)$$

We then set the parameters (m, σ, q, α) as follows.

1) Assume that δ is a real such that $n^{1+\delta} > \lceil (n+1)\log q + \omega(\log n) \rceil$. We set $m = 6n^{1+\delta}$ in terms of Theorem 2.
2) According to the above condition 1) and 2), we integrate the result of two sampling algorithms into the standard deviation $\sigma = m \log m/2 > m \cdot \omega(\sqrt{\log(2m)})$ if for any $c = 2$, there exists $m_0 = 2^4$, such that $\log m > c \cdot \sqrt{\log(2m)}$ for every $m \geq m_0$.
3) According to Equation (16), the modulus q satisfies

$$\begin{aligned} 2^8\sigma \cdot \Gamma &< 2^8(m \log m/2) \cdot (2m \log m + 1) \\ &< 2^8 m^2 \cdot \log m \cdot \log(2m) \\ &< 2^8 m^2 \log^2(2m) = q. \end{aligned} \quad (18)$$

4) According to Equation (17), the parameter α satisfies

$$\begin{aligned} & [\tfrac{6}{\sqrt{2}}\sigma \cdot \omega(\sqrt{\Gamma \log \Gamma})]^{-1} \\ > & [\tfrac{6}{\sqrt{2}}\sigma \cdot \sqrt{2m \log m + 1} \log(m \log m)/2]^{-1} \\ > & [\tfrac{3}{\sqrt{2}}\sigma \cdot \sqrt{2m \log(2m)} \log(m \log m)]^{-1} \\ > & [\tfrac{3}{2} m \log m \sqrt{m \log(2m)} \log(m \log m)]^{-1} \\ > & [\tfrac{3}{2} m^{3/2} \log m \cdot \log(2m) \cdot \log(m \log m)]^{-1} \\ > & [3m^{3/2} \log^3 m]^{-1} = \alpha, \end{aligned} \quad (19)$$

where $\log m \cdot \log(2m) \cdot \log(m \log m) < 2\log^3 m$, and $\omega(\sqrt{\log(2m \log m + 1)}) < \log(m \log m)/2$ if for any $c = 2$, there exists $m_0 = 2^2$, such that $\log(m \log m) > c \cdot \sqrt{\log(2m \log m + 1)}$ for every $m \geq m_0$.

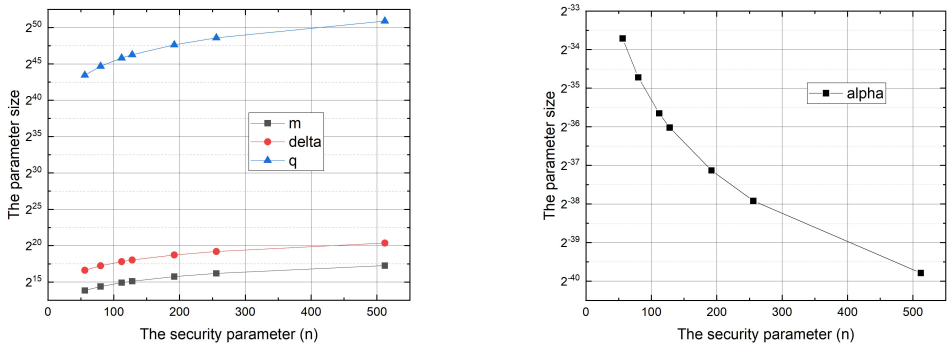
TABLE 2
Parameters set for our improved scheme.

	n	δ	m	σ	q	α
1	56	0.94	$2^{13.83}$	$2^{16.62}$	$2^{43.45}$	$2^{-33.71}$
2	80	0.87	$2^{14.39}$	$2^{17.24}$	$2^{44.68}$	$2^{-34.72}$
3	112	0.81	$2^{14.91}$	$2^{17.81}$	$2^{45.81}$	$2^{-35.65}$
4	128	0.79	$2^{15.12}$	$2^{18.04}$	$2^{46.26}$	$2^{-36.02}$
5	192	0.73	$2^{15.75}$	$2^{18.72}$	$2^{47.62}$	$2^{-37.13}$
6	256	0.70	$2^{16.19}$	$2^{19.20}$	$2^{48.58}$	$2^{-37.92}$
7	512	0.63	$2^{17.26}$	$2^{20.37}$	$2^{50.89}$	$2^{-39.79}$

As shown in TABLE 2, we combine all the above results to provide several optional parameter sets for 7 common security strengths in which n is from 56 to 512. Meanwhile, we use $q > 2\sqrt{2m}/\alpha$ in condition 3) as a test criterion to check whether the parameters conform to the requirements. Then, according to TABLE 2, we show the parameter change trends under the different security strengths (n) in Fig. 5. In Fig. 5 (a), it is easy to see that the parameters, q, δ and m, increase slowly with the increase of n. On the contrary, the parameter α decreases rapidly with the increase of n in Fig. 5 (b). Thus, the developers are allowed to choose a prime modulus q of size at no more than 64 bits. For microcontrollers with 64-bit structure, it offers fast implementations of lattice arithmetic operations.

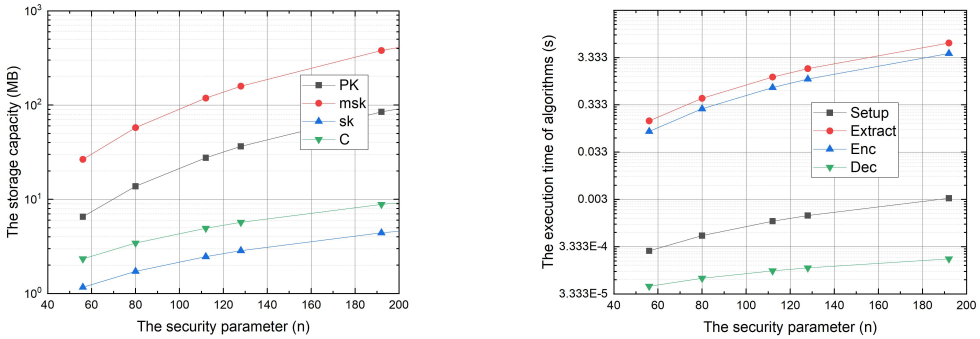
5.2 Storage Complexity

In this section, we analyze the storage complexity of the policy matrix generation algorithm and our improved CP-ABE-L scheme, as follows.



(a) (b)

Fig. 5. The parameter change trends for our construction under the different security strengths (n): (a) The trend curves of the lattice dimension m, the Gaussian parameter δ, the modulus q; and (b) the trend curve of the parameter α.



(a) (b)

Fig. 6. The complexity of our scheme, where (a) and (b) represent the storage complexity and computational complexity,respectively.

1) Storage complexity of SPMGen$(\cdot)$.

For a complete binary policy tree with depth h, we invoke the **SPMGen**$(\cdot)$ algorithm to generate policy matrix $\mathbf{W}$ by traversing the nodes of the tree. Exactly, the **SPMGen**$(\cdot)$ requires $O(h)$ stack spaces to implement the variable storage in the recursion. Thus, the storage complexity of **SPMGen**$(\cdot)$ is proportional to the depth of tree, i.e., $O(h)$.

2) Storage complexity of policy matrix $\mathbf{W}$.

For a given policy matrix $\mathbf{W} \in \mathbb{Z}_q^{k\times l}$, its storage complexity is $O(kl)$. In the worst case, the number of rows in $\mathbf{W}$ generated by **SPMGen**$(\cdot)$ is 2^{h-1} (i.e., $k \leq 2^{h-1}$), which is equal to the number of leaf nodes. Moreover, the number of columns is related to the number of AND nodes in the policy. Exactly, the number of columns is equal to the number of AND nodes plus 1 (i.e., $l \leq 2^{h-1}$), where $2^{h-1}-1$ is the maximal number of internal nodes. Thus, the storage complexity of the generated policy matrix $\mathbf{W}$ is $O(2^{2h-2})$ bits for the matrix with coefficients {-1,0,1}.

3) Storage complexity of CP-ABE-L.

The modulus q is an important parameter for storage complexity because our CP-ABE-L scheme will be developed under $\mathbb{Z}_q$. According to TABLE 2, q is a prime at most 2^{51}, i.e., $\log q \leq 51$, under all candidate security strengths. Therefore, the block of 8-byte (called 64-bit-word) can be effectively allocated for the element of lattice vector under $\mathbb{Z}_q$. Another important parameter is the security strength n. According to the NIST Special Publication 800-57 [27], 112-bit security strength (which corresponds to 2048-bit RSA keys) is considered to be secure until 2030.

In TABLE 3, we show the storage overheads of main entries, including pk, msk, sk, C and M, output by four algorithms. Assume the size of q be a fixed 64 bit, the space complexity remains quadratic on m, n, s or k for all entries except M. Moreover, the master secret key msk is a short basis, such that it can be stored on unit bit rather than byte. For clarity, we present the trend curve of storage complexity under different security strengths in Fig. 6 (a). According to the NIST specification, when $n = 112$ bit and $k = s = 10$, the length of user's private key sk and ciphertext C is about 2-4MB, and the system public key pk is 20MB. Thus, the storage overheads of our scheme against quantum attacks are larger than those of traditional cryptosystems, but it is still acceptable for the current PC storage capacity and cost. In addition, msk is 100MB, which is the heaviest cost for space. The msk is generated only once, and is handled by the system administrator, not users.

TABLE 3
Storage overheads of main entries in our scheme.

Algorithm	Entry	Size (bit)	Space Complexity
Setup	pk	$(3nm+n)\log q$	$O(nm)$
	msk	m^2	$O(m^2)$
Extract	sk	$2ms\log q$	$O(ms)$
ABE-Enc	C	$[(2m+1)k+1]\log q$	$O(km)$
ABE-Dec	M	1	$O(1)$

5.3 Computational Complexity

We next discuss the computational complexity of the policy matrix generation algorithm and our improved CP-ABE-L scheme, as follows.

1) Computational complexity of SPMGen$(\cdot)$.

For a complete binary tree with depth h, there are at most $2^{h-1}-1$ internal nodes. The **SPMGen**$(\cdot)$ algorithm is invoked as many times as the number of nodes. In the

pre-order traversal process, the algorithm will increase 1 cycle every time a internal node is processed. It can be seen from Section 4.1 that the **SPMGen**$(\cdot)$ is a recursion but does not operate on leaf nodes. Assuming that the processing efficiency of AND node is the same as that of OR node, it is easy to obtain that the computational cost of **SPMGen**$(\cdot)$ is $O(2^{h-1}-1)$.

2) Computational complexity of CP-ABE-L.

Similar to storage complexity analysis, considering the fixed 64-bit storage of q, a CPU with 64-bit word can effectively complete the algebraic operations in CP-ABE-L. This ensures that only one CPU instruction/operation is needed to perform one algebraic operation for each element in lattice vector under $\mathbb{Z}_q$. Let t_m denote the time of one multiplication or addition under $\mathbb{Z}_q$, t_s denote the time of one random number generation under $\mathbb{Z}_q$, t_{sample} denote the time of one **TrapGen** or **SampleLeft** operation. In TABLE 4, we show the computational complexity of the four algorithms in our scheme. It is easy to see that the time complexity of **Extract** and **ABE-Enc** is the fourth power formula, but one of **Setup** and **ABE-Dec** is the quadratic formula. This means that the time consumptions of the former two (**Extract** and **ABE-Enc**) are higher than that of the latter two (**Setup** and **ABE-Dec**).

TABLE 4
Execution time of each algorithm in our scheme.

Algorithm	Time Complexity
Setup	$O(nm \cdot t_s + t_{sample})$
Extract	$O(n^2ms \cdot t_m + t_{sample})$
ABE-Enc	$O(kn^2m \cdot t_m + (ln + 2mk) \cdot t_s)$
ABE-Dec	$O(tm \cdot t_m)$

Furthermore, we show the trend curve of time complexity under different security strengths in Fig. 6 (b), where a 64-bit multi-core CPU with 3.6GHz is used to perform our CP-ABE-L scheme. This means that the number of operations is 3×10^9 per second in terms of the previous assumption [3]. Under the 112-bit security strength and the same settings in the above storage complexity, the time costs of **Setup** and **ABE-Dec** algorithm are less than 1ms, and that of **Extract** and **ABE-Enc** algorithm are less than 3s. Therefore, the overall performance of our CP-ABE-L scheme is acceptable and reasonable in the context of post-quantum security, especially for the rapid decryption.

5.4 Performance Comparison

As shown in TABLE 5, we compare the performance of our scheme with that of other related works [12], [14], [16]. These schemes are lattice-based CP-ABE and argued to resist post-quantum attacks. The communication complexity takes the size of public-key (pk), secret-key (sk), ciphertext (C) and message (M) into account. Further, each of these schemes is distinguished from three aspects, Operation, Post-quantum, and Hardness. From TABLE 5, it is easy to see that the schemes in [12], [16] support single threshold operation, which would lead to low availability of the system. Moreover, the scheme [14] also has a low availability since it only supports AND operations. In addition, the PK size in the scheme [14] is relatively long, thus it would lead to high storage and communication costs. As a whole, the overall performance of our scheme is better than those of [12] and [16], but is close to that of [14].

However, the comparison results are inaccurate and incomplete in TABLE 5 if we only compare them from the perspective of mathematical formulas. The reason is that the parameters m and q in our scheme are far less than those of the schemes derived from [21], including [12], [14], [16] under the same security strength n. According to the Agrawal's Fuzzy IBE/ABE scheme in [21], the modulus q is conservatively estimated to be $q \geq m^3 \log m \cdot 2^{5T}$, where $T = O(\log m)$. When the security strength is 112-bit ($n = 112$), the modulus q is greater than $1.88 \times 2^{123} \approx 2^{124}$ for $m = 2^{15(>14.91)}$. This means that each element in $\mathbb{Z}_q$ must be stored in two 64 bits ($128 > \log q$) rather than one 64-bit in our scheme. This also implies that one 64-bit CPU instruction cannot perform one algebraic operation of each element in $\mathbb{Z}_q$ for [12], [14], [16]. Compared with these schemes, our scheme achieves shorter parameters, such as $q \geq 2^{46(>45.81)}$ for $m = 2^{15}$, from the results in TABLE 2. The benefits gained from it is that a 64-bit CPU is able to realize the storage and computation of elements in $\mathbb{Z}_q$. Thus, the whole performance of our scheme has a significant improvement in comparison with others.

6 Conclusion

In this paper, we address the problem of CFS-based data leakage on employees' remote work. We propose a new solution that integrates multiple existing technologies, including ABAC/XACML, CP-ABE-L and CFS. However, it is still a serious problem for improving the performance of these technologies. To do it, we construct a SPM that has small coefficients and generates an all-one reconstruction vector. Based on these advantages of SPM, we improve the existing CP-ABE-L scheme to guarantee data privacy against quantum attacks in the CFS. Therefore, our work significantly improves the efficiency and security of increasingly popular employees' remote work.

Acknowledgments

This work was supported by the National Key Technologies R&D Programs of China (2018YFB1402702) and the National Natural Science Foundation of China (61972032).

References

[1] NSA, "Cryptography today," *August 2015, archived on 23 November 2015, tinyurl.com/SuiteB*.
[2] N. Koblitz and A. Menezes, "A riddle wrapped in an enigma," *IEEE Security & Privacy*, vol. 14, no. 6, pp. 34–42, 2016.
[3] L. Chen, L. Chen, S. Jordan, Y.-K. Liu, D. Moody, R. Peralta, R. Perlner, and D. Smith-Tone, *Report on post-quantum cryptography*. US Department of Commerce, National Institute of Standards and Technology, 2016.
[4] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *2007 IEEE symposium on security and privacy (SP'07)*. IEEE, 2007, pp. 321–334.
[5] V. Goyal, A. Jain, O. Pandey, and A. Sahai, "Bounded ciphertext policy attribute based encryption," in *International Colloquium on Automata, Languages, and Programming*, vol. 5126. Springer, 2008, pp. 579–591.

3. The fact that the Intel Core i7-6850K Processor (15M Cache, 3.8GHz) is round 345.6GFlOPS has far more than our assumption.

TABLE 5
Performance comparison with our scheme and other related works.

Scheme	[12]	[14]	[16]	Ours
pk size	$[(2+d+s)nm+n]\log q$	$(3nm+n+ns)\log q$	$[m(d+s)+1]\log q$	$(3nm+n)\log q$
msk size	m^2	m^2	$m(d+s)$	m^2
sk size	$2m(d+s)\log q$	$2ms\log q$	$m(d+s)\log q$	$2ms\log q$
C size	$[m(d+2+k-t)+1]\log q$	$(2m+kl+1)\log q$	$[m(d+1+k-t)+1]\log q$	$[(2m+l)k+1]\log q$
M size	$\{0,1\}$	$\{0,1\}$	$\{0,1\}^n$	$\{0,1\}$
Operation	Threshold	AND	Threshold	AND,OR
Post-quantum	$\surd$	$\surd$	$\surd$	$\surd$
Hardness	LWE	LWE	Ring-LWE	LWE

[6] J. Herranz, F. Laguillaumie, and C. Ràfols, "Constant size ciphertexts in threshold attribute-based encryption," in *International Workshop on Public Key Cryptography*. Springer, 2010, pp. 19–34.

[7] S. Wang, H. Wang, J. Li, H. Wang, J. Chaudhry, M. Alazab, and H. Song, "A fast cp-abe system for cyber-physical security and privacy in mobile healthcare network," *IEEE Transactions on Industry Applications*, vol. 56, no. 4, 2020.

[8] D. Sethia, A. Shakya, R. Aggarwal, and S. Bhayana, "Constant size cp-abe with scalable revocation for resource-constrained iot devices," in *2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*. IEEE, 2019, pp. 0951–0957.

[9] Z. Liu, Z. Cao, and D. S. Wong, "Efficient generation of linear secret sharing scheme matrices from threshold access trees," *Cryptology ePrint Archive: Listing*, 2010.

[10] Y. Zhu, H. Hu, G.-J. Ahn, D. Huang, and S. Wang, "Towards temporal access control in cloud computing," in *2012 Proceedings IEEE INFOCOM*. IEEE, 2012, pp. 2576–2580.

[11] Y. Zhu, R. Yu, D. Ma, and W. C.-C. Chu, "Cryptographic attribute-based access control (ABAC) for secure decision making of dynamic policy with multiauthority attribute tokens," *IEEE Transactions on Reliability*, vol. 68, no. 4, pp. 1330–1346, 2019.

[12] J. Zhang, Z. Zhang, and A. Ge, "Ciphertext policy attribute-based encryption from lattices," in *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*. ACM, 2012, pp. 16–17.

[13] X. Boyen, "Attribute-based functional encryption on lattices," in *Theory of Cryptography Conference*. Springer, 2013, pp. 122–142.

[14] Y. Wang, "Lattice ciphertext policy attribute-based encryption in the standard model." *IJ Network Security*, vol. 16, no. 6, pp. 444–451, 2014.

[15] W. Dai, Y. Doröz, Y. Polyakov, K. Rohloff, H. Sajjadpour, E. Savaş, and B. Sunar, "Implementation and evaluation of a lattice-based key-policy abe scheme," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 5, pp. 1169–1184, 2017.

[16] Z. Chen, P. Zhang, F. Zhang, and J. Huang, "Ciphertext policy attribute-based encryption supporting unbounded attribute space from r-lwe," *TIIS*, vol. 11, no. 4, pp. 2292–2309, 2017.

[17] J. Alwen and C. Peikert, "Generating shorter bases for hard random lattices," in *26th International Symposium on Theoretical Aspects of Computer Science STACS*. IBFI Schloss Dagstuhl, 2009, pp. 75–86.

[18] N. Genise and D. Micciancio, "Faster gaussian sampling for trapdoor lattices with arbitrary modulus," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2018, pp. 174–203.

[19] S. Agrawal, D. Boneh, and X. Boyen, "Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE," in *Annual Cryptology Conference – CRYPTO'10*, vol. 6223. Springer, 2010, pp. 98–115.

[20] K. D. Gür, Y. Polyakov, K. Rohloff, G. W. Ryan, and E. Savas, "Implementation and evaluation of improved gaussian sampling for lattice trapdoors," in *Proceedings of the 6th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, 2018, pp. 61–71.

[21] S. Agrawal, X. Boyen, V. Vaikuntanathan, P. Voulgaris, and H. Wee, "Functional encryption for threshold functions (or fuzzy IBE) from lattices," in *International Workshop on Public Key Cryptography–PKC' 12*. Springer, 2012, pp. 280–297.

[22] A. Takayasu and Y. Watanabe, "Lattice-based revocable identity-based encryption with bounded decryption key exposure resistance," in *Australasian Conference on Information Security and Privacy*. Springer, 2017, pp. 184–204.

[23] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," in *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing – STOC'05*, 2005, pp. 84–93.

[24] D. Micciancio and O. Regev, "Worst-case to average-case reductions based on gaussian measures," *SIAM Journal on Computing*, vol. 37, no. 1, pp. 267–302, 2007.

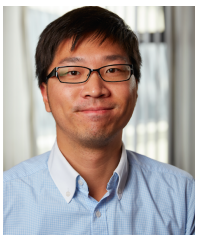
[25] R. R. Al-Dahhan, Q. Shi, G. M. Lee, and K. Kifayat, "Survey on revocation in ciphertext-policy attribute-based encryption," *Sensors*, vol. 19, no. 7, p. 1695, 2019.

[26] L. Chen, Z. Zhang, and X. Wang, "Batched multi-hop multi-key fhe from ring-lwe with compact ciphertext extension," in *Theory of Cryptography Conference*. Springer, 2017, pp. 597–627.

[27] E. Barker and Q. Dang, "NIST special publication 800-57 part 1, revision 4," *NIST, Tech. Rep*, 2016.

E Chen received the B.S. degree from the department of School of Mathematics and Physics, University of Science and Technology Beijing, China. She is currently a Ph.D. candidate with the department of School of Computer and Communication Engineering, University of Science and Technology Beijing, China. Her research interests include attribute based system and lattice cryptography. (Email: chene5546@163.com)

Yan Zhu was an Associate Professor of Computer Science with the Institute of Computer Science and Technology, Peking University, China, from 2007 to 2013. He was a Visiting Associate Professor with the Department of Computer Science and Engineering, Arizona State University, from 2008 to 2009. He was a Visiting Research Investigator with the Department of Computer and Information Science, University of Michigan-Dearborn, in 2012. He is currently a Professor with the School of Computer and Communication Engineering, University of Science and Technology Beijing, China. His research interests include cryptography, secure computation, and network security. (Email:zhuyan@ustb.edu.cn)



Kaitai Liang joined the Cybersecurity group at Delft University of Technology in 2020. Before joining TU Delft, he was an Assistant Professor in Secure Systems at the University of Surrey, UK, and an academic member of the Surrey Centre for Cyber Security. He received his PhD degree in computer science from City University of Hong Kong. His research interests are applied cryptography and information security in particular, encryption, network security, big data security, privacy-enhancing technology, and security in cloud computing (Email: Kaitai.Liang@tudelft.nl)

Hongjian Yin received the M.S. degree from the department of School of Mathematics and Statistics, Xidian University, Xi'an, China. He is currently a PhD candidate with the department of School of Computer and Communication Engineering, University of Science and Technology Beijing. His current research interests include information security and cryptography. (Email: honjanyin@163.com)